COMPUTER SCIENCE

WITH PYTHON

CLASS - XII

CENTRAL BOARD OF SECONDARY EDUCATION

Shiksha Kendra, 2, Community Centre, Preet Vihar, Delhi-110 092 India

नया आगाज़

आज समय की माँग पर
आगाज़ नया इक होगा
निरंतर योग्यता के निर्णय से
परिणाम आकलन होगा।

परिवर्तन नियम जीवन का
नियम अब नया बनेगा
अब परिणामों के भय से
नहीं बालक कोई डरेगा

निरंतर योग्यता के निर्णय से
परिणाम आकलन होगा।

बदले शिक्षा का स्वरूप
नई खिले आशा की धूप
अब किसी कोमल-से मन पर
कोई बोझ न होगा

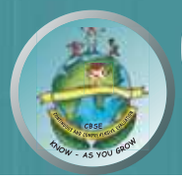
निरंतर योग्यता के निर्णय से
परिणाम आकलन होगा।
नई राह पर चलकर मंज़िल को हमें पाना है
इस नए प्रयास को हमने सफल बनाना है
बेहतर शिक्षा से बदले देश, ऐसे इसे अपनाए
शिक्षक, शिक्षा और शिक्षित
बस आगे बढते जाएँ
बस आगे बढते जाएँ
बस आगे बढते जाएँ.....

COMPUTER SCIENCE

Class-XII

CENTRAL BOARD OF SECONDARY EDUCATION

Shiksha Kendra, 2, Community Centre, Preet Vihar, Delhi-110 092 India



Computer Science

Computer Science Class–XII

Price: ₹

First Edition 2014, CBSE, India

Copies:

PUBLISHED BY : The Secretary, Central Board of Secondary Education
Shiksha Kendra, 2, Community Centre,
Preet Vihar, Delhi-110092

DESIGN, LAYOUT : Multi Graphics, 8A/101, W.E.A. Karol Bagh,
New Delhi-110005, Phone: 011-25783846

COVER DESIGN : Ankit Barodiya, Air Force Golden Jubilee Institute,
Subroto Park, New Delhi

PRINTED BY :

भारत का संविधान

उद्देशिका

हम, भारत के लोग, भारत को एक सम्पूर्ण [1][प्रभुत्व-संपन्न समाजवादी पंथनिरपेक्ष लोकतंत्रात्मक गणराज्य] बनाने के लिए, तथा उसके समस्त नागरिकों को:

सामाजिक, आर्थिक और राजनैतिक न्याय,
विचार, अभिव्यक्ति, विश्वास, धर्म
और उपासना की स्वतंत्रता,
प्रतिष्ठा और अवसर की समता
प्राप्त कराने के लिए
तथा उन सब में व्यक्ति की गरिमा
और [2][राष्ट्र की एकता और अखंडता]
सुनिश्चित करने वाली बंधुता बढ़ाने के लिए

दृढ़संकल्प होकर अपनी इस संविधान सभा में आज तारीख 26 नवम्बर, 1949 ई॰ को एतद् द्वारा इस संविधान को अंगीकृत, अधिनियमित और आत्मार्पित करते हैं।

1. संविधान (बयालीसवां संशोधन) अधिनियम, 1976 की धारा 2 द्वारा (3.1.1977) से "प्रभुत्व-संपन्न लोकतंत्रात्मक गणराज्य" के स्थान पर प्रतिस्थापित।
2. संविधान (बयालीसवां संशोधन) अधिनियम, 1976 की धारा 2 द्वारा (3.1.1977) से "राष्ट्र की एकता" के स्थान पर प्रतिस्थापित।

भाग 4 क

मूल कर्त्तव्य

51 क. मूल कर्त्तव्य – भारत के प्रत्येक नागरिक का यह कर्त्तव्य होगा कि वह –

(क) संविधान का पालन करे और उसके आदर्शों, संस्थाओं, राष्ट्रध्वज और राष्ट्रगान का आदर करे;

(ख) स्वतंत्रता के लिए हमारे राष्ट्रीय आंदोलन को प्रेरित करने वाले उच्च आदर्शों को हृदय में संजोए रखे और उनका पालन करे;

(ग) भारत की प्रभुता, एकता और अखंडता की रक्षा करे और उसे अक्षुण्ण रखे;

(घ) देश की रक्षा करे और आह्वान किए जाने पर राष्ट्र की सेवा करे;

(ङ) भारत के सभी लोगों में समरसता और समान भ्रातृत्व की भावना का निर्माण करे जो धर्म, भाषा और प्रदेश या वर्ग पर आधारित सभी भेदभाव से परे हों, ऐसी प्रथाओं का त्याग करे जो स्त्रियों के सम्मान के विरुद्ध हैं;

(च) हमारी सामासिक संस्कृति की गौरवशाली परंपरा का महत्त्व समझे और उसका परिरक्षण करे;

(छ) प्राकृतिक पर्यावरण की जिसके अंतर्गत वन, झील, नदी, और वन्य जीव हैं, रक्षा करे और उसका संवर्धन करे तथा प्राणिमात्र के प्रति दयाभाव रखे;

(ज) वैज्ञानिक दृष्टिकोण, मानववाद और ज्ञानार्जन तथा सुधार की भावना का विकास करे;

(झ) सार्वजनिक संपत्ति को सुरक्षित रखे और हिंसा से दूर रहे;

(ञ) व्यक्तिगत और सामूहिक गतिविधियों के सभी क्षेत्रों में उत्कर्ष की ओर बढ़ने का सतत प्रयास करे जिससे राष्ट्र निरंतर बढ़ते हुए प्रयत्न और उपलब्धि की नई उंचाइयों को छू ले;

[1](ट) यदि माता-पिता या संरक्षक है, छह वर्ष से चौदह वर्ष तक की आयु वाले अपने, यथास्थिति, बालक या प्रतिपाल्य के लिये शिक्षा के अवसर प्रदान करे।

1. संविधान (छयासीवां संशोधन) अधिनियम, 2002 की धारा 4 द्वारा (12.12.2002) से अंत: स्थापित।

THE CONSTITUTION OF INDIA

PREAMBLE

WE, THE PEOPLE OF INDIA, having solemnly resolved to constitute India into a [1]**[SOVEREIGN SOCIALIST SECULAR DEMOCRATIC REPUBLIC]** and to secure to all its citizens :

JUSTICE, social, economic and political;

LIBERTY of thought, expression, belief, faith and worship;

EQUALITY of status and of opportunity; and to promote among them all

FRATERNITY assuring the dignity of the individual and the[2] [unity and integrity of the Nation];

IN OUR CONSTITUENT ASSEMBLY this twenty-sixth day of November, 1949, do **HEREBY ADOPT, ENACT AND GIVE TO OURSELVES THIS CONSTITUTION.**

1. Subs, by the Constitution (Forty-Second Amendment) Act. 1976, sec. 2, for "Sovereign Democratic Republic" (w.e.f. 3.1.1977)
2. Subs, by the Constitution (Forty-Second Amendment) Act. 1976, sec. 2, for "unity of the Nation" (w.e.f. 3.1.1977)

THE CONSTITUTION OF INDIA

Chapter IV A

FUNDAMENTAL DUTIES

ARTICLE 51A

Fundamental Duties - It shall be the duty of every citizen of India-

(a) to abide by the Constitution and respect its ideals and institutions, the National Flag and the National Anthem;

(b) to cherish and follow the noble ideals which inspired our national struggle for freedom;

(c) to uphold and protect the sovereignty, unity and integrity of India;

(d) to defend the country and render national service when called upon to do so;

(e) to promote harmony and the spirit of common brotherhood amongst all the people of India transcending religious, linguistic and regional or sectional diversities; to renounce practices derogatory to the dignity of women;

(f) to value and preserve the rich heritage of our composite culture;

(g) to protect and improve the natural environment including forests, lakes, rivers, wild life and to have compassion for living creatures;

(h) to develop the scientific temper, humanism and the spirit of inquiry and reform;

(i) to safeguard public property and to abjure violence;

(j) to strive towards excellence in all spheres of individual and collective activity so that the nation constantly rises to higher levels of endeavour and achievement;

[1](k) who is a parent or guardian to provide opportunities for education to his/her child or, as the case may be, ward between age of six and forteen years.

1. Ins. by the constitution (Eighty - Sixth Amendment) Act, 2002 S.4 (w.e.f. 12.12.2002)

Foreword

This century is characterized with the emergence of knowledge based society wherein ICT plays a pivotal role. In its vision, the National Policy on ICT in School Education by MHRD, Govt. of India, states "The ICT Policy in School Education aims at preparing youth to participate creatively in the establishment, sustenance and growth of a knowledge society leading to all round socio economic development of the nation and global competitiveness". The policy envisages three stages of ICT implementations at school level - ICT literacy and Competency Enhancement, IT enabled teaching-learning, and introduction of ICT related elective subjects at Senior Secondary level.

With this backdrop a major paradigm shift is imperative in imparting ICT- enabled instructions, collaborative learning, multidisciplinary problem-solving and promoting critical thinking skills as envisaged in the National curriculum framework 2005. Foundation of these skills is laid at school level.

Ever since the invention of Charles Babbage's difference engine in 1822, computers have required a means of instructing them to perform a specific task. This is known as a programming language. Programs in computer programming language prepare people to write and design computer software. Computer languages were first composed of a series of steps to wire a particular program; these morphed into a series of steps keyed into the computer and then executed; later these languages acquired advanced features such as logical branching and object orientation.

Syllabus of Computer Sciences has been revisited accordingly with a focus on generic concepts with domain specific practical experiments and projects to ensure conceptual knowledge with practical skills. Learning to write programs stretches your mind, and helps you think better, creates a way of thinking about things that is helpful in all domains. Since Computers have permeated in every walk of life such as launching of satellites, e-trading, e-business and social networking therefore it is imperative to study programming languages.

I am happy to release Computer Science Book for Class - XII. I would like to express my deep appreciation to the text book development team for their contribution. Appreciation is also due to Dr. Sadhana Parashar, Prof. & Director (Academics, Research, Training and Innovation) and Dr. Kshipra Verma, Education Officer, CBSE in bringing out this publication.

It is hoped that all students and teachers will benefit by making best use of this publication. Their feedback will be highly appreciated for further improvement.

Vineet Joshi

Chairman, CBSE

Acknowledgements

CBSE ADVISORS

❖ **Shri Vineet Joshi, Chairman, CBSE**

❖ **Dr. Sadhana Parashar, Prof. & Director**
(Academics, Research, Training and Innovation)

DEVELOPMENT TEAM

❖ **Ms. Anju Gupta, Rukmini Devi Public School, Pitam Pura, New Delhi.**

❖ **Ms. Mohini Arora, Air Force Golden Jubilee Institute, Subroto Park, New Delhi.**

❖ **Ms. S. Meena, Sachdeva Public School, Pitam Pura, New Delhi.**

❖ **Ms. Shally Arora, Delhi Public School, Gurgaon.**

❖ **Ms. Kshipra Verma, Education Officer, CBSE, New Delhi.**

CHIEF EDITOR & MEMBER COORDINATOR

❖ **Ms. Kshipra Verma, Education Officer, CBSE, New Delhi.**

Computer Science

Content

Unit-1 : Review of Python & Concept of Oops

Chapter-1 : Review of Python

Learning Objectives:

At the end of this chapter the students will be able to understand:

- Interactive Mode
- Script Mode
- Data Types
- Functions in Python
- Sequential Statement
- Selective Statements
- Looping Statements
- String and String Methods
- List and List Methods
- Tuple and Tuple Methods
- Dictionary and Dictionary Methods

Introduction:

We have learnt Python programming language in the 11th class and continue to learn the same language program in class 12th also. We also know that Python is a high level language and we need to have Python interpreter installed in our computer to write and run Python program. Python is also considered as an interpreted language because Python programs are executed by an interpreter. We also learn that Python shell can be used in two ways, viz., interactive mode and script mode.

Interactive Mode: Interactive Mode, as the name suggests, allows us to interact with OS. Hear, when we type Python statement, interpreter displays the result(s) immediately. That means, when we type Python expression / statement / command after the prompt (>>>), the Python immediately responses with the output of it. Let's see what will happen when we type print "WELCOME TO PYTHON PROGRAMMING" after the prompt.

```
>>>print "WELCOME TO PYTHON PROGRAMMING"
WELCOME TO PYTHON PROGRAMMING
```

Example:

```
>>> print 5+10
15
>>> x=10
```

```
>>> y=20
>>> print x*y
200
```

Script Mode: In script mode, we type Python program in a file and then use interpreter to execute the content of the file. Working in interactive mode is convenient for beginners and for testing small pieces of code, as one can test them immediately. But for coding of more than few lines, we should always save our code so that it can be modified and reused.

Python, in interactive mode, is good enough to learn, experiment or explore, but its only drawback is that we cannot save the statements and have to retype all the statements once again to re-run them.

Example: Input any two numbers and to find Quotient and Remainder.

Code: (Script mode)

```
a = input ("Enter first number")
b = input ("Enter second number")
print "Quotient", a/b
print "Remainder", a%b
```

Output: (Interactive Mode)

```
Enter first number10
Enter second number3
Quotient 3
Remainder 1
```

Variables and Types: One of the most powerful features of a programming language is the ability to manipulate variables. When we create a program, we often like to store values so that it can be used later. We use objects (variables) to capture data, which then can be manipulated by computer to provide information. By now, we know that object/ variable is a name which refers to a value.

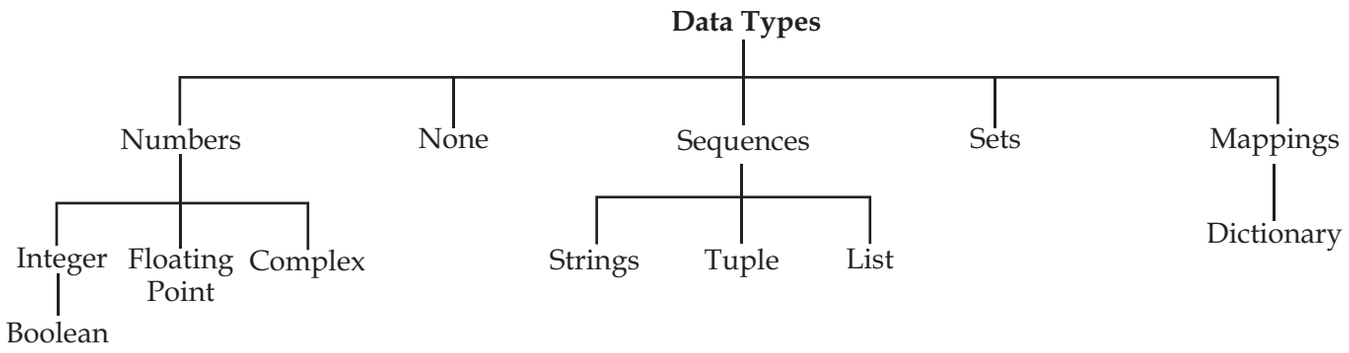
Every object has:

- An Identity,
- A type, and
- A value.

A. **Identity** of the object is its address in memory and does not get change once it is created. We may know it by typing id (variable)

We would be referring to objects as variable for now.

B. **Type** (i.e data type) is a set of values, and the allowable operations on those values. It can be one of the following:



1. **Number:** Number data type stores Numerical Values. This data type is immutable i.e. value of its object cannot be changed. Numbers are of three different types:
   - Integer & Long (to store whole numbers i.e. decimal digits without fraction part)
   - Float/floating point (to store numbers with fraction part)
   - Complex (to store real and imaginary part)
2. **None:** This is special data type with a single value. It is used to signify the absence of value/false in a situation. It is represented by None.
3. **Sequence:** A sequence is an ordered collection of items, indexed by positive integers. It is a combination of mutable (a mutable variable is one, whose value may change) and immutable (an immutable variable is one, whose value may not change) data types. There are three types of sequence data type available in Python, they are Strings, Lists & Tuples.
   - 3.1 **String-** is an ordered sequence of letters/characters. They are enclosed in single quotes (' ') or double quotes (" "). The quotes are not part of string. They only tell the computer about where the string constant begins and ends. They can have any character or sign, including space in them. These are **immutable.** A string with length 1 represents a character in Python.
   - 3.2 **Lists:** List is also a sequence of values of any type. Values in the list are called elements / items. These are **mutable** and indexed/ordered. List is enclosed in square brackets ([]).
   - 3.3 **Tuples:** Tuples are a sequence of values of any type and are indexed by integers. They are immutable. Tuples are enclosed in ().
4. **Sets:** Set is unordered collection of values of any type with no duplicate entry. It is immutable.
5. **Mapping:** This data type is unordered and mutable. Dictionaries fall under Mappings.
   - 5.1 **Dictionaries:** It can store any number of python objects. What they store is a **key -value** pairs, which are accessed using key. Dictionary is enclosed in curly brackets ({}).

C. **Value:** Value is any number or a letter or string. To bind value to a variable, we use assignment operator (=).

Keywords - are used to give some special meaning to the interpreter and are used by Python interpreter to recognize the structure of program.

A partial list of keywords in Python 2.7 is

and	del	from	not
while	as	elif	global
or	with	assert	else
if	pass	Yield	break
except	import	print	class
exec	in	Raise	continue
finally	is	return	def
for	lambda	try	

Operators and Operands

Operators are special symbols that represent computation like addition and multiplication. The values that the operator is applied to are called operands. Operators when applied on operands form an expression. Operators are categorized as Arithmetic, Relational, Logical and Assignment. Following is the partial list of operators:

Mathematical/Arithmetic operators: +, -, *, /, %, ** and //.

Relational operators: <, <=, >, >=, != or <> and ==.

Logical operators: or, and, and not

Assignment Operator: =, +=, -=, *=, /=, %=, **= and //=

Input and Output

Program need to interact with end user to accomplish the desired task, this is done using Input-Output facility. Input means the data entered by user (end user) of the program. In python, raw_input() and input ( ) functions are available for input.

Syntax of raw_input() is:

Variable = raw_input ([prompt])

Example:

```
>>>x = raw_input ('Enter your name: ')
Enter your name: ABC
```

Example:

```
y = int(raw_input ("enter your roll no"))
```

will convert the accepted string into integer before assigning to 'y'.

Syntax for input() is:

Variable = input ([prompt])

Example:

x = input ('enter data:')

Enter data: 2+ ½.0

Will supply 2.5 to x

Print: This statement is used to display results.

Syntax:

print expression/constant/variable

Example:

>>> print "Hello"

Hello

Comments: As the program gets bigger and more complicated, it becomes difficult to read it and difficult to look at a piece of code and to make out what it is doing by just looking at it. So it is good to add notes to the code, while writing it. These notes are known as comments. In Python, comments start with '#' symbol. Anything written after # in a line is ignored by interpreter. For more than one line comments, we use the following;

- Place '#' in front of each line, or
- Use triple quoted string. (""" """)

Functions in Python: A function is named sequence of statement(s) that performs a computation. It contains line of code(s) that are executed sequentially from top to bottom by Python interpreter. They are the most important building block for any software in Python. For working in script mode, we need to write the Python code in functions and save it in the file having .py extension. Functions can be categorized as belonging to

- Modules
- Built in
- User Defined

1. **Module:**

   A module is a file containing Python definitions (i.e. functions) and statements. Standard library of Python is extended as module(s) to a Programmer. Definitions from the module can be used into code of Program. To use these modules in a program, programmer needs to import the module. Once we import a module, we can reference (use) to any of its functions or variables in our code. There are two ways to import a module in our program, they are

- import
- from

Import: It is simplest and most common way to use modules in our code.

Syntax:

import modulename1 [, module name 2, ---------]

Example: Input any number and to find square and square root.

Example:

```
import math
x = input ("Enter any number")
y = math.sqrt(x)
a = math.pow(x,2)
print "Square Root value=",y
print "square value=",a
```

output:

```
Enter any number25
Square Root value= 5.0
square value= 625.0
```

From statement: It is used to get a specific function in the code instead of complete file. If we know beforehand which function(s), we will be needing, then we may use 'from'. For modules having large number of functions, it is recommended to use from instead of import.

Syntax

>>> from modulename import functionname [, functionname…..]

from modulename import *

will import everything from the file.

Example: Input any number and to find square and square root.

Example:

```
from math import sqrt,pow
x=input("Enter any number")
y=sqrt(x)         #without using math
a=pow(x,2)        #without using math
print "Square Root value =",y
print "square value =",a
```

Enter any number100

Square Root value = 10.0

square value = 10000.0

The functions available in math module are:

ceil() floor() fabs() exp() log() log10() pow() sqrt() cos() sin() tan() degrees() radians()

Some functions from random module are:

random() randint() uniform() randrange()

2. Built in Function:

Built in functions are the function(s) that are built into Python and can be accessed by Programmer. These are always available and for using them, we don't have to import any module (file). Python has a small set of built-in functions as most of the functions have been partitioned to modules. This was done to keep core language precise.

abs() max() min() bin() divmod() len() range() round() bool() chr() float() int() long() str( ) type( ) id( ) tuple( )

3. User Defined Functions:

In Python, it is also possible for programmer to write their own function(s). These functions can then be combined to form module which can be used in other programs by importing them. To define a function, keyword 'def' is used. After the keyword comes an identifier i.e. name of the function, followed by parenthesized list of parameters and the colon which ends up the line, followed by the block of statement(s) that are the part of function.

Syntax:

def NAME ([PARAMETER1, PARAMETER2, …..]):

#Square brackets include optional part of statement

statement(s)

Example: To find simple interest using function.

Example:

```
def SI(P,R,T):
    return(P*R*T)
```

Output:

```
>>> SI(1000,2,10)
20000
```

Parameters and Arguments

Parameters are the value(s) provided in the parenthesis when we write function header. These are the values required by function to work. If there is more than one value required by the function to work on, then, all of them will be listed in parameter list separated by comma.

Example: def SI (P,R,T):

Arguments are the value(s) provided in function call/invoke statement. List of arguments should be supplied in same way as parameters are listed. Bounding of parameters to arguments is done 1:1, and so there should be same number and type of arguments as mentioned in parameter list.

Example: Arguments in function call

```
>>> SI (1000,2,10)
```

1000,2,10 are arguments. An argument can be constant, variable, or expression.

Example: Write the output from the following function:

```
def SI(p,r=10,t=5):
   return(p*r*t/100)
```

if we use following call statement:

```
SI(10000)
SI(20000,5)
SI(50000,7,3)
```

Output

```
>>> SI(10000)
5000
>>> SI(20000,5)
5000
>>> SI(50000,7,3)
10500
```

Flow of Execution

Execution always begins at the first statement of the program. Statements are executed one after the other from top to bottom. Further, the way of execution of the program shall be categorized into three ways; (i) sequence statements, (ii) selection statements, and (iii) iteration or looping statements.

Sequence statements: In this program, all the instructions are executed one after another.

Example:

Program to find area of the circle.

```
r = input("enter any radius of the circle")
a = 3.14*r*r
print "Area=",a
```

output:

enter any radius of the circle7

Area = 153.86

In the above program, all three statements are executed one after another.

Selective Statements: In this program, some portion of the program is executed based upon the conditional test. If the conditional test is true, compiler will execute some part of the program, otherwise it will execute other part of the program. This is implemented in python using if statement.

Syntax:

```
if (condition):
    statements
else
    statements
```

(or)

```
if (condition):
    statements
elif (condition):
    statements
else:
    Statements
```

Example:

1. Program to find the simple interest based upon number of years. If number of years is more than 12 rate of interest is 10 otherwise 15.

   Code:

```
p = input("Enter any principle amount")
t = input("Enter any time")
if (t>10):
    si = p*t*10/100
else:
    si = p*t*15/100
print "Simple Interest = ",si
```

   output:

   Enter any principle amount 3000

   Enter any time12

   Simple Interest = 3600

2. Write a program to input any choice and to implement the following.

Choice	Find
1.	Area of square
2.	Area of rectangle
3.	Area of triangle

Code:

```
c = input ("Enter any Choice")
if(c==1):
    s = input("enter any side of the square")
    a = s*s
    print"Area = ",a
elif(c==2):
    l = input("enter length")
    b = input("enter breadth")
    a = l*b
    print"Area = ",a
elif(c==3):
     x = input("enter first side of triangle")
      y = input("enter second side of triangle")
      z = input("enter third side of triangle")
      s = (x+y+z)/2
      A = ((s-x)*(s-y)*(s-z))**0.5
      print"Area=",A
else:
    print "Wrong input"
```

Output:

```
Enter any Choice2
enter length4
enter breadth6
Area = 24
```

Iterative statements: In some programs, certain set of statements are executed again and again based upon conditional test. i.e executed more than one time. This type of execution is called looping or iteration. Looping statement in python is implemented by using 'for' and 'while' statement.

Syntax: (for loop)

```
for variable in range(start,stop+1,step):
        statements
```

Syntax: (while loop)

```
while (condition):
        Statements
```

Example:

1. Write a program to input any number and to print all natural numbers up to given number.

   Code:

```
n = input("enter any number")
for i in range(1,n+1):
     print i,
```

   Output:

```
enter any number10
1 2 3 4 5 6 7 8 9 10
```

2. Write a program to input any number and to find sum of all natural numbers up to given number.

   Code:

```
n = input("Enter any number")
sum= 0
for i in range(1,n+1):
    sum = sum+i
print "sum=",sum
```

   Output:

```
Enter any number5
sum = 15
```

3. Write a program to input any number and to find reverse of that number.

   Code:

```
n = input("Enter any number")
r = 0
while(n>0):
      r = r*10+n%10
      n = n/10
```

```
print "reverse number is", r
```

Output:

```
>>>
Enter any number345
reverse number is 543
>>>
```

Example: Write the output from the following code:

1.
```
sum = 0
for i in range(1,11,2):
    sum+ = i
print "sum = ", sum
```
output:

sum = 25

2.
```
sum = 0
i = 4
while (i<=20):
    sum+=i
    i+= 4
print "Sum = ",sum
```
output:

Sum = 60

Example: Interchange for loop into while loop

1.
```
for i in range(10,26,2):
    print i
```
Ans:
```
i=10
while(i<26):
    print i
    i+=2
```

2.
```
s=0
for i in range(10,50,10):
    s + =i
```

```
print " Sum= ", s
```

Ans:

```
s = 0
i = 10
while(i<50):
    s+ = i
    i+ = 10
print "Sum=",s
```

Example: Interchange while loop in to for loop.

```
i = 5
s = 0
while (i<25):
    s+ = i
    i + = 5
print " Sum =", s
```

Ans:

```
s = 0
for i in range(5,25,5):
    s+=i
print "Sum = ", s
```

Example: How many times following loop will execute.

1.
```
for i in range(10,50,5):
    print i
```
Ans:

i values are 10,15,20,25,30,35,40,45

8 times

2.
```
i=4
while(i<25):
    print i
    i+=4
```
Ans:

i values are 4,8,12,16,20,24

6 times

String:

In python, consecutive sequence of characters is known as a string. An individual character in a string is accessed using a subscript (index). The subscript should always be an integer (positive or negative) and starts from 0. A literal/constant value to a string can be assigned using a single quotes, double quotes or triple quotes. Strings are immutable i.e. the contents of the string cannot be changed after it is created.

Strings Operations:

+ (Concatenation)

* (Repetition)

in (Membership)

not in

range (start, stop[,step])

slice[n:m]

Example: Write the output from the following code:

1. A = 'Global'
   B = 'warming'
   print A+B
   Ans: Globalwarming
2. A = 'Global'
   Print 3*A
   Ans: 'GlobalGlobalGlobal'
3. A='Global'
   'o' in A
   Ans: True
4. A='Global'
   'g' in A
   Ans: False
5. A='Global'
   'o' not in A
   Ans: False
6. A='Global'
   'g' not in A
   Ans: True

String methods & built in functions:

len() capitalize() find(sub[,start[, end]]) isalnum() isalpha()

isdigit() lower() islower() isupper() upper() lstrip()
rstrip() isspace() istitle() replace(old,new) join ()
swapcase() partition(sep) split([sep[,maxsplit]])
Example:

```
>>> s='Congratulations'
>>> len(s)
15
>>> s.capitalize()
'Congratulations'
>>> s.find('al')
-1
>>> s.find('la')
8
>>> s[0].isalnum()
True
>>> s[0].isalpha()
True
>>> s[0].isdigit()
False
>>> s.lower()
'congratulations'
>>> s.upper()
'CONGRATULATIONS'
>>> s[0].isupper()
True
>>> s[1].isupper()
False
>>> s.replace('a','@')
'Congr@tul@tions'
>>> s.isspace()
False
>>> s.swapcase()
'cONGRATULATIONS'
>>> s.partition('a')
('Congr', 'a', 'tulations')
>>> s.split('ra',4)
```

```
['Cong', 'tulations']
>>> s.split('a')
['Congr', 'tul', 'tions']
>>> a=' abc '
>>> a.lstrip()
'abc '
>>> a.rstrip()
' abc'
```

Examples:

Example: Write a program to input any string and count number of uppercase and lowercase letters.

Code:

```
s=raw_input("Enter any String")
rint s
u=0
l=0
i=0
while i<len(s):
    if (s[i].islower()==True):
        l+=1
    if (s[i].isupper()==True):
        u+=1
        i+=1
    print "Total upper case letters :", u
    print "Total Lower case letters :", l
```

Output:

```
Enter any String Python PROG
Python PROG
Total upper case letters: 5
Total Lower case letters: 5
```

Example:

Write the output from the following code:

```
s = 'Indian FESTIVALS'
i = 0
while i<len(s):
```

```
if (s[i].islower()):
        print s[i].upper(),
if (s[i].isupper()):
        print s[i].lower(),
i + =1
```

Ans:

iNDIANfestivals

List:

Like a string, list is a sequence of values. In a string, the values are characters, whereas in a list, they can be of any type. The values in the list are called elements or items or members. It is an ordered set of values enclosed in square brackets [ ]. Values in the list can be modified, i.e. it is mutable. As it is set of values, we can use index in square brackets [ ] to identify a value belonging to it.

List Slices: Slice operator works on list. This is used to display more than one selected values on the output screen. Slices are treated as boundaries and the result will contain all the elements between boundaries.

Syntax:

Seq = L [start: stop: step]

Where start, stop & step - all three are optional. If you omit first index, slice starts from '0' and omitting of stop will take it to end. Default value of step is 1.

Example:

```
>>> L=[10,20,30,40,50]
>>> L1=L[2:4]
>>> print L1
[30, 40]
```

List Methods:

append() extend () pop() del() remove()
insert() sort() reverse() len()

Example:

```
>>> L=[500,1000,1500,2000]
>>> L.append(2500)
>>> print L
[500, 1000, 1500, 2000, 2500]
>>> L1=[3000,3500]
>>> L.extend(L1)
```

```
>>> print L
[500, 1000, 1500, 2000, 2500, 3000, 3500]
>>> L.pop()
3500
>>> L.pop(3)
2000
>>> print L
[500, 1000, 1500, 2500, 3000]
>>> del L[2]
>>> print L
[500, 1000, 2500, 3000]
>>> L.remove(1000)
>>> print L
[500, 2500, 3000]
>>> L.insert(3,3500)
>>> print L
[500, 2500, 3000, 3500]
>>> L.reverse()
>>> print L
[3500, 3000, 2500, 500]
>>> L.sort()
>>> print L
[500, 2500, 3000, 3500]
>>> print len(L)
4
```

Note: Operator + & * can also be applied on the lists. + is used to concatenate the two lists and * is used to repeat the list given number of times.

Example:

```
>>> l=[10,20,30]
>>> m=[40,50]
>>> l=l+m
>>> print l
[10, 20, 30, 40, 50]
>>> b=m*3
```

```
>>> print b
[40, 50, 40, 50, 40, 50]
```

Dictionaries:

A dictionary is like a list, but more in general. In a list, index value is an integer, while in a dictionary index value can be any other data type and are called keys. The key will be used as a string as it is easy to recall. A dictionary is an extremely useful data storage construct for storing and retrieving all key value pairs, where each element is accessed (or indexed) by a unique key. However, dictionary keys are not in sequences and hence maintain no left-to-right order.

Key-value pair: We can refer to a dictionary as a mapping between a set of indices (which are called keys) and a set of values. Each key maps a value. The association of a key and a value is called a key-value pair.

Syntax:

my_dict = {'key1': 'value1','key2': 'value2','key3': 'value3'…'keyn': 'valuen'}

Note: Dictionary is created by using curly brackets(ie. {}).

Dictionary methods:

cmp() len() clear() get() has_key()
items() keys() values() update() dict()

Example:

```
>>> month=dict()
>>> print month
{}
>>> month["one"]="January"
>>> month["two"]="Feb"
>>> print month
{'two': 'Feb', 'one': 'January'}
>>> len(month)
2
>>> month.get("one")
'January'
>>> month.get("one","feb")
'January'
>>> month.keys()
['two', 'one']
>>> month.has_key("one")
True
```

```
>>> month.has_key("three")
False
>>> month.items()
[('two', 'Feb'), ('one', 'January')]
>>> month.values()
['Feb', 'January']
>>> m=month
>>> cmp(month,m)
0
>>> n=dict()
>>> cmp(m,n)
1
>>> cmp(n,m)
-1
>>> m.clear()
>>> print m
{}
```

Tuples:

A tuple is a sequence of values, which can be of any type and they are indexed by integer. Tuples are just like list, but we can't change values of tuples in place. Thus tuples are immutable. The index value of tuple starts from 0.

A tuple consists of a number of values separated by commas. For example:

```
>>> T=10, 20, 30, 40
>>> print T
(10, 20, 30, 40)
```

But in the result, same tuple is printed using parentheses. To create a tuple with single element, we have to use final comma. A value within the parenthesis is not tuple.

Tuple Slices: Slice operator works on Tuple also. This is used to display more than one selected value on the output screen. Slices are treated as boundaries and the result will contain all the elements between boundaries.

Syntax:

Seq = T [start: stop: step]

Where start, stop & step - all three are optional. If we omit first index, slice starts from '0'. On omitting stop, slice will take it to end. Default value of step is 1.

Example:

```
>>> T=(10,20,30,40,50)
>>> T1=T[2:4]
>>> print T1
(30, 40)
```

In the above example, starting position is 2 and ending position is 3(4-1), so the selected elements are 30 & 40.

Tuple functions:

cmp( ) len( ) max( ) min( ) tuple( )

Example:

```
>>> T=tuple()
>>> print T
()
>>> T=["mon","tue","wed","thu","fri","sat","sun"]
>>> print T
['mon', 'tue', 'wed', 'thu', 'fri', 'sat', 'sun']
>>> len(T)
7
>>> min(T)
'fri'
>>> max(T)
'wed'
>>> T1=T
>>> T2=(10,20,30)
>>> cmp(T,T1)
0
>>> cmp(T2,T1)
1
>>> cmp(T1,T2)
-1
```

LET'S REVISE

Interactive Mode: Interactive Mode, as the name suggests, allows us to interact with OS.

Script Mode: In script mode, we type Python program in a file and then use interpreter to execute the content of the file.

Number: Number data type stores Numerical Values.

Sequence: A sequence is an ordered collection of items, indexed by positive integers.

Arithmetic operators: +, -, *, /, %, ** and //.

Relational operators: <, <=, >, >=, != or <> and ==.

Logical operators: or, and, and not

Assignment Operator: =, +=, -=, *=, /=, %=, **= and //=

Functions in Python: A function is named sequence of statement(s) that performs a computation.

Module: A module is a file containing Python definitions (i.e. functions) and statements. Standard library of Python is extended as module(s) to a Programmer.

String: In python, consecutive sequence of characters is known as a string. An individual character in a string is accessed using a subscript (index).

List: Like a string, list is a sequence of values. List can be of any type.

Dictionaries: A dictionary is like a list, but more in general. In a list, index value is an integer, while in a dictionary index value can be any other data type and are called keys.

Tuples: A tuple is a sequence of values, which can be of any type and they are indexed by integer.

EXERCISE

1. Write the output from the following code:

   a)
   ```
   x = 10
   y = 20
   if (x>y):
       print x+y
   else:
       print x-y
   ```

   b)
   ```
   print "Inspirational stories \n for \t Children"
   ```

   c)
   ```
   s = 0
   for I in range(10,2,-2):
       s+=I
   print "sum= ",s
   ```

   d)
   ```
   n = 50
   i = 5
   s = 0
   while i<n:
       s+ = i
       i+ = 10
   print "i=",i
   print "sum=",s
   ```

   e)
   ```
   y = 2000
   if (i%4==0):
       print "Leep Year"
   else:
       print "Not leep year"
   ```

2. Write for statement to print the following series:
   a) 10,20,30…….300
   b) 105,98,91,….7

3. Write the while loop to print the following series:
   a) 5,10,15,…100
   b) 100,98,96,…2

4. How many times are the following loop executedσ

   a) for a in range(100,10,-10):
      print a

   b) i = 100
      while(i<=200):
      print i
      i + =20

   c) for b in (1,10):
      print b

   d) i = 4
      while (i>=4):
      print i
      i+ = 10

   f) i=2
      while (i<=25)
      print i

5. Rewrite the following for loop into while loop:

   a) for a in range(25,500,25):
      print a

   b) for a in range(90,9,-9):
      print a

6. Rewrite the following while loop into for loop:

   a) i = 10
      while i<250:
      print i
      i = i+50

   b) i=88
      while(i>=8):
      print i
      i- = 8

7. Which command is used to convert text into integer valueσ

8. Find the errors from the following code.

a.
```
T=[a,b,c]
Print T
```

b.
```
for i in 1 to 100 :
    print I
```

c.
```
 i=10 ;
while [i<=n] :
    print i
   i+=10
```

d.
```
 if (a>b)
    print a:
else if (a<b)
 print b:
 else
 print "both are equal"
```

9. Find the output from the following code:

```
L=[100,200,300,400,500]
L1=L[2:4]
print L1
L2=L[1:5]
print L2
L2. .extend(L1)
print L2
```

10. Write program to input any number and to print all factors of that number.

11. Write a program to input any number and to check whether given number is Armstrong or not. (Armstrong 1,153,etc. 13 =1 , 13+53 +33 =153)

12. Write a program to input employee no, name basic pay and to find HRA, DA and netpay.

Basic pay	Hra	Da
>100000	15%	8%
<=100000&>50000	10%	5%
<=50000	5%	3%

13. Write a program to find all prime numbers up to given number.
14. Write a program to convert decimal number to binary.
15. Write a program to convert binary to decimal.
16. Write a program to input two complex numbers and to find sum of the given complex numbers.
17. Write a program to input two complex numbers and to implement multiplication of the given complex numbers.
18. Write a program to find sum of two distances with feet and inches.
19. Write a program to find difference between two times with hours, minutes and seconds.
20. Write a program to find the sum of all digits of the given number.
21. Write a program to find the reverse of that number.
22. Write a program to input username and password and to check whether the given username and password are correct or not.
23. Which string method is used to implement the following:
    a) To count the number of characters in the string.
    b) To change the first character of the string in capital letter.
    c) To check whether given character is letter or a number.
    d) To change lower case to upper case letter.
    e) Change one character into another character.
24. Write a program to input any string and to find number of words in the string.
25. Write a program to input any two strings and to check whether given strings are equal are not.
26. Differentiate between tuple and list.
27. Write a program to input n numbers and to insert any number in a particular position.
28. Write a program to input n numbers and to search any number from the list.
29. Write a program to input n customer name and phone numbers.
30. Write a program to search input any customer name and display customer phone number if the customer name is exist in the list.
31. Explain in detail about cmp() function.
32. Write a program to input n numbers and to reverse the set of numbers without using functions.

Chapter-2: Concept of Object Oriented Programming

Learning Objectives:

At the end of this chapter the students will be able to:

- Understand about Object Oriented Programming(OOP) classes and objects
- Know the concepts related to OOP
  - Objects
  - Classes
  - Encapsulation
  - Data Hiding
  - Abstraction
  - Polymorphism
  - Inheritance
- Know about the advantages of OOP over earlier programming methodologies

An object-oriented programming (OOP) is a programming language model which is organized around "objects" rather than "actions" and data rather than logic. Before the introduction of the Object Oriented Programming paradigm, a program was viewed as a logical procedure that takes input data, processes it, and produces output. But in case of OOP a problem is viewed in terms of objects rather than procedure for doing it. Now the question arises what is an objectσ

An object can be anything that we notice around us. It can be a person (described by name, address, date of Birth etc, his typing speed), a cup (described by size , color , price etc.) , a car (described by model , color , engine etc., its mileage, speed) and so on. In fact it can be an identifiable entity. The whole idea behind an object oriented model is to make programming closer to they real world thereby making it a very natural way of programming. The core of pure object-oriented programming is to combine into a single unit both data and functions or methods that operate on that data.

Simula was the first object-oriented programming language. Java, Python, C++, Visual Basic, .NET and Ruby are the most popular OOP languages today.

Basic Concepts of Object Oriented Programming

The basic concepts related to OOP are as follows:

1. Objects
2. Classes

3. Encapsulation
4. Abstraction
5. Data Hiding
6. Polymorphism
7. Inheritance

Object

An object is the basic key concept of Object Oriented Programming. As mentioned before it can be anything around us - a person, place, any activity or any other identifiable entity. Every object is characterised by:

- ***Identity:*** This is the name that identifies an object. For example a Student is the name given to anybody who is pursuing a course. Or an i-phone is a mobile phone that has been launched by Apple Inc.
- ***Properties:*** These are the features or attributes of the object. For example a student will have his name age, class, date of birth etc. as his attributes or properties. A mobile phone has model, color, price as its properties.
- ***Behaviour:*** The behaviour of an object signifies what all functions an object can perform. For example a student can pass or fail the examination. A mobile phone can click and store photographs (behave like a camera).

So an object clearly defines an entity in terms of its properties and behaviour. Consider an example of an object - Windows mobile phone. This phone has certain properties and certain functions which are different from any other mobile phone- say an Android phone. Both are mobile phones and so possess common features that every mobile phone should have but yet they have their own properties and behaviours. The data of a particular object can be accessed by functions associated with that object only. The functions of one object cannot access the data of another object.

Classes

A class is group of objects with same attributes and common behaviours. It is basically a blueprint to create objects. An object is a basic key concept of OOP but classes provide an ability to generalize similar type of objects. Both data and functions operating on the data are bundled as a unit in a class for the same category of objects. Here to explain the term 'same category of object', let us take the example of mobile phone. A Windows phone, Android phone and i-phone, all fall into the category of mobile phones. All of these are instances of a class, say Mobile_phone and are called objects.

Similarly we can have another example where students named Rani and Ravish are objects. They have properties like name, date of birth, address, class, marks etc. and the behaviour can be giving examinations. Anybody pursuing any course, giving any type of examination will come into the category of students. So a student is said to be a class as they share common properties and behaviours. Although a

student can be a school student, a college student or a university student or a student pursuing a music course and so on, yet all of these have some properties and behaviours in common which will form a class. An analogy is that you can have variables of type int which translates to saying that variables that store integers are variables which are instances (objects) of the int class.

A real instance of a class is called an object and creating the new object is called instantiation. Objects can also have functionality by using functions that belong to a class. Such functions are called methods of the class. This terminology is important because it helps us to differentiate between functions and variables which are independent and those which belong to a class or object. Collectively, the fields and methods can be referred to as the attributes of that class. Let us take the example of the class Mobile_phone which is represented in the block diagram below:

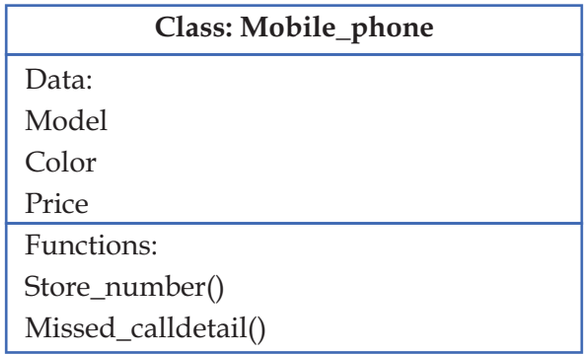


Fig 1: Class Mobile_phone

A class is defined before the creation of objects of its type. The objects are then created as instances of this class as shown in the figure below.

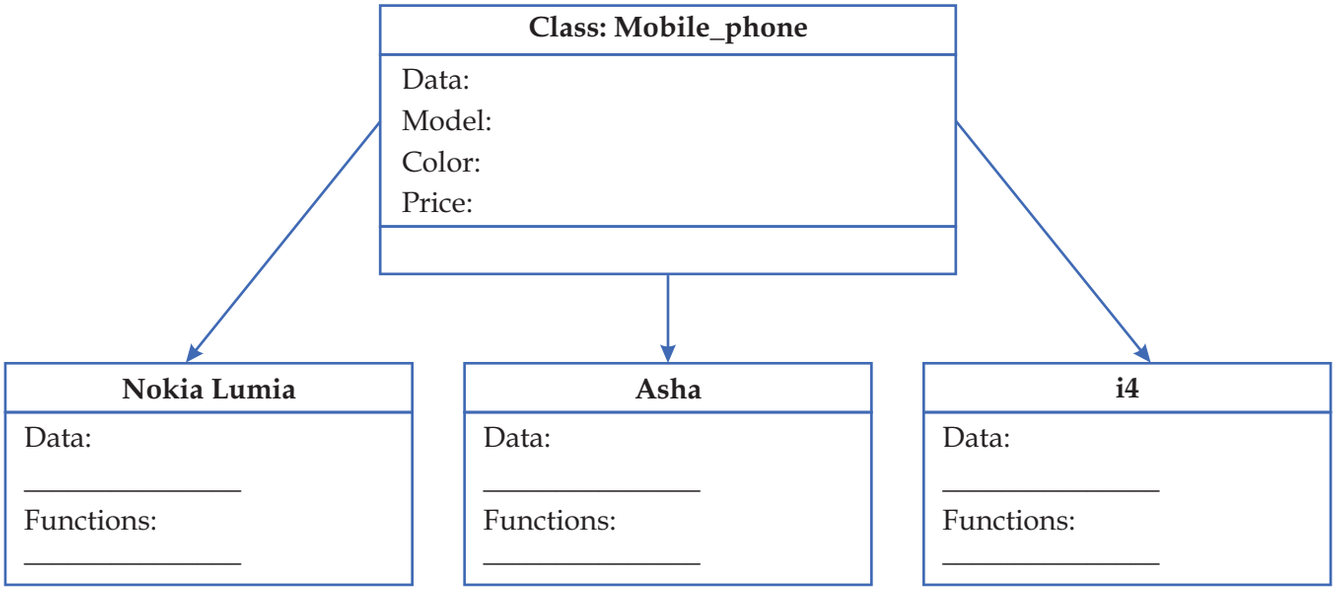


Fig 2: Class and Objects

In the above example, Nokia Lumia, Asha and i4 are all instances of the class Mobile_phone. All these instances are similar in the sense that all have basic features that a mobile phone should have. So all of these are objects of the class Mobile_phone

The general form of class definition in Python and creation of objects will be discussed in the next chapter.

Encapsulation

Encapsulation is the most basic concept of OOP. It is the combining of data and the functions associated with that data in a single unit. In most of the languages including python, this unit is called a class. In Fig -1 showing class Mobile_phone, given under the subtopic Classes, we see that the name of the class, its properties or attributes and behaviours are all enclosed under one independent unit. This is encapsulation, implemented through the unit named class.

In simple terms we can say that encapsulation is implemented through classes. In fact the data members of a class can be accessed through its member functions only. It keeps the data safe from any external interference and misuse. The only way to access the data is through the functions of the class. In the example of the class Mobile_phone, the class encapsulates the data (model, color, price) and the associated functions into a single independent unit.

Data Hiding

Data hiding can be defined as the mechanism of hiding the data of a class from the outside world or to be precise, from other classes. This is done to protect the data from any accidental or intentional access.

In most of the object oriented programming languages, encapsulation is implemented through classes. In a class, data may be made private or public. Private data or function of a class cannot be accessed from outside the class while public data or functions can be accessed from anywhere. So data hiding is achieved by making the members of the class private. Access to private members is restricted and is only available to the member functions of the same class. However the public part of the object is accessible outside the class. (You will study about private and public members in detail in the next chapter.)

Data Abstraction

Do you know the inner details of the monitor of your PC or your mobile phoneσ What happens when you switch ON the monitor or when any call is received by you on your phoneσ Does it really matter to you what is happening inside these devicesσ No, it does not. Rightσ Important thing for you is whether these devices are working as per your requirement or notσ You are never concerned about their inner circuitry. This is what we call abstraction.

The process of identifying and separating the essential features without including the internal details is abstraction. Only the essential information is provided to the outside world while the background details are hidden. Classes use the concept of abstraction. A class encapsulates the relevant data and functions that operate on data by hiding the complex implementation details from the user. The user needs to focus on what a class does rather than how it does.

Let us have a look at the Mobile_phone class. The case or body of the mobile phone is abstraction. This case is the public interface through which the user interacts. Inside the case there are numerous components such as memory, processor, RAM etc. which are private and so are hidden behind the public interface called case/body. Thus this case/body is the abstraction which has separated the essential components from implementation details. So when you purchase a mobile, you are given information about only the functions and operations of the mobile. The inside mechanism of the working of the mobile is of no concern to you. As long as the mobile is functioning properly, you are not bothered about the inner circuitry of the mobile phone.

Abstraction and Encapsulation are complementary concepts. Through encapsulation only we are able to enclose the components of the object into a single unit and separate the private and public members. It is through abstraction that only the essential behaviours of the objects are made visible to the outside world. So we can say that encapsulation is the way to implement data abstraction. In another example of class Student, only the essential information like roll no, name, date of birth, course etc. of the student are visible. The secret information like calculation of grades, allotment of examiners etc. is hidden.

Inheritance

Inheritance is one of the most useful characteristic of object-oriented programming as it enforces reusability of code. Inheritance is the process of forming a new class (derived class) from an existing class (called the base class). The data members and the methods associated with the data are accessible in the inherited class.

Let us understand this characteristic with the help of the class Mobile_phone.

An i-phone is a class in itself. It is a type of mobile phone. So we can have Mobile_phone as the base class and i_phone as its derived class as shown in the figure below:

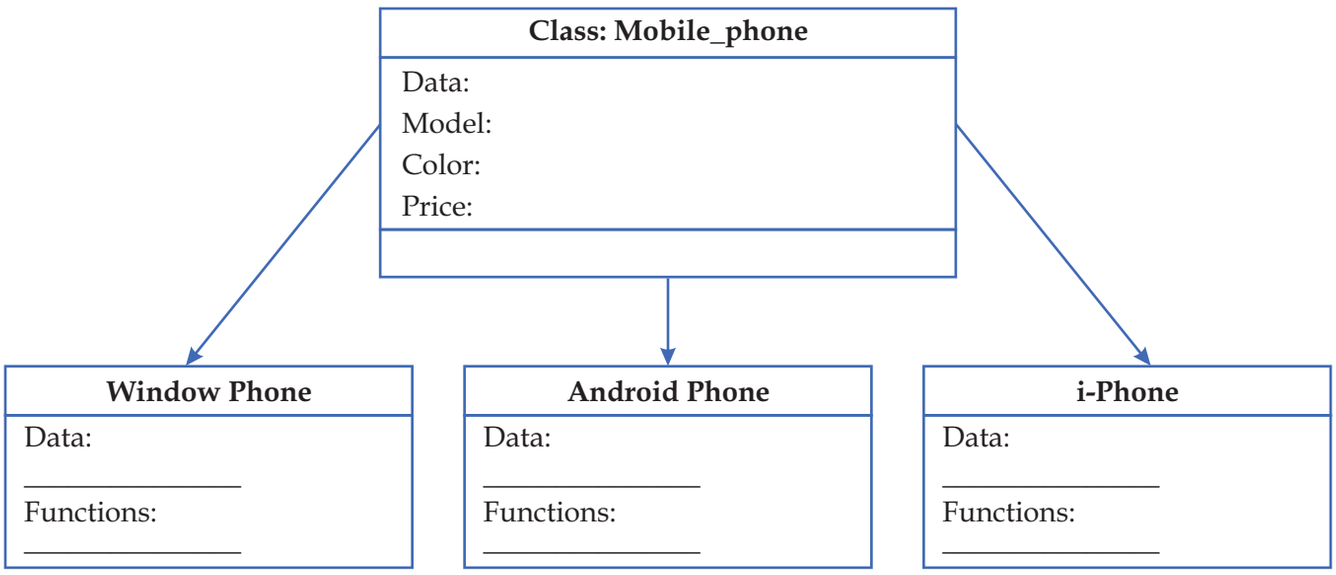


Fig 3: Inheritance

Such hierarchical classification helps to obtain a new class from an existing class. The derived class can also contain some new attributes of itself. So the derived class contains features of the base class as well as of itself. For example an i-phone will have all the features of a Mobile_phone class in addition to its own characteristics. Such a relationship between the two classes is known as "a kind of "relationship. For example an i-phone is a kind of mobile phone.

So we see that the base class can be reused again and again to define new classes. Another advantage of inheritance is its transitive nature. If a class i_phone inherits properties of another class Mobile_phone, then all derived classes of i_phone will inherit properties of the class Mobile_phone. All these factors make inheritance a very important characteristic of object oriented programming.

Polymorphism

The word Polymorphism is formed from two words - poly and morph where poly means many and morph means forms. So polymorphism is the ability to use an operator or function in various forms. That is a single function or an operator behaves differently depending upon the data provided to them. Polymorphism can be achieved in two ways:

1. **Operator Overloading**

   In class XI you have worked with '+ 'operator. You must have noticed that the '+' operator behaves differently with different data types. With integers it adds the two numbers and with strings it concatenates or joins two strings. For example:

   Print 8+9 will give 17 and

   Print "Python" + "programming" will give the output as Pythonprogramming.

   This feature where an operator can be used in different forms is known as Operator Overloading and is one of the methods to implement polymorphism.

2. **Function Overloading**

   Polymorphism in case of functions is a bit different. A named function can also vary depending on the parameters it is given. For example, we define multiple functions with same name but different argument list as shown below:

```
def test():                    #function 1
        print "hello"
def test(a, b):                #function 2
        return a+b
def test(a, b, c):             #function 3
    return a+b+c
```

   In the example above, three functions by the same name have been defined but with different number of arguments. Now if we give a function call with no argument, say test(), function 1 will be called. The

statement test(10,20) will lead to the execution of function 2 and if the statement test(10,20,30) is given Function 3 will be called. In either case, all the functions would be known in the program by the same name. This is another way to implement polymorphism and is known as Function Overloading.

As we see in the examples above, the function called will depend on the argument list - data types and number of arguments. These two i.e. data types and the number of arguments together form the function signature. Please note that the return types of the function are no where responsible for function overloading and that is why they are not part of function signature.

Here it must be taken into consideration that Python does not support function overloading as shown above although languages like Java and C/C++ do. If you run the code of three test functions, the second test() definition will overwrite the first one. Subsequently the third test() definition will overwrite the second one. That means if you give the function call test(20,20) , it will flash an error stating, "Type Error: add() takes exactly 3 arguments (2 given)". This is because, Python understands the latest definition of the function test() which takes three arguments.

Static and Dynamic Binding

Binding is the process of linking the function call to the function definition. The body of the function is executed when the function call is made. Binding can be of two types:

Static Binding: In this type of binding, the linking of function call to the function definition is done during compilation of the program.

Dynamic Binding: In this type of binding, linking of a function call to the function definition is done at run time. That means the code of the function that is to be linked with function call is unknown until it is executed. Dynamic binding of functions makes the programs more flexible. You will learn more on dynamic binding in the next chapter.

Advantages of OOP

Object Oriented programming has following advantages:

- ***Simplicity:*** The objects in case of OOP are close to the real world objects, so the complexity of the program is reduced making the program structure very clear and simple. For example by looking at the class Mobile_phone, you can simply identify with the properties and behaviour of an actual mobile phone. This makes the class Mobile_phone very simple and easy to understand.
- ***Modifiability:*** It is easy to make minor changes in the data representation or the procedures in an OO program. Changes inside a class do not affect any other part of a program, since the only public interface that the external world has to a class is through the use of methods.
- ***Extensibility and Maintainability:*** It is quite easy to add new features and extend the program in case of object oriented programming. It can be simply done by introducing a few new objects and modifying some existing ones. The original base class need not be modified at all. Even objects can be

maintained separately. There by making locating and fixing problems easier. For example if a new version of i-phone is introduced, a new derived class of the class i_phone for the new version may be created and no other class in the class hierarchy need to be modified. Similarly if any behaviour of a Windows phone changes, maintenance has to be done only for the class Windows phone.

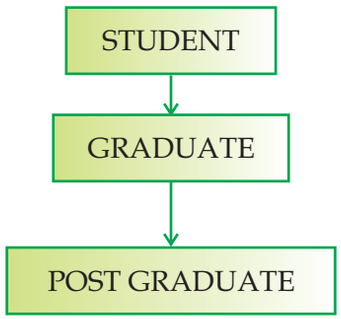
- ***Re-usability:*** Objects can be reused in different programs. The class definitions can be reused in various applications. Inheritance makes it possible to define subclasses of data objects that share some or all of the main class characteristics. It forces a more thorough data analysis, reduces development time, and ensures more accurate coding.
- ***Security:*** Since a class defines only the data it needs to be concerned with, when an instance of that class (an object) is run, the code will not be able to accidentally access other program data. This characteristic of data hiding provides greater system security and avoids unintended data corruption.

LET'S REVISE

- **Object:** clearly defines an entity in terms of its properties and behaviour.
- **Class:** a blueprint of an object.
- **Encapsulation:** combining of data and the functions associated with that data in a single unit
- **Data Hiding:** the mechanism of hiding the data of a class from the outside world
- **Abstraction:** providing only essential information to the outside world and hiding their background details
- **Inheritance:** forming a new class (derived class) from an existing class (called the base class).
- **Polymorphism:** ability to use an operator or function in various forms.
- **Static Binding:** the linking of function call to the function definition is done during compilation of the program.
- **Dynamic Binding:** the linking of function call to the function definition is done during the execution of the program.

EXCERCISE

- Fill in the blanks:
  a. Act of representing essential features without background detail is called _____.
  b. Wrapping up of data and associated functions in to a single unit is called_____.
  c. _____is called the instance of a class.
- What is Object Oriented Programmingσ List some of its advantages.
- Differentiate between an object and a class.
- What is inheritanceσ Explain with an example.
- List its three features that make it an important characteristic of OOP.
- Consider the figure given below and answer the questions that follow:



  a. Name the base class and the derived class.
  b. Which concept of OOP is implemented in the figure given aboveσ
- How do abstraction and encapsulation complement each otherσ
- Explain polymorphism with an example.
- Explain Data Hiding with respect to OOP.
- Explain Function overloading with an example.
- Is function overloading supported by Pythonσ Give reasons.
- Write the overloaded function definitions of add()- on adds two numbers and other concatenates two strings.
- Predict the output of the following program. Also state which concept of OOP is being implementedσ

```
def sum(x,y,z):
```

```
        print "sum= ", x+y+z
def sum(a,b):
        print "sum= ", a+b
sum(10,20)
sum(10,20,30)
```

- State whether the following are function overloading or not. Explain why or why not.

a. def (a,b):
   def(x,y) :

b. def(x,y,z)
   def(e,f)

- Define binding.
- Differentiate between static and dynamic binding.
- Write a program to find area of following using function overloading.
- Area of circle (function with one parameter)
- Area of rectangle (function with two parameters)
- Area of triangle (function with three parameters)
- Write a program to find out volume of the following using function overloading.
- volume of cube
- volume of cuboid
- volume of cylinder

Chapter-3: Classes in Python

Learning Objectives:

At the end of this chapter the students will be able to:

- Understand name spaces and scope rules
- Define classes (attributes , methods)
- Use __init__()
- Undersatnd the importance of "self"
- Create instance objects
- Distinguish between class attributes and instance attributes
- Add methods dynamically
- Access attributes and methods
- Use built-in class attributes (dict , doc , name , module , bases)
- Use __del__() and __str__() in a class
- Understand data hiding
- Understand static methods
- Destroy objects (garbage collection)

In the previous chapter you have studied that classes and objects are one of the most important characteristic of Object Oriented Programming. It is through classes that all the characteristics of OOP are implemented - may it be encapsulation, abstraction or inheritance.

This chapter deals with classes in Python. As Python is fully object-oriented, you can define your own classes, inherit from your own or built-in classes, and instantiate the classes that you've defined. But before we start with our study on classes let us understand about namespaces and scope rules in Python.

Namespaces

In Class XI, you had studied that variables refer to an object and they are created when they are first assigned a value. In fact the variables are bound to their values using the assignment operator(=). So a namespace is a place where a variable's name is stored and the value of the variable is bound to this namespace.

A namespace is a mapping from names to objects. It is a thing which associates the names with its values. In simple terms, it is a place where name lives. They are created at different moments and have different lifetimes. The examples of namespaces are:

- **Built-in names**

  These consist of functions such as max() , min() and built-in exception names. This namespace is created when the Python interpreter starts up, and is never deleted. The built-in names actually also live in a module called__ builtin__.

- **Global names in a module**

  The global namespace for a module is created when the module definition is read in and normally lasts until the interpreter quits. The statements executed by the top-level invocation of the interpreter, either read from a script file or interactively, are considered to be part of a module called__main__ and they have their own global namespace.

- **Local names in a function invocation**

  The local namespace for a function is created when the function is called, and deleted when the function returns or raises an exception that is not handled within the function. Even each recursive invocation has its own local namespace.

  If we talk about classes and objects, the set of attributes of an object also form a namespace. It must be noted that there is absolutely no relation between names in different namespaces. Two different modules may both define same function without any confusion because the functions are prefixed with the module name. That means module1.cmp() has no relation with module2.cmp().

Scope Rules

A scope is a region of a Python program where a namespace is directly accessible.The location where the names are assigned in the code determines the scope of visibility of the name in that code. Although scopes are determined statically i.e. during creation of the program, yet they are used dynamically i.e. during execution of the program. At any time during execution, there are at least four main things to remember in the context of scope rules:

i) In Python, names of all types of variables are treated in same manner. That means numbers, strings, functions, types, modules, classes - all are treated in the same way. Also a name can refer to only one thing at a time. For example, consider the following program:

```
var = 10 + 5
print var
def var(y):
    return y*10
    print var
var = "Hello"
print var
```

In the code given above, the variable var is bound to 15(10 + 5). Then def var(y) binds var to a function. The previous binding of var to 15 is lost and is replaced by the function. Thereafter var is bound to a string, so its binding with the function is no longer existing.

ii) The scope of a variable is its enclosing function or class or file. As discussed before, each name belongs to a namespace. For example, if a variable is created in a particular function, then its scope is that function only, since that function creates its own namespace where it resides. So any variable inside the function will be local to that namespace. In the following example, the scope of the variable x is the test function.

```
def test():
    x = 5
    print x
```

Now let us modify the program -

```
x = 10
def exam():
    print x
def test():
    x = 5
    print x
def marks(x):
    print x
print x
exam()
test()
marks(20)
```

On executing the above code, the output will be

```
10
10
5
20
```

The first line creates a variable x that belongs to the namespace of the file, so its scope is the entire file. Hence 10 is displayed. The exam function creates its namespace, but that namespace doesn't have an x in it. As Python doesn't find x there, it checks the next larger enclosing namespace and finds x. So exam uses the variable x defined at the top and displays 10.

However, the test function defines its own variable named x with value 5, which has higher priority over the first definition of x. So any mention of x within the test function will refer to that x, hence displaying 5. The marks function also has an x in its own namespace, just like test function has. So x gets bound to whatever value is passed as an argument to marks function (20 in the given example). Hence the outer x is shadowed again in this function displaying the output as 20.

iii) The names always belong to the namespace where they are bound, irrespective of whether they are bound before or after they are referred. This is the reason which makes Python a lexically scoped language. The variable scopes are determined entirely by the locations of the variables in the source code of your program files, not by function calls. If a binding for a variable appears anywhere inside a function, the variable name is local to that function. Let us understand this with the help of an example:

```
x = 10
def func1():
    x=50
    print x
def func2():
    print x
    x=25
def func3(p):
    if p<10:
        x=2
    print x
func1()
func2()
func3(20)
func3(5)
```

In the above example, the func1 function creates a local variable x in its own namespace, shadowing the outer variable x. So the line print x prints 50. The func2 function also has a local variable x in its namespace but the assignment to x is after the print statement. The local variable x shadows the outer x, even though the local x is initially not bound to anything. The line print x looks for x in the local namespace, finds that it is not bound to anything, and so the reference to x leads to an error (an Unbound Local Error occurs). Similarly, in func3(), the variable x is local.

When we call func3(20), the line x = 2 is not executed, so print x causes an error. But when we call func3(5), the line x = 2 is executed , so print x prints 2.

iv) Names declared with global keyword have to be referred at the file level. This is because the global statement indicates that the particular variable lives in the global scope. If no global statement is being

used, the assignment to the name is always in the innermost local scope. Consider the following example:

```
x=5
def func1():
    x=2
    x=x+1
def func2():
    global x
    x=x+1
print x
func1()
print x
func2()
print x
```

The above example prints 5; then calling func1()it prints 3. This is because func1 only increments a local x. Then func2()increments the global x and prints 6.

LEGB Rule

From the examples discussed above, we come up to the LEGB rule. According to this rule, when a name is encountered during the execution of the program, it searches for that name in the following order:

L. Local - It first makes a local search i.e. in current def statement. The import statements and function definitions bind the module or function name in the local scope. In fact, all operations that introduce new names use the local scope.

E. Enclosing functions - It searches in all enclosing functions, form inner to outer.

G. Global (module) - It searches for global modules or for names declared global in a def within the file.

B. Built-in (Python) - Finally it checks for any built in functions in Python.

The examples given above give the output according to the LEGB rule only.

Defining Classes

We have already studied in the previous chapter that a class is a unit that encapsulates data and its associated functions. In this section we will learn how to define classes.

To define a class in Python, we need to just define the class and start coding. A Python class starts with the reserved word 'class', followed by the class name and a colon(:). The simplest form of class definition looks like this:

```
class Class Name:
    <statement-1>
    ....
    ...
    <statement-2>
```

Everything in a class is indented after the colon, just like the code within a function, if statement or for loop. The first thing not indented is not part of the class definition. A class may contain attributes which can be data members or/and member functions i.e. methods. In fact the word attribute is used for any name following a dot. Let us take the example of class Test:

```
class Test:
    var = 50
    marks = 10
    def display():
        print var
```

In the above class definition, marks, var, display(), all are attributes of the class Test and also of all its objects. Similarly when we import modules, in the expression module1.fuctionname, module1 is a module object and function name is a method but also referred to as an attribute of module 1. The module's attributes and the global names are defined in the same namespace. You will learn about object creation and usage later in this chapter.

Class definitions, like function definitions using def statements must be given before they are referenced for use. When a class definition is entered a new namespace is created and then used as local scope. Hence all assignments to the local variables are attached with this namespace. The function definitions are also bound to the same namespace.

Attributes of a class may be read-only or writable. In the latter case, assignment to attributes is possible. That means the following statement is valid:

```
test1.marks=10
```

Writable attributes may also be deleted using the del statement. For example:

```
del test1.marks
```

The above statement will remove the attribute marks from the object named test1.In fact the del statement removes the binding of the attribute (marks) from the namespace (test1) referenced by the class's local scope.

When a class definition is left normally (via the end), a class object is created. This is basically a wrapper around the contents of the namespace created by the class definition. We will learn more about class objects in the next section.

Note that calling a method on an object can also change the object. This implies that an object is mutable. A function can modify an outer mutable object by calling a method on it. Consider the example below:

```
x= [10]
def List_ex():
    x.append(20)
def add_list():
    x=[30,40]
    x.append(50)
print x
list_ex()
print x
add_list()
print x
```

The above example prints

[10]

[10,20]

[30,40,50]

The list_ex()calls the append method of the global x, whereas the add_list(), x refers to a local x.

Also data attributes override method attributes with the same name. That means if the data attribute of the class and the method attribute are in same scope, then the data attribute will be given higher priority. So to avoid accidental name conflicts, we may use conventions like:

- capitalizing method names
- prefixing data attribute names with a small unique string(generally an underscore)
- using verbs for methods and nouns for data attributes.

If the class does not contain any statements i.e. it is a class without any attributes or methods , then a keyword 'pass' is given within the body of the class as shown below :

```
class mobile:
    pass
```

In the above example 'pass' is a keyword. Giving pass in the class definition means that the class doesn't define any methods or attributes. But since there needs to be something in the definition, so you use pass. It's a statement that does nothing.

Constructors in Python (Using __init__)

A constructor is a special method that is used to initialize the data members of a class. In python, the built in method __init__ is a sort of constructor. Notice the double underscores both in the beginning and end of init. In fact it is the first method defined for the class and is the first piece of code executed in a newly created instance of the class. But still it should also be remembered that the object has already been constructed by the time __init__ is called, and you already have a valid reference to the new instance of the class through the first argument, self of the __init__ method. Consider the following example:

```
class Initialize:
'''An example of __init__'''
    int var
    def __init__(self, var=10):  #double underscore before and after init
        Initialize.var=var
    def display():
    print var
```

__init__ method can take any number of arguments, and just like functions, the arguments can be defined with default values, making them optional to the caller. Initial values for attributes can be passed as arguments and associated to attributes. A good practice is to assign them default values, even None. In this case, var has a default value of 10. After the class definition, object.__init__(self[, ...]) is called when the instance is created. The arguments are those passed to the class constructor expression. This means the statements given below will give the output 20.

```
P = Initialize(20)
P.display()
```

Also note that if no argument was passed while creating the object, then the __init__ would have taken the default value of var and the output would have been 10.

In Python, the first argument of every class method, including __init__, is always a reference to the current instance of the class and by convention, this argument is always named 'self'. In case of __init__, self refers to the newly created object or the instance whose method was called. Note that the __init__ method never returns a value.

Importance of self

Class methods have only one specific difference from ordinary functions - they must have an extra argument in the beginning of the parameter list. This particular argument is self which is used for referring to the instance. But you need not give any value for this parameter when you call the method. Python provides it automatically. self is not a reserved word in Python but just a strong naming convention and it is always convenient to use conventional names as it makes the program more readable. So while defining your class methods, you must explicitly list self as the first argument for each method, including __init__.

To understand why you don't need to give any value for self during the method call, consider an example. Say you have a class called My_Photo and an instance of this class called My_Object. When you call a method of this object as My_Object.method(arg1, arg2), this is automatically converted by Python into My_Photo.method (My_Object, arg1, arg2). This feature makes self special and it also implies that if you have a method which takes no arguments, then you still have to define the method to have a self argument.

Self is an instance identificator and is required so that the statements within the methods can have automatic access to the current instance attributes. Here is the example showing a class definition using__init__ and self.

```
class Mobile:
    '"A sample class definition"'
    price = 0
    model = "Null"
    def __init__(self, price, model = None):
        self.price=price
        self.model="Nokia Lumia 720"
    def displaydata(self):
        print self. price, self. model
```

In the above example:

- The variables price and model are the class variables whose value would be shared among all instances of this class. This can be accessed as Mobile.price, Mobile. model from inside the class or outside the class.
- The first method __init__() is a special method, which is called class constructor or initialization method that Python calls when you create a new instance of this class.
- You declare other class methods like normal functions with the exception that the first argument to each method is self. While giving a call to the method, the instance name is automatically taken as the first argument for self.

If after the given class definition of class Mobile, the following statements are executed

```
M= Mobile(1000, 'Samsung')
M.displaydata()
```

the output is

```
1000 Samsung
```

Class instances (Objects)

Having a class defined, you can create as many objects as required. These objects are called instances of this class. In fact after the class definition is made, a class instance is created automatically once the definition is

left normally i.e. the indentation of statements is removed and the class object is called. All the instances created with a given class will have the same structure and behaviour. They will only differ regarding their state, i.e regarding the value of their attributes.

Classes and instances have their own namespaces, that is accessible with the dot ('.') operator. These namespaces are implemented by dictionaries, one for each instance, and one for the class.A class object is bound to the class name given in the class definition header. A class object can be used in two ways - by Instantiation and attribute references.

i) Instantiation: Creating instance objects

To create instances of a class, you call the class using class name and pass in whatever arguments its __init__ method accepts.

Test = T(1,100)

In the above example T is the instance of class Test.

ii) Attribute Reference: Accessing attributes of a class

This is the standard syntax used for all attribute references which is

Object Name. Attribute Name.

As discussed before all the names that were given during the class definition and hence were in the class's namespace are valid attribute names. You access the object's attributes using the dot operator with object as shown in the example below:

```
test.display()
unit_test.display()
print "Marks =", test. marks
```

The search for the referenced attribute is done in the following sequence:

a) A class instance has a namespace implemented as a dictionary which is the first place in which attribute references are searched.

b) When an attribute is not found there, and the instance's class has an attribute by that name, the search continues with the class attributes.

c) If no class attribute is found, the object's __getattr__() method is called to satisfy the lookup. You will study about this method later in the chapter.

Attribute assignments and deletions update the instance's dictionary, never a class's dictionary. If the class has a __setattr__() or __delattr__() method, this is called instead of updating the instance dictionary directly. You will learn about these methods later in this chapter.

Class Atttributes v/s Instance Attributes

Attributes can be classified into - Class Attributes and Instance attributes

Class Attributes

These belong to the class itself. These attributes will be shared by all the instances. Such attributes are defined in the class body part, usually at the top, for legibility. Consider the following example:

```
class Health_profile:
        ...
    weight = 89
    blood_group= 'B+'
    ...
```

To access this attribute, you use the dot notation:

```
>>>Health_profile.weight
89
>>>Health_profile.blood_group
B+
```

Instances attributes

As we have learnt, a class may define attributes for its instances. These are called instance attributes and they belong to each instance/object of a class. For example, for the class Health_profile given above, let H1 be an instance. So, the attributes of H1, such as the weight, are directly available through the dot operator:

```
>>>H1.weight
89
```

The dictionary for the instance attributes is also accessible by its __dict__ variable about which you will learn in the next section. To list the attributes of an instance, we have two functions:

i) vars() : This function displays the attributes of the instance in the form of a dictionary. Consider the following example:

```
>>>vars(H1)
{'weight': '89', 'blood group': 'B+'}
```

ii) dir(): This function lists more attributes than vars()because it is not limited to the dictionary of instance. It also displays the class attributes. For example

```
>>>dir(H1)
['__doc__', '__init__', '__module__', 'weight', 'blood_group',]
```

You can add, remove or modify attributes to an instance that were not defined by the class, such as the height in the following:

```
>>> H1.height = 197          # adds 'height' as attribute
>>>vars(H1)
```

```
{'weight': '89', 'blood group': 'B+',height='197'}
>>>H1. height=180          #modifies the value of height
>>>vars(H1)
{'weight': '89', 'blood group': 'B+',height='180'}
>>>del H1.height        #deleted the attribute height
>>>vars(H1)
{'weight': '89', 'blood group'}
```

Here it should always be remembered that this feature of adding and deleting attributes should be used carefully, since by doing this, you start to have instances that have different behaviour than that is specified in the class.

Adding methods dynamically

As you can add, modify and delete the attributes of a class dynamically i.e. at run time, similarly, you can add methods dynamically to an object or class. Consider the code given below:

```
class Health_profile:
                ...
                weight = 89
                blood_group= 'B+'
def play():
    print " Come on lets play"
H=Health_profile()
H.play=play()
H.play()
```

In the above example, play is just a function which does not receive self. There is no way by which H can know that play is a method. If you need self, you have to create a method and then bind it to the object. For this you have to import MethodType from types module as shown in the example below:

```
from types import MethodType
class Health_profile(object):
        weight = 89
        blood_group= 'B+'
        def __init__(self,name):
            self.name=name

        def play():
            print " Come on lets play", self. name
```

```
H=Health_profile("Shalini")
H.play=MethodType(play,H)
H.play()
```

In the above code, the built in function Method Type from the types module takes two arguments - the name of the function which has to be bound dynamically and the instance with which it has to bind the function. In the above example the play method will be bound only with the instance, H. No other instances of the class Health_profile will have play method. If we want the other instances also to have play method, then we have to add the method to the class and for that we make use of self as shown in the example below:

```
class Health_profile(object):
        weight = 89
        blood_group= 'B+'
        def __init__(self,name):
            self.name=name

def play(self):
        print " Come on lets play", self.name
Health_profile.play=play()
H1=Health_profile("Shalini")
H1.play()
H2=Health_profile("Ritu")
H2.play()
```

In the above example, note that no method is created with types.MethodType. This is because all functions in the body of the class will become methods and receive self unless you make it a static method about which you will study later in the chapter.

Accessing Attributes and methods

Attributes of a class can also be accessed using the following built in methods / functions:

- **getattr(obj, name[, default]):** This function is used to access the attribute of object.It is called when an attribute lookup has not found the referenced attribute in the class. The built in method for the same is object. __getattr__(self , name)which is called automatically if the referenced attribute is not found. For example:

```
        getattr(H1,weight)
Built in method for the same will be
        H1.__getattr__(self,weight)
```

The above statement returns the value of the weight attribute otherwise raises an Attribute Error exception. Note that if the attribute is found through the normal mechanism, __getattr__() is not called.

- **hasattr (obj,name):** It is used to check if an attribute exists or not. For example hasattr(H1,weight)

  #will return a True if 'weight' attribute exists

- **setattr (obj, name, value):** It is used to set an attribute. Alternatively object.__setattr__(self, name, value) built in method is called when an attribute assignment is attempted , where name is the name of the attribute and value is its value that is to be assigned. If an attribute does not exist, then it would be created. For example:

  setattr(H1,weight,90)

  The built in method for the same will be

  H1.__setattr__(self,weight,90)

  Either of the above statements set the value of the attribute weight as 90.

- **delattr(obj, name):** It is used to delete an attribute.The built in method for the same is object.__ delattr__(self , name). For example :

  delattr(H1,weight) # deletes the attribute weight

  The built in method for the same will be

  H1.__delattr__(self,weight)

Accessing Methods

When an instance attribute other than the data attribute is referenced, the corresponding class is searched. If it is a valid class attribute (a function object), a method is created by pointing to the instance object and the function object. When this method object is called with an argument list, a new argument list is constructed from the instance object and the argument list. The function object is then called with this new argument list. The methods of a class can be accessed in the same manner as the data attributes i.e.

ObjectName.Methodname

Now, putting all the concepts together, let us see the following example:

```
class Health_profile:
    weight=0
    blood_group='B+'
    def __init__(self,weight,blood_group):
        self.weight=weight
        self.blood_group=blood_group

    def display(self):
        print " Weight :", self.weight
```

```
        print "Blood Group : " , self.blood_group
```

The following statement will create an object (instance) of class Health_profile

```
H2=Health_profile(61 ,'A+')        #Assuming weight is 61 and blood group is A+
```

On executing the statement H1.display(), the output will be :

```
Weight :61
Blood Group :A+
```

A function object need not always be textually enclosed in the class. We can also assign a function object to a local variable in a class. For example

```
def test ( a ,b):
    return x+y

class myclass:
        F=test(10,20)
        def G(self):
            return F     " Using a function that is defined outside the class"
        H=G
```

In the above example F, G and H are all attributes of class myclass. They refer to the function objects and hence are all methods of instances of myclass.

Methods may also be referenced by global names in the same way as ordinary functions. The global scope associated with a method is the module containing its definition. Although global data in a method is rarely used, functions and modules imported into a global scope can be used by methods, functions and classes defined in it. Usually a class containing the method is itself defined in this global scope.For example

```
class myclass:
    Yf=x.f()
    while true:
        printYf()
```

In the above example, you will notice that it is not necessary to call a method right away. x.f() is a method object and can be stored(in Yf) and called later (in the while loop).

In case of methods, the object is passed as the first argument of the function. So even if no argument is given in the function call while it was defined in the function definition, no error is flashed. In the above example x.f() is exactly equivalent to myclass.f(x). So we can say that calling a method with a list of n arguments is equivalent to calling the corresponding function with an argument list that is created by inserting the method's object before the first argument.

Built in class attributes

Every Python class keeps the following built-in attributes and they can be accessed using dot operator like any other attribute:

i) **__dict__ :** It gives the dictionary containing the class's namespace.

ii) **__doc__ :** It returns the class's documentation string(also called docstring) and if no docstring is defined for a class this built in attribute returns None

iii) **__name__:** It gives the class name.

iv) **__module__:** It specifies the module name in which the class is defined. This attribute is called "__main__" in interactive mode.

v) **__bases__ :** It gives a possibly empty tuple containing the base classes, in the order of their occurrence in the base class list. (You will learn about base classes in the next chapter on Inheritance)

For the previously defined class Test let's try to access all the above built in attributes:

```
class Test:
'"A sample class to demonstrate built in attributes"'
    rollno=1
    marks=75
    def __init__(self,rollno,marks):
        self.rollno=rollno
        self.marks=marks
    def display(self):
        print " Roll No : ", self.rollno
        print "Marks : ", self.marks
print "Test.__doc__:", Test.__doc__
print "Test.__name__:", Test.__name__
print "Test.__module__:", Test.__module__
print "Test.__bases__:", Test.__bases__
print "Test.__dict__:", Test.__dict__
```

When the above code is executed, it produces the following result:

```
Test.__doc__: A Sample class to demonstrate built in attributes
    Test.__name__: Test
    Test.__module__: __main__
    Test.__bases__: ()
```

Test.__dict__: {'__module__' : '__main__' , 'display' :
<function display at 0xb8a9872> , 'rollno' :1 , 'marks':75,
'__doc__ : 'A Sample class to demonstrate built in attributes',
'__init__ : <function __init__at 0xb8a89432c>}

Using __ del()__

This function is called when the instance is about to be destroyed. This is also called a destructor. It calls the method - object.__del__(self)

When __del()__ is invoked in response to the module being deleted (for example , when the execution of the program is done), the other global variables referenced by __del()__ method may already have been deleted or must be in the process. Let us understand the concept through the class Test whose instance is T1.Consider the following command is given

>>>del T1

The above command doesn't directly call T1.__del__(). First the reference count is decremented for T1 by one and __del()__ is called only when T1's reference count reaches zero i.e. when all the variables referenced by T1 have been deleted.

Using __ str()__

It is a special function which returns the string representation of the objects. It calls the method object.__str__(self). If the class defines a __str__ method, Python will call it when you call the str() or use print statement. The str() built-in function, when used along with the print statement computes the "informal" string representation of an object. Consider the following example of the class Test defined above.

class Test:

…….

……..

def __str__(self):

return "Hello, How are youσ"

Now give the following command on the Python interpreter:

>>> T=Test()

>>> print T

Hello, How are youσ

When you give the command to "print T", Python calls str(T) to get the string representation of T. If the class of T has a __str__ method, str(T) becomes a call to T.__str__(). This returns the string to print.

Private Members - Limited Support

"Private" instance variables are those that cannot be accessed from outside the class. These attributes, may it be data or methods, can only be accessed from inside an object. These are used if we want to hide an attribute but these types of members don't exist in Python. However, if you still want to hide a member, Python provides a limited support called name mangling. According to this,a name is prefixed with two leading underscores and no more than one trailing underscore will be considered as a private member. For example __pvt , __pvt_will be considered as mangled and so will be treated as private members of the class but not __pvt__ or _pvt__ .

On encountering name mangled attributes, Python transforms these names by a single underscore and the name of the enclosing class, for example:

```
class Test:
    Weight=89
    Blood_group='B+'
    __BP=None                    #private member
```

On execution of dir() , notice that the attribute BP if prefixed with an underscore and the class name.

```
>>>dir(Test)
['__doc__', '__init__', '__module__', 'weight', 'blood_group',_Test__BP]
```

Data Hiding

Data hiding is a means to procure encapsulation. Python doesn't really enforce data-hiding in the true sense. The Python approach to getting encapsulation is by simply not using "private" data members. As discussed before, if something is particularly "private" you can use the double leading underscores to mark it as such, but this of course is nothing more than a marking. Basically it just acts as a reminder that such data is intended to be used only within the class. Remember that Python is a dynamic language and that you can add attributes or methods dynamically on an object. In the example given below the attribute pulse is dynamically added to the class Health_profile through H1.

```
class Health_profile:
    pass
>>H1 =Health_profile()
H1.pulse = 75
```

Python's sense of encapsulation means that you don't have to worry about implementation details. Also it does not matter whether or not an attribute has public or private access. As explained in the previous topic, a leading double underscore means that an attribute is private in the sense that if you try to access the attribute directly (via it's name), you won't be able to and this is done by secretly name-mangling the variable with a leading underscore, the name of the class, and then the name of the variable. If you try to access this hidden attribute, then an error message will be displayed.

Consider the following example of class Health_profile:

```
class Health_profile:
        Weight=89
        Blood_group='B+'
        __BP=None                       #private member
        def __init__(self):
                self.__BP = "Hidden attribute"
        def display_BP(self):
                print self.__BP
```

```
>>>H =Health_profile()
>>>H.__BP
Traceback (most recent call last):
 File "<input>", line 1, in <module&gt;
AttributeError: Example instance has no attribute '__BP'
H.display__BP()
"Hidden attribute"
```

In the above example, __BP has been defined as private and so an error message is displayed when an attempt is made to access it from outside the class. Even when you try to import using the command- 'from mymodule import *', anything with a double leading underscore doesn't get imported. But still there is an alternative way to access the private members as shown in the following example :

```
H2=Health_profile()
H2.Heath_profile__BP="180/90"
```

I know you must be wondering are private members actually hiddenσThe answer is no.

Static methods

A static method of a Python class is a method that does not obey the usual convention in which self, an instance of the class, is the first argument to the method. It is just like any other function defined in the class but it is callable without creating the instance of the class. They belong to the class but they don't use the object self at all. To declare a static method, declare the method normally but precede it with the built-in staticmethod decorator called@staticmethod. Good candidates for static methods are methods that do not reference the self variable. Consider the following example

```
class Test:
    @staticmethod
```

```
    def Displaytest():
        print "This is a static method"
>>>Test.Displaytest()
This is a static method
```

Note in the above example that the function Displaytest() is not having self as the first argument and that it is being called without the instance. It eases the readability of the code as seeing @staticmethod, we know that the method does not depend on the state of the object. Once you have declared a method to be static, the arguments you pass to it are exactly the arguments it receives. Static methods return the underlying functions with no changes at all. As a result, the function becomes identically accessible from either an object or a class. For example

```
class Test:
    @staticmethod
    def square(x):
        Test.result= x*x

>>>Test.square(5)
>>>Test.result
25
>>>Test().square(6)
>>>Test.result
36
>>>T1=Test()
>>>T1.square(2)
>>>T1.result
>>>4
```

Destroying Objects (Garbage Collection)

Python automatically allocates and de-allocates memory. The user does not have to preallocate or deallocate memory by hand as one has to when using dynamic memory allocation in languages such as C or C++. Python uses two strategies for memory allocation-reference counting and automatic garbage collection.

i) Reference Counting

Prior to Python version 2.0, the Python interpreter only used reference counting for memory management. Reference counting works by counting the number of times an object is referenced by other objects in the system. Python's garbage collector runs during program execution and is triggered when an object's reference count reaches zero. An object's reference count changes as the number of

aliases that point to it change. An object's reference count increases when it is assigned a new name or placed in a container (list, tuple or dictionary). The object's reference count decreases when it is deleted with del, its reference is reassigned, or its reference goes out of scope. When an object's reference count reaches zero, Python collects it automatically. Consider the code given below:

```
X=50          # an object X is created which is bound to 50.
Y=X           # increase in reference count of 50
Z[0]={Y}      # increase in reference count of 50
del X         # decrease in reference count of 50
Y=10          # decrease in reference count of 50
```

You can see that a class can implement the special method __del__(), called a destructor, that is invoked when the instance is about to be destroyed. This method might be used to clean up any non-memory resources used by an instance.

Reference counting is extremely efficient but it does have some caveats. One such caveat is that it cannot handle reference cycles. A reference cycle is when there is no way to reach an object but its reference count is still greater than zero. The easiest way to create a reference cycle is to create an object which refers to itself as in the example below:

```
def reference_cycle():
    x=[ ]
    x.append(x)
reference_cycle()
```

In the above example since reference_cycle( ) creates an object x which refers to itself (statement x.append(x)), x will not automatically be freed when the function returns. This will cause the memory that x is using to be held onto until the Python garbage collector is invoked.

ii) Automatic Garbage Collection

In this case garbage collection is a scheduled activity. Python schedules garbage collection based upon a threshold of object allocations and object de-allocations. Python deletes the objects which are not required, may it be built-in types or class instances, through the process named garbage collection. When the number of allocations minus the number of de-allocations are greater than the threshold number, the garbage collector is run and the unused block of memory is reclaimed. One can inspect the threshold for new objects by loading the gc module and asking for garbage collection thresholds.

Automatic garbage collection will not run if your Python device is running out of memory. In such case, your application will throw exceptions, which must be handled otherwise your application will crash. Also, the automatic garbage collection occurs more for the number of free objects than the size of objects.

LET'S REVISE

- **Namespace:** A mapping from names to objects. Examples of namespaces are built-in names, global names in a module and local names in function invocation
- **Scope:** A region of Python program where a namespace is directly accessible.
- In Python a name, storing any type of data, can refer to only one thing at a time.
- The scope of a variable is its enclosing function, class or file.
- The names always belong to the namespace where they are bound.
- Names declared with global keyword have to be referred at the file level.
- **LEGB rule:** when a name is encountered during the execution of the program , it searches for that name in the following order:

  L. Local - It first makes a local search i.e. in current def statement.

  E. Enclosing functions - It searches in all enclosing functions, form inner to outer.

  G. Global (module) - It searches for global modules or for names declared global

  B. Built-in (Python) - Finally it checks for any built in functions in Python.
- Class definitions should be given before it is referenced.
- __init__ is a special method used to initialize the members of a class.
- *self* is the first argument that is passed to the methods of a class.
- A class object can be used in two ways - Instantiation and Attribute reference
- Class attributes belong to the class and will be shared by all instances
- Instance attributes belong to a particular instance of a class only.
- The attributes - data and methods can be added to the class dynamically.
- getattr(obj, name,[default]): is used to access the attribute of the object
- hasattr(obj,name): is used to check if an attribute exists or not
- setattr(obj,name,value): is used to set an attribute with a value.
- delattr(obj,name) : is used to delete an attribute
- __dict__ : gives the dictionary containing class namespace
- __doc__: returns the docstring of a class
- __name__: it gives the class name
- __module__: specifies the module name in which the class is defined

- __bases__: it gives a tuple containing base classes
- __del__: is invoked when the module is being deleted
- __str__: returns the string representation of the objects
- Private variables can only be accessed from inside the objects.
- **Name Mangling:** A name is prefixed with two leading underscores and no more than one trailing underscore.
- **Static Method:** is a method that does not obey the usual convention in which self, an instance of the class, is the first argument to the method.
- Python uses two strategies for memory allocation- Reference counting and Automatic garbage collection.
- **Reference Counting:** works by counting the number of times an object is referenced by other objects in the system. When an object's reference count reaches zero, Python collects it automatically.
- **Automatic Garbage Collection:** Python schedules garbage collection based upon a threshold of object allocations and object de-allocations. When the number of allocations minus the number of de-allocations are greater than the threshold number, the garbage collector is run and the unused block of memory is reclaimed.

EXERCISE

1. Give one word for the following:
   a. A sort of constructor in Python _____
   b. A region of a Python program where a namespace is directly accessible. _____
   c. It returns the docstring of a class. _____
   d. It returns the string representation of the object. _____
   e. A method used to delete an attribute. _____
2. Define a namespace. Give examples of namespaces with respect to Python.
3. Is data of different types treated differently in PythonσSupport your answer with an example.
4. Explain LEGB rule.
5. Is object of a class mutableσWhy/why notσ
6. Explain the usage of keyword 'pass' in class definition.
7. What is the use of __init__σWhen is it calledσExplain with an example.
8. Explain the importance of self in Python classes.
9. Differentiate between class attributes and instance attributes.
10. Explain __str__ with an example.
11. What do you mean by name manglingσSupport your answer with relevant example.
12. Differentiate between reference counting and automatic garbage collection with respect to Python.
13. If we want to make the method that is added dynamically to a class available to all the instances of the class, how can it be doneσExplain with the help of an example.
14. Predict the output of the following code snippet

(i)
```
ptr=40
def result():
    print ptr
    ptr=90
def func(var):
    if var<=60:
    ptr=30
```

```
        print ptr
    result()
    func(60)
    func(70)
(ii) ptr=50
    def result():
        global ptr
        ptr=ptr+1
    print ptr
    result()
    print ptr
```

15. Name the methods that can be used to
    a. access attribute of an object
    b. delete an attribute of an object
16. Give the statement to
    a. Check whether the attribute str exists in the class Test whose object is T1
    b. Assign a value "Hello" to the attribute str of class Test and object T1.
17. Consider the following class definition:

```
class Yourclass
        marks=10
        name= "ABC"
        def __init__(self,marks,name):
                self.marks=marks
                self.name=name
        def display(self):
                print marks
                print name
```

Give the statement to create an object of class Yourclass.

18. In the code of Q-13 what will be output if the following command is given: (Assuming YC is the object)
YC.display()

19. Predict the output of the following code :

```
class Match:
    ' "Runs and Wickets" '
    runs=281
    wickets=5
    def __init__(self,runs,wickets):
            self.runs=runs
            self.wickets=wickets
        print " Runs scored are : " ,runs
        print "Wickets taken are : " ,wickets
print "Test.__do__:" , Match.__doc__
print "Test.__name__:" , Match.__name__
print "Test.__module__:" , Match.__module__
print "Test.__bases__:" , Match.__bases__
print "Test.__dict__:" , Match.__dict__
```

20. Create the class SOCIETY with following information:

society_name

house_no

no_of_members

flat

income

Methods

- An __init__ method to assign initial values of society_name as "Surya Apartments", flat as "A Type", house_no as 20, no_of_members as 3, income as 25000.
- Inputdata( ) - to read data members(society,house_no,no_of_members&income) and call

allocate_flat().

- allocate_flat( ) - To allocate flat according to income

Income	Flat
>=25000	A Type
>=20000 and <25000	B Type
<15000	C Type

- Showdata( ) - to display the details of the entire class.

21. Define a class ITEMINFO in Python with the following description:

ICode (Item Code)

Item (Item Name)

Price (Price of each item)

Qty (quantity in stock)

Discount (Discount percentage on the item)

Netprice (Final Price)

Methods

- A member function FindDisc( ) to calculate discount as per the following rule:

If Qty<=10 Discount is 0

If Qty (11 to 20) Discount is 15

If Qty>=20 Discount is 20

- A constructor(__init__ method) to assign the value with 0 for ICode, Price, Qty, Netprice and Discount and null for Item respectively
- A function Buy( ) to allow user to enter values for ICode, Item, Price, Qty and call function FindDisc( ) to calculate the discount and Netprice(Price*Qty-Discount).
- A Function ShowAll( ) to allow user to view the content of all the data members.

Chapter-4: Inheritance

Learning Objectives:

At the end of this chapter the students will be able to:

- Understand the concept of Inheritance
- Understand the types of Inheritance
- Understand Single Inheritance
- Use super() in derived class to invoke _init_()
- Understand Multiple Inheritance
- Understand Overriding methods
- Override methods of parent class
- Learn about abstract methods

In the previous chapter you have studied how to implement concept of classes and object using python. This chapter deals with Inheritance in Python.

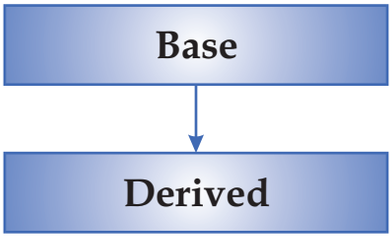
In object oriented programming, inheritance is a mechanism in which a new class is derived from an already defined class. The derived class is known as a subclass or a child class. The pre-existing class is known as base class or a parent class or a super class.The mechanism of inheritance gives rise to hierarchy in classes. The major purpose of inheriting a base class into one or more derived class is code reuse. The subclass inherits all the methods and properties of the super class. The subclass can also create its own methods and replace methods of the superclass. The process of replacing methods defined in super with new methods with same name in the derived class is known as overriding.

Before we learn how we implement the concept of inheritance in Python, let us learn about types of inheritance.

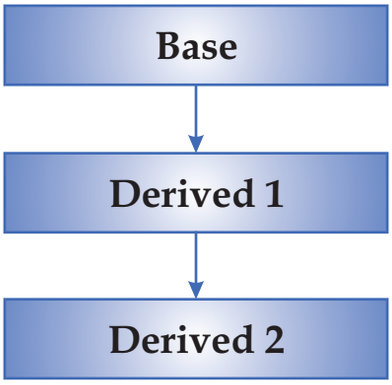
Inheritance can be categorised into five types:

- Single Inheritance
- Multilevel Inheritance
- Multiple Inheritance
- Hierarchical Inheritance
- Hybrid Inheritance

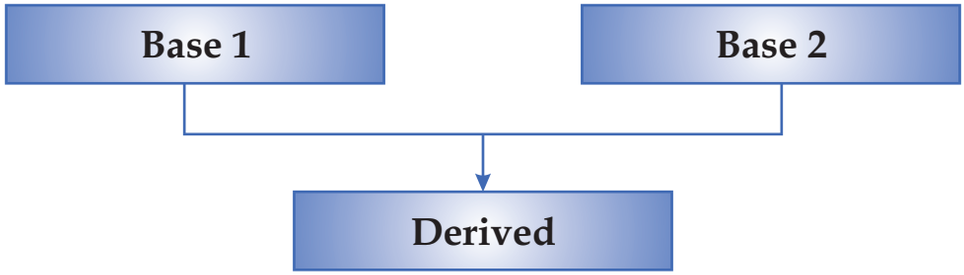
Single Inheritance: This is the simplest kind of inheritance. In this, a subclass is derived from a single base class.



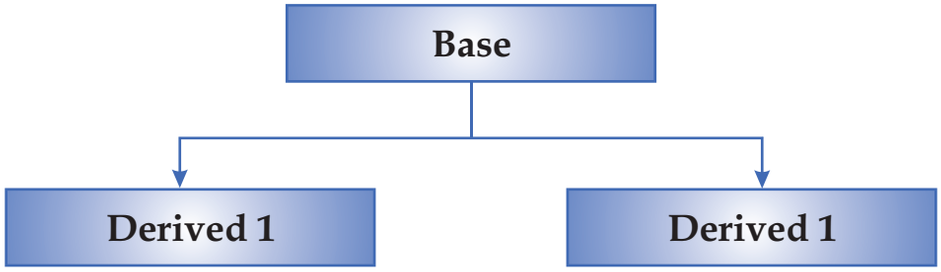
Multilevel Inheritance: In this type of inheritance, the derived class becomes the base of another class.



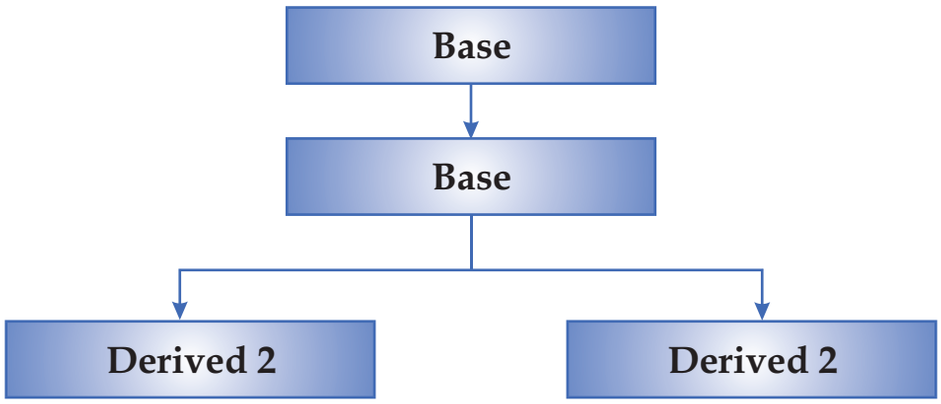
Multiple Inheritance: In this type of inheritance, the derived class inherits from one or more base classes.



Hierarchical Inheritance: In this type of inheritance, the base class is inherited by more than one class.



Hybrid Inheritance: This inheritance is a combination of multiple, hierarchical and multilevel inheritance.



Now let us see how inheritance can be established in Python .This can be done by using the following syntax.

Syntax: class subclass (super):

For Example

```
class person(object):
    def __init__(self,name,age):
        self.name=name
        self.age=age
    def getName(self):
            return self.name
    def getAge(self):
            return self.Age
class student(person):
    def __init__(self,name,age,rollno,marks):
        super(student,self)._init_(self, name, age)
        self.rollno=rollno
        self.marks=marks
    def getRoll(self):
            return self.rollno
    def getMarks(self):
            return self.Marks
```

The above example implements single inheritance. The class student extends the class person. The class student adds two instance variables rollno and marks. In order to add new instance variables, the _init_() method defined in class person needs to be extended. The __init__() function of subclass student initializes name and age attributes of superclass person and also create new attributes rollno and marks.

```
>>p=person("Ram",15)
>>p.getName()
Ram
>>P.getAge( )
15
>> s=student("Sita",16,15,78)
>>s.getName( )
Sita
>>S.getAge( )
16
>>s.getMarks( )
78
```

In the above code segment, the object of class person takes values for instance variables name and age. The object of class student takes values for instance variables name, age, rollno and marks. The object of class student can invoke getName() function because it inherits this function from the base class. Similarly, object of class student can also access instance variables of class person.

In python the above task of extending __init__ () can be achieved the following ways:

i) By using super() function

ii) By using name of the super class.

Method-I

By using super() function

In the above example, the class student and class person both have __init__ () method. The __init__ () method is defined in cIass person and extended in class student. In Python, super() function is used to call the methods of base class which have been extended in derived class

Syntax:

super(type, variable) bound object

Example

```
class student(person):
    def __init__(self,name,age,rollno,marks):
        super(student, self).__init__ (self, name, age)
        self.rollno=rollno
        self.marks=marks
```

Method-II

By using name of the super class

As discussed above, the class student and class person both have __init__ () method. The __init__ () method is defined in cIass person and extended in class student. In Python, name of the base class can also be used to access the method of the base class which has been extended in derived class.

Example

```
class student(person):
    def __init__(self,name,age,rollno,marks):
        person.__init__ (self, name, age)
        self.rollno=rollno
        self.marks=marks
```

Multiple Inheritance

Python supports a limited form of multiple inheritance as well. A class definition with multiple base classes looks like this:

```
class SubClassName( Base1, Base2, Base3):
<statement1>
.
.
.
.
<statement N>
```

For old-style classes, the only rule is depth-first, left-to-right. Thus, if an attribute is not found in SubClassName, it is searched in Base1, then (recursively) in the base classes of Base1, and only if it is not found there, it is searched in Base2, and so on.

For Example:

```
class student(object):
    def __init__(self,Id,name):
        self.Id=Id
        self.name=name
    def getName(self):
        return self.name
    def getId(self):
        return self.Id
    def show(self):
        print self.name
```

```
        print self.Id
class Teacher(object):
    def __init__(self,tec_Id,tec_name, subject):
        self.tec_Id=tec_Id
        self.tec_name=tec_name
        self.subject=subject
    def getName(self):
        return self.tec_name
    def getId(self):
        return self.tec_Id
    def getSubject(self):
        return self.ubject
    def show(self):
        print self. tec_name
        print self.tec_Id
        print self.subject
class school(student,Teacher):
    def __init__(self, ID, name, tec_Id, tec_name, subject, sch_Id):
        student.__init__ (self,ID,name)
        Teacher. __init__(self,tec_Id,tec_name, subject)
        self.sch_Id= sch_Id
    def getId(self ):
        return self.sch_Id
    def display(self):
        return self.sch_Id
```

In above example class school inherits class student and teacher.

Let us consider these outputs

```
>>> s1=school(3,"Sham",56,"Ram","FIT",530)
>>> s1.display()
530
>>> s1.getId()
530
>>> s1.show()
Sham
3
```

The object of class school takes six instance variables. The first five instance variables have been defined in the base classes (school and Teacher). The sixth instance variable is defined in class school.

s1.display()displays the sch_Id and s1.getId() also returns the sch_id. The classes school and Teacher both have the method show (). But as shown in above, the objects1 of class school access the method of class student. This because of depth-first, left-to-right rule.

Also, method getId()of class school is overriding the methods with same names in the base classes.

Overriding Methods

The feature of overriding methods enables the programmer to provide specific implementation to a method in the subclass which is already implemented in the superclass. The version of a method that is executed will be determined by the object that is used to invoke it. If an object of a parent class is used to invoke the method, then the version in the parent class will be executed, but if an object of the subclass is used to invoke the method, then the version in the child class will be executed.

Example 1:

```
class student(person):
    def __init__(self,name,age,rollno,marks):
        super(student,self).__init__(self, name, age)
        self.rollno=rollno
        self.marks=marks
def getRoll(self):
        return self.rollno
def getMarks(self):
        return self.Marks
def show(self):
        print self.rollno
        print self.marks
```

As shown in the above example class student inherits class person. Both the classes have a function show( ). The function show() in student is overriding function show() in person(). The object of class person will print name and age. The object of class student will print rollno and marks. In case it is required that function show of class student should display name, age, rollno and marks, we should make the following change in class student

```
class student(person):
    def __init__(self,name,age,rollno,marks):
        super(student,self).__init__(self,name,age)
```

```
        self.rollno=rollno
        self.marks=marks
def getRoll(self):
        return self.rollno
def getMarks(self):
        return self.Marks
def show(self):
    person.show()
    print self.rollno
    print self.marks
```

Example 2:

```
class student(object):
    def __init__(self,Id,name):
        self.Id=Id
        self.name=name
    def getName(self):
        print self.name
    def getId(self):
        print self.Id
    def show(self):
        print self.name
        print self.Id
class Teacher(object):
    def __init__(self,tec_Id,tec_name, subject):
        self.tec_Id=tec_Id
        self.tec_name=tec_name
        self.subject=subject
    def getName(self):
        print self.tec_name
    def getId(self):
        print self.tec_Id
    def getSubject(self):
        print self.subject
```

```
    def show(self):
        print self. tec_name
        print self.tec_Id
        print self.subject
class school(student,Teacher):
    def __init__(self, ID, name, tec_Id, tec_name, subject, sch_Id):
        student.__init__ (self,ID,name)
        Teacher. __init__(self,tec_Id,tec_name, subject)
        self.sch_Id= sch_Id
    def getId(self ):
                student.getId(self)
                Teacher.getId(self)
                return self.sch_Id
    def getName(self):
        student.getName(self)
        Teacher.getName(self)
    def show(self):
        student.show(self)
        Teacher.show(self)
        return self.sch_Id
```

Abstract Methods

An abstract method is a method declared in a parent class, but not implemented in it. The implementation of such a method can be given in the derived class.

Method to declare an abstract method

Method

```
>>> class circle(object):
        def get_radius(self):
            raise NotImplementedError
>>> c=circle()
>>> c.get_radius()
Traceback (most recent call last):
File "<pyshell#2>", line 1, in <module>
c.get_radius()
```

```
File "<pyshell#0>", line 3, in get_radius
raise NotImplementedError
NotImplementedError
```

In the above example, the class circle has a unimplemented method get_radius().

As soon as we call get_radius function using the object of class circle, it raises a "Not Implemented Error".

Now Let us see a class circle1() which implements the abstract method of class circle

```
>>> class circle1(circle):
        def __init__(self,radius):
            self.radius=radius
        def get_radius(self):
            return self.radius
>>> c=circle1(10)
>>> c.get_radius()
10
```

Any class inheriting the class circle canimplement and override the get_radius() method. In case the derived class doesn't implement get_radius() method, it will have to raise"NotImplementedError" exception. This has been shown in the example shown below

```
>>> class circle1(circle):
        def __init__(self,radius):
            self.radius=radius
        def get_radius(self):
             raise NotImplementedError
>>> c1=circle1(10)
>>> c1.get_radius()
Traceback (most recent call last):
File "<pyshell#34>", line 1, in <module>
c1.get_radius()
File "<pyshell#32>", line 6, in get_radius
raise NotImplementedError
NotImplementedError
```

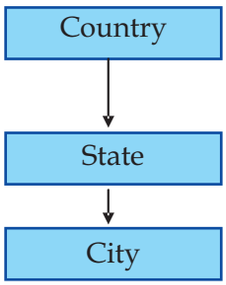
The method of implementing abstract methods by using the "NotImplementedError" exception has a drawback.

If you write a class that inherits from circle and forget to implement get_radius, the error will only be raised when you'll try to use that method.

LET'S REVISE

- **Inheritance:** In object oriented programming, inheritance is a mechanism in which a new class is derived from an already defined class. The derived class is known as a subclass or a child class. The pre-existing class is known as base class or a parent class or a super class.
- **Single Inheritance:** In single inheritance a subclass is derived from a single base class.
- **Multilevel Inheritance:** In multilevel inheritance, the derived class becomes the base of another class.
- **Multiple Inheritance:** In this type of inheritance, the derived class inherits from one or more base classes.
- **Hierarchical Inheritance:** In this type of inheritance, the base class is inherited by more than one class.
- **Hybrid Inheritance:** This inheritance is a combination of multiple, hierarchical and multilevel inheritance.
- **Overriding Methods:** The feature of overriding methods enables the programmer to provide specific implementation to a method in the subclass which is already implemented in the superclass.
- **Abstract Methods:** An abstract method is a method declared in a parent class, but not implemented in it. The implementation of such a method can be given in the derived class.

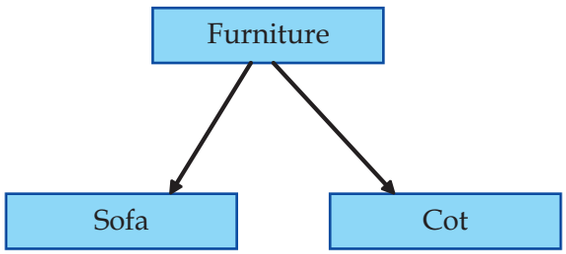
EXERCISE

1. Define the term inheritance.
2. Give one example for abstract methodσ
3. What is single inheritanceσ
4. What is multiple inheritanceσExplain with an example.
5.. Give one example of multilevel inheritance.
6. 

   Based on the above diagram, answer the following;

   (i) Write the name of the base class and derived class of state.

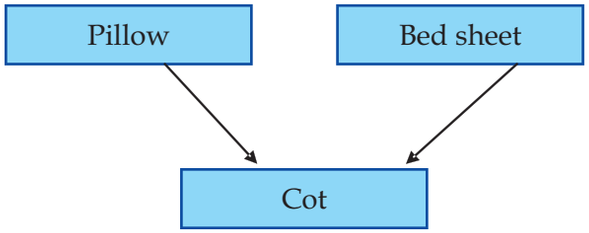
   (ii) Write the type of inheritance depicted in the diagram.

7. 

   Based on the above diagram, answer the following;

   (i) Write the name of the base class and derived classes.

   (ii) Write the type of inheritance depicted in the above diagram.

8. 

Based on the above diagram, answer the following;

(i) Write the name of the base classes and derived class

(ii) Name the type of inheritance depicted in the above example.

9. Explain the significance of super( ) function.
10. What are abstract methodsσ
11. Explain the concept of overriding methods.
12. Explain multiple inheritance with the help of an example.
13. How do we implement abstract method in Python. Support your answer with an example.
14. Find the output of the following code and write the type of inheritance:

a)

```
class person(object):
    def __init__(self,name,age):
    self.name=name
    self.age=age
    def display1(self):
    print "Name :",self.name
    print "Age  :",self.age
class student(person):
 def __init__(self,name,age,rollno,marks):
    super(student,self).__init__(name,age)
    self.rollno=rollno
    self.marks=marks
    def display(self):
     self.display1()
    print " Roll No:",self.rollno
     print " Marks  :",self.marks
    p=student('Mona',20,12,99)
p.display()
```

b)

```
class person(object):
def __init__(self,name,age):
      self.name=name
      self.age=age
def display1(self):
      print "Name :",self.name
      print "Age  :",self.age
class student(person):
def __init__(self,name,age,rollno,marks):
      super(student,self).__init__(name,age)
      self.rollno=rollno
      self.marks=marks
def display(self):
      self.display1()
      print " Roll No:",self.rollno
      print " Marks  :",self.marks
class Gstudent(student):
def __init__(self,name,age,rollno,marks,stream):
      super(Gstudent,self).__init__(name,age,rollno,marks)
 self.stream=stream
def display2(self):
      self.display()
      print " stream:",self.stream
p=Gstudent('Mona',20,12,99,'computer')
p.display2()
```

c)

```
class person(object):
    def __init__(self,name,age):
    self.name=name
    self.age=age
def display1(self):
    print "Name :",self.name
    print "Age  :",self.age
class student(object):
def __init__(self,rollno,marks):
    self.rollno=rollno
    self.marks=marks
def display(self):
    print " Roll No:",self.rollno
    print " Marks  :",self.marks
class Gstudent(person,student):
def __init__(self,name,age,rollno,marks,stream):
    super(Gstudent,self).__init__(self,stream)
    person.__init__(self,name,age)
    student.__init__(self,rollno,marks)
    self.stream=stream
def display2(self):
     self.display1()
    self.display()
    print " stream:",self.stream
p=Gstudent('Mona',20,12,99,'computer')
p.display2()
```

15. Rewrite the following code after removing errors. Underline each correction and write the output after correcting the code:

```
class First():
def __init__(self):
    print "first":
class Second(object):
def __init__(self):
    print "second"
class Third(First, Second):
def __init__(self):
    First.__init__(self):
    Second.__init__(self):
    print "that's it"
t=Third()t=Third()
```

16. Complete the following code:

```
class employee(object):
def __init__(self,no,name,age):
    self.no=no
    self.name=_____          #complete the statement
    self.age=_____           #complete the statement
def printval(self):
    print "Number:",self.no
    print "Name :",self.name
    print "Age  :",self.age
class pay(object):
    def __init__(self,dept,salary):    #complete the definition
    _____
    _____
```

```
def display(self):          #complete the definition
_____                       # call printval()
_____                       # print dept
_____                       # print salary
```

17. Define a class furniture in Python with the given specifications:

    Instance variables:

    - Type
    - Model

    Methods

    - getType() To return instance variable Type
    - getModel() To return instance variable Model
    - show() To print instance variable Type and Model

    Define a class sofa:

    - No_of_seats
    - Cost

    Methods

    - getSeats() To return instance variable No_of_seats
    - getCost() To return instance variable Cost
    - show() To print instance variable No_of_seats and Cost

    This class inherits class furniture

18. Define a class trainer in Python with the given specifications:

    Instance variables:

    - Name
    - T_ID

    Methods

    - getName()To print instance variable Name

2 getTID()To print instance variable T_ID

Define a class learner in Python with the given specifications:

Instance variables:

- L_Name
- L_ID

Methods

- getName()To print instance variable L_Name
- getLID()To print instance variable L_ID

Define a class Institute in Python with the given specifications:

Instance variables:

- I_Name
- I_Code

Methods

- getName()To print instance variable I_Name
- getICode()To print instance variable I_Code

The class Institute inherits the class Trainer and Learner

19. Define a class student in Python with the given specifications:

Instance variables:

Roll number, name

Methods:

Getdata()- To input roll number and name

Printdata()- To display roll number and name

Define another class marks, which is derived from student class

Instance variable

Marks in five subjects

Methods:

Inputdata() - To call Getdata() and input 5 subjects marks.

Outdata() - To call printdata() and to display 5 subjects marks.

Implement the above program in python.

20. Define a class employee in Python with the given specifications:

Instance variables:

Employee number, name

Methods:

Getdata()- To input employee number and name

Printdata()- To display employee number and name

Define another class Teaching, which is derived from employee

Instance variable

Department name

Methods:

Inputdata() - To call Getdata() and input department name.

Outdata() - To call printdata() and to display department name.

Define another class Non_teaching, which is derived from employee

Instance variable

Designation

Methods:

Inputdata() - To call Getdata() and input designation.

Outdata() - To call printdata() and to display designation.

Implement the above program in python.

21. Define a class employee in Python with the given specifications:

Instance variables:

Employee number, name

Methods:

Getdata()- To input employee number and name

Printdata()- To display employee number and name

Define another class payroll, which is derived from employee

Instance variable

Salary

Methods:

Inputdata() - To call Getdata() and input salary.

Outdata() - To call printdata() and to display salary.

Define another class leave, which is derived from payroll.

Instance variable

No of days

Methods:

acceptdata() - To call Inputdata() and input no of days.

showdata() - To call Outdata() and to display no of days.

Implement the above program in python.

22. Pay roll information system:

- Declare the base class 'employee' with employee's number, name, designation, address, phone number.
- Define and declare the function getdata() and putdata() to get the employee's details and print employee's details.
- Declare the derived class salary with basic pay, DA, HRA, Gross pay, PF, Income tax and Net pay.
- Declare and define the function getdata1() to call getdata() and get the basic pay,
- Define the function calculate() to find the net pay.
- Define the function display() to call putdata() and display salary details .
- Create the derived class object.
- Read the number of employees.
- Call the function getdata1() and calculate() to each employees.
- Call the display() function.

23. Railway reservation System

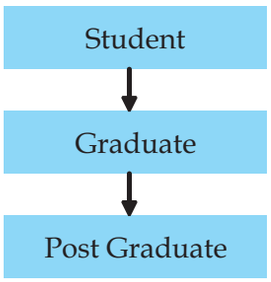
- Declare the base class Train with train number, name, Starting station, destination, departure time, arrival time, etc.
- Define and declare the function getdata() and putdata() to get the train details and print the details.
- Define search() function to search train detail using train number.

- Declare the derived class passenger with ticket number, PNR name of the passenger, gender, age, address, phone number, etc.
- Declare and define the function getdata1() to call search() function to get the train details and get the passenger's information.
- Define the function display() to call putdata() and display passenger's details.
- Create base class object.
- Read the number of trains.
- Create the derived class object.
- Read the number of passengers.
- Call the function getdata1() to each passenger.
- Call the display() function.

24. SKP Hotel offers accommodation, meals facilities.

- Create a class Accommodation with Room Number, type of room, and rent, etc..
- Create a class meals services includes: meals code, name, price, etc..
- Create a class customer with customer number, name, address, etc.
- Customer class is derived by using Accommodation and meals classes.

25. Implement the following using multilevel information.



- Create a student class with student number and name.
- Class graduate is created by using student.
- Graduate class is created using subject code and subject name.
- Class Post Graduate is created by using Graduate.
- Post Graduate class is created using master subject code and master subject name

Unit-2: Advanced Programing with Python

Chpater-1 : Linear List Manipulation

Learning Objectives

At the end of this chapter the student will be able to:

- Understand data structures
- Understand sequential memory allocation
- Learn basic list operations -
  - Traversal in a list
  - Insertion in a sorted list
  - Deletion of an element from a list
- Learn Searching Techniques in a list-
  - Linear Search
  - Binary Search
- Learn Sorting a list
  - Selection Sort
  - Bubble Sort
  - Insertion Sort

Data Structures

A data structure is a group of data which can be processed as a single unit. This group of data may be of similar or dissimilar data types. Data Structures are very useful while programming because they allow processing of the entire group of data as a single unit. This makes managing of data simpler. The simplest form of data structure is an array which combines finite data of same data type.

Data structures are of two types: Linear and Non - Linear. In a linear data structure, the elements are stored in a sequential order. On the other hand, in a non linear data structure no sequential order is followed. It is a sort of multilevel data structure. Arrays, lists, stacks, queues, linked lists etc. are examples of linear data structure while tree, graph etc. is a non - linear data structure.

List is one of the simplest and most important data structures in Python. In class XI, you have studied that a list is a sequence of values of any data type. These ordered set of values are called elements or members of a list and are enclosed in square brackets[ ]. To identify a value of a list, we use index. In this chapter we will study how to traverse, edit and sort the list.

Implementation of List in memory

In any programming language, an array is defined as a set of contiguous data of similar data type. In most of the old languages, the length of the array is fixed at the time of declaration, although now recent languages provide the facility of variable length arrays. Python lists are actually arrays of variable length. The elements of a list are of heterogeneous types which means they are of different data types. Hence [4,"Good", 8.45, 'k'] is a valid list. This is possible because in Python language, technically pointers., and not objects are stored in the form of an array. So a list in Python is an array that contains elements (pointers to objects) of a specific size only and this is a common feature of all dynamically typed languages. For implementation of a list, a contiguous array of references to other objects is used. Python keeps a pointer to this array and the array's length is stored in a list head structure. This makes indexing of a list independent of the size of the list or the value of the index. When items are appended or inserted, the array of references is resized.

Sequential Memory Allocation

As you have studied before, a list is allocated memory in sequential manner. This means that the elements of the list are stored in memory in sequence of their declaration. So if you want to view the fourth element of the list, you have to first traverse through first three elements of the list. This is called sequential allocation of memory.

The list is allocated memory in accordance with its members and their types. Also if memory is being shared by two or more members of the list, then memory consumption will be different from the situation where no sharing is involved. In Python, the length of the lists is not fixed as elements can be dynamically added and removed from the lists.

Secondly memory consumption of the list is referenced by a pointer, which will take 4 bytes in 32 bit mode and 8 bytes in 64 bit mode. Here mode refers to the word size of the processor (Remember Unit- I of class XI). Hence the size of list will be 4 times the number of objects in the list in the 32 bit mode and 8 times the number of objects in the list in 64 bit mode. Other than these every list has a fixed header overhead and some other over allocations involved for all Python lists.

Considering all the given factors, calculation of address of individual elements is beyond the scope of this book.

List Operations

Traversal, insertion and deletion are three main operations of any list in Python. Let us study all these operations in detail.

> Please note that in all algorithms and programs that follow, DATA_LIST is the list containing size elements. U is the upper bound or the maximum index of the list and is equal to size-1. Also for convenience, we also assume that DATA_LIST contains only integer elements.

Traversal in a List

Traversal means to move in a list in sequential manner starting from first element to the last element. The algorithm and code for traversal in the list is as follows:

Algorithm

1. ctr=0
2. Repeat steps 3 through 4 until ctr>U
3. print DATA_LIST[ctr]
4. ctr=ctr+1
   #End of repeat
5. END

Program Code

```
def traversal(DATA_LIST):
    for x in DATA_LIST:
        print x
    print
```

In the above code, the for loop runs till the time all elements of the list are displayed.

Insertion of an element in a sorted list

If insertion is done in an unsorted list, the new element is inserted at the end of the list, using append(). You have studied about this in class XI. But if the list is sorted, then the new element has to be inserted at an appropriate position so that it falls in order with the other elements of the list. To insert the new element at the proper position, the rest of the elements have to be shifted as shown in the figure:

Original List		Space created for the new element	List after insertion
2	Element to be inserted	2	2
6	12	6	6
7		7	7
10		10	10
14			12
15		14	14
16		15	15
19		16	16
		19	19

Fig: Insertion in a sorted list

If the list is sorted in ascending order, then the appropriate position for the new element, say ELEM has to be determined first. If the appropriate position is i+1, then DATA_LIST[i]<=ELEM<=DATA_LIST[i+1].

Algorithm

1. Set ctr=0, U=size-1
2. if DATA_LIST[ctr]>ELEM then

   position=1

   else
3. { Repeat steps 5 and 6 until ctr>=U
4. if DATA_LIST[ctr]<=ELEM and ELEM<=DATA_LIST[ctr+1] then

   { position=ctr+1

   break;

   }
5. ctr=ctr+1

   #End of repeat
6. if ctr=U then

   position= U+1

   }
7. ctr=U
8. while ctr>=position perform steps 10 through 11
9. { DATA_LIST[ctr+1]=DATA_LIST[ctr]
10. ctr=ctr-1

    }
11. DATA_LIST[position]=ELEM
12. END

The above algorithm can be implemented in Python to develop a program to insert a new element in the sorted list. But Python provides a more efficient and simple way to add a new element into the sorted list. It is through the bisect module which implements the given algorithm in Python for inserting elements into the list while maintaining list in sorted order. It is found to be more efficient than repeatedly or explicitly sorting a list.

Consider the following program which generates a random number and inserts them into a sorted list.

```
#Generate random numbers and inserts them into a list sorted in ascending order.
import bisect
import random
random.seed(1)
print 'New Element---Location---Contents of the list'
print
 DATA_LIST = []
for i in range(1, 15):
     var = random.randint(1, 100)
    loc = bisect.bisect(DATA_LIST, var)
    bisect.insort(DATA_LIST, var)
    print '%3d      %3d' % (var, loc), DATA_LIST
```

The above program uses the random module and bisect module. The seed method of the random module uses a constant seed to ensure that the same pseudo random numbers are used each time a loop is run. The randint method of this module generates a random number between 1 and 100.

The bisect method of the bisect module used in the program given above gives the appropriate location of the new element and the insort method inserts the new element in the proper position.

The output of the above program will be

```
New Element--- Location---Contents of the list
    19          0       [19]
    4           0       [4, 19]
    54          2       [4,19,54]
    26          2       [4,19,26,54]
    80          4       [4,19,26,54,80]
    32          2       [4,19,26,32,54,80]
    75          5       [4,19,26,32,54,75,80]
    49          4       [4,19,26,32,49,54,75,80]
    2           0       [2,4,19,26,32,49,54,75,80]
```

In the above output, the first column of the output shows the new random number. The second column shows the position or location where the number will be inserted into the list. The last column shows the content of the list.

Deletion of an element from the sorted array

To delete the element from the sorted list, the element to be deleted has to be first searched for in the array using either linear search or binary search algorithm (discussed in the next section of this chapter). If the search is successful, the element is removed from the list and the rest of the elements of the list are shifted such that the order of the array is maintained. Since the elements are shifted upwards, the free space is available at the end of the array.

Algorithm

1. ctr=position # position is the position of the element to be deleted
2. Repeat steps 3 and 4 until ctr<=U
3. DATA_LIST[ctr]=DATA_LIST[ctr+1]
4. ctr=ctr+1

   #end of repeat

Python code to delete an element at the position pos, from a sorted list

```
def delete_element(DATA_LIST , pos):
ctr=pos
while(ctr<=U):          #U= size of list-1
    DATA_LIST[ctr]=DATA_LIST[ctr+1]
    ctr=ctr+1
```

Searching Techniques

There are many searching algorithms. We shall be discussing two of them - Linear Search and Binary Search.

Linear search

In linear search, the search element is compared with each element of the list, starting from the beginning of the list. This continues till either the element has been found or you have reached to the end of the list. That is why this type of searching technique is also called linear search.

The algorithm given below searches for an element, ELEM in the list named DATA_LIST containing size number of elements.

Algorithm

1. Set ctr=0 , U= size -1 # size is the size of the list L
2. Repeat steps 3 through 4 until ctr>U
3. if DATA_LIST[ctr]==ELEM then
   { print " Element found at "
   print ctr +1
   break
   }
4. ctr= ctr+1 # repeat ends
5. if ctr>U then
   print "Element not found"
6. END

Python Program to search an element ELEM from the list called DATA_LIST

```
def linear_search(DATA_LIST, ELEM):
    flag=0
for ctr in DATA_LIST:
    if DATA_LIST[ctr]==ELEM :
        print " Element found at "
        print ctr +1
        flag=1
        break
 if flag==0:
    print "Search not successful----Element not found"
```

Linear Searching technique is simple to use but has few drawbacks. If the element to be searched is towards the end of the array, the searching process becomes very time consuming. This is because the algorithm searches for the element in a sequential manner starting form the first element. This drawback is overcome to a large extent in the next sorting technique i.e. binary search.

Binary search

This searching technique reduces the number of comparisons and hence saves on processing time. For binary search, the condition that has to be fulfilled is that the array should be sorted in either ascending or descending order.

To search for an element, ELEM in a list that is sorted in ascending order, the ELEM is first compared with the middle element. If ELEM is greater than the middle element, latter part of the list is searched. So, this latter part becomes the new sub-array or the new segment of the array to be scanned further. On the contrary, if ELEM is smaller than the middle element, former part of the list becomes the new segment. For the first stage, the segment contains the entire array. In the next stage the segment is half of the list, in next stage it becomes one-fourth of the entire list and so on. So, the number of comparisons to be made, in this case are reduced proportionately, thereby decreasing the time spent in searching the element.

Let us assume that the element, ELEM has to be searched in a list that is sorted in ascending order. ELEM is first compared with the middle element of the list. If ELEM is greater than the middle element of the list, the second half of the list becomes the new segment to be searched. If the ELEM is less than the middle element, then the first half of the list is scanned for the ELEM. The process is repeated till either the element is found or we are left with just one element in the list to be checked and the ELEM is still not found.

Algorithm

In the algorithm given below, low and high are upper bound and lower bound respectively of DATA_LIST.

1. Set low=0 , high=size-1
2. Repeat steps 3 through 6 until low<high
3. mid=(int) (low+high)/2
4. if DATA_LIST[mid]==ELEM then

```
{
    print "Element found at"
    print mid
    break;
}
```

5. if DATA_LIST[mid]<ELEM then

```
    low=mid + 1
```

6. if DATA_LIST[mid]>ELEM then

```
        high=mid-1                    #End of Repeat
7.  if low >= high
        Print "ELEMENT NOT FOUND"
8.  END
```

Python Program

```
def binary_search(DATA_LIST, ELEM, low, high):
    low=0
    high=len(DATA_LIST)
while(low<high):
    mid=(int)(low+high/2)
    if DATA_LIST[mid]==ELEM:
        print "Element found at"
        print mid
        break
    if DATA_LIST[mid]<ELEM:
        low=mid+1
    if DATA_LIST[mid]>ELEM:
        high=mid-1
    if low >= high
        print "ELEMENT NOT FOUND"
```

Sorting a list

Sorting is to arrange the list in an ascending or descending order. There are three sorting techniques that can be used to sort the list in ascending or descending order. These are selection sort, bubble sort and insertion sort.

Selection Sort

The basic logic of selection sort is to repeatedly select the smallest element in the unsorted part of the array and then swap it with the first element of the unsorted part of the list. For example, consider the following

array

10 5 19 6 80 4 15

Step 1: The smallest element of the array, i.e. 4 is selected. This smallest element is then swapped with the first element of the array. So now the array becomes

4 5 19 6 80 10 15

sorted unsorted

Step 2: Now again the smallest element is picked up from the unsorted part of the array. The smallest element in the unsorted part is 5. Since 5 is already at the right position, so the array does not reflect any change.

4 5 19 6 80 10 15

Sorted Unsorted

Step 3: The next shortest element from the unsorted part is 6 which is swapped with 19. The array now becomes

4 5 6 19 80 10 15

Sorted Unsorted

Step 4: Next 10 is swapped with 19

4 5 6 10 19 80 15

Sorted Unsorted

Step 5: Now 15 is swapped with 19

4 5 6 10 15 80 19

Sorted Unsorted

Step 6: 19 is swapped with 80

4 5 6 10 15 19 80

Sorted

The array is finally sorted in ascending order.

The code for a selection sort involves two nested loops. The outside loop tracks the current position that the code wants to swap the smallest value with. The inside loop starts at the current location and scans the rest of the list in search of the smallest value. When it finds the smallest value, swapping of elements takes place.

Algorithm

1. small = DATA_LIST[0] #initialize small with the first element
2. for i=0 to U do
   {
3. small=DATA_LIST[i]
   position=i
4. for j=i to U do
   {
5. if DATA_LIST[j]<small then
6. { small = DATA_LIST[j]
7. position=j
   }
   j=j+1
   }
8. temp=DATA_LIST[i]
9. DATA_LIST[i]=small
10. DATA_LIST[position]=temp
    }
11. END

Selection Sort Program

```
def selection_sort(DATA_LIST):
    for i in range(0, len (DATA_LIST)):
    min = i
    for j in range(i + 1, len(DATA_LIST)):
        if DATA_LIST[j] < DATA_LIST[min]:
            min = j
    temp= DATA_LIST[min];
```

```
DATA_LIST[min] = DATA_LIST[i]
DATA_LIST[i]=temp            # swapping
```

Bubble Sort

In case of bubble sort algorithm, comparison starts from the beginning of the list. Every adjacent pair is compared and swapped if they are not in the right order (the next one is smaller than the former one). So, the heaviest element settles at the end of the list. After each iteration of the loop, one less element (the last one) is needed to be compared until there are no more elements left to be compared. Say for example the following array is to be sorted in ascending order

90 11 46 110 68 51 80

Step 1: Compare the first two elements, i.e. 90 and 11. Since they are not in order, so both these elements are swapped with each other. The array now becomes

11 90 46 110 68 51 80

Step 2: Compare second and third element, i.e. 90 and 46. Since they are not in order, so they are swapped. The array now becomes

11 46 90 110 68 51 80

Step 3: Compare 90 and 110. Since they are in order, so no change in the array.

Step 4: Compare 110 and 68. Since 68 is less than 110, so both are swapped

11 46 90 68 110 51 80

Step 5: Compare 110 and 51 . They need to be swapped.

11 46 90 68 51 110 80

Step 6: 110 and 80 are swapped since they are not in order.

11 46 90 68 51 80 110

After these six steps, the largest element is settled at its proper place i.e. at the end. So the first six elements form the unsorted part while the last element forms the sorted part. The above steps have to be repeated again to get the following array:

11 46 68 51 80 90 110

Unsorted sorted

Repeating the steps again, the array becomes

11 46 51 68 80 90 110

Unsorted sorted

The steps will be repeated again till all the elements are sorted in ascending order. The final sorted array is

11 46 51 68 80 90 110

Algorithm

```
1. for i=L to U
2. {    for j=L to ((U-1)-i)   #the unsorted array reduces with every iteration
3.      {    if(DATA_LIST[j]>DATA_LIST[j+1] then
4.           {    temp=DATA_LIST[j]
5.                DATA_LIST[j]=DATA_LIST[j+1]
6.                DATA_LIST[j+1]=temp
           }
      }
   }
7. END
```

Bubble Sort Program

```
#Bubble Sort
def bubble_Sort(DATA_LIST):
i = 0
j = 0
for i in range(len(DATA_LIST)):
    for j in range(len(DATA_LIST) - i):
        if DATA_LIST[j] < DATA_LIST[j-1]:
            temp = DATA_LIST[j-1]
            DATA_LIST[j-1] = DATA_LIST[j]
            DATA_LIST[j] = temp
    print DATA_LIST
print DATA_LIST
```

Insertion Sort

As far as the functioning of the outer loop is considered, insertion sort is similar to selection sort. The difference is that the insertion sort does not select the smallest element and put it into place; rather it selects the next element to the right of what was already sorted. Then it slides up each larger element until it gets to the correct location to insert into the sorted part of the array.

Consider the following unsorted array:

70 49 31 6 65 15 51

Unsorted array

Step 1: First element, i.e. 70 is compared with second element i.e.49 and swapped

49 70 31 6 65 15 51

Step 2: Third element, 31 compared with 70 and swapped.

49 31 70 6 65 15 51

Step 3: Further 31 is also swapped with 49 because 31 is less than 49

31 49 70 6 65 15 51

Step 4: Next element, 6 is swapped with 70, then 49, then 31

6 31 49 70 65 15 51

Step 4: 65 swapped with 70

6 31 49 65 70 15 51

Step 5: 15 swapped with 70, then 65, then 49, and then 31 to be inserted in the appropriate position

6 15 31 49 65 70 51

Step 6: 51 is inserted at proper position after being swapped by 70 and then by 65

6 15 31 49 51 65 70

As is visible from the above example, the insertion sort also breaks the list into two sections, the "sorted" half and the "unsorted" half. In each round of the outside loop, the algorithm will grab the next unsorted element and insert it into the sorted part of the list.

In the code below, the variable K marks the boundary between the sorted and unsorted portions of the list. The algorithim scans to the left of K using the variable ptr. Note that in the insertion short, ptr goes down to the left, rather than up to the right. Each cell location that is larger than keyvalue, temp gets moved up (to the right) one location. When the loop finds a location smaller than temp, it stops and puts temp to the left of it.

The algorithm and program for insertion sort is given below:

Algorithm

1. ctr=0
2. repeat steps 3 through 8 for K=1 to size-1
   {
3. temp=DATA_LIST[k]
4. ptr=K-1
5. repeat steps 6 to 7 while temp<DATA_LIST[ptr]
   {
6. DATA_LIST[ptr+1]=DATA_LIST[ptr]
7. ptr=ptr-1
   }
8. DATA_LIST[ptr+1]=temp
   }
9. END

Program

```
def insertion_sort(DATA_LIST):
    for K in range (1, len(DATA_LIST)):
        temp=DATA_LIST[K]
        ptr=K-1
        while(ptr>=0)AND DATA_LIST[ptr]>temp:
            DATA_LIST[ptr+1]=DATA_LIST[ptr]
            ptr=ptr-1
        DATA_LIST[ptr+1]=temp
```

LET'S REVISE

- Data structure: A group of data which can be processed as a single unit.
- There are two types of data structures - Linear and Non linear.
- Array: a set of contiguous data of similar data type.
- Python lists are actually arrays of variable length and have elements of different data types.
- Sequential allocation of memory: Elements stored in sequence of their declaration.
- Traversal: To move in a list in a sequential manner starting from first element to the last element.
- Insertion of a new element in a sorted list has to be inserted at an appropriate position so that it falls in order with the other elements of the list.
- Searching algorithms - Linear Search and Binary Search.
- In linear search, the search element is compared with each element of the list, starting from the beginning of the list to the end of the list.
- Binary Search: This searching technique reduces the number of comparisons and hence saves on processing time.
- For binary search, the array should be sorted in either ascending or descending order.
- Sorting is to arrange the list in an ascending or descending order.
- Sorting techniques - Selection, Bubble, Insertion
- Selection Sort: To repeatedly select the smallest element in the unsorted part of the array and then swap it with the first element of the unsorted part of the list.
- In case of bubble sort algorithm, every adjacent pair is compared and swapped if they are not in the right order
- Insertion Sort- Selects the next element to the right of what was already sorted, slides up each larger element until it gets to the correct location

EXERCISE

1. Define a data structure.
2. Name the two types of data structures and give one point of difference between them.
3. Give one point of difference between an array and a list in Python.
4. How are lists implemented in memoryσ
5. What is sequential allocation of memoryσWhy do we say that lists are stored sequentiallyσ
6. How is memory allocated to a list in Pythonσ
7. Write a function that takes a list that is sorted in ascending order and a number as arguments. The function should do the following:

a. Insert the number passed as argument in a sorted list.

b. Delete the number from the list.

8. How is linear search different from binary searchσ
9. Accept a list containing integers randomly. Accept any number and display the position at which the number is found is the list.
10. Write a function that takes a sorted list and a number as an argument. Search for the number in the sorted list using binary search.
11. In the following list containing integers, sort the list using Insertion sort algorithm. Also show the status of the list after each iteration.

    15 -5 20 -10 10

12. Consider the following unsorted list

    Neena Meeta Geeta Reeta Seeta

Sort the list using selection sort algorithm. Show the status of the list after every iteration.

13. Consider the following unsorted list

    90 78 20 46 54 1

Write the list after:

a. 3rd iteration of selection sort

b. 4th iteration of bubble ort

c. 5th iteration of insertion sort

14. Consider the following unsorted list:

10 5 55 13 3 49 36

Write the position of elements in the list after:

a. 5th iteration of bubble sort

b. 7th iteration of insertion sort

c. 4th iteration of selection sort

15. Sort a list containing names of students in ascending order using selection sort.

16. A list contains roll_no, name and marks of the student. Sort the list in descending order of marks using Selection Sort algorithm.

17. A list contains Item_code, Item_name and price. Sort the list :

a. In ascending order of price using Bubble sort.

b. In descending order of qty using Insertion sort.

18. Accept a list containing numbers. Sort the list using any sorting technique. Thereafter accept a number and display the position where that number is found. Also display suitable message, if the number is not found in the list.

Chpater-2: Stacks and Queues in List

Learning Objectives:

At the end of this chapter the students will be able to :

- Understand a stack and a queue
- Perform Insertion and Deletion operations on stacks and queues
- Learn Infix and Postfix expressions
- Convert an infix to postfix
- Evaluation of Postfix Expression

In the previous chapter you studied about manipulation of lists in Python. In this chapter you further study about stacks and queues and their implementation in Python using lists.

Stack

A stack is a data structure whose elements are accessed according to the Last-In First-Out (LIFO) principle. This is because in a stack, insertion and deletion of elements can only take place at one end, called top of the stack. Consider the following examples of stacks:

1. Ten glass plates placed one above another. (The plate that is kept last has to be taken out first)
2. The tennis balls in a container. (You cannot remove more than one ball at a time)
4. A pile of books
6. A stack of coins

http://us.123rf.com

In the above picture coins are kept one above the other and if any additional coin is to be added, it can be added only on the top. If we want to remove any coin from the stack, the coin on the top of the stack has to be removed first. That means, the coin that was kept last in the stack has to be taken out first.

The two operations performed on the stack are:

1. Push: Adding(inserting) new element on to the stack.
2. Pop: Removing (deleting) an element from the stack

Push operation

Adding new element to the stack list is called push operation. When the stack is empty, the value of top is

1. Basically, an empty stack is initialized with an invalid subscript.Whenever a Push operation is performed, the top is incremented by one and then the new value is inserted on the top of the list till the time the value of top is less than or equal to the size of the stack. Let us first have a look at the logic to program the Push operation for a stack through the following algorithm:

    1. Start
    2. Initialize top with -1.

    Step 3: Input the new element.

    Step 4: Increment top by one.

    Step 5: stack[top]=new element

    Step 6: Print "Item Inserted"

    Step 7: Stop

Pop operation

Removing existing elements from the stack list is called pop operation. Here we have to check if the stack is empty by checking the value of top. If the value of top is -1, then the stack is empty and such a situation is called Underflow. Otherwise Pop operation can be performed in the stack. The top is decremented by one if an element is deleted from the list. The algorithm for pop operation is as follows:

Algorithm for Pop operation:

Step 1: Start

Step 2: If the value of top is -1 go to step 3 else go to step 4

Step 3: Print "Stack Empty" and go to step 7

Step 4: Deleted item = Stack[top]

Step 5: Decrement top by 1

Step 6: print "Item Deleted"

Step 7: Stop

Traversal in a stack

Traversal is moving through the elements of the stack. If you want to display all the elements of the stack, the algorithm will be as follows:

Step 1: Start
Step 2: Check the value of top. If top=-1 go to step 3 else go to step 4
Step 3: Print "Stack Empty" and go to step 7
Step 4: Print the top element of the stack.
Step 5: Decrement top by 1
Step 6: If top= -1 go to step 7 else go to step 4
Step 7: Stop

In Python, we already have pop() and append() functions for popping and adding elements on to the stack. Hence, there is no need to write the code to add and remove elements in the stack. Consider the following programs that perform the Push and Pop operation on the stack through append() and pop().

Program to implement stack (without classes)

```
s=[]
c="y"
while (c=="y"):
   print "1. PUSH"
   print "2. POP "
   print "3. Display"
   choice=input("Enter your choice:  ")
   if (choice==1):
      a=input("Enter any number      :")
      s.append(a)
   elif (choice==2):
      if (s==[]):
         print "Stack Empty"
      else:
         print "Deleted element is :   ",s.pop()
   elif (choice==3):
      l=len(s)
      for i in range(l-1,-1,-1):
         print s[i]
   else:
      print("Wrong Input")
   c=raw_input("Do you want to continue or notσ   ")
```

Output:

```
  >>>
1. PUSH
2. POP
3. Display
Enter your choice: 2
Stack Empty
Do you want to continue or notσ  y
1.PUSH
2.POP
3.Display
Enter your choice: 1
Enter any number     :100
Do you want to continue or notσ  y
1.  PUSH
2. POP
3. Display
Enter your choice: 1
Enter any number     :200
Do you want to continue or notσ  y
1. PUSH
2. POP
3. Display
Enter your choice: 1
Enter any number     :300
Do you want to continue or notσ  y
1. PUSH
2. POP
3. Display
Enter your choice: 3
300
200
100
```

```
Do you want to continue or notσ  y
1. PUSH
2. POP
3. Display
Enter your choice:  2
Deleted element is :    300
Do you want to continue or notσ  y
1. PUSH
2. POP
3. Display
Enter your choice:  3
200
100
Do you want to continue or notσ  n
>>>
```

The same program can also be implemented using classes as shown below:

Program to implement a stack(Using classes)

```
class stack:
   s=[]
   def push(self):
      a=input("Enter any number     :")
      stack.s.append(a)
   def display(self):
      l=len(stack.s)
      for i in range(l-1,-1,-1):
          print stack.s[i]
a=stack()
c="y"
while (c=="y"):
  print "1. PUSH"
  print "2. POP "
  print "3. Display"
  choice=input("Enter your choice:  ")
  if (choice==1):
```

```
    a.push()
  elif (choice==2):
    if (a.s==[]):
      print "Stack Empty"
    else:
      print "Deleted element is :  ",a.s.pop()
  elif (choice==3):
    a.display()
  else:
    print("Wrong Input")
  c=raw_input("Do you want to continue or notσ   ")output:
```

Output:

```
>>>
1. PUSH
2. POP
3. Display
Enter your choice:  1
Enter any number     :100
Do you want to continue or notσ   y
1. PUSH
2. POP
3. Display
Enter your choice:  1
Enter any number     :200
Do you want to continue or notσ   y
1. PUSH
2. POP
3. Display
Enter your choice:  3
200
100
Do you want to continue or notσ   y
1. PUSH
2. POP
```

```
3. Display
Enter your choice: 2
Deleted element is : 200
Do you want to continue or notσ y
1. PUSH
2. POP
3. Display
Enter your choice: 2
Deleted element is : 100
Do you want to continue or not: y
1. PUSH
2. POP
3. Display
Enter your choice: 2
Stack Empty
Do you want to continue or notσ n
>>>
```

Expression

You have already studied about expressions in class XI. An expression is a combination of variables, constants and operators. Expressions can be written in Infix, Postfix or Prefix notations.

Infix Expression: In this type of notation, the operator is placed between the operands.
For example: A+B, A*(B+C), X*Y/Z, etc

Postfix Expression: In this type of notation, the operator is placed after the operands.
For example: AB+, ABC+*,XYZ/*, etc.

Prefix Expression: In this type of notation, the operator is placed before the operands.
For example: +AB,*A+BC,*X/YZ, etc.

Conversion of an infix expression to postfix expression

The following algorithm shows the logic to convert an infix expression to an equivalent postfix expression:

Step 1: Start

Step 2: Add "("(left parenthesis) and ")" (right parenthesis) to the start and end of the expression(E).

Step 3: Push "("left parenthesis onto stack.

Step 4: Check all symbols from left to right and repeat step 5 for each symbol of 'E' until the stack becomes empty.

Step 5: If the symbol is:

i) an operand then add it to list.

ii) a left parenthesis "("then push it onto stack.

iii) an operator then:

a) Pop operator from stack and add to list which has the same or higher precedence than the incoming operator.

b) Otherwise add incoming operator to stack.

iv) A right parenthesis")" then:

a) Pop each operator from stack and add to list until a left parenthesis is encountered.

b) Remove the left parenthesis.

Step 6: Stop

Example 1:

A*(B+C)

Operator/Operand	Stack	Result (List)	Meaning
A	A		Operand moved to result list
*	*	A	Operator pushed to stack
(	*(	A	Open Parentheses pushed to stack
B	*(	AB	Operand moved to result list
+	*(+	AB	Operator pushed to stack
C	*(+	ABC	Operand moved to result list
)	*	ABC+	Close Parentheses encountered,pop operators up to open Parentheses and add it to the result list. Remove open parenthesis from the stack
		ABC+*	Expression empty - Final Result

The resultant postfix expression is A B C + *

Example 2:

A/B^C-D

Opeartor/Operand	Stack	Result (List)	Meaning
A	A		Operand moved to result list
/	/	A	Operator pushed to stack
B	/	AB	Operand moved to result list
^	/^	AB	Operator pushed to stack
C	/^	ABC	Operand moved to result list
-	-	ABC^/	As compared to - operator, ^and / has higher priority, so pop both operators and add them to the Result list
D	-	ABC^/D	Operand moved to result list
		ABC^/D-	Expression empty - Final Result

The resultant postfix expression is ABC^/D-

Evaluation of Postfix Expression

The algorithm to evaluate a postfix expression is as follows:

Step 1: Start

Step 2: Check all symbols from left to right and repeat steps 3 & 4 for each symbol of expression 'E' until all symbols are over.

i) If the symbol is an operand, push it onto stack.

ii) If the symbol is an operator then

a) Pop the top two operands from stack and apply an operator in between them.

b) Evaluate the expression and place the result back on stack.

Step 3: Set result equal to top element on the stack.

Step 4: Stop

Example 1:

6,5,2,*,10,4,+,+.-

Operator/Operand	Stack	Calculation	Meaning
6	6		Operand pushed tostack
5	65		Operand pushed to stack
2	652		Operand pushed tostack

*	6 10	5*2=10	Pop last two operands, perform the operation and push the result in stack
10	6 10	10	Operand pushed tostack
4	6 10	10 4	Operand pushed to stack
+	6 10 14	10+4=14	Pop last two operands, perform the operation and push the result on to stack
+	6 24	10+14=24	Pop last two operands, perform the operation and push the result on to stack.
-	-18	6-24=-18	Pop last two operands, perform the operation and push the result on to stack.

The answer is - 18

Example 2:

True False AND True True NOT OR AND

Operator /Operand	**Stack**	**Calculation**	**Meaning**
True	True	Operand pushed to stack	
False	True False		Operand pushed to stack
AND	False	TRUE AND FALSE = FALSE	Pop last two operands, perform the operation and push the result on to stack.
True	False True		Operand pushed to stack
True	False True True	Operand pushed to stack	
NOT	False True False	NOT TRUE=FALSE	Pop last two operands, perform the operation and push the result on to stack.
OR	FALSE TRUE	TRUE OR FALSE= TRUE	Pop last two operands, perform the operation and push the result on to stack.
AND	FALSE	FALSE AND TRUE = FALSE	Pop last two operands, perform the operation and push the result on to stack.

The answer is **False**

Queue

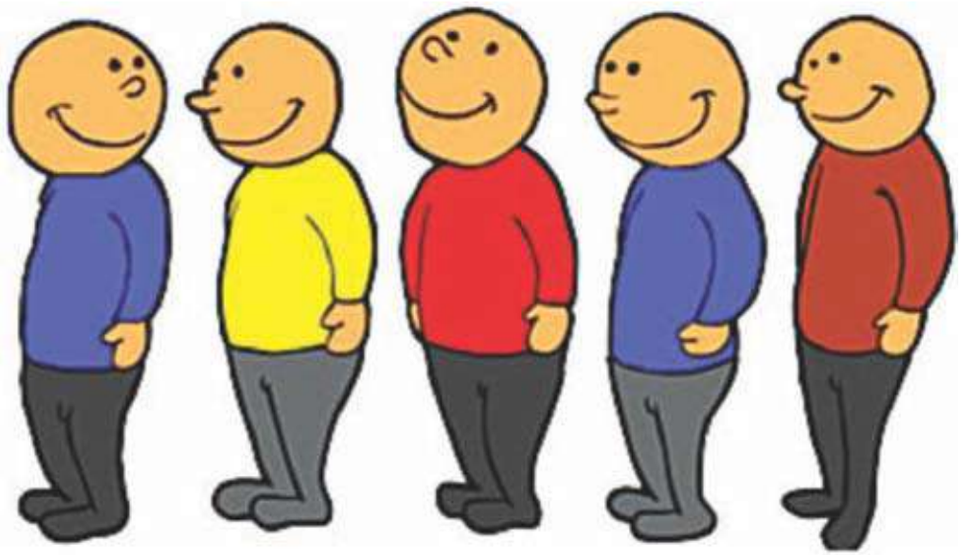
Another most common data structure found in computer algorithm(s) is queue. We are already familiar with it, as we run into enough of them in our day to day life. We queue up__

at the bank

at fee counter

at shopping centre etc.

A queue is a container of elements, which are inserted and removed according to the first-in first-out (FIFO) principle.



http://3.bp.blogspot.com/-

In a queue, persons who stand in the queue will carry out their work one by one. That means those who stands first in the queue will be allowed to carry out his work first and the person who stands at the second position will be allowed to carry out his work second only. At the same time those who come late will be joining the queue at the end. In simple terms it is called 'first come first out'.

Technically speaking a queue is a linear list, to keep an ordered collection of elements / objects. The principle operations, which can be performed on it are

- Addition of elements &
- Removal of elements.

Addition of element is known as INSERT operation, also known as enqueu-ing. It is done using rear terminal position, i.e. tail end. Removal of element is known as DELETE operation also know as dequeue-

ing. It is done using front terminal position, i.e. head of the list. As the two operations in the queue are performed from different ends, we need to maintain both the access points. These access points are known as FRONT, REAR. FRONT is first element of the queue and REAR is last element. As queue is FIFO implementation, FRONT is used for delete operation and REAR is used for insert operation.

Let's find out some applications of queue in computers:

- In a single processor multi tasking computer, job(s) waiting to be processed form a queue. Same happens when we share a printer with many computers.
- Compiling a HLL code
- Using down load manager, for multiple files also uses queue for ordering the files.
- In multiuser OS - job scheduling is done through queue.

Queue operations

Various operations, which can be performed on a queue are:

Create a queue having a data structure to store linear list with ordering of elements

Insert an element will happen using REAR, REAR will be incremented to hold the new value in queue.

Delete an element will happen using FRONT and FRONT will also be incremented to be able to access next element

Let's understand this with the help of an example:

1. We will use list data type to implement the queue.

2. As initially queue is empty, front and rear should not be able to access any element. The situation can be represented by assigning -1 to both REAR and FRONT as initial value.

F(front) = -1, **R**(rear) = -1

Once the queue is created, we will perform various operations on it. Following is the list of operations with its affect on queue:

INSERT(5)

F = F + 1

R = R + 1 (as this is the first element of the queue)

F R

INSERT(15)

R = R + 1

DELETE()

F = F + 1

INSERT(25)

R = R + 1

INSERT(35)

R = R + 1

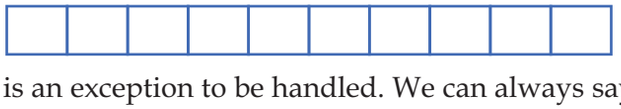
DELETE()

F = F + 1

DELETE()

F + F + 1

DELETE()



As the queue is empty, this is an exception to be handled. We can always say that deletion is attempted from an empty queue, hence not possible. The situation is known as **underflow** situation. Similarly when we work with fixed size **list,** insertion in a full list results into **overflow** situation. In python as we don't

have fixed size list, so don't need to bother about overflow situation.

Following are the formal steps for INSERT and DELETE operations

Algorithm for insertion:

Step 1: Start

Step 2: Check FRONT and REAR value, if both the values are -1, then
FRONT and REAR are incremented by 1
other wise
Rear is incremented by one.

Step 3: Add new element at Rear. (i.e.) queue[Rear]=new element.

Step 4: Stop

Algorithm for deletion:

Step 1: Start

Step 2: Check for underflow situation by checking value of Front = -1
If it is display appropriate message and stop
Otherwise

Step 3: Deleted item = queue [Front]

Step 4: If Front = Rear then Front =Rear = -1
Otherwise
Front is incremented by one

Step 5: Print "Item Deleted"

Step 6: Stop

Althogh principle operations in queue are Insert and Delete, but as a learner, we need to know the contents of queue at any point of time. To handle such requirement we will add traversal operation in our program. Following is the algorithm for same.

Algorithm for Display (Traversal in stack):

1. Start
2. Store front value in I
3. Check I position value, if I value is -1 go to step 4 else go to step 5
4. Print "Queue Empty" and go to step 8
5. Print queue[I]
6. I is incremented by 1
7. Check I position value, if I value is equal to rear+1 go to step 8 else go to step 5

Step 8: Stop

Note: In Python already we have **del()** and **append()** functions for deletion of elements at the front and addition of elements at the rear. Hence, no need of writing special function for add and remove elements in the queue. Likewise, 'Front and Rear positions' are also not required in the Python programming while implementing queue.

Example:

Write a program to implement Queue using list.

Code: (without using class)

```
a=[]
c='y'
while c=='y':
   print "1. INSERT"
   print "2. DELETE "
   print "3. Display"
   choice=input("enter your choice    ")
   if (choice==1):
         b=input("enter new number   ")
         a.append(b)
   elif (choice==2):
      if (a==[]):
         print("Queue Empty")
      else:
         print "deleted element is:",a[0]
         del a[0]
   elif (choice==3):
      l=len(a)
      for i in range(0,l):
          print a[i]
   else:
      print("wrong input")
   c=raw_input("do you want to continue or not   ")
```

Program: (Using class)

```
class queue:
```

```
    q=[]
    def insertion(self):
        a=input("enter any number:     ")
        queue.q.append(a)
    def deletion(self):
        if (queue.q==[]):
            print "Queue empty"
        else:
            print "deleted element is: ",queue.q[0]
            del queue.q[0]
    def display(self):
        l=len(queue.q)
        for i in range(0,l):
            print queue.q[i]
a=queue()
c="y"
while (c=="y"):
    print "1. INSERTION"
    print "2. DELETION "
    print "3. DISPLAY"
    choice=input("enter your choice:    ")
    if (choice==1):
        a.insertion()
    elif (choice==2):
        a.deletion()
    elif (choice==3):
        a.display()
    else:
        print("wrong input")
    c=raw_input("do you want to continue or not :")
```

LET'S REVISE

- **LIFO:** Last-In First-Out
- **FIFO:** First-In First-Out
- **Stack:** A stack is a container of elements that are inserted and removed according to the last-in first-out (LIFO) law.
- **Queue:** A queue is a container of elements, which are inserted and removed according to the first-in first-out (FIFO) law.
- **Infix Expression:** Operator is in between the operand.
- **Postfix Expression:** Operators are written after the operand.
- **Prefix Expression:** Operators are written before operand.

EXERCISE

1. Expand the following:
   (i) LIFO (ii) FIFO
2. What is stackσ
3. What is Queueσ
4. What are all operations possible in data structureσ
5. Give one example of infix expression.
6. Give one example of postfix expression.
7. Give one example of prefix expression.
8. Convert (A+B)*C in to postfix form.
9. Evaluate using stack 10, 3,*,30,2,*,-
10. Converting following Infix expression to Postfix notation:
    a) (A+B)*C+D/E-F
    b) P+Q*(R-S)/T
    c) (True And False) || (False And True)
11. Evaluation the following Postfix Expression:
    a) 20,8,4,/,2,3,+,*,-
    b) 15,3,2,+,/,7,+,2,*
    c) False, Not, True, And, True, False, Or, And
    d) True, False, Not, And, False, True, Or, And
12. Write the push operation of stack containing names using class.
13. Write the pop operation of stack containing numbers using class.
14. Write the insertion operation of queue containing character using class.
15. Write the deletion operation of queue containing numbers using class.
16. Write any two example of stack operation.
17. Write any two example of pop operation.
18. Write an algorithm to evaluate postfix expression.
19. Write an algorithm to convert infix to postfix.
20. Write an algorithm to implement push operation.
21. Write an algorithm to implement pop operation.
22. Write an algorithm to implement insertion operation of queue.
23. Write an algorithm to implement deletion operation of queue.
24. Write a function to push any student's information to stack.
25. Write a function to add any customer's information to queue.

Chapter-3: Data File Handling

Learning Objective

After going through the chapter, student will be able to:

- Understand the importance of data file for permanent storage of data
- Understand how standard Input/Output function work
- Distinguish between text and binary file
- Open and close a file (text and binary)
- Read and write data in file
- Write programs that manipulate data file(s)

Programs which we have done so far, are the ones which run, produce some output and end. Their data disappears as soon as they stop running. Next time when you use them, you again provide the data and then check the output. This happens because the data entered is stored in primary memory, which is temporary in nature. What if the data with which, we are working or producing as output is required for later useσResult processing done in Term Exam is again required for Annual Progress Report. Here if data is stored permanently, its processing would be faster. This can be done, if we are able to store data in secondary storage media i.e. Hard Disk, which we know is permanent storage media. Data is stored using file(s) permanently on secondary storage media.You have already used the files to store your data permanently - when you were storing data in Word processing applications, Spreadsheets, Presentation applications, etc. All of them created data files and stored your data, so that you may use the same later on. Apart from this you were permanently storing your python scripts (as .py extension) also.

A file (i.e. data file) is a named place on the disk where a sequence of related data is stored. In python files are simply stream of data, so the structure of data is not stored in the file, along with data. Basic operations performed on a data file are:

- Naming a file
- Opening a file
- Reading data from the file
- Writing data in the file
- Closing a file

Using these basic operations, we can process file in many ways, such as

Creating a file

Traversing a file for displaying the data on screen

Appending data in file

Inserting data in file

Deleting data from file

Create a copy of file

Updating data in the file, etc.

Python allow us to create and manage two types of file

- Text
- Binary

A text file is usually considered as sequence of lines. Line is a sequence of characters (ASCII), stored on permanent storage media. Although default character coding in python is ASCII but using constant u with string, supports Unicode as well. As we talk of lines in text file, each line is terminated by a special character, known as End of Line (EOL). From strings we know that \n is newline character. So at the lowest level, text file will be collection of bytes. Text files are stored in human readable form and they can also be created using any text editor.

A binary file contains arbitrary binary data i.e. numbers stored in the file, can be used for numerical operation(s). So when we work on binary file, we have to interpret the raw bit pattern(s) read from the file into correct type of data in our program. It is perfectly possible to interpret a stream of bytes originally written as string, as numeric value. But we know that will be incorrect interpretation of data and we are not going to get desired output after the file processing activity. So in the case of binary file it is extremely important that we interpret the correct data type while reading the file. Python provides special module(s) for encoding and decoding of data for binary file.

To handle data files in python, we need to have a file object. Object can be created by using open() function or file() function. To work on file, first thing we do is open it. This is done by using built in function open(). Using this function a file object is created which is then used for accessing various methods and functions available for file manipulation.

Syntax of open() function is

file_object = open(filename [, access_mode] [,buffering])

open() requires three arguments to work, first one (filename) is the name of the file on secondary storage media, which can be string constant or a variable. The name can include the description of path, in case, the file does not reside in the same folder / directory in which we are working. We will know more about this in later section of chapter. The second parameter (access_mode) describes how file will be used throughout the program. This is an optional parameter and the default access_mode is reading. The third parameter (buffering) is for specifying how much is read from the file in one read. The function will return an object of file type using which we will manipulate the file, in our program. When we work with file(s), a buffer (area in memory where data is temporarily stored before being written to file), is automatically

associated with file when we open the file. While writing the content in the file, first it goes to buffer and once the buffer is full, data is written to the file. Also when file is closed, any unsaved data is transferred to file. flush() function is used to force transfer of data from buffer to file.

File access modes:

r will open the text file for reading only and **rb** will do the same for binary format file. This is also the default mode. The file pointer is placed at the beginning for reading purpose, when we open a file in this mode.

w will open a text file for writing only and **wb** for binary format file. The file pointer is again placed at the beginning. A non existing file will be created using this mode. Remember if we open an already existing file (i.e. a file containing data) in this mode then the file will be overwritten as the file pointer will be at the beginning for writing in it.

a mode allow us to append data in the text file and ab in binary file. Appending is writing data at the end of the file. In this mode, file pointer is placed at the end in an existing file. It can also be used for creating a file, by opening a non existing file using this mode.

r+ will open a text file and **rb+** will open a binary file, for both reading and writing purpose. The file pointer is placed at the beginning of the file when it is opened using r+ / **rb+** mode.

w+ opens a file in text format and **wb+** in binary format, for both writing and reading. File pointer will be placed at the beginning for writing into it, so an existing file will be overwritten. A new file can also be created using this mode.

a+ opens a text file and ab+ opens a binary file, for both appending and reading. File pointer is placed at the end of the file, in an already existing file. Using this mode a non existing file may be created.

Example usage of open

file= open("Sample.txt","r+")

will open a file called Sample.txt for reading and writing purpose. Here the name (by which it exists on secondary storage media) of the file specified is constant. We can use a variable instead of a constant as name of the file. Sample file, if already exists, then it has to be in the same folder where we are working now, otherwise we have to specify the complete path. It is not mandatory to have file name with extension. In the example .txt extension is used for our convenience of identification. As it is easy to identify the file as text file. Similarly for binary file we will use . dat extension.

Other function, which can be used for creation of a file is file(). Its syntax and its usage is same as open().

Apart from using open() or file() function for creation of file, **with statement** can also be used for same purpose. Using **with** ensures that all the resources allocated to file objects gets deallocated automatically once we stop using the file. Its syntax is :

with open() as fileobject :

Example:

with open("Sample.txt","r+") as file :

file manipulation statements

Let's know about other method's and function's which can be used with file object.

fileobject. close() will be used to close the file object, once we have finished working on it. The method will free up all the system resources used by the file, this means that once file is closed, we will not be able to use the file object any more. Before closing the file any material which is not written in file, will be flushed off. So it is good practice to close the file once we have finished using it. In case, if we reassign the file object to some other file, then python will automatically close the file.

Methods for reading data from the file are:

readline() will return a line read, as a string from the file. First call to function will return first line, second call next line and so on. Remember file object keeps the track of from where reading / writing of data should happen. For readline() a line is terminated by \n (i.e. new line character). The new line character is also read from the file and post-fixed in the string. When end of file is reached, readline() will return an empty string.

It's syntax is

fileobject.readline()

Since the method returs a string it's usage will be

>>>x = file.readline()

or

>>>print file.readline()

For reading an entire file using readline(), we will have to loop over the file object. This actually is memory efficient, simple and fast way of reading the file. Let's see a simple example of it

>>>for line in file:

… print line

Same can be achieved using other ways of looping.

readlines()can be used to read the entire content of the file. You need to be careful while using it w.r.t. size of memory required before using the function. The method will return a list of strings, each separated by \n. An example of reading entire data of file in list is:

It's syntax is:

fileobject.readlines()

as it returns a list, which can then be used for manipulation.

read() can be used to read specific size string from file. This function also returns a string read from the file. At the end of the file, again an empty string will be returned.

Syntax of read() function is

fileobject.read([size])

Here size specifies the number of bytes to be read from the file. So the function may be used to read specific quantity of data from the file. If the value of size is not provided or a negative value is specified as size then entire file will be read. Again take care of memory size available before reading the entire content from the file.

Let's see the usage of various functions for reading data from file. Assuming we have a file **data.txt** containing hello world.\n this is my first file handling program.\n I am using python language

Example of readlines():

```
>>>lines = []
>>>lines = file.readlines()
```

If we print element of lines (which can be done by iterating the contents of lines) we will get:

hello world.

this is my first file handling program.

I am using python language.

Can you notice, there are two blank lines in between every string / sentence. Find out the reason for it.

Example of using read():

```
lines = []
content = file.read()          # since no size is given, entire file will be read
lines = content.splitlines()
print lines
```

will give you a list of strings:

['hello world.', 'this is my first file handling program.', 'I am using python language.']

For sending data in file, i.e. to create / write in the file, **write() and writelines()** methods can be used. write() method takes a string (as parameter) and writes it in the file. For storing data with end of line character, you will have to add \n character to end of the string. *Notice addition of \n in the end of every sentence while talking of data.txt.* As argument to the function has to be string, for storing numeric value, we have to convert it to string.

Its syntax is

fileobject.write(string)

Example

```
>>>f = open('test1.txt','w')
>>>f.write("hello world\n")
>>>f.close()
```

For numeric data value conversion to string is required.

Example

```
>>>x = 52
>>>file.write(str(x))
```

For writing a string at a time, we use write() method, it can't be used for writing a list, tuple etc. into a file. Sequence data type can be written using writelines() method in the file. It's not that, *we can't write a string using writelines() method.*

It's syntax is:

fileobject.writelines(seq)

So, whenever we have to write a sequence of string / data type, we will use writelines(), instead of write().

Example:

```
f = open('test2.txt','w')
str = 'hello world.\n this is my first file handling program.\n I am using python language"
f.writelines(str)
f.close()
```

let's consider an example of creation and reading of file in interactive mode

```
>>>file = open('test.txt','w')
>>>s = ['this is 1stline','this is 2nd line']
>>>file.writelines(s)
>>>file.close()
>>>file.open('test.txt') # default access mode is r
>>>print file.readline()
>>>file.close()
```

Will display following on screen

this is 1stline this is 2nd line

Let's walk through the code. First we open a file for creation purpose, that's why the access mode is **w**. In the next statement a list of 2 strings is created and written into file in 3rd statement. As we have a list of 2 strings, writelines() is used for the purpose of writing. After writing the data in file, file is closed.

In next set of statements, first one is to open the file for reading purpose. In next statement we are reading the line from file and displaying it on screen also. Last statement is closing the file.

Although, we have used **readline()** method to read from the file, which is suppose to return a line i.e. string at a time, but what we get is, both the strings. This is so, because writelines() does not add any EOL character to the end of string. You have to do it. So to resolve this problem, we can create s using following statement

s = ['this is 1st line\n', 'this is 2nd line\n']

Now using readline(), will result into a string at a time.

All reading and writing functions discussed till now, work sequentially in the file. To access the contents of file randomly **- seek** and **tell** methods are used.

tell() method returns an integer giving the current position of object in the file. The integer returned specifies the number of bytes from the beginning of the file till the current position of file object.

It's syntax is

fileobject.tell()

seek()method can be used to position the file object at particular place in the file. It's syntax is :

fileobject.seek(offset [, from_what])

here offset is used to calculate the position of fileobject in the file in bytes. Offset is added to from_what (reference point) to get the position. Following is the list of from_what values:

Value	reference point
0	beginning of the file
1	current position of file
2	end of file

default value of from_what is 0, i.e. beginning of the file.

Let's read the second word from the test1 file created earlier. First word is 5 alphabets, so we need to move to 5th byte. Offset of first byte starts from zero.

```
f = open('test1.txt','r+')
f.seek(5)
fdata = f.read(5)
print fdata
f.close()
```

will display **world** on screen.

Let's write a function to create and display a text file using one stream object.

```
def fileHandling():
   file = open("story.txt","w+")
   while True:
      line = raw_input("enter sentence :")
      file.write(line)
      choice = raw_input("want to enter more data in file  Y / N")
      if choice.upper() == 'N' : break
   file.seek(0)
   lines = file.readlines()
   file.close()
   for l in lines:
      print l
```

in this function after opening the file, while loop allow us to store as many strings as we want in the file. once that is done, using seek() method file object is taken back to first alphabet in the file. From where we read the complete data in list object.

We know that the methods provided in python for writing / reading a file works with string parameters. So when we want to work on binary file, conversion of data at the time of reading, as well as writing is required. Pickle module can be used to store any kind of object in file as it allows us to store python objects with their structure. So for storing data in binary format, we will use pickle module.

First we need to import the module. It provides two main methods for the purpose, dump and load. For creation of binary file we will

use pickle.dump() to write the object in file, which is opened in binary access mode.

Syntax of dump() method is:

dump(object, fileobject)

Example:

```
def fileOperation1():
      import pickle
      l = [1,2,3,4,5,6]
      file = open('list.dat', 'wb')     # b in access mode is for binary file
      pickle.dump(l,file)           # writing content to binary file
      file.close()
```

Example:

Example of writing a dictionary in binary file:

```
MD = {'a': 1, 'b': 2, 'c': 3}
file = open('myfile.dat', 'wb')
pickle.dump(MD,file)
file.close()
```

Once data is stored using dump(), it can then be used for reading. For reading data from file we will: use pickle.load() to read the object from pickle file.
Syntax of load() is :

object = load(fileobject)

Note : we need to call load for each time dump was called.

```
# read python dict back from the file
ifile = open('myfile.dat', 'rb')
MD1 = pickle.load(ifile)          # reading data from binary file
ifile.close()
print MD1
```

Results into following on screen:

```
{'a': 1, 'c': 3, 'b': 2}
```

To distinguish a data file from pickle file, we may use a different extension of file. **.pk / .pickle are** commonly used extension for same.

Example of storing multiple integer values in a binary file and then read and display it on screen:

```
def binfile():
    import pickle                                    # line 1
    file = open('data.dat','wb')                     # line 2
    while True:
        x = int(raw_input())                         # line 3
        pickle.dump(x,file)                          # line 4
        ans = raw_input('want to enter more data Y / N')
        if ans.upper()== 'N' : break
    file.close()                                     # line 5
    file = open('data.dat','rb')                     # line 6
```

```
try :                                   # line 7
   while True :                         # line 8
      y = pickle.load(file)             # line 9
      print y                           # line 10
except EOFError :                       # line 11
   pass
file.close()                            # line 12
```

Let's walk through the code

Line 1 is importing pickle module, required to work on binary file. Line 2 is opening a binary file(data.dat) for writing in it. Using a loop we read integer value and then put it in file using line no. 3 & 4. In line number 5 we are closing the file - data.dat, this will de allocate all the resources being used with file. In line number 6 we are again associating data.dat to file stream for reading from it. Line number 7 & 11 is used for detection of end of file condition. This helps us in reading the content of entire file. try & except allow us to handle errors in the program. You will learn about them in details in next chapter. For now we know that, reading at end of file will result into error. Which is used here to terminate the loop used for reading the data from the file. Line no. 8 allow us to make an infinite loop, as the same will be handled using end of file condition. In line no. 9 we are getting data from the binary file and storing same in y, which is printed on screen in next line. Remember one load will get data for one dump. Last statement will again close the file.

Files are always stored in current folder / directory by default. The os (Operating System) module of python provides various methods to work with file and folder / directories. For using these functions, we have to import os module in our program. Some of the useful methods, which can be used with files in os module are as follows:

1. getcwd() to know the name of current working directory
2. path.abspath(filename) will give us the complete path name of the data file.
3. path.isfile(filename) will check, whether the file exists or not.
4. remove(filename) will delete the file. Here filename has to be the complete path of file.
5. rename(filename1,filename2) will change the name of filename1 with filename2.

Once the file is opened, then using file object, we can derive various information about file. This is done using file attributes defined in os module. Some of the attributes defined in it are

1. file.closed returns True if file is closed
2. file.mode returns the mode, with which file was opened.
3. file.name returns name of the file associated with file object while opening the file.

We use file object(s) to work with data file, similarly input/output from standard I/O devices is also performed using standard I/O stream object. Since we use high level functions for performing input/output through keyboard and monitor such as - raw_input(), input() and print statement we were not required to explicitly use I/O stream object. But let's learn a bit about these streams also.

The standard streams available in python are

- Standard input stream
- Standard output stream and
- Standard error stream

These standard streams are nothing but file objects, which get automatically connected to your program's standard device(s), when you start python. In order to work with standard I/O stream, we need to import sys module. Methods which are available for I/O operations in it are

- read() for reading a byte at a time from keyboard
- write() for writing data on terminal i.e. monitor

Their usage is

```
import sys
print ' Enter your name :'
name = ''
while True:
    c = sys.stdin.read()
    if c == '\n' :
      break
    name = name + c
    sys.stdout.write( 'your name is ' + name)
```

same can be done using high level methods also

```
name = raw_input('Enter your name :')
print 'your name is ',name
```

As we are through with all basic operations of file handling, we can now learn the various processes which can be performed on file(s) using these operations.

Creating a file

Option 1 : An empty file can be created by using open() statement. The content in the file can be stored later on using append mode.

Option 2 : create a file and simultaneously store / write the content also.

Algorithm

1. Open a file for writing into it
2. Get data to be stored in the file (can be a string at a time or complete data)
3. Write it into the file (if we are working on a string at a time, then multiple writes will be required.)
4. Close the file

Code :

```
def fileCreation():
    ofile = open("story.txt","w+")
    choice = True
    while True:
        line = raw_input("enter sentence :")
        ofile.write(line)
        choice = raw_input("want to enter more data in file  Y / N")
        if choice == 'N' : break
    ofile.close()
```

At the run time following data was provided

this is my first file program

writing 2nd line to file

this is last line

Traversal for display

Algorithm

1. Open file for reading purpose
2. Read data from the file (file is sequentially accessed by any of the read methods)
3. Display read data on screen
4. Continue step 2 & 3 for entire file
5. Close the file.

Program Code:

```
def fileDisplaying1():
  for l in open("story.txt","r").readlines():
      print l
file.close()
```

The above code will display

this is my first file programwriting 2nd line to filethis is last line.

Remember we didn't add new line character with string, while writing them in file.

```
def fileDisplaying2():
    file = open("story.txt","r")
    l = file.readline()
    while l:
        print l
        l = file.readline()
file.close()
```

The above code will also display the same content, i.e.

this is my first file programwriting 2nd line to filethis is last line.

```
def fileDisplaying3():
    file = open("story.txt","r")
    l = "123"
    while l != '':
        l = file.read(10)
        print l
file.close()
```

The above code will give the following output

this is my
first fil
e programw
riting 2nd
line to f
ilethis is
last line

Till now we were creating file object to either read from the file or to write in the file. Let's create an object capable of both reading and writing :

```
def fileHandling():
    file = open("story.txt","w+")          # both reading & writing can be done
```

```
choice = True
while True:
  line = raw_input("enter sentence :")
  file.write(line)                    #  creation of file
  choice = raw_input("want to enter more data in file  Y / N")
  if choice == 'N' : break
file.seek(0)                    # transferring file object to beginning of the file
lines = file.readlines()
file.close()
for l in lines:
      print l
```

The following code performs same operation a binary file

```
def binfile():
    import pickle
    file = open('data.dat','wb')
    while True:
      x = int(raw_input())
      pickle.dump(x,file)
      ans = raw_input('want to enter more data Y / N')
      if ans.upper()== 'N' : break
    file.close()
    file = open('data.dat','rb')
    try :
      while True :
          y = pickle.load(file)
          print y
    except EOFError :
      pass
    file.close()
```

Creating a copy of file

Algorithm

1. Open the source file for reading from it.
2. Open a new file for writing purpose
3. Read data from the first file (can be string at a time or complete data of file.)
4. Write the data read in the new file.
5. Close both the files.

Program Code for creating a copy of existing file:

```
def fileCopy():
  ifile = open("story.txt","r")
  ofile = open("newstory.txt","w")
  l = file.readline()
while l:
  ofile.write(l)
  l = file.readline()
ifile.close()
ofile.close()
```

Similarly a binary file can also be copied. We don't need to use dump() & load()methods for this. As we just need to pass byte strings from one file to another.

Deleting content from the file

It can be handled in two ways

Option 1 (for small file, which can fit into memory i.e. a few Mb's)

Algorithm

1. Get the data value to be deleted. (Nature of data will be dependent on type of file)
2. Open the file for reading from it.
3. Read the complete file into a list
4. Delete the data from the list
5. Close the file
6. Open same file for writing into it
7. Write the modified list into file.
8. Close the file.

Program code for deleting second word from story.txt is:

```
def filedel():
    with open('story.txt','r') as file :
      l = file.readlines()
    file.close()
    print l
    del l[1]
    print l
    file.open('story.txt','w')
    file.writelines(l)
    file.close()
```

Similarly we can delete any content, using position of the data. If position is unknown then serach the desired data and delete the same from the list.

Option 2 (for large files, which will not fit into memory of computer. For this we will need two files)

Algorithm

1. Get the data value to be deleted
2. Open the file for reading purpose
3. Open another (temporary) file for writing into it
4. Read a string (data value) from the file
5. Write it into temporary file, if it was not to be deleted.
6. The process will be repeated for entire file (in case all the occurrence of data are to be deleted)
7. Close both the files.
8. Delete the original file
9. Rename the temporary file to original file name.

Program code for deletion of the line(s) having word (passed as argument) is :

```
import os
def fileDEL(word):
    file = open("test.txt","r")
    newfile = open("new.txt","w")
    while True:
      line = file.readline()
      if not line :
          break
```

```
    else :
    if word in line :
        pass
    else :
        print line
        newfile.write(line)
file.close()
newfile.close()
os.remove("test.txt")
os.rename("new.txt","test.txt")
```

Inserting data in a file

It can happen in two ways. Insertion at the end of file or insertion in the middle of the file.

Option 1 Insertion at the end. This is also known as appending a file.

Algorithm

1. open the file in append access mode.
2. Get the data value for to be inserted in file, from user.
3. Write the data in the file
4. Repeat set 2 & 3 if there is more data to be inserted (appended)
5. Close the file.

Program code for appending data in binary file

```
def binAppend():
    import pickle
    file = open('data.dat','ab')
    while True:
        y = int(raw_input())
        pickle.dump(y,file)
        ans = raw_input('want to enter more data Y / N')
        if ans.upper()== 'N' : break
file.close()
```

Option 2 Inserting in the middle of file

There is no way to insert data in the middle of file. This is a limitation because of OS. To handle such insertions re-writing of file is done.

Algorithm

1. Get the data value to to be inserted with its position.
2. Open the original file in reading mode.
3. Open another (temporary) file for writing in it.
4. Start reading original file sequentially, simultaneously writing it in the temporary file.
5. This is to be repeated till you reach the position of insertion of new data value.
6. Write the new value in temporary file.
7. Repeat step 4 for remaining data of original file.
8. Delete original file
9. Change the name of temporary file to original file.

Code for inserting data in the file with the help of another file

It will be similar to the code used for deletion of content using another file. Here instead of not writing the content add it in the file.

An alternative to this is, first read the complete data from file into a list. Modify the list and rewrite the modified list in the file.

Updating a file

File updation can be handled in many ways. Some of which are

Option 1 - Truncate write

Algorithm

1. Open the file for reading from it
2. Read the content of file in an object (variable) usually list
3. Close the file
4. Get the details of data to be modified
5. Update the content in the list
6. Re open the file for writing purpose (we know that now opening the file for writing will truncate the existing file)
7. Write the list back to the file.

Program Code for this will be similar to following:

```
with open("sample.txt","r") as file:
  content = file.read()
file.close()
```

```
content.process()
wit h open ("sample.txt","w") as file :
  file.writelines(content)
file.close()
```

Option 2 - Write replace

Algorithm

- Open the original file for reading
- Open temporary file for writing
- Read a line / record from the file
- If this was not to be modified copy it in temporary file otherwise copy the modified line / record in the temporary file.
- Repeat previous two steps for complete file.

This way of processing a file using python has been handled earlier.

Option 3 - In place updation

Algorithm

1. Open file for reading and writing purpose
2. Get the details of data value to be modified
3. Using linear search, reach to the record / data to be modified
4. Seek to the start of the record
5. Re write the data
6. Close the file.

Updating a text file in this manner is not safe, as any change you make to the file may overwrite the content you have not read yet. This should only be used when the text to be replaced is of same size. In place updation of a binary file is not possible. As this requires placing of fileobject to the beginning of the record, calculating size of data in dump file is not possible. So updating a data file using third option is not recommended in python.

Let's create a data file storing students record such as Admission number, Name, Class and Total marks. Data to be stored contains numeric data, hence will be stored in binary file. We will use dictionary data type to organize this information.

```
from pickle import load, dump
import os
import sys
```

```
def bfileCreate(fname):
    l = []
    sd = {1000:['anuj',12,450]}
    with open(fname,'wb') as ofile :
      while True :
          dump(sd,ofile)
          ans = raw_input("want to enter more data Y / N")
          if ans.upper() == 'N' : break
          x = int(raw_input("enter admission number of student"))
          l = input("enter name class and marks of student enclosed in []")
          sd[x] = l
    ofile.close()

    def bfileDisplay(fname):
      if not os.path.isfile(fname) :
          print "file does not exist"
      else:
          ifile = open(fname,'rb')
          try :
              while True:
                  sd = {}
                  sd = load(ifile)
                  print sd
          except EOFError:
                  pass
    ifile.close()
```

Use the code to store records of your class mates. Once the file is created, use bfileDisplay() to see the result. Do you find some problem in the content displayedσFind and resolve the problemσ

LET'S REVISE

Files are used to store huge collection of data permanently. The stored data can later be used by performing various file operations like opening, reading, writing etc.

Access modes specify the type of operations to be performed with the opened file.

read(), readline() and readlines() methods are available for reading data from the file.

write() and writelines() are used for writing data in the file.

There are two functions which allow us to access a file in a non-sequential or random mode. They are seek() and tell()

Serialization is the process of converting a data structure / object that can be stored in non string format and can be resurrected later. Serialization can also be called as deflating the data and resurrecting as inflating it.

Pickle module is used in serialization of data. This allow us to store data in binary form in the file. Dump and load functions are used to write data and read data from file.

os module provide us various functions and attributes to work on files.

EXERCISE

1.
```
file = open('textfile.txt','w')
word = ''
while word.upper() != 'END':
    word = raw_input('Enter a word use END to quit')
    file.write(word + '\n')
file.close()
```

The above program is to create a file storing a list of words. What is the name of file on hard disk containing list of wordsσ

2. Human readable form of file is called ----------.
3. Write a try …. except statement that attempts to open a file for reading and catches the exception thrown when the file does not exist.
4. Compare & contrast read(), readline() and readlines().
5. How is write() different from writelines()σ
6. In how many ways can end of file be detectedσ
7. How many file modes can be used with the open() function to open a fileσState the function of each mode.
8. What does the seekg() and seekp() functions doσ
9. Explain the use of output functions write() and writeline() with an example each.
10. Write a function that writes a structure to disk and then reads it back and display on screen.
11. Using the file in mentioned in question no. 1, write a function to read the file and display numbered list of words.
12. In the code (given in question no. 1) the word END used to indicate end of word list is also stored in the file. Modify the code so that end is not stored in the file.
13. Write a function that takes three argument 1st input file, 2nd output file and 3rd transformation function. First argument is the file opened for reading, second argument is the file opened for writing and third argument is a function, which takes single string and performs a transformation of your choice and returns the transformed string. The function should reed each line in the input file, pass the line through transformation function and then write transformed line to output file. A transformation can be - capitalize first alphabet of every word.

14. Write a function to create a text file containing following data

    Neither apple nor pine are in pineapple. Boxing rings are square.

    Writers write, but fingers don't fing. Overlook and oversee are opposites. A house can burn up as it burns down. An alarm goes off by going on.

    i) Read back the entire file content using read() or readlines () and display on screen.
    ii) Append more text of your choice in the file and display the content of file with line numbers prefixed to line.
    iii) Display last line of file.
    iv) Display first line from 10th character onwards
    v) Read and display a line from the file. Ask user to provide the line number to be read.

15. Create a dictionary having decimal equivalent of roman numerals. Store it in a binary file. Write a function to convert roman number to decimal equivalent using the binary file data.

16. Write a program to delete the content provided by user from the binary file. The file is very large and can't fit in computers memory.

17. Write a function to insert a sentence in a text file, assuming that text file is very big and can't fit in computer's memory.

18. Write a program to read a file 'Story.txt' and create another file, storing an index of Story.txt telling which line of the file each word appears in. If word appears more than once, then index should show all the line numbers containing the word.

Hint : Dictionary with key as word(s) can be used to solve this.

19. Write a function called replace file(), that takes pattern string, replacement string and two file names as argument. The function should read the first file and write the content into second file (creating it, if necessary). If the pattern string appear anywhere in first file, it should be replaced by replacement string in second file.

20. Write a program to accept a filename from the user and display all the lines from the file which contain python comment character '#'.

21. Reading a file line by line from beginning is a common task, what if you want to read a file backward. This happens when you need to read log files. Write a program to read and display content of file from end to beginning.

22. Create a class item to store information of different items, existing in a shop. At least following is to be stored w.r.t. each item code, name, price, qty. Write a program to accept the data from user and store it permanently in the file. Also provide user with facility of searching and updating the data in file based on code of item.

Chapter-4: Exception Handling & Generator Functions

Learning Objective

At the end of this chapter the students will be able to understand:

- How does Python deal with errorsσ
- What an exception is and its terminology.
- Why we use themσ
- What are the different types of exceptionσ
- Generating exceptions and handling multiple exceptions
- Distinguish between iterators and generators
- Create generator functions

When we plan our code/program, we always work for situations that are normally expected, and our program works very well in those situations. But, we all understand that programs have to deal with errors. Here errors are not syntax errors instead they are the unexpected condition(s) that are not part of normal operations planned during coding. Partial list of such kinds of errors are:

- Out of Memory
- Invalid filename
- Attempting to write into read only file
- Getting an incorrect input from user
- Division by zero
- Accessing an out of bound list element
- Trying to read beyond end of file
- Sending illegal arguments to a method

If any of such situations is encountered, a good program will either have the code check for them and perform some suitable action to remedy them, or at least stop processing in a well defined way after giving appropriate message(s). So what we are saying is if an error happened, there must be code written in program, to recover from the error. In case if it is not possible to handle the error then it must be reported in user friendly way to the user.

Errors are exceptional, unusual and unexpected situations and they are never part of the normal flow of a program. We need a process to identify and handle them to write a good program. Exceptions handling is the process of responding in such situations. Most of the modern programming languages provide support with handling exceptions. They offer a dedicated exception handling mechanism, which simplifies the way in which an exception situation is reported and handled.

Before moving ahead with exception handling, let's understand, some of the terms associated with it - when an exception occurs in the program, we say that exception was raised or thrown. Next, we deal with

it and say it is handled or caught. And the code written to handle it is known as exception handler.

For handling exceptional situations python provides

1. **raise statement** to raise exception in program
2. **try..... except** statement for catching and handling the errors.

Raise statement allows the programmer to force a specified exception to occur. Once an exception is raised, it's up to caller function to either handle it using try/except statement or let it propagate further.

Syntax of raise is:

raise [exception name [, argument]]

A raise statement that does not include an exception name will simply re raise the current exception.

Here exception name is the type of error, which can be string, class or object and argument is the value. As argument is optional its default value **None** is used in case the argument is not provided.

Before moving further, let's talk of some of the python's predefined error types, which we can use with raise statement. This does not mean that raise can't be used to raise user defined errors.

S. No.	Error type	Description
1.	IOError	is raised when I/O operator fails. Eg. **File not found, disk full**
2.	EOFError	is raised when, one of the file method i.e. read(), readline() or readlines(), try to read beyond the file.
3.	ZeroDivisionError	is raised when, in division operation, denominator is zero
4.	ImportError	is raised when import statement fails to find the module definition or file name
5.	IndexError	is raised when in a sequence - index/subscript is out of range.
6.	NameError	is raised when a local or global name is not found
7.	IndentationError	is raised for incorrect indentation
8.	TypeError	is raised when we try to perform an operation on incorrect type of value.
9.	ValueError	is raised when a built in function/method receives an argument of correct type but with inappropriate value.

Example:

```
>>>l = [1,2,3]
>>>i = 5
>>> if i > len(l):
```

```
        raise IndexError
    else:
        print l[i]
```

As the value assigned to i is 5, the code will raise an error, during execution. The error will be index error as mentioned in raise statement.

Let's see an example of raise, with user defined error type:

```
>>>def menu(choice):
        if choice < 1:
            raise "invalid choice", choice
```

In order to handle the errors raised by the statement, we can use try except statement.

try.....except is used when we think that the code might fail. If this happens then we are able to catch it and handle same in proper fashion. Let's re-consider the menu() function defined above to handle the error also:

```
while True:
    try:
        x = int(raw_input("Enter a number"))
        break
    except ValueError:
        print " This was not a valid number. Enter a valid number"
```

This code will ask user to input a value until a valid integer value is entered.

Now let's see how try and except work?

Once while loop is entered execution of try clause begins, if no error is encountered, i.e. the value entered is integer, except clause will be skipped and execution of try finishes. If an exception occurs i.e. an integer value is not entered then rest of try clause is skipped and except cause is executed.

Following is the syntax of try...except statement

```
try:
    statements which might go wrong
    except error type1:
    statements to be executed, if error type1 happens
[except error type2:
    statements to be executed, if error type 2 happens
    .
    .
    .
```

else:

statements to be executed, if no exception occurs

finally:

statements to be executed]

remember error type can be user defined also.

You can see a single try statement can have multiple except statements. This is required to handle more than one type of errors in the piece of code. In multiple except clauses, a search for except is taken up. Once a match is found it is executed. You may not specify an error type in exception clause. If that is done, it is to catch all exceptions. Such exceptions should be listed as last clause in the try block.

The **else** clause written after all except statements, is executed, if code in try block does not raise any exception.

finally clause is always executed before leaving the try statement irrespective of occurrence of exception. It is also executed if during execution of try and except statement any of the clause is left via break, continue or return statement.

In try statement the block of code written in try is main action statement, except clause defines handlers for exceptions raised during execution of main statements. Else clause, if written, provides a handler to be run if no exception occurs, and finally is used to provide clean up action.

Example using user defined exception, which was created earlier in raise statement

```
>>>try:
        Statement(s)
    except "Invalid choice ":
        exception handling statements
    else:
        rest of the code
```

Example of except clause without error type.

```
    >>>try:
    x = y
    except:
        print " y not defined "
```

Example code having except, else and finally clause

```
def divide(x, y):
...    try:
...        result = x / y
...    except ZeroDivisionError:
...        print "division by zero!"
```

```
... else:
...     print "result is", result
... finally:
...     print "executing finally clause"
```

Let's apply exception handling in data file.

```
try:
    fh = open("testfile", "w")
    fh.write("This is my test file for exception handling!!")
    except IOError:
    print "Error: can\'t find file or read data"
else:
    print "Written content in the file successfully"
```

This will produce the following result, if you are not allowed to write in the file :

Error: can't find file or read data

Another example of error handling in data file:

```
lists = [ ]
infile = open('yourfilename.pickle', 'r')
while 1:
    try:
        lists.append(pickle.load(infile))
    except EOFError:
        break
infile.close()
```

This will allow us to read data from a file containing many lists, a list at a time.

Generator Functions

Iteration is the repetition of a process in computer program. This is typically done using looping constructs. These loops will repeat a process until certain number / case is reached. Recursive function is other way of implementing iteration in the program. Sequence data type, in python is iterable objects.

Here itratable objects have a specific protocol, which allow us to iterate (loop) over different type of objects. Such as - list, strings, dictionaries, files and others. The purpose of the protocol is to allow a user to process every element of container, without knowing the internal structure of container. So irrespective of whether container is list, string, tuple or file we can iterate in a similar fashion on all of them. This you have already done in class XI using for statement, for could be used to iterate a string, list, dictionary and tuple.

Iteration protocol specifies that, an iterator should have at least three operators:

1. Increment

2. De referencing
3. Non equality comparison

We are not going to cover their details in the chapter instead we will talk of generators, which allow us to write user defined iterator, without worrying about the iterator protocol.

Before moving further into generator functions let's walk through the execution of normal function. When a normal python function is called, in a program, the control from calling function is shifted to called function. Execution of called function starts from first line and continues until a return statement or exception or end of function is encountered. In all these situations, the control is returned back to caller function. That means any work done by the called function for providing the result is lost. To use it again we will again call the function and its execution will start from scratch.

Sometimes in programming we need to have functions which are able to save its work, so that instead of starting from scratch the function starts working from the point where it was left last. In other words, a function should be able to yield a series of values instead of just returning a single value. Here returning a value also implies returning of control.

Such functions are Generator functions. These functions can send back a value and later resume processing from the place, where they left off. This allows the function to produce a series of values - over time, rather than computing them all at once and returning a list of values.

Generator functions are not much **different** from normal functions, they also use **def** to define a function. The primary difference between generator and normal function is that generator will yield a value instead of returning a value. It is the **yield** statement which allows the generator function to suspend the processing and send a value, simultaneously retaining the existing state of generator to resume processing over time.

Let's take a simple example illustrating yield

```
def testGenerator():
    yield 1
    yield 2
    yield 3
```

For using the generator function, following will be done

```
>>> g = testGenerator()
>>> g.next()
1
>>>g.next()
2
>>>g.next()
3
```

Let's consider another example:

```
1.  def counter (n):
2.      i = 1
3.      while i< = n:
4.          yield i
5.          i+=1
6.  >>> x= counter (3)
7.  >>> x.next ( )
8.  1
9.  >>> x. next ( )
10. 2
11. >>> x.next ( )
12. 3
```

Let's walk through the code, except for yield statement remaining statements are just like normal function

1. Presence of yield statement (at line no.4) means that the function is not normal function but a generator
2. Calling the function, does not actually execute the function, but creates an instance of the function. As done in line no. 6. Here counter ( ) is actually creating an object x.
3. In line no, 7, the next ( ) method, executes the code written in counter up to first yield statement and then returns the value generated. In our case, it will be 1, which is displayed in line no. 8.
4. Calling next ( ) again in line no. 9, will resume the processing of function counter, from the place it was last left till the yield statement is encountered again. So in our example processing will resume from line no. 5, and will continue to line no. 3 & 4. At 4, there is yield which will again return the generated value back, after pausing the processing.

Since counter is called with argument value 3, so the process will be repeated three times.

Trace the following code, and explain what is happening and whyσ

```
def Count ( ) :
    n = 1
    while True:
        yield n
        n+ = 1
```

Let's take an example of producing squares of **n** numbers using iterator and generator.

Squares using generator

```
def Square (n):
    for i in range (n):
        yield i**2
```

The function can be invoked in following way

```
>>> for k in square (6):
        print k,
```

Let's do same thing using iterator, i.e., we will replace yield with return

```
def Square (n):
    for i in range (n):
        return i**2
```

Calling the function square

```
>>> for k in square (6):
        print k
```

results into Type Error, saying int object is not iterable. It's because here Square( ) will just return an integer object not a series of values.

Remember using generator you can iterate on generated data only once.

Let's use generator to produce Fibonacci series.

```
def Fibonacci (max):
    a, b = 0, 1
    while a <= max:
        yield a
        a, b = b, a + b
>>> for i in Fibonacci (100):
        print i,
0 1 1 2 3 5 8 13 21 34 55 89
```

The generator can also be used to return a list of values

```
>>> L = list (Fibonacci (100))
>>> print L
[0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89]
```

Advantages of using generator

- These functions are better w.r.t. memory utilization and code performance, as they allow function to avoid doing all work at a time.
- They provide a way to manually save the state between iterations. As the variables in function scope are saved and restored automatically.

LET'S REVISE

- An exception is a rarely occurring condition that requires deviation from the program's normal flow.
- We can raise and handle the errors in our program
- For raising errors statement used is raise[exception name [, argument]]
- For handling errors statement used is try …. except…… else finally.
- Iterable is an object capable of returning its member one at a time.
- An iterator is an object that provides sequential access to an underlying sequential data. The underlying sequence of data is not stored in memory, instead computed on demand.
- A generator is user defined iterator.
- A generator is a function that produces a sequence of results instead of simple value using yield statement

EXERCISE

1. What all can be possible output's of the following code

```
def myfunc(x=None):
    result = ""
    if x is None:
    result = "No argument given"
    elif x == 0:
    result = "Zero"
    elif 0 < x <= 3:
    result = "x is between 0 and 3"
else:
    result = "x is more than 3"
return result
```

2. We will try to read input from the user. Press ctrl-d and see what happens

```
>>> s = raw_input('Enter something --> ')
```

3. What will happen when following functions are executedσ

```
def fib():
x,y = 1,1
while True:
yield x
x,y = y, x+y
def odd(seq):
for number in seq:
if number % 2:
yield number
def under Four Million(seq):
for number in seq:
```

```
    if number > 4000000:
        break
    yield number
        print sum(odd(underFourMillion(fib())))
```

4. Find out the situation(s) in which following code may crash

```
while loop == 1:
try:
    a = input('Enter a number to subtract from > ')
    b = input ('Enter the number to subtract > ')
    except NameError:
    print "\nYou cannot subtract a letter"
    continue
except  SyntaxError:
print  "\nPlease enter a number only."
continue
print a - b
try:
loop = input('Press 1 to try again > ')
except (NameError,SyntaxError):
loop = 0
```

5. Describe, what is the following module doing:

```
def test(start = 0):
c = start
while True:
value = yield c
if value != None:
c = value
    else:
        c += 1
```

6. When do we get type errorσ
7. List the situation(s) in which the code may result in IOErrorσ
8. What is the purpose of yield statementσ
9. How is yield different from returnσ
10. Write syntax of raise statement.
11. Write a function called oops that explicitly raises a Index Error exception when called. Then write another function that calls oops inside a try/except statement to catch the error. What happens if you change oops to raise Key Error instead of Index Errorσ
12. range() function in python does not include the stop value. Write a generator function equivalent to range() function that includes the stop value also. The function will take three arguments start, stop and step and will generate the desired list.
13. Write a function to find average of a list of numbers. Your function should be able to handle an empty list and also list containing string.
14. Write a function to generate cube's of numbers over time.
15. Write a program, to accept a date as day, month & year from user and raise appropriate error(s), if legal value(s) is not supplied. Display appropriate message till user inputs correct value(s).
16. Create a class Person to store personal information (of your choice) for a person. Ensure that while accepting the data incorrect entry is properly handled.

Unit-3: Database Management Systems and SQL

Chapter-1: Database Concepts and SQL

Learning Objectives

At the end of this chapter the students will be able to understand:

- What is DBMSσ
- What is relational database modelσ
- Relation
- Tuples
- SQL
- DDL
- DML
- Relational Algebra
- Selection
- Projection
- Union
- Cartesian Product

Introduction

Database is a collection of related information that is organized in such a way that supports for easy access, modify and maintain data. The contents of a database are obtained by combining data from all the different sources in an organization. Generally, the database is managed by some special software packages known as Database Management Systems (DBMSs). DBMSs are specially designed applications to create connection between user and program, and to store data in an organized manner. The purpose of DBMSs software is to allow the user to create, modify and administration of database. Examples of database management systems are: Ms-Access, MySQL, PostgreSQL, SQLite, Microsoft SQL Server, Oracle, SAP, dBASE, FoxPro, etc.

Relational data model

The relational data model is a database model based on first-order predicate logic(First Order Predicate Logic is one where the quantification is over simple variables), formulated and proposed by Edgar F. Codd. in 1969. The first-order predicate logic is a symbolised reasoning, in which statement is broken down into a subject and a predicate. The predicate modifies the properties of the subject, while in the first-order logic, a predicate can only refer to a single subject. In the relational data model, database is represented as collection of related tables. Each table is termed as relation and has its unique name in the relational data model. Tables are formed by using rows and columns. A row (horizontal subset) of a table represents a tuple or record, while column (vertical subset) of a table represents an attribute.

Relation

In database, a relation means a 'table', in which data are organized in the form of rows and columns. Therefore in database, relations are equivalent to tables.

For example

Relation: Student

Ad No	Name	Class	Section	Average
101	Anu	12	A	85
105	Balu	12	D	65
203	Leena	11	B	95
205	Madhu	10	B	75
305	Surpreeth	9	C	70
483	Usha	6	A	60

Domain

A domain is the original sets of atomic values used to model data. In data base management and database, a domain refers to all the possible unique values of a particular column.

For example:

i) The domain of gender column has a set of two possible values i.e, Male or Female.

ii) The domain of marital status has a set of four possible values i.e, Married, Unmarried, Widows and Divorced.

Therefore, a domain is a set of acceptable values of a particular column, which is based on various properties and data types. We will discuss data types later in this chapter.

Tuple

Horizontal subset/information in a table is called tuple. The tuple is also known as a 'record', which gives particular information of the relation (table).

For example:

i) In customer table, one row gives information about one customer only.

ii) In student table, one row gives information about one student only.

Key

Keys are an important part of a relational database and a vital part of the structure of a table. They help enforce integrity and help identify the relationship between tables. There are three main types of keys -

candidate keys, primary keys, foreign keys and alternate keys.

Primary Key: A column or set of columns that uniquely identifies a row within a table is called primary key.

Candidate Key: Candidate keys are set of fields (columns with unique values) in the relation that are eligible to act as a primary key.

Alternate Key: Out of the candidate keys, after selecting a key as primary key, the remaining keys are called alternate key.

Foreign Key: A foreign key is a field (or collection of fields) in one table that uniquely identifies a row of another table. In other words, a foreign key is a column or a combination of columns that is used to establish a link between two tables.

Degree

The degree is the number of attributes (columns) in a table.

Cardinality

Cardinality is number of rows (tuples) in a table.

Example:

Relation: Student

Ad No	Name	Class	Section	Average
101	Anu	12	A	85
105	Balu	12	D	65
203	Leena	11	B	95
205	Madhu	10	B	75
305	Surpreeth	9	C	70
483	Usha	6	A	60

Fields (Attributes/Columns):- AdNo, Name, Class, Section and Average.

Tuples (Rows/Records):

101	Anu	12	A	85

Domain: Possible values of section are ('A','B','C','D')

Degree: 5 (Number of columns).

Cardinality: 6 (Number of rows).

Candidate Key: In the above table, AdNo and Name has unique values. Therefore, AdNo and Name are candidate keys.

Primary Key: Out of the AdNo and Name, AdNo is the primary key.

Alternate Key: In the candidate key, AdNo is the primary key and the Name is the Alternate key.

Structured Query Language (SQL)

Structured Query Language (SQL) is a standard language used for accessing databases. This is a special purpose programming language used to create a table, manage data and mainipulate data.

SQL provides statements for a variety of tasks, including:

i) Querying data
ii) Inserting, updating, and deleting rows in a table
iii) Creating, replacing, altering, and dropping objects (tables)
iv) Controlling access to the database and its objects (tables)
v) Guaranteeing database consistency and integrity

SQL unifies all of the preceeding tasks in one consistent language.

Advantages of using SQL:

1) **SQL is portable:** SQL is running in all servers, mainframes, PCs, laptops, and even mobile phones.
2) **High speed:** SQL queries can be used to retrieve large amounts of records from a database quickly and efficiently.
3) **Easy to learn and understand:** SQL generally consists of English statements and as such, it is very easy to learn and understand. Besides, it does not require much coding unlike in programming languages.
4) **SQL is used with any DBMS system with any vendor:** SQL is used by all the vendors who develop DBMS. It is also used to create databases, manage security for a database, etc. It can also be used for updating, retrieving and sharing data with users.
5) **SQL is used for relational databases:** SQL is widely used for relational databases.
6) **SQL acts as both programming language and interactive language:** SQL can do both the jobs of being a programming language as well as an interactive language at the same time.
7) **Client/Server language:** SQL is used for linking front end computers and back end databases. It provides client server architecture (Email, and the World Wide Web - all apply the client-server architecture).
8) **Supports object based programming:** SQL supports the latest object based programming and is highly flexible.

Types of SQL Statements

The SQL statements are categorized into different categories based upon the purpose. They are;

i) Data Definition Language (DDL) statement

ii) Data Manipulation Language (DML) statement

iii) Transaction Control Statement

iv) Session Control Statement

v) System Control Statement

vi) Embedded SQL Statement

Out of these six, we will be studying only the first two types in this course.

Data Definition Language (DDL) Statements

Data Definition Language (DDL) or Data Description Language (DDL) is a standard for commands that defines the different structures in a database. DDL statements are used to create structure of a table, modify the existing structure of the table and remove the existing table. Some of the DDL statements are CREATE TABLE, ALTER TABLE and DROP TABLE.

Data Manipulation Language (DML) Statements

Data Manipulation Language (DML) statements are used to access and manipulate data in existing tables. The manipulation includes inserting data into tables, deleting data from the tables, retrieving data and modifying the existing data. The common DML statements are SELECT, UPDATE, DELETE and INSERT.

Data Types

Each value manipulated by SQL Database has a data type. The data type of a value associates a fixed set of properties with the value. In SQL there are three main data types: Character, Number, and Date types.

Character

Character data types stores character (alphanumeric) data, which are words and free-form text. They are less restrictive than other data types and consequently have fewer properties. For example, character columns can store all alphanumeric values, but number columns can store only numeric values. Character data types are;

i) CHAR

ii) VARCHAR

iii) VARCHAR2

CHAR: CHAR should be used for storing fix length character strings. String values will be space/blank padded (The adding of meaningless data [usually blanks] to a unit of data to bring it up to some fixed size) before they are stored on the disk. If this type is used to store variable length strings, it will waste a lot of disk space (always allocate fixed memory) . If we declare data type as CHAR, then it will occupy space for

NULL values.

Format: CHAR(n)

Fixed-length character string having maximum length n.

VARCHAR: Varchar is a variable character string. If we declare data type as VARCHAR, then it will occupy space for NULL values. It can have maximum of 2000 characters.

Format: VARCHAR (n)

Variable-length character string having maximum length n.

VARCHAR2: VARCHAR2 is used to store variable length character strings. The string value's length will be stored on disk with the value itself. VARCHAR2 can store up to 4000 bytes of characters. Thus, the difference between VARCHAR and VARCHAR2 is that VARCHAR is ANSI standard but takes up space for variables, whereas the VARCHAR2 is used only in Oracle but makes more efficient use of space.

Format: VARCHAR2 (n)

Example:

CHAR(10) has fixed length, right padded with spaces.

VARCHAR(10) has fixed length, right padded with NULL

VARCHAR2(10) has variable length.

Name char (10): Suppose if we store Name is as "Ravi", then first four places of the ten characters are filled with Ravi and the remaining 6 spaces are also allocated to Name. Thus, the size of name is always ten.

Name varchar (10): Suppose if we store Name is as "Ravi", then first four places of the ten characters are filled with Ravi and the remaining 6 spaces are filled with NULL.

Name varchar2 (10): Suppose if we store Name is as "Ravi", then only first four places are filled with Ravi.

The following table gives possible string data types used in different DBMS

Data type	Access	SQL Server	Oracle	My SQL	Postgre SQL
string (fixed)	N/A	Char	Char	Char	Char
string (variable)	Text (<256) Memo (65k+)	Varchar	Varchar Varchar2	Varchar	Varchar

Numeric data type: Numeric data types are mainly used to store number with or without fraction part. The numeric data types are:

1. NUMBER
2. DECIMAL
3. NUMERIC

4. INT
5. FLOAT

NUMBER: The Number data type stores fixed and floating-point numbers. The Number data type is used to store integers (negative, positive, floating) of up to 38 digits of precision.

The following numbers can be stored in a Number data type column:

- Positive numbers in the range 1 x 10-130 to 9.99...9 x 10125 with up to 38 significant digits.
- Negative numbers from -1 x 10-130 to 9.99...99 x 10125 with up to 38 significant digits.
- Zero

Format:

NUMBER (p, s)

Where;

- 'p' is the precision or the total number of significant decimal digits, where the most significant digit is the left-most nonzero digit and the least significant digit is the right-most known digit.
- 's' is the scale or the number of digits from the decimal point to the least significant digit.

DECIMAL and NUMERIC: Decimal and numeric data types have fixed precision and scale.

Format:

DECIMAL[(p[, s])] and NUMERIC[(p[, s])]

Square brackets ([ ]) are option.

where;

- 'p' is the precision or the total number of significant decimal digits, where the most significant digit is the left-most nonzero digit and the least significant digit is the right-most known digit.
- 's' is the scale or the number of digits from the decimal point to the least significant digit.

INT/INTEGER: The int data type is the integer data type in SQL. This used to store integer number (without any fraction part).

FLOAT: This data type is used to store number with fraction part(real numbers).

The following table gives possible numeric data types used in difference DBMS

Data type	Access	SQL Server	Oracle	My SQL	Postgre SQL
Integer	Number	Int	Number (integer)	Int Integer Numeric	Int Integer
Float	Number (single)	Float Real	Number	Float Decimal Numeric	Numeric

DATE

Date is used to store valid date values, which is ranging from January 1, 4712 BC to December 31, 9999 AD. The date formats are: YYYY-MM-DD or DD/MON/YY or YYYY/MM/DD or MM/DD/YY or DD-MON-YYYY.

Format: DATE

Relational Algebra

An algebra is a combination of set of operands and a set of operators. We can form algebraic expressions by applying operators to operands. Relational algebra consists of a set of operations that take one or two relations as input and produces a new relation as output. Operators map values are taken from the domain and put it into other domain values. If domain is produced from more than one relation, then we get relational algebra.

Operation in relational algebra:

1. Selection
2. Projection
3. Union
4. Cartesian Product

Selection

Selection in relational algebra returns those tuples(records) in a relation that fulfil a condition(Produce table containing subset of rows).

Syntax:

σ condition (relation)

Example: The table S (for STUDENT).

Relation: Student

AdNo	Name	Class	Section	Average
101	Anu	12	A	85
105	Balu	12	D	65
203	Leena	11	B	95
205	Madhu	10	B	75
305	Surpreeth	9	C	70
483	Usha	6	A	60

σ class=12 (S)

Output:

AdNo	Name	Class	Section	Average
101	Anu	12	A	85
105	Balu	12	D	65

Projection

Projection in relational algebra returns those columns in a relation that given in the attribute list (Produce table containing subset of columns).

Syntax:

π attribute list(relation)

Example:

π Adno,Name (S)

Output:

AdNo	Name
101	Anu
105	Balu
203	Leena
205	Madhu
305	Surpreeth
483	Usha

Union

The union operator is used to combine two or more tables. Each table within the UNION should have the same number of columns, similar data types and also the columns must be in the same order. In the union operation, duplicate records will be automatically removed from the resultant table.

For example:

Table: Student 1

Roll no	Name
11	Kumar
22	Mohan
33	Rohit

Table: Student2

Roll no	Name
22	Mohan
11	Rahul
77	Kavita

Query is

σ (Students 1)

Union

σ (Students 2)

Or

π rollno, Name (Students 1)

Union

π rollno, Name (Students 2)

Resultant table is:

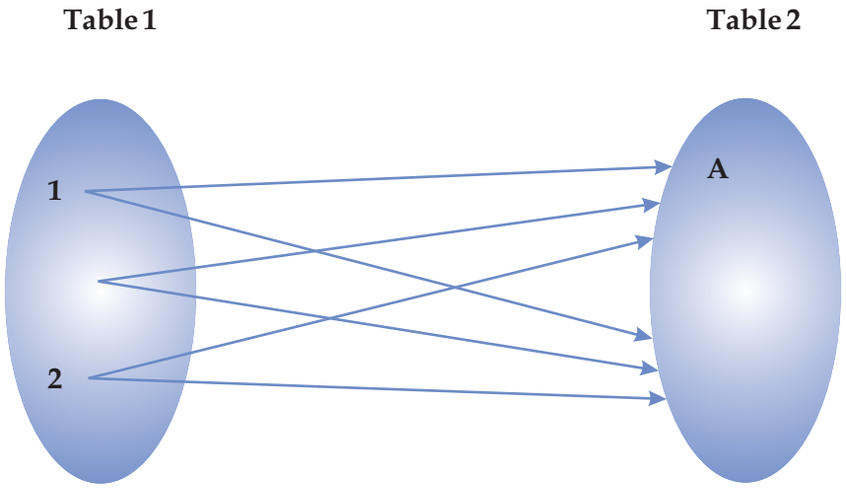
Roll no	Name
11	Kumar
22	Mohan
33	Rohit
11	Rahul
77	Kavita

In the above resultant table, student1 is copied as it is, but in student2, roll no 22 Mohan's information is same as student1. So, that is not copied in the resultant table again. Roll no 11 is same as student1, but name is different. So, that is copied in the resultant table. Roll no 77 is not in student1 table, so that is also copied in the resultant table.

Cartesian product

SQL joins are used to relate information in different tables. It combines fields from two or more tables by comparing values of common columns (join condition). A join condition is a part of the SQL query that retrieves rows from two or more tables. If join condition is ommitted or if it is invalid, then join operation will result in a Cartesian product. Cartesian product is a binary operation and is denoted by (x) Cartesian product returns a number of rows equal to number of rows in the first table multiply by number of rows in

the second table. At the same time, number of columns equal to number of columns in the first table added by number of columns in the second table.



For example:

Table 1: Product

Product_no	Product_name	Price
111	Computer	50000
222	Printer	10000
333	Scanner	12000
444	Modem	500

Table 2: Customer

Cust_no	Cust_name	City	Product_no
101	Kavitha	Delhi	333
201	Mohan	Mumbai	222
301	Rohan	Bangalore	111
401	Sahil	Mumbai	333
501	Rohita	Delhi	444

Query is

σ (Product, customer)

Product_no	Product_name	Price	Cust_no	Cust_name	City	Product_no
111	Computer	50000	101	Kavitha	Delhi	333
111	Computer	50000	201	Mohan	Mumbai	222
111	Computer	50000	301	Rohan	Bangalore	111
111	Computer	50000	401	Sahil	Mumbai	333
111	Computer	50000	501	Rohita	Delhi	444
222	Printer	10000	101	Kavitha	Delhi	333
222	Printer	10000	201	Mohan	Mumbai	222
222	Printer	10000	301	Rohan	Bangalore	111
222	Printer	10000	401	Sahil	Mumbai	333
222	Printer	10000	501	Rohita	Delhi	444
333	Scanner	12000	101	Kavitha	Delhi	333
333	Scanner	12000	201	Mohan	Mumbai	222
333	Scanner	12000	301	Rohan	Bangalore	111
333	Scanner	12000	401	Sahil	Mumbai	333
333	Scanner	12000	501	Rohita	Delhi	444
444	Modem	500	101	Kavitha	Delhi	333
444	Modem	500	201	Mohan	Mumbai	222
444	Modem	500	301	Rohan	Bangalore	111
444	Modem	500	401	Sahil	Mumbai	333
444	Modem	5003	501	Rohita	Delhi	444

Table 1:

Number of rows (cardinality) = 4

Number of columns (degree) = 3

Table 2:

Number of rows (cardinality) = 5

Number of columns (degree) = 4

Cartesian product:

Number of rows (cardinality) = 4*5 = 20

Number of columns (degree) = 3+4 = 7

LET'S REVISE

- **Relation:** In database, a relation means a 'table' (form of rows and columns).
- **Domain:** A domain is the original sets of atomic values used to model data.
- **Tuple:** A row in a table.
- **Attribute:** A column in a table.
- **Primary Key:** A column or set of columns that uniquely identifies a row within a table is called primary key.
- **Candidate Key:** Candidate keys are set of fields (columns with unique values) in the relation that are eligible to act as a primary key.
- **Alternate Key:** Out of the candidate keys, after selecting a key as primary key, the remaining keys are called alternate key.
- **DDL:** Data Definition Language (DDL) or Data Description Language (DDL).
- **DML:** Data Manipulation Language (DML).
- **String datatypes:** CHAR, VARCHAR, VARCHAR2
- **Numeric datatype:** NUMBER, NUMERIC, INT, FLOAT, DECIMAL
- **Date:** DATE
- **Selection:** Selection in relational algebra returns those tuples (records) in a relation that fulfil a condition(Produce table containing subset of rows).
- **Projection:** Projection in relational algebra returns those columns in a relation that given in the attribute list (Produce table containing subset of columns).
- **Union:** The union operator is used to combine two or more tables. In the union operation, duplicate records will be automatically removed from the resultant table.
- **Cartesian product:** SQL joins are used to relate information in different tables. Cartesian product returns a number of rows equal to number of rows in the first table multiply by number of rows in the second table. At the same time, number of columns equal to number of columns in the first table added by number of columns in the second table.

EXERCISE

1. Expand the following:
   (i) SQL (ii) DBMS
2. What is relational database modelσ
3. What is relationσ
4. Define the following:
   a) Cardinality
   b) Degree
   c) Tuple
   d) Field
5. Define the following keys.
   a) Primary key
   b) Candidate key
   c) Alternate key
6. What is DDLσ
7. What all character types are possible in sqlσ
8. What all numeric data types are possible in sqlσ
9. Write all character data types in SQL.
10. Write all number data types. In SQL
11. Define the following:
    a) Projection
    b) Selection
    c) Union
    d) Cartesian product
12. Differentiate between char and varchar.
13. Write the similarity between decimal and numeric data types.
14. What is the importance of primary key in a tableσExplain with suitable example.

15. Differentiate between primary key and candidate key.
16. Differentiate between candidate key and alternate key.
17. What all are domain name possible in genderσ
18. Write any four advantages of SQL.
19. In which situation one can apply union operation of two tables.
20. Differentiate between union and Cartesian product.
21. A table 'customer' has 10 columns but no row. Later, 10 new rows are inserted and 3 rows are deleted in the table. What is the degree and cardinality of the table 'customerσ
22. A table 'game1' has 3 columns and 20 rows and another table 'game2' has the same column as game1 (ie 3) and 15 rows. 5 rows are common in both the table. If we take union, what is the degree and cardinality of the resultant tableσ
23. A table 'student1' has 4 columns and 10 rows and 'student2' has 5 columns and 7 rows. If we take Cartesian product of these two tables, what is the degree and cardinality of the resultant tableσ
24. From the following two tables, write the output of the Cartesian product. Also write primary key of customer and product table.

Customer

Cust. No.	Name	Address	Phone No.
111	Rohan Aggarwal	Delhi	28756389
222	Kanika Jain	Delhi	29807654
333	Keshav Gupta	Mumbai	25678945
444	Dharna	Bambay	24675678

Product

P. No.	P. Name	Price	Qty
101	Computer	35000	3
103	Scanner	20000	2
105	Printer	15000	1

25. In the following two tables, find the union value of employee and emp.

EMPLOYEE

Emp. No.	Name	Salary
1000	Abishek Garg	25000
222	Prachi Goal	30000
1002	Simran Dua	25000
1003	Rishika Pal	40000
1004	Mohit Batra	23000

EMP

Emp. No.	Name	Salary
1002	Simran Dua	25000
1004	Mohit Batra	23000
1007	Sonal Gupta	26000
1009	Rohit Batia	50000

Chapter-2: Structure Query Language

Learning Objectives

At the end of this chapter the students will be able to understand:

- What is SQLσ
- Need for SQL
- How to create tables in SQLσ
- How to add information to tablesσ
- SELECT … FROM…WHERE (with aggregate functions)
- GROUP BY ….HAVING
- ORDER BY
- UPDATE AND DELETE Command
- ALTER TABLE AND DROP TABLE Command
- EQUI JOIN

Introduction

SQL (Structured Query Language) is a standard language for accessing and manipulating databases. SQL commands are used to create, transform and retrieve information from Relational Database Management Systems and also used to create interface between user and database. By using SQL commands, one can search any data in the database and perform other functions like, create tables, add records, modify data, remove rows, drop table etc. SQL commands are used to implement the following;

- SQL can retrieve data from a database
- SQL can insert records in a database
- SQL can update records in a database
- SQL can delete records from a database
- SQL can create new databases
- SQL can create new tables in a database
- SQL can create views in a database

CREATE TABLE Command

CREATE TABLE command is used to create table structure. In this command, we need to give full information about table such as number of columns, type of each column and constraints (primary key).

The CREATE TABLE command requires:

- Name of the table,
- Names of fields,
- Definitions and constrains for each field.

Constrains

In SQL, we have the following constraints:

- NOT NULL - To check a column cannot store NULL value.
- PRIMARY KEY - To check that a column have an unique identity which helps to find a particular record in a table.

Syntax:

CREATE TABLE<table name>
(<column name1> <data type>[size][constraints],
<column name2> <data type>[size][constraints],
.
.
.
<column name n> <data type>[size][constraints]);

Example:

Create the following table:

Table: Student

Column Name	Data Type	Size	Constraints
Adno	Numeric	3	Primary key
Name	Varchar	20	NOT NULL
Class	Numeric	2	
Section	Char	1	
Fees	Numeric	10, 2	

Command:

CREATE TABLE student
(Adno Numeric (3) Primary Key,
Name varchar (20) not null,
Class Numeric (2),
Section char (1),
Fees numeric (10, 2));

INSERT INTO Command:

This command is used to add rows in the table, but can add only one row at a time.

Syntax:

INSERT INTO <table name> [Column_name1, Column_name2, ...Column_name n] VALUES (value1,value2,value3,….,value n);

OR

INSERT INTO <table name> VALUES (value1,value2,value3,….,value n);

Note: [] Option

Example:

Insert the following information to the table student:

Adno	Name	Class	Section	Fees
111	Anu Jain	12	A	2500
222	Mohit Sharma	11	B	4500
333	K.P.Gupta	12	B	3000
444	Ajit Kumar	10	A	2000
555	Nandini	12	C	3000
666	Rohan Sharma	11	B	2500

INSERT INTO student VALUES (111,"Anu Jain", 12,"A", 2500);

INSERT INTO student VALUES (222,"Mohit Sharma", 11,"B", 4500);

[**Note:** If we want to insert values from the selective columns then we have to use this method INSERT INTO student (ADNO, Name, CLASS) VALUES (777,' LEENA', 'B');]

SELECT Command

This command is used to view table information from SQL database. By using SELECT command, we can get one or more fields information, while using *, one can get all fields information.

Syntax:

SELECT (*/field list)

FROM <table name>

[WHERE <condition>];

We can specify any condition using where clause. Where clause is optional.

Example:

1. Display student table information.
   SELECT *
   FROM student;
   This will display all information of the particular table (student) in the database.
2. To display name and class of student table information.
   SELECT name, class
   FROM student;
3. To display name of 10th class student information.
   SELECT name
   FROM student
   WHERE class = 10;

Operators used in SQL commands:

Arithmetic operators:

Arithmetic operator takes two operands and performs a mathematical calculation on them. However, they can be used only in SELECT command. The arithmetic operators used in SQL are:

+ Addition

- Subtraction

* Multiplication

/ Division

Example (string join)

1) **Table:** Name

First Name	Second Name
Anu	Jain
Madhu	Bhattia

Display first name with second name.

SELECT FirstName + SecondName

FROM Name;

Output:

FirstName + SecondName
Anu Jain
Madhu Bhattia

2) **Table:** Salary

Basic	DA
25000	5000
35000	7000

SELECT Basic + DA

FROM Salary;

Output:

Basic + DA
30000
42000

SELECT Basic + DA as "NET PAY"

FROM Salary;

[Note: If we want to give new name of the column then we have to use above format]

Output:

NET PAY
30000
42000

Select DA-100

From salary;

Output:

DA-100
4900
6900

Select DA*100

From salary;

Output:

DA*100
500000
700000

Select DA/100

From salary;

Output:

DA/100
50
70

Relational operators:

Relational operators are used to implement comparison between two operands. These operators can be used only in 'where clause'. Relational operators are -

< less than

> greater than

< = less than or equal to

> = greater than or equal to

= equal to

! = not equal to

Example:

Table: Student

Adno	Name	Class	Section	Fees
111	Anu Jain	12	A	2500
222	Mohit Sharma	11	B	4500
333	K.P.Gupta	12	B	3000
444	Ajit Kumar	10	A	2000
555	Nandini	12	C	3000
666	Rohan Sharma	11	B	2500

1. Display students' name, who are paying below 3000 fees.

```
SELECT name
FROM student
WHERE fees<3000;
```

Output:

Name
Anu Jain
Ajit Kumar
Rohan Sharma

2. Display students' name, who are paying above or equal to 3000 fees.

```
SELECT name
FROM student
WHERE fees>=3000;
```

Output:

Name
Mohit Sharma
Nandini

3. Display students' information, who are not in class 10

```
SELECT *
FROM student
WHERE class! = 10;
```

Adno	Name	Class	Section	Fees
111	Anu Jain	12	A	2500
222	Mohit Sharma	11	B	4500
333	K.P.Gupta	12	B	3000
555	Nandini	12	C	3000
666	Rohan Sharma	11	B	2500

Logical operators:

Logical operators are also possible only in 'where clause' and are used to merge more than one condition. Logical operators are:

AND
OR
NOT

Example:

1. Display information of students in class 11B.

```
SELECT *
FROM student
WHERE class = 11 AND section = 'B';
```

Adno	Name	Class	Section	Fees
222	Mohit Sharma	11	B	4500
666	Rohan Sharma	11	B	2500

2. Display 11th and 12th class students' information.

```
SELECT *
FROM student
WHERE class =11 OR class=12;
```

Adno	Name	Class	Section	Fees
111	Anu Jain	12	A	2500
222	Mohit Sharma	11	B	4500
333	K.P.Gupta	12	B	3000
555	Nandini	12	C	3000
666	Rohan Sharma	11	B	2500

3. Display students' information, who are not in 10th class.

```
SELECT *
FROM student
WHERE NOT class = 10;
```

Adno	Name	Class	Section	Fees
111	Anu Jain	12	A	2500
222	Mohit Sharma	11	B	4500
333	K.P.Gupta	12	B	3000
555	Nandini	12	C	3000
666	Rohan Sharma	11	B	2500

LIKE OPERATOR

LIKE OPERATOR is used to search a value similar to specific pattern in a column using wildcard operator. There are two wildcard operators - percentage sign (%) and underscore (_). The percentage sign represents zero, one, or multiple characters, while the underscore represents a single number or character. The symbols can be used in combinations.

For example:

1. Display the names that start with letter "A".

   SELECT name

   FROM student

   WHERE name LIKE "A%";

   Here, % replaces one or more characters.

Name
Anu Jain

2. Display names, whose name's second letter is 'o'.

   SELECT name

   FROM student

   WHERE name LIKE "_o%";

   Here, % replaces one or more than one character and _ replaces only one character.

Name
Mohit Sharma
Rohan Sharma

3. Display names, whose name has 7 characters.

   SELECT name

   FROM student

   WHERE name LIKE "_____";

   Here, _ replaces only one character. As such, 7 underscores replace 7 characters.

Name
Nandini

IN Operator

The IN operator allows us to specify multiple values in a WHERE clause

For example:

Display students' information, who are in section A and B.

```
SELECT *
FROM student
WHERE class IN ("A","B");
```

Adno	Name	Class	Section	Fees
111	Anu Jain	12	A	2500
222	Mohit Sharma	11	B	4500
333	K.P.Gupta	12	B	3000
444	Ajit Kumar	10	A	2000
666	Rohan Sharma	11	B	2500

BETWEEN Operator

The BETWEEN operator is used to test whether or not a value (stated before the keyword BETWEEN) is "between" the two values stated after the keyword BETWEEN.

For example:

Display students' information, who are paying fees between 2500 and 3500.

```
SELECT *
FROM student
WHERE fees BETWEEN 2500 AND 3500;
```

[Note: In the above Query 2500 and 3500 is also included]

Adno	Name	Class	Section	Fees
111	Anu Jain	12	A	2500
333	K.P.Gupta	12	B	3000
444	Ajit Kumar	10	A	2000
555	Nandini	12	C	3000
666	Rohan Sharma	11	B	2500

ORDER BY

This command is used to arrange values in ascending or descending order.

For example:

SELECT *
FROM student
ORDER BY fees ASC;

'asc' for ascending order. Without asc also the list is displayed with ascending order only.

Adno	Name	Class	Section	Fees
444	Ajit Kumar	10	A	2000
111	Anu Jain	12	A	2500
666	Rohan Sharma	11	B	2500
333	K.P.Gupta	12	B	3000
555	Nandini	12	C	3000
222	Mohit Sharma	11	B	4500

SELECT *
FROM student
ORDER BY fees DESC;

'desc' for descending order. If the 'desc' is not given, the list will be displayed with ascending order.

Adno	Name	Class	Section	Fees
222	Mohit Sharma	11	B	4500
555	Nandini	12	C	3000
333	K.P.Gupta	12	B	3000
666	Rohan Sharma	11	B	2500
111	Anu Jain	12	A	2500
444	Ajit Kumar	10	A	2000

Aggregate functions

Aggregate functions are used to implement calculation based upon a particular column. These functions always return a single value.

Aggregate functions are:

1. SUM()
2. AVG()
3. MAX()

4. MIN()
5. COUNT()

SUM()

This function is used to find the total value of a particular column.

Example:

SELECT SUM (fees)
FROM student;

SUM (fees)
17500

AVG()

This function is used to find the average value of a particular column.

Example:

SELECT AVG (fees)
FROM student;

AVG (fees)
2916.6666

MAX()

This function is used to find the maximum value of a particular column.

Example:

SELECT MAX (fees)
FROM student;

MAX (fees)
4500

MIN()

This function is used to find the minimum value of a particular column.

Example:

SELECT MIN (fees)
FROM student;

MIN(fees)
2000

COUNT()

This function is used to find the number of values (ie. number of rows) of a particular column.

Example:

SELECT COUNT (fees)
FROM student;
(or)
SELECT COUNT (*)
FROM student;

COUNT (fees)
6

(or)

COUNT (*)
6

GROUP BY

The SQL GROUP BY is a clause that enables SQL aggregate functions for grouping of information. (ie. GROUP BY clause is used in collaboration with the SELECT statement to arrange identical data into groups.). This clause is used whenever aggregate functions by group are required.

For example:

1. Display number of students in each class.

   SELECT count (*), class

   FROM student

   GROUP BY class;

Count (*)	Class
2	11
3	12
1	10

2. Display sum of fees for each class.

SELECT class, sum (fees)
FROM student
GROUP BY class;

Class	Sum (fees)
11	7000
12	8500
10	2000

Having clause

As mentioned earlier, the 'where' clause is used only to place condition on the selected columns, whereas the 'HAVING' clause is used to place condition on groups created by 'group by' clause, because here the 'WHERE' clause is not useable.

Example:

Display sum of fees which is more than 5000 for each class

SELECT class, sum (fees)
FROM student
GROUP BY class
HAVING sum (fees)>5000;

Class	Sum (fees)
11	7000
12	8500

DISTINCT

The DISTINCT keyword is used to remove duplicate values in a particular column.

For example:

Display class in student table.

SELECT class
FROM student;

Class
12
11

12
10
12
11

Display different classes from student table.

SELECT DISTINCT class

FROM student;

Class
12
11
10

UPDATE Command

This command is used to implement modification of the data values.

Syntax:

UPDATE <table name>

SET <column name1>=new value, <column name>=new value etc

[WHERE <condition>];

Example:

1. Increase fees value by 500.

   UPDATE student

   SET fees = fees + 500;

Adno	Name	Class	Section	Fees
111	Anu Jain	12	A	3000
222	Mohit Sharma	11	B	5000
333	K.P.Gupta	12	B	3500
444	Ajit Kumar	10	A	2500
555	Nandini	12	C	3500
666	Rohan Sharma	11	B	3000

2. Increase the fees value by 100 for adno 222.

```
UPDATE student
SET fees = fees+100
WHERE adno = 222;
```

Adno	Name	Class	Section	Fees
111	Anu Jain	12	A	3000
222	Mohit Sharma	11	B	5100
333	K.P. Gupta	12	B	3500
444	Ajit Kumar	10	A	2500
555	Nandini	12	C	3500
666	Rohan Sharma	11	B	3000

DELETE Command

This command is used to remove information from a particular row or rows. Please remember that this command will delete only row information but not the structure of the table.

Syntax:

```
DELETE
FROM <table name>
[WHERE <condition>];
```

For example:

1. Remove adno 444 information.

```
DELETE
FROM student
WHERE adno = 444;
```

Adno	Name	Class	Section	Fees
111	Anu Jain	12	A	3000
222	Mohit Sharma	11	B	5100
333	K.P.Gupta	12	B	3500
555	Nandini	12	C	3500
666	Rohan Sharma	11	B	3000

2. Remove all records.

   DELETE

   FROM student;

Adno	Name	Class	Section	Fees

ALTER TABLE command

This command is used to implement modification of the structure of the table. This is a DDL command. Using this command, we can add a new column, remove the existing column and modify data type of existing column.

Syntax:

ALTER TABLE <table name>

[ADD/MODIFY/DROP] <column name>;

For example:

1. Add one new column totalfees with number (10, 2).

   ALTER TABLE student

   ADD totalfees number(10,2);

2. Change totalfees datatype as number(12,2).

   ALTER TABLE student

   MODIFY totalfees number(12,2);

3. Remove totalfees column.

   ALTER TABLE student

   DROP totalfees;

DROP TABLE Command

This command is used to remove the entire structure of the table and information. This is also from the DDL command.

Syntax:

DROP TABLE<table name>;

For example:

Remove the whole structure of student table.

DROP TABLE student;

Equi Join

Equi Joins are used to give information in different tables. It is a special type of join in which we use only equality operator.

For example

```
SELECT *
FROM product, customer
WHERE product.product_no = customer. procuct_no;
(or)
SELECT *
FROM product p, customer c
WHERE p.product_no=c.procuct_no;
```

Product_no	Product_name	Price	Cust_no	Cust_name	City	Product_no
111	Computer	50000	301	Rohan	Bangalore	111
222	Printer	10000	201	Mohan	Mumbai	222
333	Scanner	12000	101	Kavitha	Delhi	333
333	Scanner	12000	401	Sahil	Mumbai	333
444	Modem	500	501	Rohita	Delhi	444

SQL Non-equi joins

The non-equi join is a type of join in which, we use only relational operators except equal operator. These include >, <, >=, >= and !=.

For example

```
SELECT *
FROM product,customer
WHERE product.product_no!=customer.procuct_no;
```

Product_no	Product_name	Price	Cust_no	Cust_name	City	Product_no
111	Computer	50000	201	Mohan	Mumbai	222
111	Computer	50000	101	Kavitha	Delhi	333
111	Computer	50000	401	Sahil	Mumbai	333
111	Computer	50000	501	Rohita	Delhi	444

222	Printer	10000	301	Rohan	Bangalore	111
222	Printer	10000	101	Kavitha	Delhi	333
222	Printer	10000	401	Sahil	Mumbai	333
222	Printer	10000	501	Rohita	Delhi	444
333	Scanner	12000	301	Rohan	Bangalore	111
333	Scanner	12000	201	Mohan	Mumbai	222
333	Scanner	12000	501	Rohita	Delhi	444
444	Modem	500	301	Rohan	Bangalore	111
444	Modem	500	201	Mohan	Mumbai	222
444	Modem	500	101	Kavitha	Delhi	333
444	Modem	500	401	Sahil	Mumbai	333

LET'S REVISE

- **CREATE TABLE:** Used to create structure of the table.
- **ALTER TABLE:** Used to implement structure modification.
- **DROP TABLE:** To remove full structure.
- **INSERT INTO:** To add row values.
- **SELECT:** To display row or column information.
- **DISTINCT:** To select different information.
- **MAX():** To select maximum value of a particular column.
- **MIN():** To select minimum value of a particular column.
- **SUM():** To find total value of a particular column.
- **AVG():** To find average value of a particular column.
- **COUNT():** Number of records in the table.

EXERCISE

1. Mr. Rohan has created a table 'student' with rollno., name, class and section. Now he is confused to set the primary key. So identify the primary key column.

2. Ms. Ravina is using a table 'customer' with custno, name, address and phonenumber. She needs to display name of the customers, whose name start with letter 'S'. She wrote the following command, which did not give the result.

   Select * from customer where name="S%";

   Help Ms. Ravina to run the query by removing the errors and write the correct query.

3. Write SQL query to add a column totalprice with data type numeric and size 10, 2 in a table product.

4. The name column of a table 'student' is given below.

Name
Anu Sharma
Rohan Saini
Deepak Sing
Kannika Goal
Kunal Sharma

Based on the information, find the output of the following queries:

- Select name from student where name like "%a";
- Select name from student where name like "%h%";

5. A customer table is created with cno,cname, and address columns. Evaluate the following statement whether it is correct or notσ

   Delete cname from customer;

6. Shopkeeper needs to change the first name of one of his customers in table 'customer'. Which command should he use for the sameσ

7. Sonal needs to display name of teachers, who have "o" as the third character in their name. She wrote the following query.

   Select name
   From teacher
   Where name ="$$oσ";

But the query is not producing the result. Identify the problems.

8. Pooja created a table 'bank' in SQL. Later on, she found that there should have been another column in the table. Which command is used to add column to the tableσ

9. Surpreeth wants to add two more records of customer in customer table. Which command is used to implement thisσ

10. Deepak wants to remove all rows from the tableBank. But he needs to maintain the structure of the table. Which command is used to implement the sameσ

11. While creating table 'customer', Rahul forgot to add column 'price'. Which command is used to add new column in the table. Write the command to implement the same.

12. Write the syntax of creating table command.

13. Write the syntax of dropping table command.

14. What all are the clause possible in select statement.

15. What is the default value of order by command.

16. Differentiate between delete and drop table command.

17. Differentiate between update and alter table command.

18. Differentiate between order by and group by command.

19. Define the following.

    a) Union

    b) Cartesian product

    c) Equi Join

    d) Non equi join.

20. What is the use of wildcardσ

21. Create the following table items.

Column name	Data type	Size	Constrain
Itemno	Number	3	Primary key
Iname	Varchar	15	
Price	Number	10,2	
Quantity	Number	3	

22. Insert the following information:

Table: Item

Itemno	Iname	Price	Quantity
101	Soap	50	100
102	Powder	100	50
103	Face cream	150	25
104	Pen	50	200
105	Soap box	20	100

23. Write queries based upon item table given in q. no 22.

a) Display all items information.
b) Display item name and price value.
c) Display soap information.
d) Display the item information whose name starts with letter 's'.
e) Display a report with item number, item name and total price. (total price = price * quantity).
f) Display item table information in ascending order based upon item name.
g) Display item name and price in descending order based upon price.
h) Display item name, whose price is in between 50 to 100.
i) Add new column totalprice with number (10, 2).
j) Increase price value by 100.
k) Fill up totalprice = price * quantity.
l) Remove powder information.
m) Remove totalprice column.
n) Remove whole item structure.

24. Write outputs based upon item table given in q. no 22.

a) select sum(price) from item;
b) select avg(price) from item;
c) select min(price) from item;
d) select max(price) from item;
e) select count(price) from item;
f) select distinct price from item;
g) select count(distinct price) from item;

h) select iname,price*quantity from item;

25. In a database there are two tables - 'Brand' and 'Item' as shown below:

BRAND:

ICODE	BNAME
100	SONY
200	HP
300	LG
400	SAMSUNG

ITEM:

ICODE	INAME	PRICE
100	TELEVISION	25000
200	COMPUTER	30000
300	REFRIGERATOR	23000
400	CELL PHONE	40000

Write MYSQL queries for the following:

a) To display Iname, price and corresponding Brand name (Bname) of those items, whose price is between 25000 and 30000 both values inclusive).

b) To display ICode, Price and BName of the item, which has IName as "Television".

c) To increase the Prices of all items by Rs. 10%.

26. Create the following table

Students

Column name	Data type	Size	Constraints
Adno	Integer	3	Primary key
Name	Varchar	20	
Average	Integer	3	
Sex	Char	1	
Scode	Integer	4	

27. Insert the following information:

Students

Adno	Name	Average	Sex	Scode
501	R.Jain	98	M	111
545	Kavita	73	F	333
705	K.Rashika	85	F	111
754	Rahul Goel	60	M	444
892	Sahil Jain	78	M	333
935	Rohan Saini	85	M	222
955	Anjali	64	F	444
983	Sneha Aggarwal	80	F	222

28. Write queries based upon item table given in q. no 27.

(i) Display all students' information.

(ii) Display Rohan Saini's information.

(iii) Display number of students in the table.

(iv) Display number of students in each sex.

(v) Display students' information in ascending order using name.

(vi) Display students' information in descending order using average marks.

(vii) Display students' name starting with letter "K".

(viii) Display students' information, whose name ends with "l".

(ix) Display a report with adno,name,average*5 as total marks from student table.

(x) Display students' information, whose average marks are in between 80 to 90.

(xi) Display students' info., who are getting average marks of more than 80 and scode 333.

(xii) Display students' name and average marks, whose scode is 222 and 333.

(xiii) Display sum of average marks.

(xiv) Display maximum average marks

(xv) Display minimum average marks.

(xvi) Display average value of average marks.

(xvii) Display maximum, minimum and sum of average marks in each scode.

(xviii) Display number of students in each scode.

29. Create the following table.

Column name	Data type	Size	Constraints
Scode	Integer	3	Primary key
Sname	Varchar	20	
Place	Varchar	10	

30. Insert the following information.

Streams

Scode	Sname	Place
111	Science	SBlock
222	Commerce	CBlock
333	Humanity	HBlock
444	Art	ABlock

31. Write queries based upon item table given in q. no 27& 30

(i) To display Adno, Name, Sex and Average from Student's table and Stream name (Sname) and place from Stream table with respect to Scode.

(ii) Add the following information into the student table.

999 Deepak Sharma 83 M 2222

(iii) Display science stream students' information.

32. Give the output of the following SQL queries.

(i) Select sum(Average)

From students

Where sex='M';

(ii) Select distinct (Scode)

From students;

33. Remove 111 scode information.
34. Add new column state with varchar(10).
35. Increment 2 marks for 444 scode info.
36. Remove column state.
37. Remove the whole table stream.

Unit-4: Introduction to Boolean Algebra

Chapter-1 : Boolean Algebra

Learning Objective

At the end of the chapter students will:

- Learn Fundamental concepts and basic laws of Boolean algebra.
- Learn about Boolean expression and will be able to generate same.
- Understand the basic operations used in computers and other digital circuits.
- Be able to represent Boolean logic problems as - truth table .
- Understand how logic relates to computing problems.

Boolean Algebra was introduced by George Boole in 1854. He was an English mathematician, who primarily worked on differential equations and the calculus of variations. But the work for which he is best known is - application of algebraic systems to non-mathematical reasoning. He proposed that logical propositions should be expressed as algebraic equations, so now logic was reduced to algebra. In his logical deductions he replaced the operation of multiplication by AND, and addition by OR.

However, it is Claude Shannon, whose work on using Boolean Algebra to design Switching Circuits in late 1930s, became the foundation of computer Logic and Design. As Boolean logic allows things to be mapped into bits and bytes, it has application in telephone switching circuits, electronics, computers (both hardware and software), etc.

Boolean Algebra, which is algebra of two values may be (True, False) or (Yes, No) or (0, 1), is an important tool in analyzing, designing and implementing digital circuits.

Boolean Algebra is made up of

- Elements - which are variables or constants with value 1 or 0.
- Operators - which are And, Or and Not.
- Axioms and Theorems.

A boolean variable is a symbol used to represent a logical quantity. It will take value from the domain {0, 1}, and boolean constant is single digit binary value (bit) viz. 0 or 1.

There are three fundamental operators- AND, OR and NOT. AND is a binary operator, to perform logical multiplication, it is represented by '.' OR is also a binary operator, to perform logical addition. It is represented by '+'. NOT is a unary operator, to complement the operand. Not is represented as ' or $\bar{}$. Complement is the inverse of a variable/ constant. In case of boolean algebra, since the variable/constant can have value 0 or 1 so complement will be 1 or 0.

Note: A binary operator is applied on two operands and unary to one.

Just like algebra, Boolean algebra also have axioms and theorems, which describe how logical quantities behave. We know that axiom is a statement which is considered to be true, and theorems are to be proved.

First axiom is Closure Property, it states that Boolean operations (+ or .) on Boolean variables or constants will always result into a Boolean value.

Following are other axioms and theorems of Boolean algebra:

Identity: states that, sum of anything and zero, or product of anything and one, is same as the original anything. So identity with respect to + is 0, and with respect to . is 1

$A + 0 = A$ and $A.1 = A$

Commutative: property tells us, we can reverse the order of variables, that are either ORed together or ANDed together without changing the truth of the expression.

$A + B = B + A$ and $A . B = B . A$

Distributive: law states that, ORing variables and then ANDing the result with single variable is equivalent to ANDing the result with a single variable with each of the several variables and then ORing the products. Vice versa with respect to operators and terms is also true.

$A . (B + C) = A . B + A . C$ and $A + (B . C) = (A + B) . (A + C)$

Complement: states that sum of a Boolean quantity with its complement or product of a Boolean quantity with its complement results into identity

$A + A' = 1$ and $A . A' = 0$

Indempotency: states that when we sum or product a boolaen quantity to itself, the resultant is original quantity.

$A + A = A$ and $A . A = A$

Null Element: No matter what the value of Boolean variable, the sum of variable and 1 will always be 1. Also the product of Boolean variable and 0 will always be 0.

$A + 1 = 1$ and $A . 0 = 0$

Involution: states that complementing a variable twice, or any even number of times, results in the original Boolean value.

$(A')' = A$

Associative: this property tells us that we can associate, group of sum or product variables together with parenthesis without altering the truth of the expression

$A + (B + C) = (A + B) + C$ and $A . (B . C) = (A . B) . C$

Absorption: is also known as covering, have three forms

1. $A + AB = A$ and $A . (A + B) = A$
2. $A + (A' . B) = A + B$ and $A . (A' + B) = A . B$
3. $(A + B) . (A' + C) . (B + C) = (A + B) . (A' + C)$ AND $(A . B) + (A' . C) + (B . C) = A . B + A' . C$

De Morgan's Theorem: states that the complement of a sum / product, equals the product / sum of the complements.

$(A + B)' = A' . B'$ and $(A . B)' = A' + B'$

Apart from these axioms and theorems, there is a principle. Duality Principle which states that starting with Boolean relation, you can device another Boolean relation by

1. Changing each OR operation to AND
2. Changing each AND operation to OR
3. Complementing the identity (0 or 1) elements.

For example dual of relation

$A + 1 = 1$ is $A . 0 = 0$.

You must have noticed that second part of axioms and theorems is actually the result of this principle only. Practical consequence of this theorem is, it is sufficient to prove half the theorem, the other half comes gratis from the principle.

If we compare Boolean Algebra with Arithmetic algebra, we will find that

- In Boolean algebra **+** can distribute over **.**
- Logical subtraction and logical division are not available in Boolean algebra, as there is no additive or multiplicative inverse available.
- Complement is available in Boolean algebra
- Two valued Boolean algebra contains 0 & 1 only.

Boolean Function - is an expression formed with boolean variable(s), boolean constant(s), boolean operator(s), parenthesis and equal sign. It will always result into a boolean value, as specified by closure property. There are three ways to represent a boolean expression/function viz.

1. Algebraic
2. Truth Table
3. Circuit Diagram

Algebraic Representation:

Axioms and Theorems written above are examples of algebraic representation, but let's have some more examples:

$F_1 = a.b$

formed using two boolean variables a & b with AND operator. This function will be equal to 1, for a = 1 and b = 1, otherwise it will be 0.

$F_2 = xyz'$

formed using three boolean variables x, y and z with AND operator. Function will be 1 if x = 1, y = 1, and z' = 1 i.e. z = 0, in all other situations F_2 will be 0.

They can also be represented as

$F_1 (a, b) = a.b$

$F_2 (x,y,z) = x.y.z'$

These are examples of some simple Boolean functions. Complex boolean functions can be formed using simple expressions. Following are the examples of such functions:

$f_3 (a, b, c) = (a+b).c$

$f_4 (x,y) = (x+y). (x'+y')$

$f_5(x, y, z) = x. (y+z). (x+y')$

$f_6 (w, x, y, z) = y'z + wxy' + wx'z$

When an expression/function contains two or more operators, the order of operations is decided by parenthesis and precedence of operators. Moving from high to low precedence, following table gives the precedence of Boolean operators.

Operator	Precedence
()	High
NOT	↓
AND	↓
OR	Low

Truth Table representation

A truth table is a chart of 1's and 0's, arranged to indicate the results of all possible input options. Number of columns in the table is decided by the number of Boolean variables in the function. Each variable will be represented in a column. For n variable expression, there will be 2^n rows in the table. For filling these rows in the table, they will be the binary representation of the integers from 0 to (2^n-1), in the same order.

Let's fill the table for one variable - **a**

a
0
1

Since the table is for one variable there will be only two rows $2^1 = 2$ rows. For two variables - a & b, the table will have 4 rows ($2^2 = 4$), listed below

a	b
0	0
0	1
1	0
1	1

Three Variables - a, b & c, 8 rows

a	b	c
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

and so on. The trick for filling the table is, starting from right most column of the table - 1st column will be combination of 0,1,0,1,…… (2^0 zeros and then same number of 1s. In 2nd column it will be 0,0,1,1,0,0,….. 2^1 zeros and then same number of ones. In 3rd column 0,0,0,0,1,1,1,1,0,0, …… (2^2 number of zeros and ones), and so on.

Now let's use one variable table, to represent a simple boolean function f (a) = a'

a	f (a)
0	1
1	0

Second column of table gives the value of Boolean function i.e. **a'**. This table tells that if a is true, a' will be false and vice versa, as is evident from 2nd column of table.

Truth table for the boolean function applied on two variables with all basic Boolean operators i.e. AND, OR will be

a	b	f = a.b	f = a + b
0	0	0	0

0	1	0	1
1	0	0	1
1	1	1	1

Now let's represent some more functions written earlier in algebraic form using truth table:

$F_2(x,y,z) = xyz'$

x	y	z	f2
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

From our earlier explanation we know that F_2 will be one for x = 1, y = 1 and z = 0.

$f_3(a, b, c) = (a+b).c$

a	b	c	(a+b)	f3 = (a+b).c
0	0	0	0	0
0	0	1	0	0
0	1	0	1	0
0	1	1	1	1
1	0	0	1	0
1	0	1	1	1
1	1	0	1	0
1	1	1	1	1

$f_4(x,y) = (x+y).(x'+y')$

x	y	x'	y'	(x+y)	(x'+y')	f4 = (x+y).(x'+y')
0	0	1	1	0	1	0
0	1	1	0	1	1	1
1	0	0	1	1	1	1
1	1	0	0	1	0	0

Third way of representation of the Boolean function will be taken up later in the Unit.

For now let's move on to proving the various theorems of Boolean algebra:

Theorems can be proved in two ways:

i) **Algebraic Method:** In this method, we use axioms & theorems. This method is similar to what you do in mathematics.

ii) **Truth Table:** Here we use truth table to show that expression given on LHS results into same values for expression on RHS. This actually is verification of theorem, not proving.

Associative:

Theorem states that A + (B+C) = (A+B) + C and A.(B.C) = (A.B).C

Let's algebraically prove it:

Let X = [A + (B + C)] [(A + B) + C]

then

$$
\begin{aligned}
X &= [(A + B) + C]\, A + [(A + B) + C]\, (B + C) \\
&= [(A + B)\, A + C\, .\, A] + [(A + B) + C]\, (B + C) \\
&= A + [\,(A + B) + C]\, (B + C) \\
&= A + [(A + B) + C]\, B + [(A + B) + C]\, C \\
&= A + (B + C)
\end{aligned}
$$

But also

$$
\begin{aligned}
X &= (A + B)\, [A + (B + C)] + C\, [A + (B + C)] \\
&= (A + B)\, [A + (B + C)] + C \\
&= A\, [A + (B + C)] + B\, [A + (B + C)] + C \\
&= (A + B) + C
\end{aligned}
$$

Thus A + (B + C) = (A + B) + C

We know, second form of the theorem will also be true. (Duality Principle)

Truth table method:

A	B	C	B+C	LHS A+(B+C)	(A+B)	RHS (A+B)+C
0	0	0	0	0	0	0
0	0	1	1	1	0	1
0	1	0	1	1	1	1
0	1	1	1	1	1	1
1	0	0	0	1	1	1
1	0	1	1	1	1	1
1	1	0	1	1	1	1
1	1	1	1	1	1	1

As LHS= RHS, hence verified

Null Element states that A + 1 = 1 and A . 0 = 0

Algebraic method:

Taking LHS

A+1	= (A+1).1	by Identity
	= (A+1). (A+A')	by Complement
	= A + (1.A')	by Distribution
	= A+ (A'.1)	by Indempotancy & Commutative
	= A+A'	by Identity
	= 1	by Complement

By applying the principle of Duality, we get A.0 = 0

Truth Table method:

A	1	LHS A+1	RHS
0	1	1	1
1	1	1	1

As LHS= RHS, hence verified

Involution states that ((A')') = A

Taking LHS

(A')'	= (A')' + 0	by Identity	
	= (A')' + (A'.A)	by Complement	
	= ((A')' + A'). ((A')' + A)	by Distribution	
	= 1. ((A')'+A)	by Complement	
	= (A')'+A	by Identity	(1)

Taking LHS again

(A')'	= (A')'. 1	by Identity	
	= (A')'. (A+A')	by Complement	
	= (A')' .A) + ((A')'.A')	by Distribution	
	= (A')' .A) + 0	by Complement	
	= (A')'. A		(2)

Substituting the value of (A')' = (A')'. A in (1), we get

(A')' + A	= (A')'.A + A	
	= A + (A')'.A	by Commuting

= A + A. (A')' by Commuting
= A by Absorption

We know, second form of the theorem will also be true. (Duality Principle)

Truth Table method:

A	(A')	LHS (A')'	RHS A
0	1	0	0
1	0	1	1

As LHS= RHS, hence verified

Idempotency states that A+A= A and A . A = A

Algebraic method:

Taking L.H.S

A+A = (A+A).1 by identity
= (A+A). (A+A') by complement
= (AA+AA') + (AA + AA') by distribution
= AA+0
= A

We know, second form of the theorem will also be true. (Duality Principle)

Truth Table method:

A	RHS A	LHS A+A
0	0	0
1	1	1

As LHS = RHS, hence verified

Absorption Law:

i) A + AB = A and A . (A + B) = A

Algebraic method:

Taking LHS

A + AB = (A.1) + (A.B) by Identity
= A. (1+B) by Distribution
= A.1 by Null Element
= A

Using Duality, we know that A.(A+B) = A is also true

Truth Table method:

A	B	A.B	LHSA+AB	RHS A
0	0	0	0	0
0	1	0	0	0
1	0	0	1	1
1	1	1	1	1

As LHS= RHS, hence verified

ii) (A + A'B) = A+ B and A . (A' + B) = A . B

Algebraic method:

Taking LHS

A + A'B	= (A+ A') . (A+B)	by Distribution
	= 1. (A+B)	by Complement
	= A+B	by Identity

Because of duality, A.(A'+B) = A.B is also true.

Truth Table method:

A	B	A'B	LHS A+A'B	RHS A+B
0	0	0	0	0
0	1	1	1	1
1	0	0	1	1
1	1	0	1	1

As LHS= RHS, hence verified

iii) ((A+B).(A'+C). (B+C)) = (A+B). (A'+C) AND ((A.B) + (A'.C) + (B.C)) = A.B + A'.C

Taking LHS

= (A+B).(A'+C). (B+C)	
= (A+B).(A'+C). (B+C). (B+C)	by Idempotency
= (A+B).(B+C).(A'+C). (B+C)	by Commuting
= (B+A).(B+C).(C+A'). (C+B)	by Commuting
= (B+A.C) . (C+ A'B)	by Distributive
= (B+AC).C + (B+AC). A'B	by Distribution
= B.C + A.C.C + B. A.'.B + A.C.A'B	by Distribution
= B.C + A.(C.C) + (B.B). A'+ C. (A. A').B	by Associativity and Idempotency

= B.C + A.C + B.A'+ 0 by Complement and Idempotency

= (B+A).C + B.A'+ A. A' by Complement and Distributive

= (B+A).C + (B+A).A' by Distributive

= (B+A).(C + A') by Distributive

= (A+B). (A'+C) by Commuting

Using Duality Principle we can say that A.B + A'+C + B. C = A. B. A'.C is also true

Truth Table method:

A	B	C	A+B	A'+C	B+C	LHS (A+B).(A'+C). (B+C)	RHS (A+B). (A'+C)
0	0	0	0	1	0	0	0
0	0	1	0	1	1	0	0
0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	1
1	0	0	1	0	0	0	0
1	0	1	1	1	1	1	1
1	1	0	1	0	1	0	0
1	1	1	1	1	1	1	1

As LHS= RHS, hence verified

De Morgan's Theorem:

States that (A+B)' = A'.B' and (A.B)' = A' + B'

Algebraic method:

Assuming that DeMorgan's theorem is true, then through this we can find complement of a function/expression.

We also know that

X+X' = 1 and X.X'=0

So by showing that

i) (A+B) + A'.B' = 1

ii) (A+B). (A'.B') = 0

We shall establish the DeMorgan's theorem.

i) (A+B) + A'.B' =1

Taking LHS

= (A+B) + (A'. B')= ((A+B)+ A'). ((A+B)+ B') By Distribution
= (A+(B+ A')). (A+(B+ B')) By Associativity
= (A+(A'+B)). (A+(B+ B')) By Commutativity
= (A+ (A'+B)). (A+1) By Complement
= ((A+ A')+B).(A+1) By Commutativity
= (1+B). (A+1) By Complement
= 1.1 By Null element
= 1

ii) (A+B). (A'.B')= 0

Taking LHS

= (A.(A'.B') + (B. (A'.B'))
= ((A.A').B') + ((B.A'). B') By Associativity
= 0 + ((B'.A'). B') By Null Element
= 0 + ((A'.B).B') By Commutativity
= 0 + (A'+(B.B')) By Associativity
= 0 + 0 By Null Element
= 0

Truth Table method:

A	B	A+B	LHS (A+B)'	A'	B'	RHS A'.B'
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

As LHS = RHS, hence verified.

Hence De Morgan's theorem is established. Using Duality we can say that (A . B)' = A' + B'

LET'S REVISE

- **Boolean Algebra:** is an algebra that deals with binary variables and logic operations
- **Variable:** A variable is a symbol, usually an alphabet used to represent a logical quantity. It can have a 0 or 1 value
- **Boolean Function:** consists of binary variable, constants 0 & 1, logic operation symbols, parenthesis and equal to operator.
- **Complement:** A complement is the inverse of a variable and is indicated by a' or bar over the variable.
- A **binary variable** is one that can assume one of the two values 0 and 1.
- **Literal:** A Literal is a variable or the complement of a variable
- **Truth table:** it provides the basic method of describing a Boolean function.
- **List of axioms and theorems:**

Identity	A + 0 = A	A. 1 = A
Complement	A + A' = 1	A. A' = 0
Commutative	A + B = B + A	A. B = B. A
Assosiative	A + (B + C) = (A + B) + C	A. (B. C) = (A. B). C
Distributive	A. (B + C) = A. B + A. C	A + (B. C) = (A + B). (A + C)
Null Element	A + 1 = 1	A. 0 = 0
Involution	(A')' = A	
Indempotency	A + A = A	A. A = A
Absorption	A + (A. B) = A	A. (A + B) = A
	A + A'. B = A + B	A. (A' + B) = A. B
	(A+B).(A'+C).(B+C) = (A+B).(A'+C)	(A.B)+(A'.C)+(B.C) = A.B+A'.C
	(A+B) (B+C) (A'+C)=(A+B) (A'+C)	
De Morgan's	(A + B)' = A'. B'	(A. B)' = A'. B'

EXERCISE

1. The commutative law of Boolean algebra states that
   A + B = A . B
   i) True
   ii) False
2. Applying Dr. Morgan's theorem to the expression (ABC)', we get
   i) A'+B'+C'
   ii) (A+B+C)'
   iii) A+B'+C.C'
   iv) A (B+C)
3. Which Boolean Algebra theorem allow us to group operands in an expression in any order without affecting the results of the operation:
   i) Associative
   ii) Commutative
   iii) Boolean
   iv) Distributive
4. Applying Dr. Morgans theorem to the expression ((x+y)'+z)', we get _____
   i) (x+y)z
   ii) (x'+y')z
   iii) (x+y)z'
   iv) (x'+y')z'
5. Applying the distributive law to the expression A (B+C'+D), we get_____
   i) AB+AC+AD
   ii) ABCD
   iii) A+B+C+D
   iv) AB+AC'+AD
6. A variable is a symbol used to represent a logical quantity that can have a value of 1 or 0
   i) True

ii) False

7. The OR operation is Boolean multiplication and the AND operation is Boolean addition

   i) True

   ii) False

8. In Boolean Algebra, A+1 = 1

   i) True

   ii) False

9. In the Commutative law, in ORing and ANDing of two variables. The order in which the variables are ORed or ANDed makes no difference

   i) True

   ii) False

10. Verify using truth table, DeMorgan's theorem for three variables.

11. Construct a truth table for the following functions

   i) F1(A,B,C) = A + BC'

   ii) F2(A,B,C) = AC + BC + AB'

   iii) F3(w,x,y,z) = y'z + wxy' + wxz' + wx'z

   iv) (x + y) . (y + z) . (z + x)

12. Expand the following expressions using De Morgan's theorem

   i) F1(A,B,C) = (AB)'(ABC)'(A'C)'

   ii) F2(A,B,C) = ((AB + B'C).(AC + A'C'))'

13. State the fundamental axioms of Boolean algebra.

14. Explain principle of Duality of Boolean algebra.

15. What do you mean by literal?

16. List the theorems of Boolean Algebra.

17. What is the difference between operator in mathematics and logic?

18. Write Dual of

   i) A + 1 = 1

ii) x + x'y = x + y

iii) x + x' = 1

vi) x'(y + z) + x'(y' + z')

19. How is Boolean Algebra different from Arithmetic algebra?

20. Prove following

i) C + A (C + B) + BC = C + AB

ii) (A + C)A + AC + C = A + C

iii) BC + A(B+C) = AB + BC + AC

iv) B + A (B + C) + BC = B + AC

v) x+ (x. y) = x

vi) (a' + b') . (a' + b) . (a + b') = a' . b'

21. For the given truth table, write Boolean expression for function G1, G2, G3, G4, G5

X	Y	Z	G1	G2	G3	G4	G5
0	0	0	0	0	0	0	1
0	0	1	0	1	0	1	0
0	1	0	1	0	0	1	0
0	1	1	1	1	0	1	0
1	0	0	1	0	0	1	1
1	0	1	0	0	1	0	1
1	1	0	0	1	1	0	0
1	1	1	0	0	1	0	1

22. Express the OR operator in the terms of AND and NOT operator.

23. List the reasons used for algebraic proof of Associative theorm.

Chapter-2: Boolean Functions & Reduce Forms

Learning Objective

At the end of the chapter students will

- Understand how logic relates to computing problems
- Handle simple Boolean function in SOP and POS form
- Apply rules to simplify / expand Boolean terms / functions
- Simplify the Boolean function using K map and algebraic method
- Establish the correspondence between Karnaugh maps, truth table and logical expressions.

Till now, we were using the available boolean function, now let's learn how to make a boolean function. In a boolean function, a binary variable can appear in any form, normal (a) or complemented (a'). Now consider a boolean expression formed on two variables a & b using AND operator. Since each variable may appear in any form, there are four possible ways of combining these two variables viz. a'b', ab', a'b, ab.

Similarly if we combine them using OR operator, again there are four possible ways viz. a+b, a'+b, a+b', a'+b'

For now we will concentrate on the terms formed using AND operator only. Each of them is called a minterm or Standard Product Term. A minterm, is obtained from AND term, with each variable being complemented, if the corresponding bit of binary number (in the truth table) is 0 and normal (i.e. non complemented) if the bit is 1. These terms are designated using **m.**

Following table provide minterm for 3 variables:

a	b	c	Minterm	Designation
0	0	0	**a'b'c'**	m_0
0	0	1	**a'b'c**	m_1
0	1	0	**a'bc'**	m_2
0	1	1	**a'bc**	m_3
1	0	0	**ab'c'**	m_4
1	0	1	**ab'c**	m_5
1	1	0	**abc'**	m_6
1	1	1	**abc**	m_7

For four variables, there will be 16 minterms designated by m_0 to m_{15}.

Similarly, OR terms are called maxterms or Standard Sum Term. A maxterm is obtained from OR operator, variable being complemented if the corresponding bit of binary number (in the truth table) is 1 and normal if the bit is 0. These terms are designated by M

We already have maxterms for 2 variables, maxterms for 3 variables will be:

a	b	c	maxterm	designation
0	0	0	**a+b+c**	M_0
0	0	1	**a+b+c'**	M_1
0	1	0	**a+b'+c**	M_2
0	1	1	**a+b'+c'**	M_3
1	0	0	**a'+b+c**	M_4
1	0	1	**a'+b+c'**	M_5
1	1	0	**a'+b'+c**	M_6
1	1	1	**a'+b'+c**	M_7

Maxterm for four variables can be obtained in similar manner and they are designated as M_0 to M_{15}. If we put both maxterms and minterms together, we will have following table

a	b	c	minterm	maxterm
0	0	0	**a'b'c'**	**a+b+c**
0	0	1	**a'b'c**	**a+b+c'**
0	1	0	**a'bc'**	**a+b'+c**
0	1	1	**a'bc**	**a+b'+c'**
1	0	0	**ab'c'**	**a'+b+c**
1	0	1	**ab'c**	**a'+b+c'**
1	1	0	**abc'**	**a'+b'+c**
1	1	1	**abc**	**a'+b'+c'**

By now you might have noted that each maxterm is complement of its corresponding minterm.

For *n* variables, there will be 2^n, minterms (i.e. Standard Product Term) denoted as m_i, $0 < i < 2^n - 1$, and 2^n maxterms (i.e. Standard Sum Term), their complement, denoted as M_i, $0 < i < 2^n - 1$.

Now let's use this information for algebraic representation of boolean function from the truth table. This is done by forwarding a minterm for each combination of variables which produces a 1 in the function, and then by ORing these minterms.

For eg.

a	b	c	F1	minterm
0	0	0	**0**	**a'b'c'**
0	0	1	**1**	**a'b'c**
0	1	0	**0**	**a'bc'**
0	1	1	**0**	**a'bc**
1	0	0	**1**	**ab'c'**
1	0	1	**0**	**ab' c**
1	1	0	**1**	**abc'**
1	1	1	**1**	**abc**

Since **a'b'c, ab'c', abc'** and **abc** minterms are resulting into 1, in function (F_1) column, so algebraic representation of the function F_1 will be

$F_1(a,b,c) = a'b'c + ab'c' + abc' + abc$

This is a boolean expression, expressed as sum of minterms and , will give value 1.

This form of expression is known as **Sum of Products or SOP** expression. F_1 in SOP form can also be represented as

i) $m_1 + m_4 + m_6 + m_7$

Instead of using Product terms, we can just mention the minterm designation.

ii) $\sum(1, 4, 6, 7)$

This one is mathematical representation of the above expression.

Let's take some more examples:

x	y	z	F_2	F_3	F_4
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	1
1	0	0	1	1	0
1	0	1	0	1	0
1	1	0	1	1	1
1	1	1	0	0	1

Algebraic representation in SOP form, of the three functions, represented in the truth table is

$F_2(x,y,z) = x'yz + xy'z' + xyz'$

OR

$m_3 + m_4 + m_6$

$F_3(x,y,z) = x'y'z + x'yz + xy'z' + xy'z + xyz'$

OR

$m_1 + m_3 + m_4 + m_5 + m_6$

$F_4(x,y,z) = x'yz' + x'yz + xyz' + xyz$

OR

$m_2 + m_3 + m_6 + m_7$

All of them will result into value 1.

What if we have to represent the function in complement form i.e. for its value 0. Simple, we will OR the minterms, from the truth table, for which function value is 0. So we have,

$F_1' = x'y'z' + x'yz' + x'yz + xy'z$ (the remaining minterms)

On evaluating it for complement, we get

F_1= (x+y+z). (x+y'+z) . (x+y'+z') . (x'+y+z')

What we get is the **Products of Sum form (POS) of the function F_1.**

POS form of the expression gives value 0, and we work on the complements. POS is formed by **ANDing maxterms.**

POS also can be represented in many ways. One way we have seen, others are

i) $F_1 = M_0.M_2.M_3.M_5$

Instead of using maxterms, we can use designation of it.

ii) $F_1 = \prod(0, 2, 3, 5)$

This one is mathematical representation of the above expression.

Let's look both, SOP and POS expression for same function

x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

SOP expression for F will be

$x'y'z + x'yz'$ **or** $\sum(1,2)$

POS equivalent of it will be found using complement of minterms with value zero, we know this

$(F')' = (x'y'z' + x'yz + xy'z' + xy'z + xyz' + xyz)'$

$= (x + y + z) . (x + y' + z') . (x' + y + z). (x' + y + z'). (x' + y' + z).(x' + y' + z')$

OR

$= \prod(0,3,4,5,6,7)$

Conversion from one form to another is easy. Short cut for it is :

i) Interchange the symbol (OR to AND / $\sum$ to $\prod$)

ii) List those terms, which are missing in original.

Expressions which were written till now, are in Canonical form. In this form,every term of the function should have all the literals. When, minterms or maxterms from truth table, are used for representation, the function is in canonical form only, as terms contains all variables (may be in any form). Since we can use either minterms or maxterms to form expression, we can have **Canonical SOP** expression or **Canonical POS** expressions.

Till now, we were given truth table and we were representing Boolean Function in algebraic form. Vice versa of it, i.e. for a given Boolean function in algebraic representation, we can represent it in Truth table. This was taken up in details in previous chapter.

What we have worked on was Canonical Boolean functions. There is another form of representation, also - both for SOP & POS. It is known as **reduced/simplified form.**

Simplified expressions are usually more efficient and effective when implemented in practice. Also simpler expressions are easier to understand, and less prone to errors. So the Canonical expressions are simplified/minimized to obtain a reduced expression (Standard form). This expression is then used for implementation of circuits in computers, *we will learn it in next chapter.*

Boolean function may be reduced in many ways, but we will be taking up the following two ways of reduction in this course.

1. Algebraic Manipulations
2. Karnaugh Map.

Algebraic reduction of Boolean function/expression: in this reduction, we apply certain algebraic axioms and theorems to reduce the number of terms or number of literals in the expression. In this way of reduction, there are no fixed rules that can be used to minimize the given expression. It is left to individual's ability to apply axioms or theorems for minimizing the expression.

Example 1:

F (a, b)	= (a+b)' + ab'	
	= a'b'+ab'	by De Morgan's theorem
	= (a'+a)b'	by Distribution
	= 1.b'	by Complement
	= b'	by Identity

Example 2:

F (a, b)	= ab+ ab'+ a'b	
	= a(b+b') + a'b	by Distribution
	= a.1 + a'b	by complement
	= a + a'b	by Identity
	= a+b	by absorption

Algebraic transformation, can always be used to produce optimal reduced form, but does not guarantee the reduced form. Also, it is not necessary to reach to same transformation, when worked on by different group of people. However, there are other methods using which, we will always reach to optimal and same solution every time.

Before moving ahead with other methods of reduction lets wait, to see how vice versa will be done, conversion of reduced from to Canonical form? Yes you guessed right, using theorems and axioms. Lets do it for 2nd example

F(a,b) = a + b	Reduced expression
= a . 1 + b . 1	by Identity
= a.(b + b') + b . 1	by Complement
= a . b + a . b' + b . 1	by Distribution
= a . b + a . b' + 1 . b	by Commuting
= a . b + a . b' + (a + a') . b	by Complement
= a . b + a . b' + a . b + a' . b	by Distribution
= a . b + a . b' + a' . b	by Indempotancy

Now let's learn other way of reduction.

Karnaugh Map (K-map):

A K-map is nothing more than a special form of truth table representation useful for reducing logic functions into minimal Boolean expressions. Karnaugh map was developed by Maurice Karnaugh, at Bell Labs in 1953. The map reduces boolean expression more quickly and easily, as compared to algebraic reduction.

Using K-map for reduction is 4 step process

1. Drawing K-map
2. Filling it, i.e. representing the boolean expression in map
3. Grouping the terms for reducing purpose
4. Reading the reduced expression from map

1. Drawing K-map

A K-map is a diagram, made up of squares, where each square represents one minterm. So in a two variable map there will be 4 squares and for three variables, map will contain 8 squares. K-map diagram for two variables - a,b, will look like:

a	b	minterm
0	0	a'b'
0	1	a'b
1	0	ab'
1	1	ab

a \ b	b'	b
a'	a'b' m_0	a'b m_1
a	ab' m_2	ab m_3

For three variables - a, b,c, we can draw it as:

a \ bc	b'c'	b'c	bc	bc'
a'	m_0	m_1	m_3	m_2
a	m_4	m_5	m_7	m_6

OR

ab \ c	c'	c
a'b'	m_0	m_1
a'b	m_2	m_3
ab	m_6	m_7
ab'	m_4	m_5

For four variables - a, b, c, d

ab \ cd	c'd'	c'd	cd	cd'
a'b'	m_0	m_1	m_3	m_2
a'b	m_4	m_5	m_7	m_6
ab	m_{12}	m_{13}	m_{15}	m_{14}
ab'	m_8	m_9	m_{11}	m_{10}

OR

cd \ ab	a'b'	a'b	ab	ab'
c'd'	m_0	m_4	m_{12}	m_8
c'd	m_1	m_5	m_{13}	m_9
cd	m_3	m_7	m_{15}	m_{11}
cd'	m_2	m_6	m_{14}	m_{10}

The cells of the map are marked by the minterms as explained in first map.

For eg.

If you look at the square assigned m_3 (minterm), it corresponds to binary no. 0100 of the truth table.

Now to understand, why the squares have been marked, as they are?

If you look at any two adjacent cell in the map, they differ only by one variable, which is normal in one cell and complemented in other or vice-versa. For eg. From m_3 to m_7 only **a** changes its state from complemented to normal. And we know, using Boolean axiom (a'+a=1) that we can eliminate the variable from the resultant reduced term.

2. Filling the K-map

Once a K-map is drawn, next is to fill the map i.e. **mark the function in the map.** For representation, the function should be in canonical SOP form. If we have the function represented in truth table, then it will be in canonical form only. Otherwise, we know how to convert it. We will mark 1 in all those cells of K-map, for which there is a minterm in the Boolean function/expression.

3. Grouping the minterms

Next is grouping the 1s in the map. We group 1s of adjacent cells.

Following are the rules for grouping. Remember our goal should be to maximize the size of group (i.e. no. of 1s included in the group) and to minimize the number of groups.

- A group must contain 1, 2, 4, 8, 16, …. cells, i.e. in the power of 2.

1	1
0	0

Correct

0	0	0	0
0	1	1	1

Incorrect

- Cells which are grouped, should be adjacent cells, i.e. horizontal or vertical, not diagonal.

0	1
1	1

Correct

0	1
1	0

Incorrect

- Groups should not include any cell containing zero

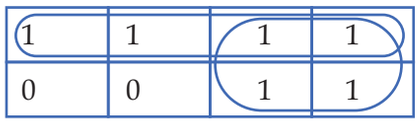
0	0
1	1

Correct

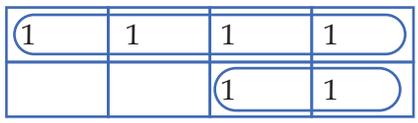
0	0
1	0

Incorrect

- Each group should be as large as possible

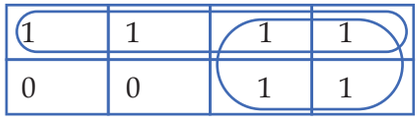


Correct

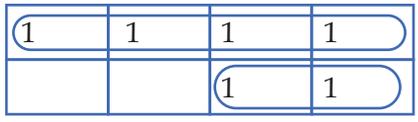


Incorrect

- Groups may overlap

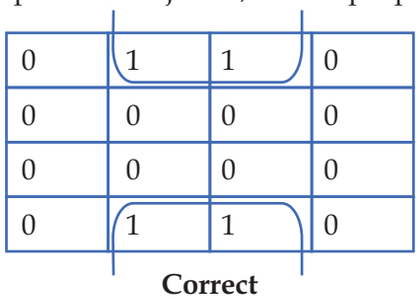


Correct

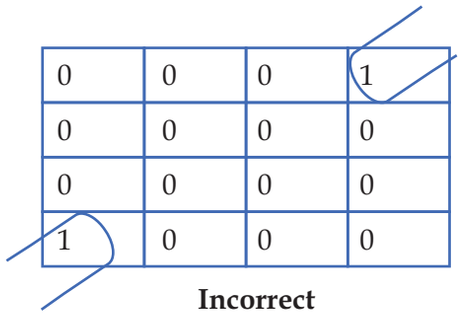


Incorrect

- Groups may wrap around the table. The left most cell(s) in a row may be grouped with rightmost cell(s), and the top cell(s) in a column may be grouped with the bottom cells(s). Cells in corners of the map are also adjacent, for that purpose



Correct



Incorrect

- There should be as few groups as possible, as long as this does not contradict any of the previous rule. Once the grouping is done, last step is

4. Reading the reduced expression from K-map

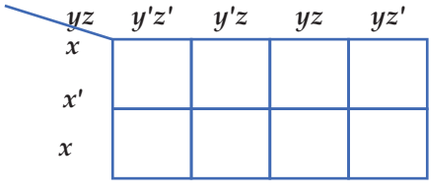
- Each group corresponds to one product term, from K-map
- The term is determined by finding the common literals in that group, i.e. the variables which change their state (from normal to complement or vice-versa) are to be omitted in resultant product term.

  What we have to do is, find the variable(s) listed on top and / or side, which are the same for entire group and ignore variable(s) which are not same in the group.

  Using the above strategy, write the result. Lets look at some examples

  Simplify the Boolean Function F(x,y,z) = x'yz + xy'z' + xyz + xyz'

  Step 1: Drawing the K-map



Step 2: Marking of function in map

yz / x	y'z'	y'z	yz	yz'
x'	0	0	1	0
x	1	0	1	1

Step 3: Grouping

yz / x	y'z'	y'z	yz	yz'
x'	0	0	1 (1)	0
x	1	0	1	1 (2)

Step 4: Reading reduced expression from map

For Group 1, variable **x** is changing its state and for Group 2, variable **y** is changing its state. So x in 1st group and y in 2nd group will be omitted. The resultant expression will contain yz, because of 1st group and xz' for 2nd. As this is SOP expression so both the terms will be joined through OR operator. Resultant reduced function will be

F = yz + xz'

Example:

F (w,x,y,z) = ∑ (5,7,13,15)

yz / wx	y'z'	y'z	yz	yz'
w'x'	0 (0)	0 (1)	0 (3)	0 (2)
w'x	0 (4)	1 (5)	1 (7)	0 (6)
wx	0 (12)	1 (13)	1 (15)	0 (14)
wx'	0 (8)	0 (9)	0 (11)	0 (10)

F (w,x,y,z) = xz

Now let's take an example where the function is represented using Truth table, instead of algebraic representation

x	y	z	f
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Using minterm designation we will represent the function in K map.

yz / *x*	*y'z'*	*y'z*	*yz*	*yz'*
x'	0	1	0	0
x	0	1	1	1

f (x,y,z) = y'z + xy

So far we were working with SOP expression/function. However, it is possible to reduce POS function also. Although we know k-map naturally reduces the expression is SOP form. In POS reduction, the approach is much the same except for, instead of minterms, we work with maxterms. We know a maxterm results into zero.

Now if we look at drawing of k-map it will be following for three variable

BC / A	B+C	B+C'	B'+C'	B'+C
A	M_0	M_1	M_3	M_2
A'	M_4	M_5	M_7	M_6

OR

C / AB	C	C'
A+B	M_0	M_1
A+B'	M_2	M_3
A'+B'	M_6	M_7
A'+B	M_4	M_5

Similarly a four variable map may be created.

Next is representing the Boolean expression/function in K- map. Representation remains same except, now we represent 0 instead of 1.

For reduction, 0's from the k-map will be grouped in same fashion as 1's were grouped, and reduced expression will be represented using maxterms.

Example:

f(w, x, y, z) = II (10, 11, 14, 15)

Using maxterm numbers, we will represent the function, i.e. 0 will be represented in map. For reduced POS expression 0's will be grouped

yz / *wx*	*y'z'*	*y'z*	*yz*	*yz'*
w'x'	1	1	1	1
w'x	1	1	1	1
wx	1	1	0	0
wx'	1	1	0	0

f(w, x, y, z) = w'+y'

f (a, b, c) = a'b'c + abc' +abc is an Sop expression, whose POS reduced form is denied.

bc / *a*	*b+c*	*b+c'*	*b'+c'*	*b'+c*
a	0	0	0	1
a'	0	0	1	1

As we are working for POS, 0's will be grouped and every group will result into sumterm.

f(a,b,c) = (a+c').b

LET'S REVISE

- A Boolean function can be expressed algebraically from a given truth table by forming a minterm and then taking the OR of all those terms.
- **Minterm:** An n variable minterm is a product term with n literals resulting into 1.
- **Maxterm:** An n variable maxterm is a sum term with n literals resulting into 0.
- A sum-of-product expression is logical OR of two or more AND terms
- A product-of-sum is logical AND of two or more OR terms
- If each term in SOP / POS form contains all the literals, then it is canonical form of expression
- To convert from one canonical form to another, interchange the symbol and list those numbers missing from the original form.
- The Karnaugh map (K-map) provides a systematic way of simplifying Boolean algebra expressions.
- For minimizing a given expression in SOP form, after filling the k map look for combination of one's. Combine theses one's in such a way that the expression is minimum.
- For minimizing expression in POS form we mark zeros, from the truth table, in the map. Combine zeros in such a way that the expression is minimum.
- **Sum Term:** is a single literal or the logical sum of two or more literals.
- **Product term:** is a single literal or the logical product of two or more literals.

EXERCISE

1. Determine the values of A, B, C and D that make the product term A'BC'D equal to 1
   i) A = 0, B = 1, C = 0, D = 1
   ii) A = 0, B = 0, C = 0, D = 1
   iii) A = 1, B = 1, C = 1, D = 1
   iv) A = 0, B = 0, C = 1, D = 0
2. The binary value of 1010 is converted to the product term A'B'C'D
   i) True
   ii) False
3. Which of the following expression is in SOP form?
   i) (A+B) (C+D)
   ii) AB (CD)
   iii) (A) B (CD)
   iv) AB+CD
4. A truth table for SOP expression ABC'+ AB'C + A'B'C has how many input combinations?
   i) 1
   ii) 2
   iii) 4
   iv) 8
5. POS equivalent of ABC+ AB'C'+AB'C+ABC'+A'B'C will be
   i) (A'+B'C') (A'+B+C') (A'+B+C)
   ii) (A'+B'+C') (A+B'+C) (A+B'+C)
   iii) (A+B+C) (A+B'+C) (A+B'+C')
   iv) (A+B+C) (A'+B+C') (A+B'+C)
6. Converting the Boolean expression LM+M (NO+PQ) to SOP form we get _____
   i) LM +MNOPQ
   ii) L+MNQ+MPQ

iii) LM+ M+NO+MPQ

iv) LM+MNO+MPQ

7. State whether AC+ABC = AC is

   i) True or

   ii) False

8. A student makes a mistake somewhere in the process of simplifying the following Boolean expression:

   i) ab + a (b+c)

   = ab+ab+c

   = ab+c

   Determine, where the mistake was made, and what proper sequence of steps should be used to simplify the original expression.

   ii) (x'y+z)'

   = (x'y)'.z'

   = (x')'+y'.z'

   = x+y'.z'

   Determine what the mistake is?

9. When grouping cells within k-map, the cells must be combined in groups of _____

   i) 2s

   ii) 1, 2, 4, 8, etc

   iii) 4s

   iv) 3s

10. Mapping the SOP expression A'B'C' + A'BC' + A'BC +ABC' we get _____

    i)

	C'	C
AB		
A'B'	1	1
A'B	1	
AB'		1

ii)

	A	B
AB	1	
A'B'	1	
A'B		1
AB'	1	1

iii)

	C'	C
AB		1
A'B'		
A'B	1	1
AB'	1	1

iv)

	C'	C
AB	1	1
A'B'		
A'B	1	
AB'		1

v)

	00	01	11	10
00	0	0	0	0
01	0	1	1	0
11	0	1	1	0
10	0	0	0	0

vi)

	C'D'	C'D	CD	CD'
A'B'	0	0	0	0
A'B	0	0	0	0
AB	1	1	1	1
AB'	0	0	0	0

11. If you look at the following k-map, you should notice that only two of the input variables- A, B, C, D change their state, in the marked group. The other two variables hold the same value '1'. Identify which variable change, and which stay the same:

	00	01	11	10
00	0	0	0	0
01	0	1	1	0
11	0	1	1	0
10	0	0	0	0

	C'D	CD	CD'
A'B'	0	0	0	0
A'B	0	1	1	0
A B	0	1	1	0
AB'	0	0	0	0

12. Give truth table for
    i) Z = x'+y'+z
    ii) Z = x(y+xz+x')
13. Use Boolean algebra to find the most simplified SOP expression for F=ABD+CD+ ACD+ABC+ABCD
    i) F = ABD+ABC+CD
    ii) F= CD+AD
    iii) F = BC+AB
    iv) F= AC+AD
14. From the truth table below, determine SOP and POS expression

A	B	C	Output X
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

15. Specify which axiom(s)/theorem(s) are being used in the following Boolean reductions:
    i) x'y' + x'y'z = x'y'
    ii) 1+A = 1
    iii) D+CD =D
    iv) a'.a'=a'
    v) (bc)' + bc = 1
    vi) xyz+zx = xz
    viii) ca'b' + ab = ab + c
16. Construct a truth table for the following functions and from the truth table obtain an expression for the inverse function
    i) F1(A,B,C) = A + BC'
    ii) F2(A,B,C) = AC + BC + AB'
17. Examine the given truth table and then write both SOP & POS boolean expressions describing the output.

A	B	C	Output
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

18. Write POS Boolean expressions for F (a, b) = ab'+a'b. Show through Boolean algebra reduction that the SOP & POS expressions are indeed equivalent to one another.
19. Minimize the following Boolean functions using algebraic method
    Z = f(a, b, c) = a'b'c' +a'b + abc' +ac
    Z = f (a,b,c) = a'b +bc' +bc+ ab'c'
    Z = f (a,b,c) = a'b'c' +
20. Reduce the following Boolean expression using k-map
    i) F (a, b, c) = a'b'c'+ a'bc'+ a 'bc' + a'bc + ab'c' + abc' to SOP form
    ii) F(w,x,y,z) = (w + x) (x + z') (w' + y' + z) to SOP form

iii)

A	B	C	F1
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

To SOP & POS form

ii)

A	B	C	D	F2
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

To SOP & POS Form

22. Obtain the minterm canonical form of the Boolean expression by algebraic method

i) xyz + xy + x'(yz' + y'z)

ii) ab + abc + a'b + ab'c

Chapter-3: Application of Boolean Logic

Learning Objective:

At the end of the chapter the students will:

- Learn about gates that process logic signals.
- Understand basic terminology, type of logic gates
- Learn about logic circuits and Boolean expression
- Learn how to design small circuits
- Learn to determine output of the gate(s) / circuit for the given input values.
- Learn about universality of NAND & NOR gate
- Explore the application of Boolean algebra in the design of electronic circuits.

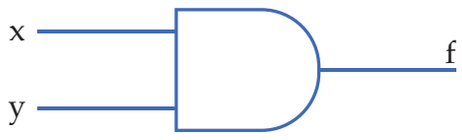
Boolean Expression through Logic gates

We know that a Boolean function is an algebraic expression formed with Boolean variables, operator OR, AND and NOT, parenthesis and equal to sign. Any Boolean function can be transformed in a straight forward manner from an algebraic expression into logic circuit diagram. This logic circuit diagram can be composed of basic gates AND, OR and NOT or advance gates, NAND and NOR, etc.

Logic circuits that perform logical operations such as AND, OR and NOT are called gates. A gate is block of hardware that produces a logic 0 or a logic 1 output, in response to the binary signal(s) applied to it as input.

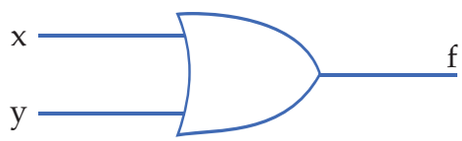
Let's look at the symbols used to represent digital logic gates

AND $f(x, y) = x.y$



For logical multiplication operation

OR $f(x, y) = x+y$

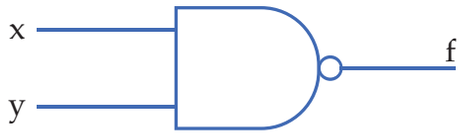


For logical addition operation

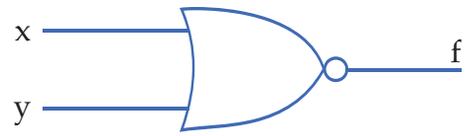
NOT $f(x) = x'$

For Complementation

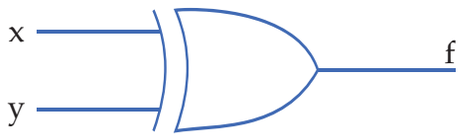
NAND $f(x,y) = (x.y)'$



NOR $f(x,y) = (x+y)'$



XOR $f(x,y) = x \oplus y = x'y + xy'$

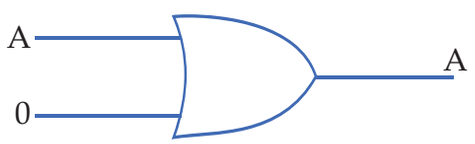


All the gates, except NOT, have two inputs represented on LHS and one output - on the RHS. Gates can have more than two inputs also. As the logic circuit is a way of representing Boolean expression, let's represent, axioms and theorems in this form.

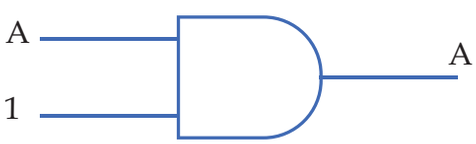
When a Boolean Function is implemented with logic gates, each literal in the function becomes an input to a gate. That's why we minimize the function - to reduce the number of inputs to gate also to reduce the total number of gates in the circuit. Also, let's assume that literal in both the forms- normal and complemented is available. This will help us in drawing neat circuit diagrams.

1. Identity

i) $A + 0 = A$



ii) $A . 1 = A$



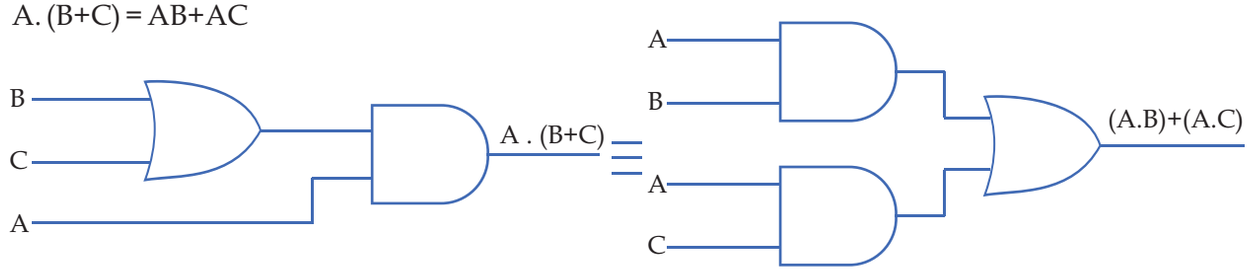
2. Commutative

i) A+B = B+A

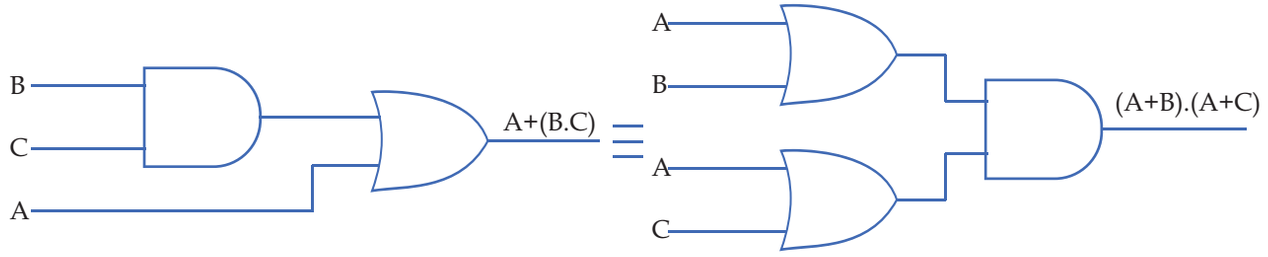
ii) A . B = B . A

3. Distributive

i) A. (B+C) = AB+AC

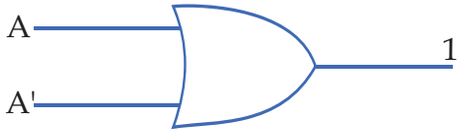


ii) A+ (B.C) = (A+B). (A+C)

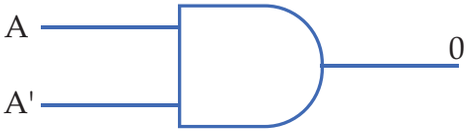


4. Complement

i) A+A' = 1

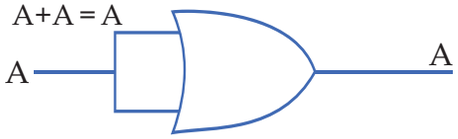


ii) A.A' = 0

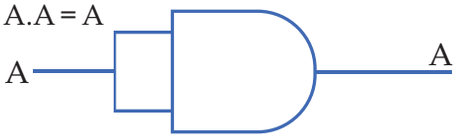


5. Indempotency

i) A+A = A

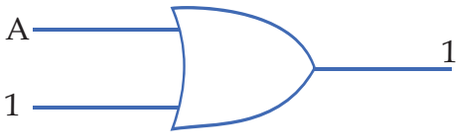


ii) A.A = A

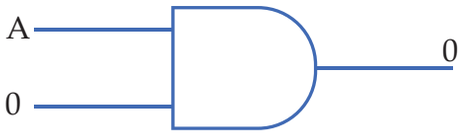


6. Null Element

i) A+1 = 1



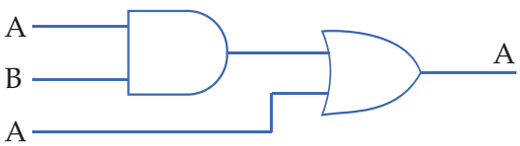
ii) A.0 = 0



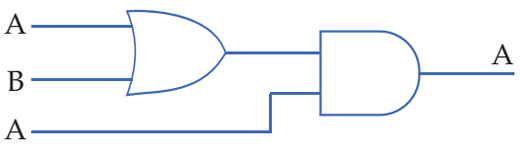
7. Involution (A')' = A

8. Absorption

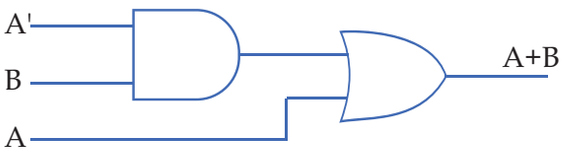
First law: i) A+ (AB) = A



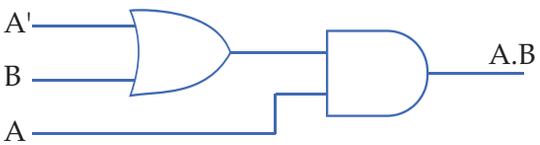
ii) A.(A+B) = A



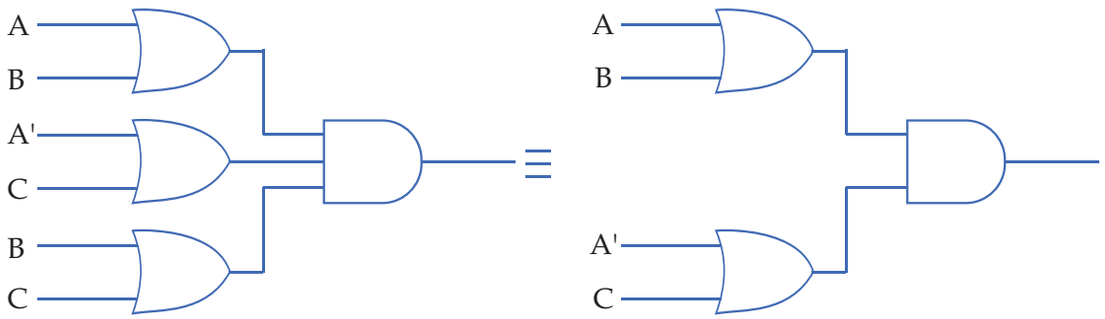
Second law: i) A+A'B = A+B



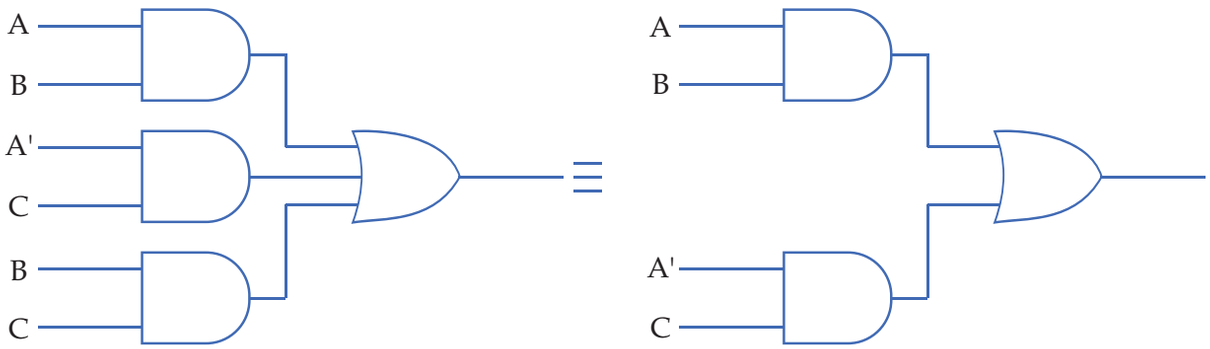
ii) A.(A'+B) = A.B



Third law: i) (A + B) (A' + C) (B + C) = (A + B) (A' + C)

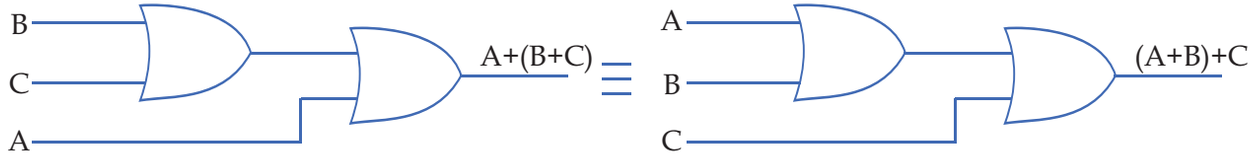


ii) (A . B) + (A' . C) + (B . C) = A . B + A' . C

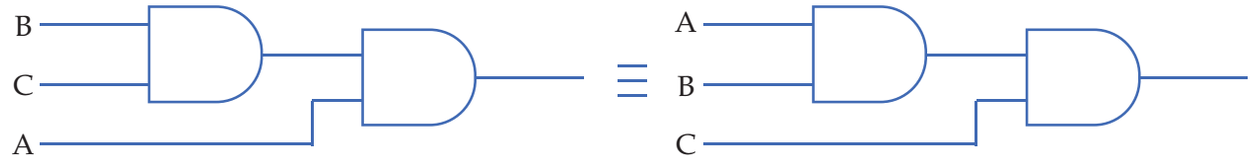


9. **Associative**

i) A + (B+C) = (A+B) + C



ii) A (B.C) = (A.B) C



10. **DeMorgan's Theorem**

i) (A+B)' = A'.B'

(NOR) **(Negative AND)**

ii) (A.B)' = A'+B'

(NAND) **(Negative OR)**

We have already verified these theorems using truth table and algebraic method in previous chapter.

Any Boolean function can be associated with a logic circuit, in which the inputs and outputs, represent the statement of Boolean algebra. Whatever is the complexity of function, they all can be constructed using three basic gates. For implementation of a Boolean function in logic circuit form:

The function will either be in SOP form or POS form. The SOP (Sum-of- Product) form is implemented in the following manner:

i) Each AND term is represented by AND gate (one gate for each AND term)

ii) Output of each AND gate is connected as input to single OR gate.

POS (Product-of-Sums) is implemented as

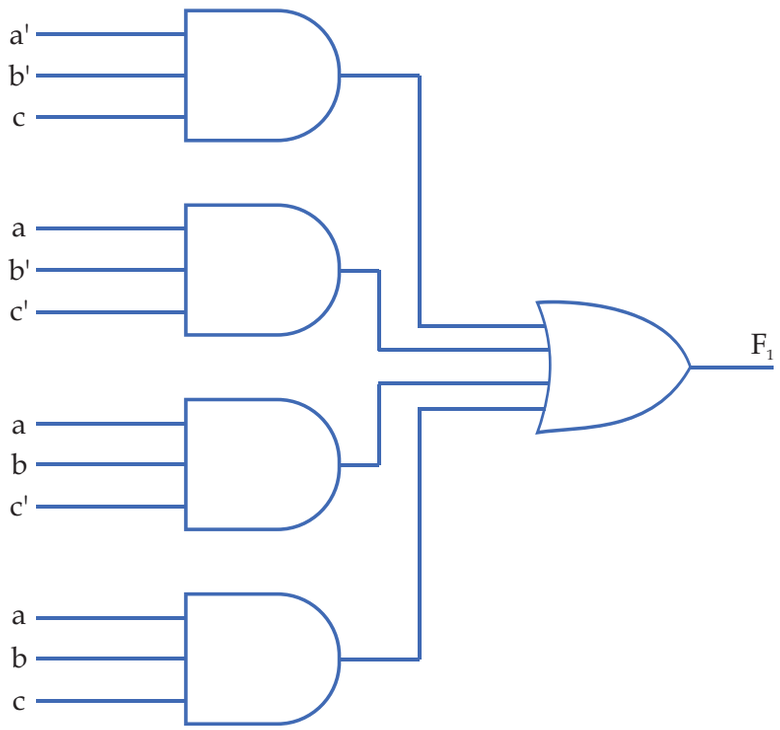
i) Each OR term is represented using an OR gate.

ii) Output of all the OR gate(s) are connected as input to single AND gate.

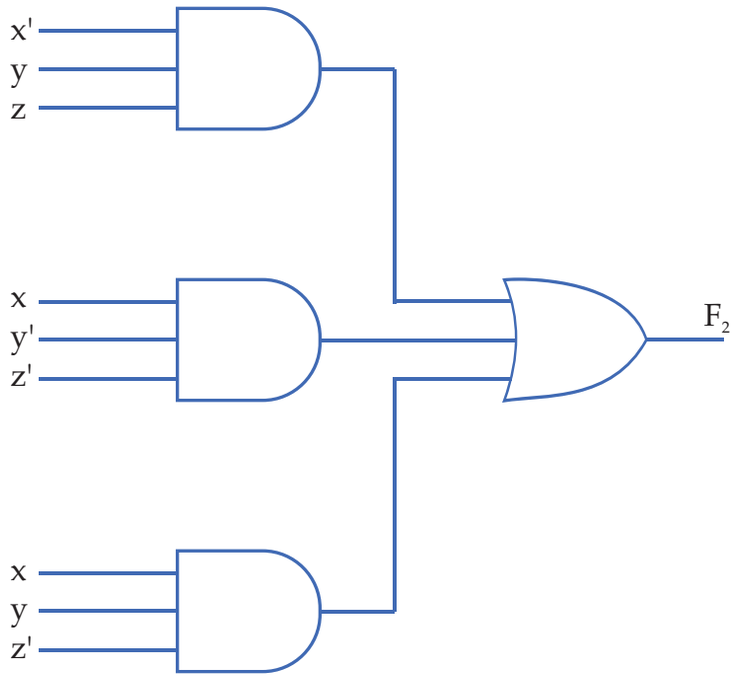
In both the representations, *it is assumed that both normal and complemented inputs, for all the variables, are available. So inverters are not needed.* Also in both the types of representation, two levels of gates are used. In SOP form AND gates are connected to OR gate and in POS form OR gates are connected to AND gates. Thus the implementation is known as two-level implementation.

Let's illustrate some Boolean functions from previous chapter in circuit form.

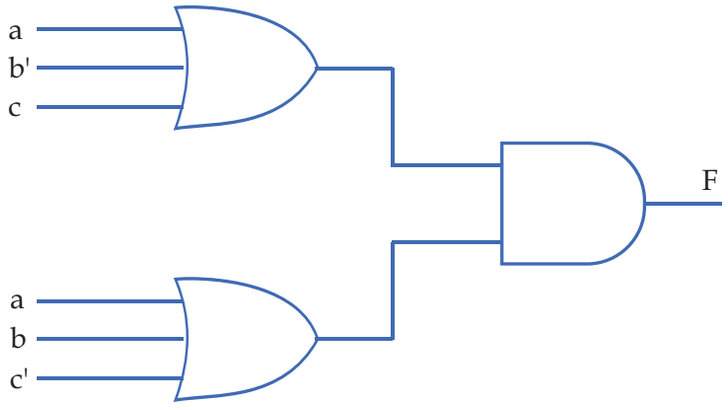
$F_1 = a'b'c + ab'c' + abc' + abc$



$F_2 = x'yz + xy'z' + xyz'$



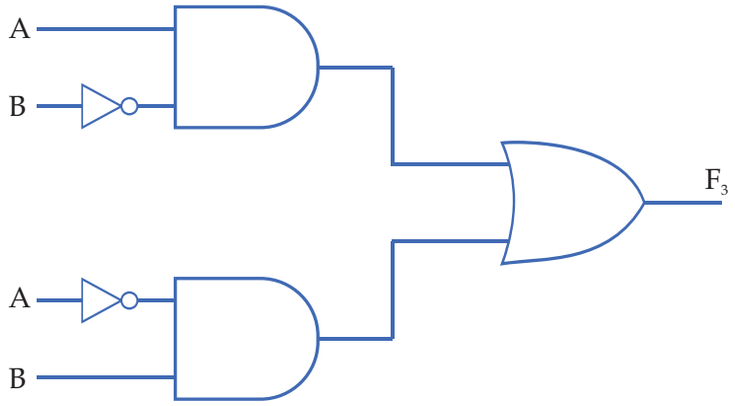
$F_3(a,b,c) = (a+b'+c) . (a+b+c')$



How to get a Boolean expression for a Logic Circuit?

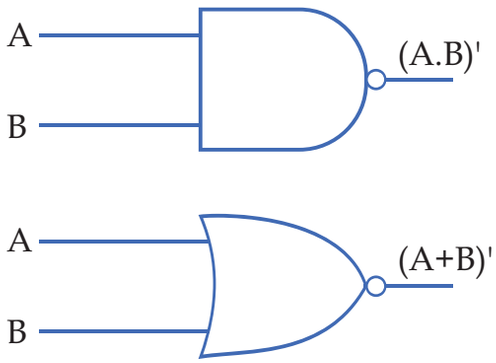
To derive the Boolean expression for a given logic circuit begin at the left most inputs and work towards the final output, by writing the expression for each gate.

For example:



- The expression for the left-most first gate will be B'
- This output along with A becomes input to AND gate so its output will be A.B'
- Similarly, working on 2nd set of gate from Left most side we will have A' and then A'.B
- Output of these two AND gates is input to OR gate, therefore the expression for the OR gate will be A.B'+ A'.B which is the final output expression for the entire circuit

The circuits made, till now have one type of gate(s) AND / OR at first level and another type of gate OR / AND at second level. As these are simple circuits, but in a complex circuit we might have a function which uses both type of gate(s) at first level. Building a really big complex digital system using many types of gates will make the job tedious. We need to have a gate, which can replace the function of all other gates, hence can be used to implement any digital circuit, such gates are also known as universal gates. NAND and NOR are such universal gates. NAND for 2 input is (A.B)' and NOR for 2 input is (A+B)'



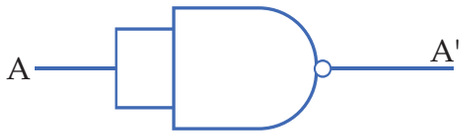
An SOP expression, represented by AND gate(s) at first level and OR gate at second level, can naturally be implemented by only NAND gate(s). Similarly a POS expression, which uses OR gates at first level and AND gate at second level can naturally be implemented through NOR gate(s) only.

Now let's see how each logic gate can be created with universal gate, so that entire circuit can be implemented through single type of gate.

Realization of all functions using NAND gate

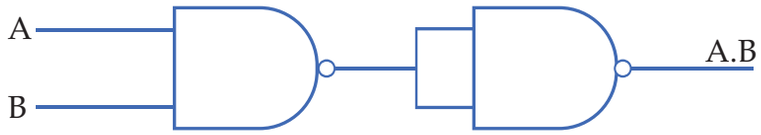
NOT A' = (A.A)'

i.e. A nand A



AND A.B = ((A.B)')'

i.e. (A nand B) nand (A nand B)



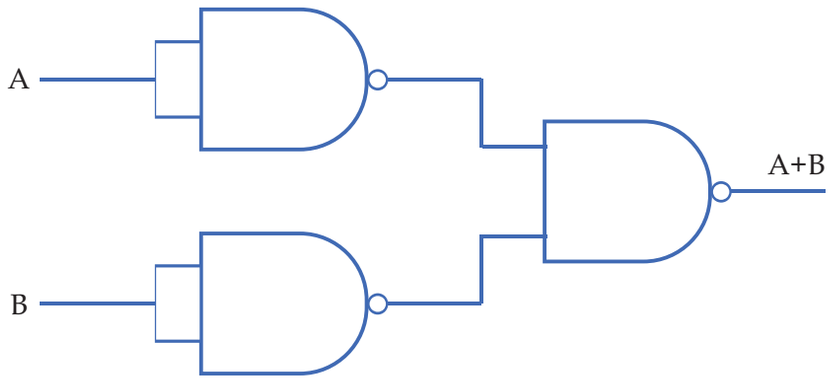
OR = A+B

= ((A+B)')'

= (A'.B')'

=((A.A)'. (B.B)')'

i.e. (A nand A) nand (B nand B)

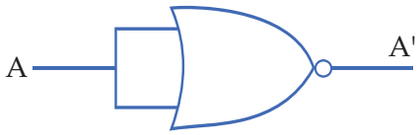


Realization of logic functions using NOR gate

NOT = (A)'

= (A+A)'

i.e. A NOR A



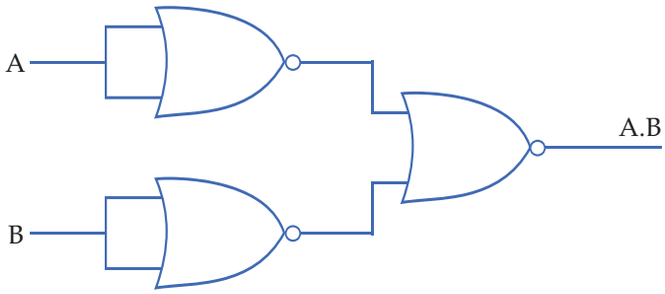
AND = A.B

= ((A.B)')'

= (A'+B')'

= ((A nor A) + (B nor B))'

i.e. (A nor A) nor (B nor B)

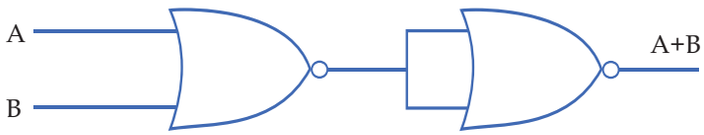


OR = A+B

= ((A+B)')'

= (A nor B)'

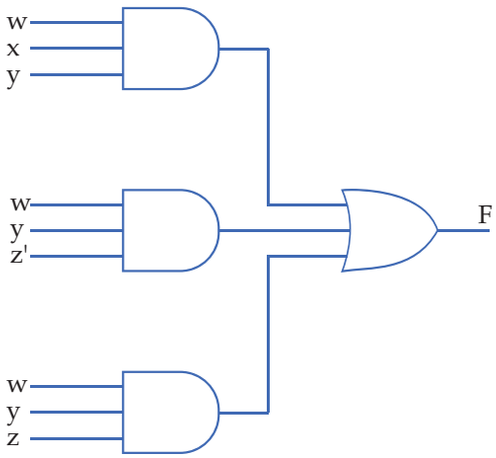
i.e. (A nor B) nor (A nor B)



Let's take some example of implementing complete circuit through universal gate

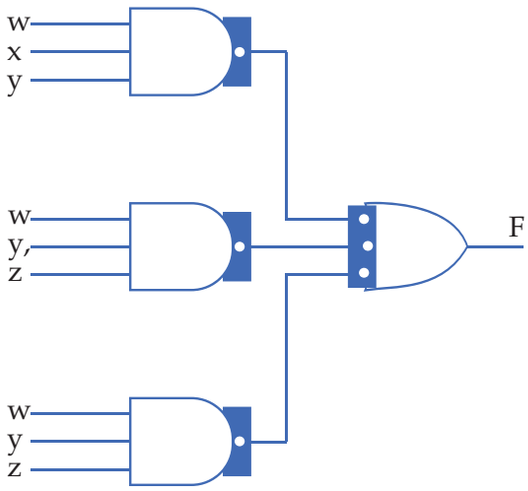
F (w,x,y,z) = wxy + wyz' + wyz

This is an SOP expression and will be represented by AND OR gate

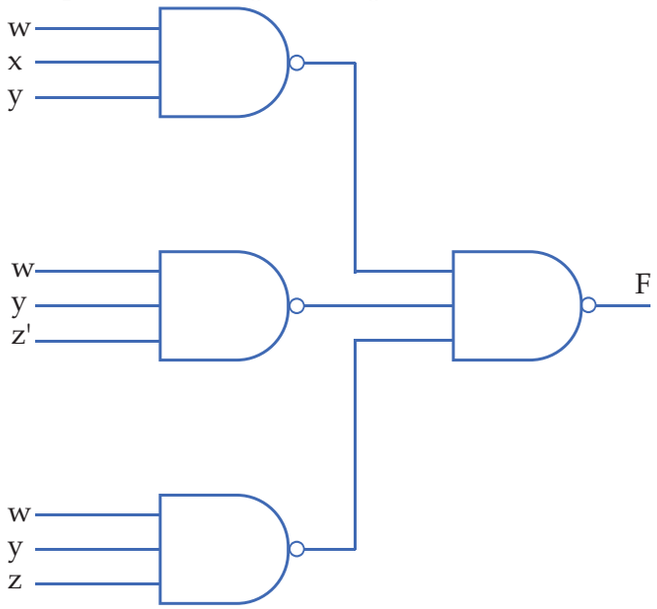


Let's apply De Morgan's theorem to implement the above circuit with minimum number of NAND gates, because if each gate is replaced by universal gate using above solution we will end up using a large number of gates and increase the size of circuit. You may try this for a Boolean function, which can then be compared by the solution given below.

If we introduce bubbles at AND gate(s) output and OR gate(s) input, the resultant will remain same as A = (A')'. Applying this, the above circuit will become



All the gates at first level are NAND gates. If you look at the second level gate it is (A'+B'+C') assuming A, B & C are output from 1st, 2nd, and 3rd gate respectively. (A'+B'+C') = (A.B.C)' using DeMorgan's Theorem. So now the complete circuit may be represented using NAND gate as:

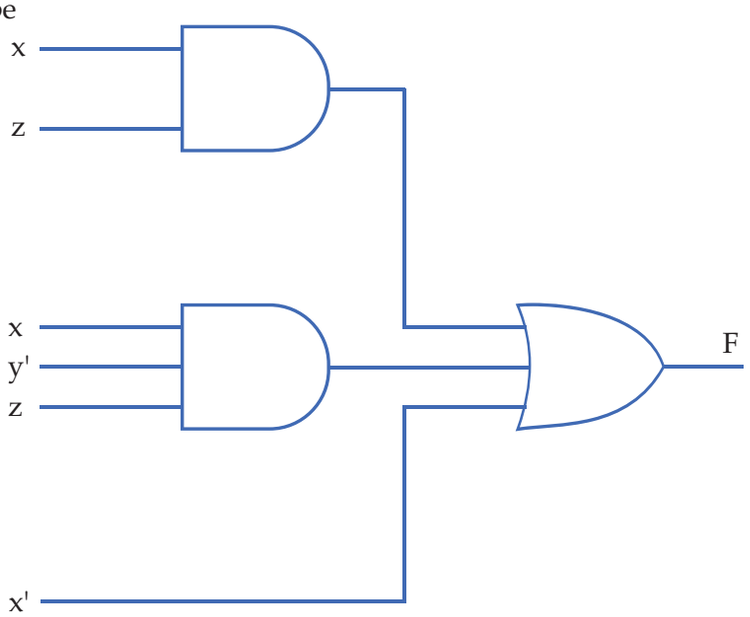


So the trick is replace AND gate at 1st level and OR gate at 2nd level by NAND gate.

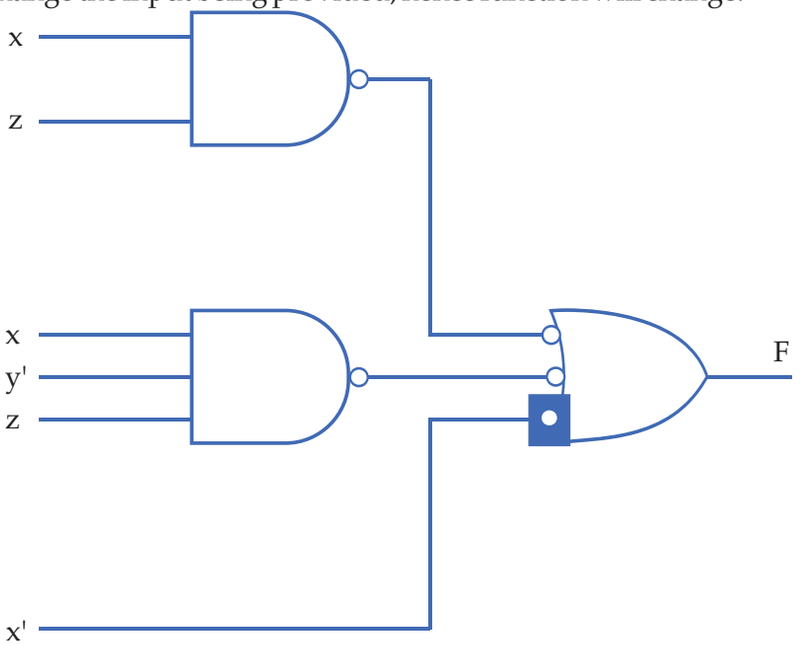
Let's take another kind of example

F (x,y,z) = xz + xy'z + x'

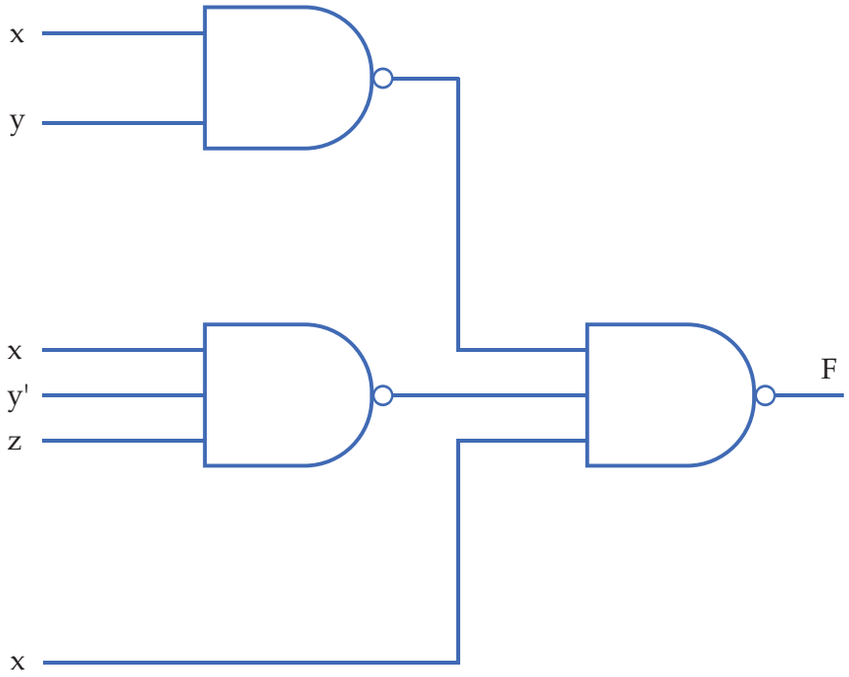
Its **AND** → **OR** diagram will be



If we replace all gates through NAND only, then the single input going to second level gate (OR) will get complemented, which will change the input being provided, hence function will change.



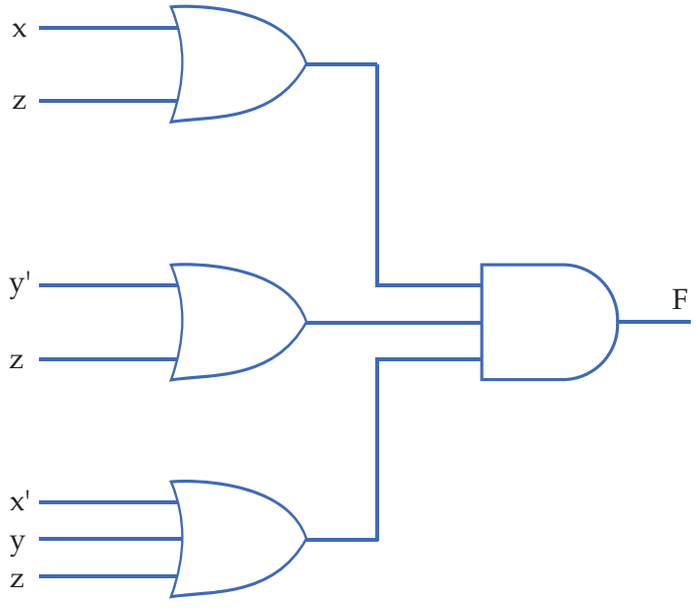
This can be handled by sending an inverted input to 2nd level gate whenever there is single input which passes directly to second level gate. So the correct solution will be



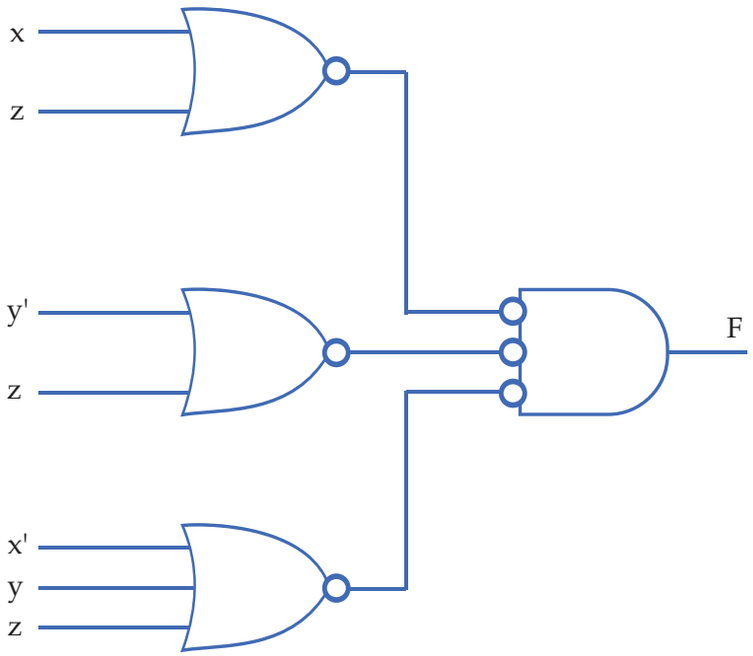
Now let's implement a complete circuit using NOR gate(s)

F (x,y,z) = (x+z) (y'+z) (x'+y+z)

Its natural representation is **OR → AND**



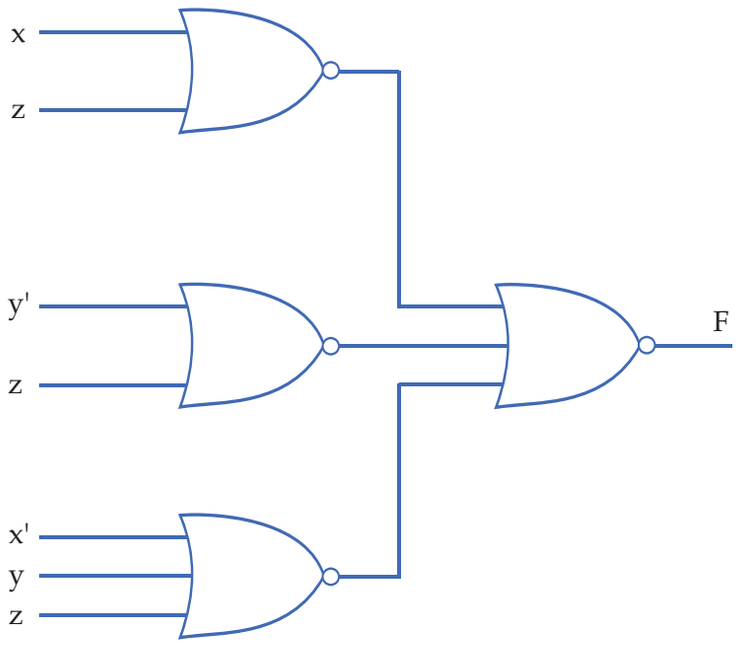
Again using the same concept, we have



bubbles to be highligted

Which is OK as the output of 1st level gate is complemented and it is again complemented before providing to next level gate.

The second level gate is equivalent to NOR gate as per DeMorgan's theorem. So the circuit implementation using NOR gate only will be



In this implementation also, if an input is directly moving to second level gate in normal OR→AND representation then, in NOR implementation the input will be complemented before moving to 2nd level gate.

Now we know SOP expression can be implemented using AND → OR circuit or universal NAND gate. POS expression can be implemented using OR→AND or universal NOR gate. Representing POS using only NAND gates will require conversion of POS to SOP form and then implement same using only NAND only gates. We have already learnt the way of converting expression from one form to another.

Let's take an example:

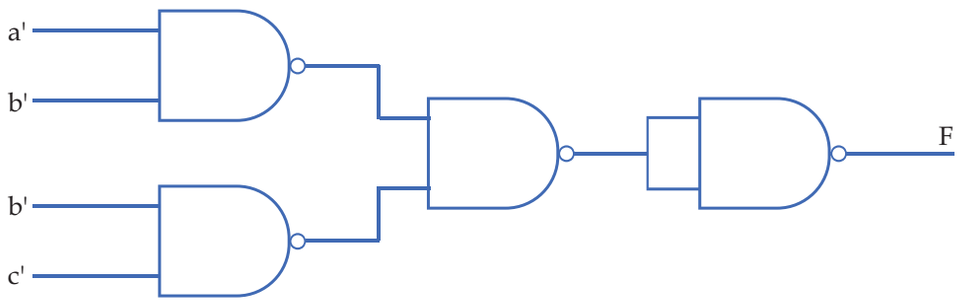
F (a,b,c) = (a+b).(b+c)

= [((a+b) . (b+c))']'

= [(a+b)' + (b+c)']'

= [(a'.b') + (b'.c')]'

So the circuit will be



For representation of SOP through only NOR gates, we can proceed in similar manner.

One of the most recent use of Boolaen logic is, Internet search engine and Database search. You have applied operator AND, OR and NOT in SQL queries and already know it's usage. Let's learn the usage of Boolean operators for searching Internet.

We know, Internet is a vast computer database and the proper use of Boolean operators - AND, OR and NOT would increase the effectiveness of web search. These operators can act as effective filters for finding just the information one need from Internet. So most of the search engine support some form of Boolean query using Boolean operators (check the help section of your favorite search engine supported by it). An engine is a search program, that allows us to search its data base.

These operators can be used in two ways:

i) You can use the words - AND, OR, NOT

ii) You can use symbols (math equivalents) - **+** for AND, **-** for NOT, OR is the default setting in many search engine.

Note: When you are using symbols, don't add space between the symbols and your query.

OR operator is used to broaden the search. It's interpreted as "at least one is required, more than one or all can be returned". So when search strings (keyword) are joined using OR, the resultant will include - any, some or all of the keywords used in the statement. The operator will ensure that you do not miss anything valuable.

AND operator is used to narrow the search. It is interpreted as "all is required ". So when search string i.e. keywords are joined using AND, the resultant will include the documents containing all the keywords.

NOT operator will again narrow down the search, this time by excluding the search string i.e. keyword.

LET'S REVISE

- Gate is an electronic system that performs a logical operation on a set of input signal(s). They are the building blocks of IC.
- An SOP expression when implemented as circuit - takes the output of one or more AND gates and OR's them together to create the final output.
- An POS expression when implemented as circuit - takes the output of one or more OR gates and AND's them together to create the final output.
- Universal gates are the ones which can be used for implementing any gate like AND, OR and NOT, or any combination of these basic gates; NAND and NOR gates are universal gates.
- Implementation of a SOP expression using NAND gates only
  1) All 1st level AND gates can be replaced by one NAND gate each.
  2) The output of all 1st level NAND gate is fed into another NAND gate.

  This will realize the SOP expression
  3) If there is any single literal in expression, feed its complement directly to 2nd level NAND gate.

  Similarly POS using NOR gate can be implemented by replacing NAND by NOR gate.
- Implementation of POS / SOP expression using NAND / NOR gates only.
  1) All literals in the first level gate will be fed in their complemented form.
  2) Add an extra NAND / NOR gate after 2nd level gate to get the resultant output.

EXERCISE

1. One of the DeMorgan's theorems states that (x=y)' = x'y'. Simply stated, this means that logically there is no difference between:
   i) A NOR and an AND gate with inverted inputs
   ii) A NAND and an OR gate with inverted inputs
   iii) An AND an a NOR gate with inverted inputs
   iv) A NOR and a NAND gate with inverted inputs

2. An AND gate with 'bubbles' on its input performs the same function as a(n) _____ gate
   i) NOT
   ii) OR
   iii) NOR
   iv) NAND

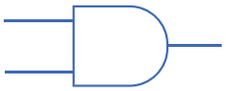
3. How many gates would be required to implement the following Boolean expression before simplification of X4 + X (X+Z) + Y (X+Z)
   i) 1
   ii) 2
   iii) 4
   iv) 5

4. The NAND and NOR gates are referred to as universal gates because either,
   i) Can be found in almost all digital circuit
   ii) Can be used to build all other types of gate.
   iii) Are used in all countries of the world
   iv) Were the first gates to be integrated

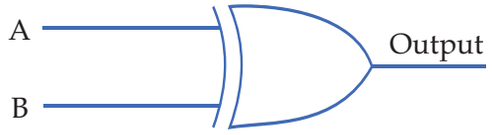
5. The boolean expression x=A'+B'+C' is logically equivalent to which single gate?
   i) NAND
   ii) NOR
   iii) AND
   iv) OR

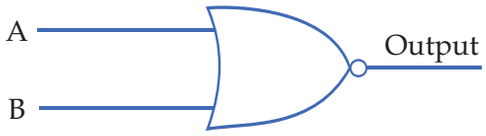
6. The symbol shown below is for a 2-input NAND gate
   i) True
   ii) False

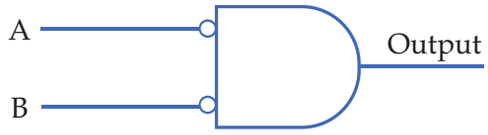
7. By applying DeMorgan's theorem to a NOR gate two identical truth tables can be produced
   i) True
   ii) False

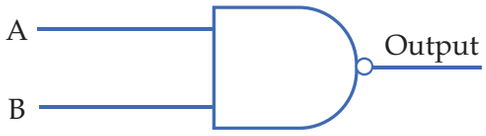


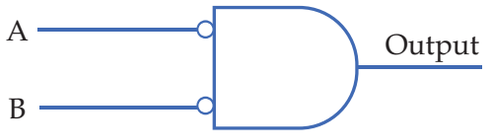
8. Give the truth table of following gate

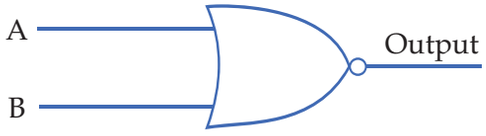
i)


ii)


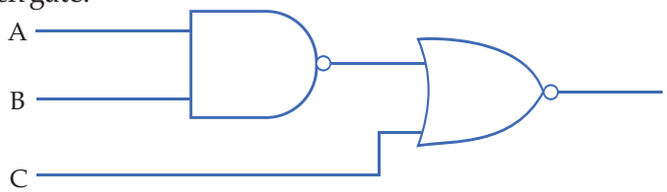
iii)


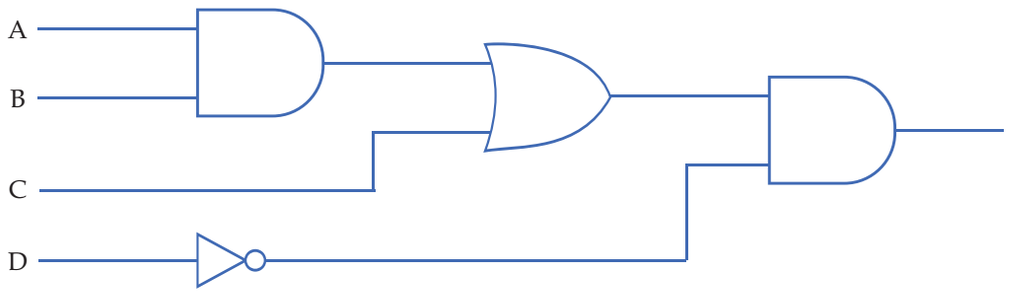
iv)


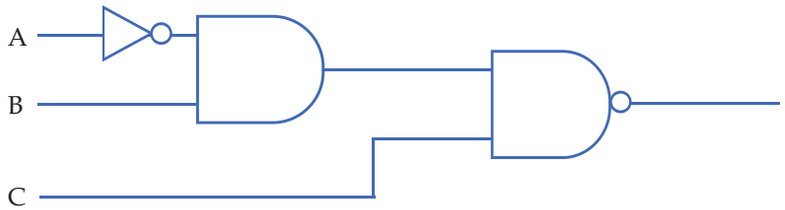
v)


vi)


9. Convert the following logic gate circuit into a Boolean expression, writing Boolean sub-expression next to each gate.

i)


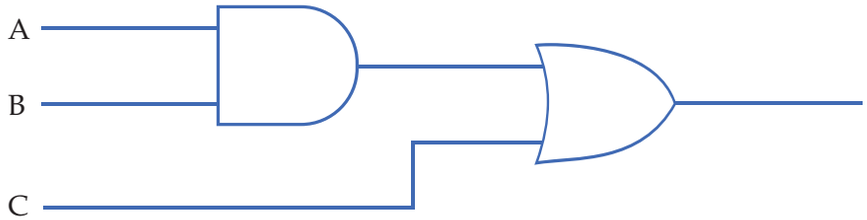
ii)


iii)


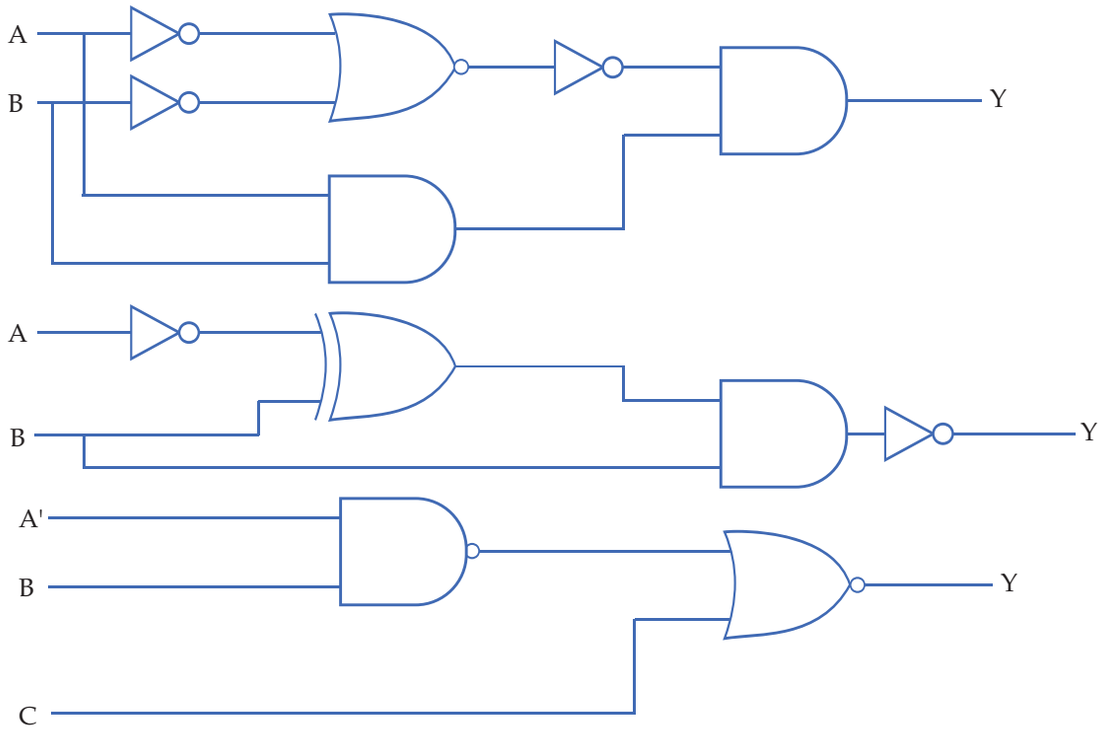
10. Draw the circuit diagram for the following using AND, OR and NOT gate(s)
    i) F(a,b,c,) = a + bc'
    ii) F(x,y,z,) = xy + yz + xy'
    iii) f(w,x,y,z) = yz' + wxy' + wxz' + wx'z

iv) f(x,y,z) = (x + y). (y + z) . (z + x)

11. Convert the following circuit diagram using NAND gate(s) only. Do not simplify the expression.



12. Draw truth table and write Boolean function for following circuit



Remove fig no.

13. Draw the logic gate diagram for following expression using NOR gates only
    i) f(a,b,c) = ?(2,4,5,7,8)
    ii) f(x,y,z) = ?(0,1,3,4)
    iii) f(w,x,y,z) = ?(2,4,5,6,9)
14. Which gates are known as universal gate? Why?
15. Show how AND, OR Not gate functions can be implemented using only
    i) NAND gate
    ii) NOR gate

Unit-5: Communication Technologies

Chapter-1 : Networking Concepts - 1

Learning Objectives:

At the end of this chapter the students will be able to:

- Learn about a network and its need
- Understand the evolution of Networking, Internet , Interspace
- Learn about requirements of a network
- Understand various communication terminologies
  - Nodes (Workstations)
  - Server (Dedicated and Non Dedicated)
  - Network Interface Unit(NIU)
- Learn about commonly used switching techniques (Circuit and Packet)
- Understand various types of Networks (PAN , LAN , WAN , MAN)
- Know about various data communication terminologies
  - Channel
  - Bandwidth
  - Data Transfer Rate

Communication means to convey. In today's fast changing world, data needs to be transferred efficiently, frequently and speedily. To move data at a fast speed and that too with minimum data loss, networking plays an important role. Computer networks have made a major impact on the society as a whole. As we are moving ahead in the 21st century, the world seems to be converging. The merging of computers and communication technology has changed the very perspective of communication today. It has had a profound influence on the way the computer systems are organized and used today. The old model of a single computer serving all of the organization's computational needs has been replaced by one in which large number of separate but interconnected computers do the same job. These systems together form a computer network.

What is a network?

We often need peripheral devices and data to be shared among various computers. In fact, in your school's computer lab, you must have seen one printer which is connected to only one computer, serving to the needs of all the computers in the lab. How does this happen? This happens because all your lab's computers and peripherals are forming a network. They are interconnected with each other enabling you to send and receive data from one computer to another. Hence it can be said that two computers are interconnected if they are able to exchange information.

A network is any collection of independent computers that communicate with one another over a shared network medium. In simple terms, a computer network is a collection of two or more computers linked together for the purpose of sharing information and resources. When these computers are joined in a network, people can share files and peripherals such as modems, printers, backup drives, or CD-ROM drives. Each computer on the network is called a node and hence, a network is a series of points or nodes interconnected by communication paths (transmission media). A network can be as small and simple as two computers that share a printer or as complex as the world's largest network, the Internet. When networks at multiple locations are connected using services available from phone companies, people can send e-mail, share links to the global Internet, or conduct video conferences in real time with other remote users. As companies rely on applications like electronic mail and database management for core business operations, computer networking becomes increasingly more important.

Need for networking

Why has networking evolved as an indispensible part of technology today? To find answer to this question, let us have a look at the advantages of networking:

1. Resource sharing - files and peripherals

i) Sharing of files and software

A network enables users to share data files with each other. For e.g. different departments of an organization may be seperated physically, being at distant places, but their data could be stored on a central computer which can be accessed by computers located in different departments. In this way latest data can be made available at all times to all users. Files and folders can be backed up to local or remote shares. Software can be installed centrally rather than on each machine which proves to be much cheaper than buying licenses for every machine.

ii) Sharing Peripherals

Laser printers and large storage media are quite expensive. Networks enable us to share such resources and hence reduce the operational cost of any organization. For e.g. a company with about fifty computers can share resources such as printers, scanners, hard disks etc, thereby reducing the cost considerably. Even fax systems can be integrated within a network. Audio and video content can also be streamed to multiple devices.

iii) Sharing storage

On a network, one can access data from any machine. Hence storage can be distributed and thus database load can be shared on the network. This even proves to be cost effective. A file can even have copies on two or three machines.

2. Improving communication

A computer network can provide a powerful, fast and reliable communication medium among the users of various computers on the network. Using a network, it is easy for two or more people, of say

different departments or different branches to prepare a presentation together in spite of being located in different cities. In fact the best example in this context can be of the use of internet (discussed later in the chapter). With the help of internet we can communicate efficiently and easily via email, instant messaging, chat rooms, telephone, video telephone calls, and video conferencing.

3. Access to remote database

This is another major use of network. It is easy for an average person to access any remote database, say for example airline reservations and thereby book tickets. Likewise databases of trains, online universities, hotels etc can be accessed as per the requirement. Remote-control/access programs can be used to troubleshoot problems or show new users how to perform a task.

Like two sides of a coin, networking also has some disadvantages associated with it. The disadvantages are as follows:

1. Threat to data

A computer network may be used by unauthorized users to steal or corrupt the data and even to deploy computer viruses or computer worms on the network. File security has to be taken care of especially if connected to WANs.

2. Difficult to set up

The systems on a network are more sophisticated and complex to run. Sometimes setting up a network, especially larger networks may turn out to be a difficult task. If systems are badly managed services can become unusable. In addition to this, larger networks may also be very costly to set up and maintain. Often a specialist may be needed to run and maintain the network.

Evolution of Networking

Networking started way back in 1969 with the development of the first network called the ARPANET. The U.S. department of defence sponsored a project named ARPANET (Advanced Research Projects Agency Network) whose goal was to connect computers at different universities and U.S. defence. Soon engineers, scientists, students and researchers who were part of this system began exchanging data and messages on it. Gradually they could play long distance games and also socialize with people. Hence ARPANET expanded rapidly. In mid 80s, the National Science Foundation created a new high capacity network called NSFnet which allowed only academic research on its network. So many private companies built their own networks, which were later interconnected along with ARPANET and NSFnet to form Internet - a network formed by linking two or more networks.

Internet

The Internet is a system of linked networks that are worldwide in scope and facilitate data communication services such as remote login, file transfer, electronic mail, the World Wide Web and newsgroups. The Internet is made up of many networks each run by a different companies and are interconnected at peering

points. It is really a network of networks spread across the globe, all of which are connected to each other. This super network is a glorified WAN in many respects. It connects many smaller networks together and allows all the computers to exchange information with each other through a common set of rules for communication. These rules are called protocols and the internet uses Transmission Control Protocol/Internet Protocol (TCP/IP). Programs such as web browsers, File Transfer Protocol (FTP) clients, and email clients are some of the most common ways through which the users work on the Internet.

With the meteoric rise in demand for connectivity, the Internet has become a communications highway for millions of users. The Internet was initially restricted to military and academic institutions, but now it is a full-fledged conduit for any and all forms of information and commerce. Internet websites now provide personal, educational, political and economic resources to every corner of the planet.

Inter space

It is a client/server software program that allows multiple users to communicate online with real time audio, video and text chat in dynamic 3D environments.

It provides the most advanced form of communication technology available today. It is a vision of what internet will become tomorrow. The users will be able to communicate in multiple ways and from multiple sources instantly.

Requirements of a Network

Every network includes:

- At least two computers - Server or Client workstation.
- Network Interface Cards (NIC)
- A connection medium, usually a wire or cable, although wireless communication between networked computers and peripherals is also possible.
- Network Operating system software, such as Microsoft Windows NT or 2000, Novell NetWare, Unix and Linux.

Network Terminologies

Before continuing our study on networks let us first learn about some terminologies commonly used in networking.

i) Nodes (Workstations)

A computer becomes a node (also called a workstation) as soon as it is attached to a network. Each user on a network works on a workstation. If there are no nodes there would be no network.

ii) Server

A computer that facilitates sharing of data, software and hardware resources on the network is known as the server. A network can have more than one server also. Each server has a unique name by which it is identified by all the nodes on the network. Servers can be of two types:

a) Dedicated and
b) Non dedicated servers

Dedicated Servers: These are generally used on big network installations where one computer is reserved for server's job. It helps all nodes access data, software and hardware resources. Since it does not double up as a workstation but only manages the network, so it is known as a dedicated server and such type of networks are called master- slave networks.

Non dedicated servers: In small networks, a workstation can double up as a server. These servers are known as non dedicated servers. The small networks using such a server are known as Peer to Peer networks.

Also on a network there may be several servers that allow workstations to share specific resources - for example a file server which takes care of files related requests, a printer server taking care of printing requirements and a modem server that helps group of users to use a modem.

iii) Network Interface Unit (NIU)

A network interface unit is a device that is attached to each of the workstations and the server which helps to establish communication between the server and workstations. As soon as a standalone computer becomes a workstation, it needs an interface to help establish connection with the network because without this the workstations will not be able to share network resources or communicate with each other. The NIC basically acts like an interpreter and is also known as Terminal Access Point (TAP) or Network Interface card(NIC).The NIC manufacturer assigns a unique physical address to each NIC card and this physical address is known as the MAC address.

Switching Techniques

Switching techniques are used to efficiently transmit data across the network. The two types of switching techniques are employed nowadays to provide communication between two computers on a network are: Circuit Switching and Packet Switching

Circuit Switching

Circuit switching is a technique in which a dedicated and complete physical connection is established between two nodes and through this dedicated communication channel, the nodes may communicate. The circuit guarantees the full bandwidth of the channel and remains connected for the duration of the communication session. Even if no communication is taking place in a dedicated circuit, that channel still remains unavailable to other users (idle channels).

The defining example of a circuit-switched network is the early analogue telephone network. When a call is made from one telephone to another, switches within the telephone exchange create a continuous wire circuit between the two telephones, for as long as the call lasts.

Packet Switching

Packet switching is a switching technique in which packets (discrete blocks of data of fixed size and of any content, type or structure) are routed between nodes over data links shared with other traffic. The term "packets" refers to the fact that the data stream from your computer is broken up into packets of about 200 bytes (on average), which are then sent out onto the network. Each packet contains a "header" with information necessary for routing the packet from source to destination. Each packet in a data stream is independent.

The main advantage of packet-switching is that the packets from many different sources can share a line, allowing for very efficient use of the communication medium. With current technology, packets are generally accepted onto the network on a first-come,

first-served basis. If the network becomes overloaded, packets are delayed or discarded ("dropped"). This method of data transmission became the fundamental networking technology behind the internet and most Local Area Networks.

Types of Networks:

A network may be a small group of interlinked computers to a chain of a few hundred computers of different types (for example personal computers, minicomputers , mainframes etc.). These computers may be localised or spread around the world. Thus networks vary in terms of their size and complexity. Various types of networks are discussed below:

PAN (Personal Area Network)

A Personal Area Network is a computer network organized around an individual person. Personal area networks typically involve a mobile computer, a cell phone and/or a handheld computing device such as a PDA. You can use these networks to transfer files including email and calendar appointments, digital photos and music.

Personal area networks can be constructed with cables or be wireless. USB and FireWire technologies often link together a wired PAN, while wireless PANs typically use bluetooth or sometimes infrared connections. Bluetooth PANs generally cover a range of less than 10 meters (about 30 feet). PANs can be viewed as a special type (or subset) of local area network (LAN) that supports one person instead of a group.

LAN (Local Area Network)

In a LAN, network devices are connected over a relatively short distance. They are generally privately owned networks within a single building or campus, of up to a few kilometres in size. LANs can be small, linking as few as three computers, but often link hundreds of computers used by thousands of people. This means that many users can share expensive devices, such as laser printers, as well as data on the LAN. Users can also use the LAN to communicate with each other, by sending mails or engaging in chat sessions.

Nowadays we also have WLAN (Wireless LAN) which is based on wireless network (covered in next chapter). One LAN can even be connected to other LANs over any distance via telephone lines and radio waves. However there is also a limit on the number of computers that can be attached to a single LAN. The development of standard networking protocols and media has resulted in worldwide proliferation of LANs throughout business and educational organizations.

MAN (Metropolitan Area Network)

This is basically a bigger version of LAN and normally uses similar technology. It might cover few buildings in a city and might either be private or public. This is a network which spans a physical area (in the range of 5 and 50 km diameter) that is larger than a LAN but smaller than a WAN. MANs are usually characterized by very high-speed connections using optical fibres or other digital media and provides up-link services to wide area networks (WANs) and the Internet. For example in a city, a MAN, which can support both data and voice might even be related to local cable television network.

The MAN, its communications links and equipment are generally owned by either a consortium of users or by a single network provider who sells the service to the users. Since MAN adopts technologies from both LAN and WAN to serve its purpose, it is also frequently used to provide a shared connection to other networks using a link to a WAN.

WAN (Wide Area Network)

As the term implies, WAN spans a large geographical area, often a country or a continent and uses various commercial and private communication lines to connect computers. Typically, a WAN combines multiple LANs that are geographically separated. This is accomplished by connecting the different LANs using services such as dedicated leased phone lines, dial-up phone lines, satellite links, high speed fibre optic cables and data packet carrier services. Wide area networking can be as simple as a modem and remote access server for employees to dial into, or it can be as complex as hundreds of branch offices globally linked using special routing protocols and filters to minimize the expense of sending data sent over vast distances.

Let us take an example of your local telephone exchange which is a part of WAN. The computers at each telephone exchange connect to other exchanges to allow you to talk to people all over the world. The internet is the largest WAN , spanning the entire earth.

Data Communication Terminologies

Before we proceed with our study of networks, let us learn about some data communication terminologies being used.

Channel: A communication channel is a medium that is used in the transmission of a message from one point to another. In simple terms we can say that it is a pathway over which data is transferred between remote devices. It may refer to the entire physical medium, such as a telephone line, optical fibre, coaxial

cable or twisted pair wire, or, it may refer to one of the several carrier frequencies transmitted simultaneously within the line.

Depending on their speed, we have three broad categories of communication channels - narrow band which is slow and used for telegraph lines and low speed terminals; voice band used for ordinary telephone communication and broad band which is fastest and is used for transmitting large volumes of data at high speeds.

Bandwidth: In electronic communication , bandwidth refers to the range of frequencies available for transmission of data. It is expressed as the difference in Hertz(Hz) between the highest frequency and the lowest frequency. For example , a typical voice signal has a bandwidth of approximately 3KHz. Wider the bandwidth of a communication system, greater is the capacity and hence greater is the amount of data that can be transmitted over a period of time.

In computer networking, bandwidth is often used as a synonym for data transfer rate about which you will read in the next subtopic.

Data Transfer rate

The data transfer rate (DTR) is the amount of data in digital form that is moved from one place to another in a given time on a network. As studied before, the greater the bandwidth of a given medium, the higher is the data transfer rate. This can also be referred to as throughput, although data transfer rate applies specifically to digital data streams. Data transfer rate is often measured in bits per second (bps), although the unit baud , which is one bit per second is also used. It is commonly used to measure how fast data is transferred from one location to another. For example, your ISP may offer an Internet connection with a maximum data transfer rate of 4Mbps.

LET'S REVISE

- **Network:** A collection of independent computers that communicate with one another over a shared network medium.
- **Node:** A computer attached to a network.
- **Server:** A computer that facilitates sharing of data, software and hardware resources on the network.
- **Network Interface Unit (NIU):** A device that helps to establish communication between the server and workstations.
- **Circuit switching:** A technique in which a dedicated and complete physical connection is established between two nodes for communication.
- **Packet switching:** A switching technique in which packets are routed between nodes over data links shared with other traffic.
- **Personal Area Network (PAN):** A computer network organized around an individual person.
- **Local Area Network (LAN):** A network in which the devices are connected over a relatively short distance.
- **Metropolitan Area Network (MAN):** A network which spans a physical area (in the range of 5 and 50 km diameter) that is larger than a LAN but smaller than a WAN.
- **Wide Area Network (WAN):** A network which spans a large geographical area, often a country or a continent.
- **Internet:** It is a network of networks spread across the globe, all of which are connected to each other.
- **Interspace:** A client/server software program that allows multiple users to communicate online with real time audio, video and text chat in dynamic 3D environments.
- **Channel:** A medium that is used in the transmission of a message from one point to another.
- **Bandwidth:** The range of frequencies available for transmission of data.

EXERCISE

1. Fill in the blanks:
   a. Two or more computers connected to each other for information exchange form a _____.
   b. The range of frequencies available for transmission of data is called_____.
   c. _____ is the network of networks.
   d. A technique in which a dedicated and complete physical connection is established between two nodes for communication is _____switching.
   e. Any computer attached on the network is called a _____.
2. Multiple Choice Questions:
   1) Choose the option, which is not included in networking.
      a. Access to remote database
      b. Resource sharing
      c. Power transferring
      d. Communication
   2) Data transfer rate is often measured in
      a. Mbps
      b. Kbps
      c. Bps
      d. gbps
   3) Which one of the following is not in the category of communication channels?
      a. narrow band
      b. broad band
      c. light band
      d. voice band
   4) The greater the bandwidth of a given medium, the _____ is the data transfer rate
      a. higher
      b. lower
      c. both a and b
      d. neither a nor b

5) What is the approximate bandwidth of a typical voice signal?
   a. 2KHz
   b. 2MHz
   c. 3KHz
   d. 3MHz

3. What is a network? Give any two uses of having a network in your school computer lab.
4. Mention any two disadvantages of a network.
5. Two students in the same class sitting inside the same room have connected their laptops using Bluetooth for working on a group presentation. What kind of network have they formed?
6. Expand the following:
   a. ARPANET.
   b. PAN
   c. NIU
   d. MAN
7. What are the requirements for setting up a network?
8. How is a dedicated server different from a non dedicated server?
9. Two companies in different states wanted to transfer information. Which type of network will be used to implement the same?
10. Two schools in the same city wanted to transfer e-learning information. Which type of network will be used to implement the same?
11. Two teachers in the same school sitting in different labs wanted to transfer information. Which type of network will be used to implement the same?
12. Define a protocol. Name any two protocols used on Internet.
13. Differentiate between :
    a. Internet and Interspace
    b. Circuit Switching and Packet Switching technique
    c. LAN , WAN and MAN
14. Define a node and an NIU?
15. Define a channel. Name the three categories of communication channel.
16. What do you mean by bandwidth and DTR?

Chapter-2: Networking Concepts - II

Learning Objectives:

At the end of this chapter the students will be able to:

- Learn about transmission media
  - Wired (Twisted Pair, Coaxial, Fibre Optic)
  - Wireless(Infrared, Radio waves, Microwaves, Satellites)
- Understand Network Topologies (Bus, Star, Tree)
- Learn about Network Devices (Modem, RJ-45, Ethernet Card, Switch, Repeater, Router, Gateway, Wi-Fi card)

Transmission Medium

A transmission medium (plural media) is one which carries a signal from one computer to another. It is also known as communication channel. Transmission medium can be wired or wireless. We also name them as Guided and Unguided Media respectively.

Wired transmission media includes twisted pair cable, Ethernet cable, coaxial cable and optical fibre whereas wireless transmission media includes microwave, radio wave, satellite, infrared, Bluetooth, WIFI etc.

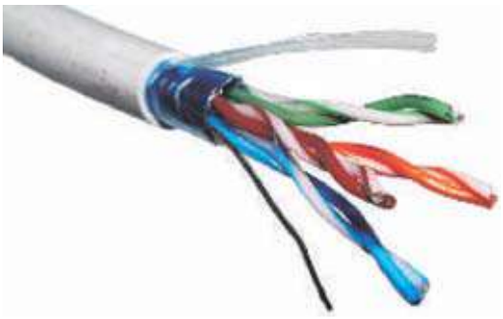
Wired Transmission Media

The wired or guided transmission media physically connects the two computers. The data signal physically gets transferred from the transmitting computer to the receiving computer through the wired transmission medium. Some of the wired transmission media are discussed below:

1. Twisted Pair Cables

This is one of the common forms of wiring in networks, especially in LANs and it consists of two insulated wires arranged in a regular spiral pattern (double helix). It is generally used for telephone communications in offices and also in modern Ethernet networks. For telephonic communication a Voice Grade Medium (VGM) cable is used but for LAN applications a higher quality cable known as Data Grade Medium (DGM) is used.

The twisting of wires reduces crosstalk which is the bleeding of a signal from one wire to another. This crosstalk can corrupt the signal and hence cause network errors. In addition to preventing internal crosstalk, the twisting of wires also protects the signal from external interference which affects both the wires and also creates unwanted signals.



A twisted pair cable

Advantages:

1. It is capable of carrying a signal over long distances without amplification.
2. It is simple, low weight, easy to install and easy to maintain.
3. It is an adequate and least expensive medium for low speed (up to 10 mbps) applications where the distance between the nodes is relatively small.

Disadvantages:

1. It can easily pick up noise signals.
2. Being thin in size, it is likely to break easily.
3. It is unsuitable for broadband applications.

Types of Twisted Pair Cables

There are two types of twisted pair cables available. These are:

i) Shielded Twisted Pair(STP) Cable.

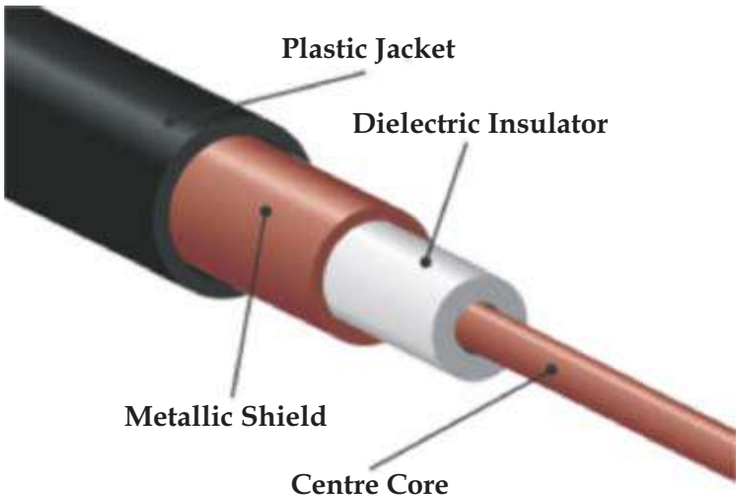
ii) Unshielded Twisted Pair(UTP) Cable

The STP cable comes with shielding of the individual pairs of wires, which further protects it from external interference and crosstalk. But STP is heavier and costlier than UTP and also requires proper grounding at both the ends.

http://btsadvancedcommunications.files.wordpress.com/2011/11/stputp.jpg

2. Coaxial Cables

It is the most commonly used transmission media for LANs. It consists of solid wire cores surrounded by one or more foil or wire shields, each separated by some kind of plastic insulator. The inner core carries the signal and the shield provides the ground. It has high electrical properties and is suitable for high speed communication. It is widely used for television signals and also by large corporations in building security systems. Multi channel television signals can be transmitted around metropolitan areas at considerably less cost.



http://upload.wikimedia.org/wikipedia/commons/thumb/f/f4/Coaxial_cable_cutaway.svg/500px-Coaxial_cable_cutaway.svg.png

A coaxial cable

Advantages

1. Data transmission characteristics are better than that of twisted pair.
2. It can be used for broadband communication i.e. several channels can be transmitted simultaneously.
3. It offers high bandwidth (up to 400 mbps)
4. It can be used as the basis for shared cable network.

Disadvantages

1. It is expensive as compared to twisted pair cables

Types of coaxial cables:

The two most common types of cables are Thicknet and Thinnet. Whereas thicknet is thicker and its cable segments can be up to 500 metres long , the thinnet on the other hand is thinner and it can have a maximum segment length of 185 metres.

3. Optical Fibres

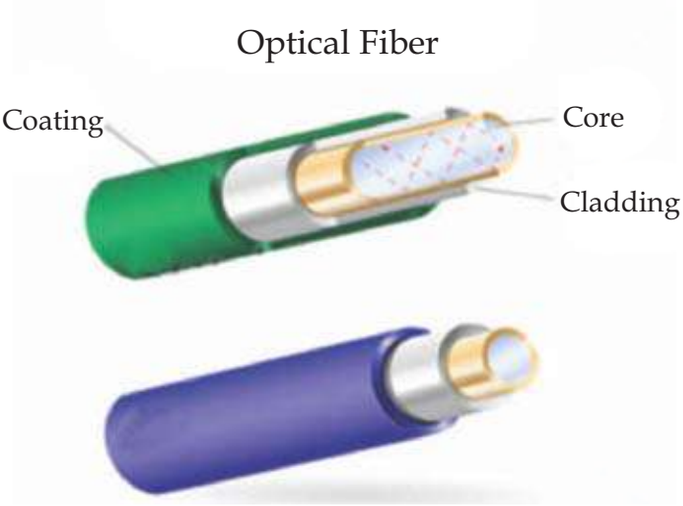
These consists of thin strands of glass or glass like material which are so constructed that they carry light from a source at one end of the fibre to a detector at the other end. The light sources used are either light emitting diodes (LEDs) or laser diodes (LDs). The data to be transmitted is modulated onto a light beam using frequency modulation techniques. At the receiver's end, the signals are demodulated. Optical fibres offer a very high bandwidth and this makes it capable of multichannel communication.

The Optical fibre consists of three layers:

i) Core - glass or plastic through which the light travels

ii) Cladding - covering of the core that reflects the light back to the core

iii) Protective (Buffer) coating-protects the fibre cable from hostile environments



Layers of an Optical Fiber

Advantages

1. It is immune to electrical and magnetic interference.
2. It is highly suitable for harsh industrial environments.
3. It guarantees secure transmission and has a very high transmission capacity.
4. It can be used for broadband transmission where several channels can be handled in parallel.

Disadvantages

1. It is difficult to install and maintain since they are quite fragile.
2. It is most expensive of all cables.
3. Connecting two fibres together or even connecting the light source with the cable is a difficult process. Hence connection loss is a common problem
4. Light can reach the receiver out of phase.

Fibre Optic cable can be of two types:

i) Single node fibre optic cable: It supports a segment length of up to 2kms and bandwidth of up to 100Mbps

ii) Multinode fibre optic cable: It has a segment length of 100kms and bandwidth of 2Gbps

Wireless Transmission Media

Wireless or unbounded or unguided media transport electromagnetic waves without using a physical conductor. The signals are broadcasted through air or water and thus are available to anyone that has a device capable of receiving them. Some of the wireless media are:

1. Infrared

Infrared is the frequency of light that is not visible to human eye. It has a range of wavelengths, just like visible light has wavelengths from red light to violet light. Far infrared waves are thermal. This is the reason we feel the heat from sunlight, a fire or a radiator. Shorter, near infrared waves are not hot at all - in fact we can't even feel them. These shorter wavelengths are the ones used by your TV remotes.

Infrared communication requires a transceiver (a combination of transmitter and receiver) in both devices that communicate. Infrared communication is playing an important role in wireless data communication due to the popularity of laptop computers , personal digital assistants(PDAs) , digital cameras , mobile phones , pagers and other devices but being a line-of-sight transmission , it is sensitive to fog and other atmospheric conditions.

Advantages

1. Since it is having short range of communication hence it is considered to be a secure mode of transmission.
2. It is quite inexpensive transmission medium.

Disadvantages

1. It can only be used for short range communication
2. Infrared wave transmission cannot pass through obstructions like walls, buildings etc.

2. Radiowaves

We all are quite familiar with radios and their working. In case of radiowave transmission, certain radio frequencies are allocated to private/government organizations for direct voice communications. Each radio signal uses a different frequency and this differentiates it from others. The transmitter takes some message, encodes it and then transmits it with radio wave. The receiver on the other hand receives the radio waves and decodes it. Both the transmitter and the receiver use antennas to radiate and capture the radio signal. Radio transmission is widely used by delivery services, policemen, security personals etc.

Advantages

1. It is easy to communicate through radio waves in difficult terrains since there is no need of digging and laying cables.
2. Radio waves can travel through long distances in all directions. Also they can easily pass through obstacles like a building so they can be used for both indoor and outdoor communication.

Disadvantages

1. It is susceptible to weather effects like rain, thunderstorm etc.
2. Data transmitted through radiowaves is not secure.

3. Microwaves

Another popular transmission medium is the microwave which permits data transmission rates of about 16 gigabits per second. This type of transmission uses high frequency radio signals to transmit data through space. Like radio waves, microwaves can pass through obstacles viz. buildings, mountains etc. Microwaves offer a line of sight method of communication. A transmitter and receiver of a microwave system are mounted on very high towers and both should be visible to each other (line of sight) In case of microwave transmission, curvature of the earth, mountains and other structures often block the line of sight. Hence several repeater stations are required for long distance transmission thereby increasing the cost considerably. It is generally used for long distance telephonic communications.

Advantages

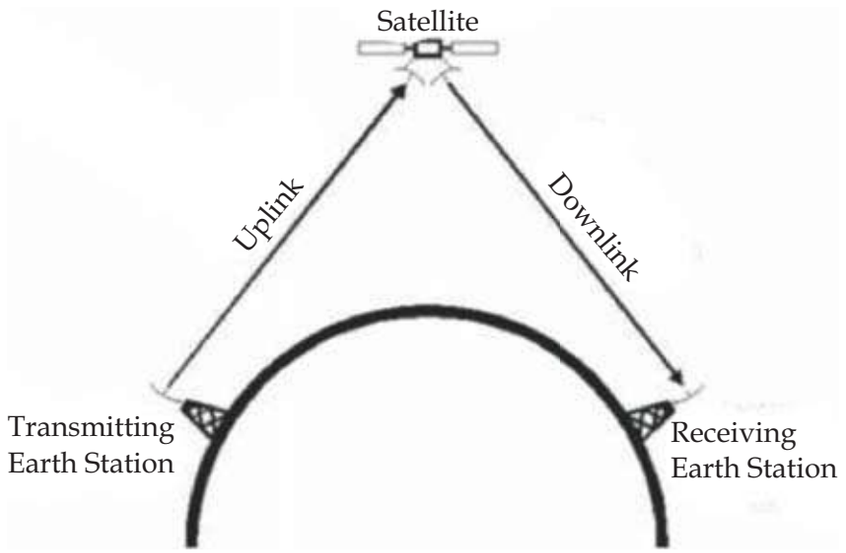
1. Microwave transmission does not require the expense of laying cables
2. It can carry 25000 voice channels at the same time.
3. Since no cables are to be laid down so it offers ease of communication over difficult terrains like hilly areas.

Disadvantages

1. Signals become weak after travelling a certain distance and so require amplification. To overcome this problem, repeaters are used at regular intervals (25-30 kms). The data signals are received, amplified and then retransmitted. This makes it a very expensive mode of communication
2. Installation and maintenance of microwave links turns out be a very expensive affair.
3. The transmission is affected by weather conditions like rain, thunderstorms etc.

4. Satellites

Satellites are an essential part of telecommunications systems worldwide today. They can carry a large amount of data in addition to TV signals.



http://www.radio-electronics.com/info/satellite/communications_ satellite/communications_satellite.gif

Fig: Satellite Communication

Satellite communication is a special use of microwave transmission system. A satellite is placed precisely at 36000 km above the equator where its orbit speed exactly matches the earth's rotation speed. Hence it always stays over the same point with respect to the earth. This allows the ground station to aim its antenna at a fixed point in the sky. The ground station consists of a satellite dish that functions as an antenna and communication equipment to transmit (called Uplink) and receive (called Downlink) data from satellites passing overhead. Such satellites can cost $60 million to build but only three of them are needed to cover the entire earth's surface. Capacity or number of channels used in satellite communications depends on the frequency used. Typical data transfer rates are 1 to 10 Mbps. Satellites are especially used for remote locations, which are difficult to reach with wired infrastructure. Also communication and data transfer on internet, is only possible through satellites.

Advantages

1. Satellite communication is very economical keeping in mind the fact that the area covered through satellite transmission is quite large. For e.g., satellites used for national transmission are visible from all parts of the country.
2. Transmission and reception costs are independent of the distance between the two points.

Disadvantages

1. Placing the satellite into its orbit involves very high cost.
2. Since signals sent to a satellite are broadcasted to all receivers, so necessary security measures have to be taken to prevent unauthorized tampering of data.
3. Transmission is affected by weather conditions like rain, thunderstorm etc.

Network Topologies

Topology is the pattern of interconnection of nodes in a local area network(LAN). The topology used helps to select the communication medium and the other network devices. While choosing a topology, care has to be taken that the installation cost is minimum, the network so designed should be reliable and flexible. In simple terms the addition or reduction of nodes should be easy and also fault detection and removal should be simple. Before we talk about topologies in detail, let us learn about point to point link which has two ends transmitter and receiver. The main characteristic of Point to Point link is that each transmitter transmits to exactly one receiver and each receiver receives exactly form one transmitter. The transmission might occur on a single medium i.e. single wire or over separate wires.

Fig: Point to Point Link

Network topologies are categorized into the following basic types:

- bus
- star
- tree

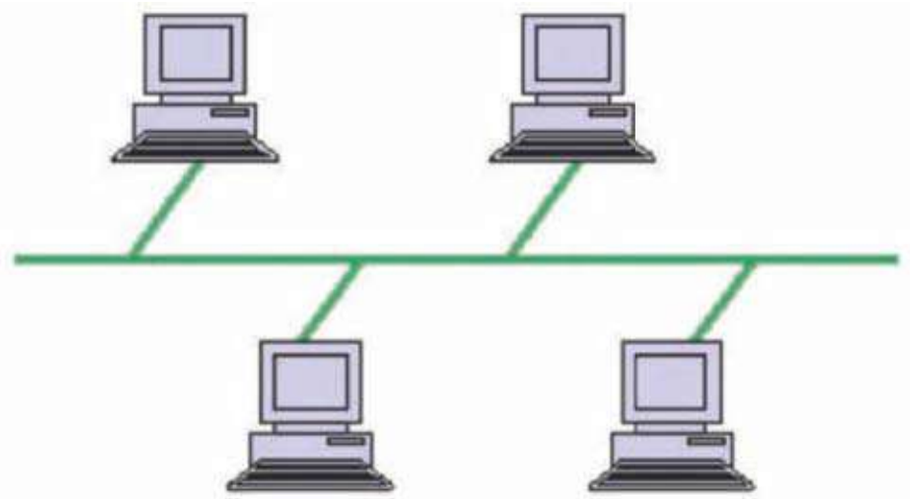
More complex networks can be built as hybrids of two or more of the above basic topologies. But right now let us study the above mentioned topologies:

Bus Topology

Bus topology is also known as Linear Topology. In this type of topology, each node attaches directly to a common cable which acts as the backbone and therefore functions as a shared communication medium onto which various nodes are attached. A device wanting to communicate with another device on the network sends a broadcast message in both directions onto the wire that all other devices see, but only the intended recipient actually accepts and processes the message. Data is transmitted in small blocks called packets. Each packet has a header containing the destination address. When data is transmitted on the cable, the destination node identifies the address on the packet and thereby processes the data.

This topology most often serves as the backbone for a network. In some instances, such as in classrooms or labs, a bus will connect small workgroups

Ethernet bus topologies are relatively easy to install and don't require much cabling compared to the alternatives. 10Base-2 ("ThinNet") and 10Base-5 ("ThickNet") both were popular Ethernet cabling options many years ago for bus topologies. However, bus networks work best with a limited number of devices. If more than a few dozen computers are added to a network bus, performance problems are likely to occur. In addition, if the backbone cable fails, the entire network effectively becomes unusable.



http://compnetworking.about.com/library/graphics/topology_bus.gif

Fig: Bus Topology

Advantages of Bus Topology

i) Since there is a single common data path connecting all the nodes, the bus topology uses a very short cable length which considerably reduces the installation cost.

ii) The linear architecture is very simple and reliable.

iii) Additional nodes can be easily connected to the existing bus network at any point along the length of the transmission medium.

Disadvantages of Bus topology

i) Fault detection and isolation is difficult. This is because control of the network is not centralized in any particular node. If a node is faulty on the bus, detection of fault may have to be performed at many points on the network. The faulty node has then to be rectified at that connection point.

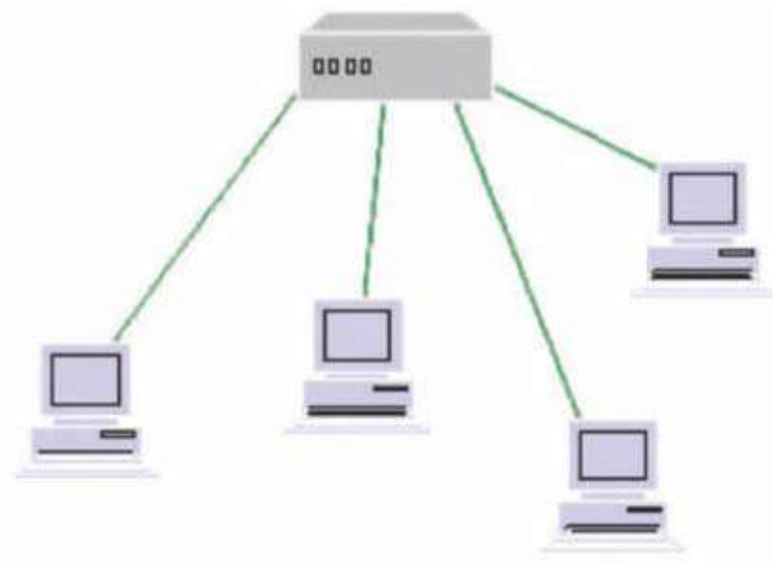
ii) If the central bus length becomes too long, then repeaters might have to be used to amplify the signal. The use of repeaters makes reconfiguration necessary.

iii) Since each node is directly connected to the central bus, so there has to be some way of deciding who can use the network at any given time.

Star Topology

A star network features a central connection point called a "hub node" to which all other nodes are connected by a single path. Each node has a dedicated set of wires connecting it to a central network hub. Since all traffic passes through the hub, the hub becomes a central point for isolating network problems and gathering network statistics. This type of topology is used in most existing information networks involving data communications or voice communications. For example in IBM370 installations, multiple 3270 terminals are connected to the host system. Many home networks also use the star topology.

Compared to the bus topology, a star network generally requires more cable, but a failure in any star network cable will only take down one computer's network access and not the entire LAN. On the other hand if the hub fails, the entire network also fails.



http://compnetworking.about.com/od/networkdesign/ig/Computer-Network-Topologies/Star-Network-Topology-Diagram.htm

Fig: Star Topology

Advantages of Star Topology

i) Failure of a single connection does not affect the entire network. It just involves disconnecting one node from an otherwise fully functional network. This also helps in easy reconfiguration of the network.

ii) Fault detection is easier.

iii) Access protocols being used in a Star network are very simple since the central node has the control of the transmission medium for data transmission

Disadvantages of Star Topology

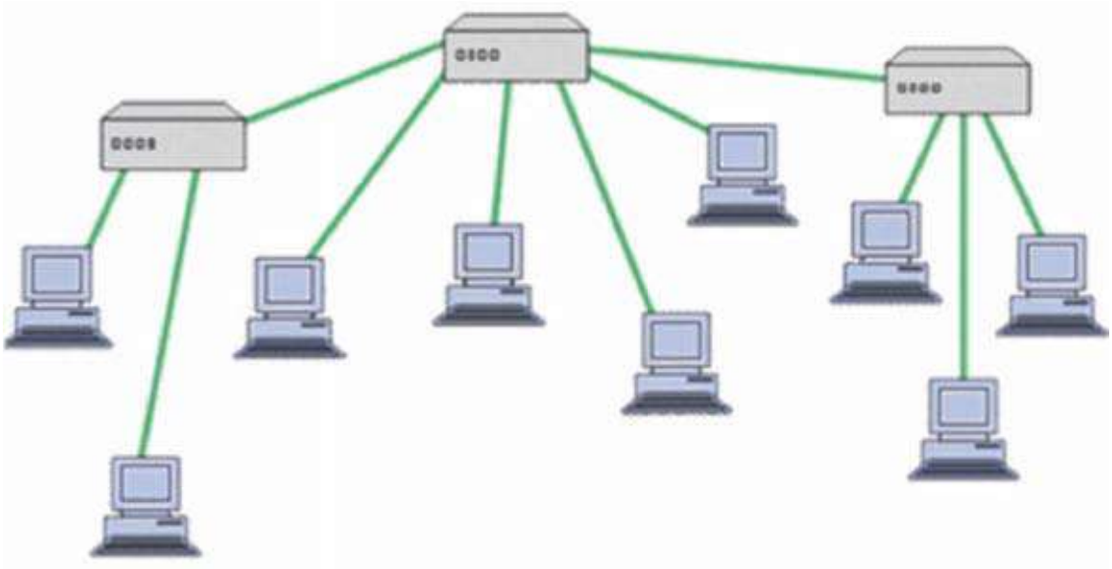
i) Since every node is directly connected to the centre, so large amount of cable is needed which increases the installation cost of the network.

ii) The entire network is dependent on the central node. If the central node fails the entire network goes down.

Tree Topology

Tree topology is a combination of bus and star topology. The network looks like an inverted tree with the central root branching and sub-branching down to the nodes. It integrates multiple star topologies together onto a bus. In its simplest form, only hub devices connect directly to the tree bus. This bus/star

hybrid approach supports future expandability of the network much better than a bus (limited in the number of devices due to the broadcast traffic it generates) or a star (limited by the number of hub connection points) alone. Data transmission takes place in the same way as in bus topology. When the signal reaches the end of the transmission medium, it is absorbed by the terminators. Tree topology is best suited for applications which have a hierarchical flow of data and control.



http://compnetworking.about.com/od/networkdesign/ig/Computer-Network-Topologies/Tree-Network-Topology-Diagram.htm

Fig: Tree Topology

Network Devices

For efficient working of any network, many devices are required. Some of the common network devices are discussed below:

Modem

A modem (Modulator - Demodulator) is a peripheral device that enables a computer to transmit data over, telephone or cable lines. The computers operate digitally using binary language (a series of zeros and ones), but transmission mediums are analogue. The digital signals when pass from one value to another, there is no middle or half way point, it's All or Nothing (one or zero). Conversely, analogue does not change "per step", it covers all the values, so you can have 0, 0.1, 0.2, 0.3 ...1.0 and all values in between. A modem converts between these two forms. It modulates an analogue carrier signal to encode digital information, and also demodulates such a carrier signal to decode the transmitted information. This is why modem is an acronym of MOdulator/DEModulator. The goal of this process of modulation - demodulation is to produce a signal that can be transmitted easily and decoded to reproduce the original digital data.

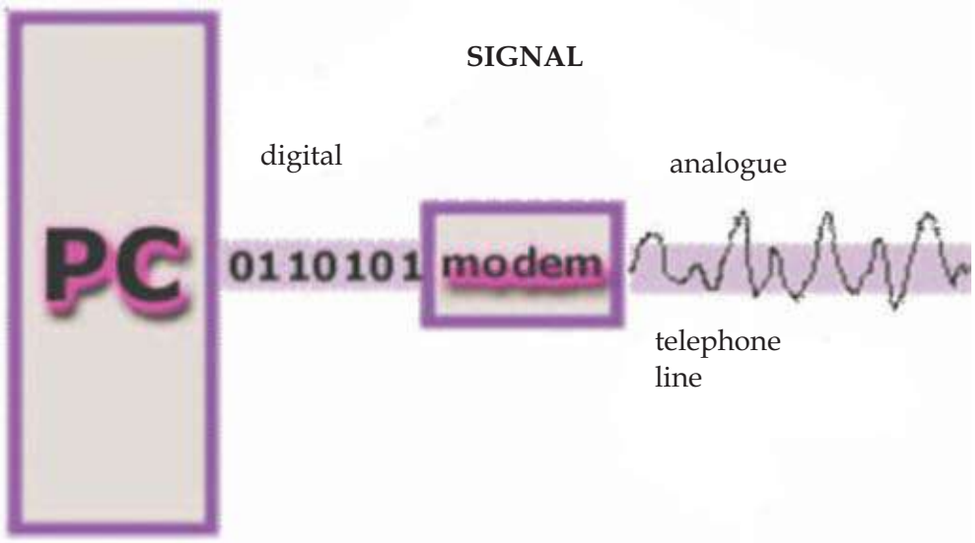


Fig: Working of a Modem

All these processes are performed by modem at extremely high speeds. The speed of the modem depends upon the number of available access lines and the technology of the modem. The amount of data that can be sent in a given unit of time, is usually expressed in bits per second (bps) or bytes per second(B/s).

RJ-45

RJ-45 , short form of Registered Jack - 45 , is an eight wired connector that is used to connect computers on a local area network(LAN), especially Ethernet. RJ-45 connectors look similar to the RJ-11 connector used for connecting telephone equipment, but they are somewhat wider.

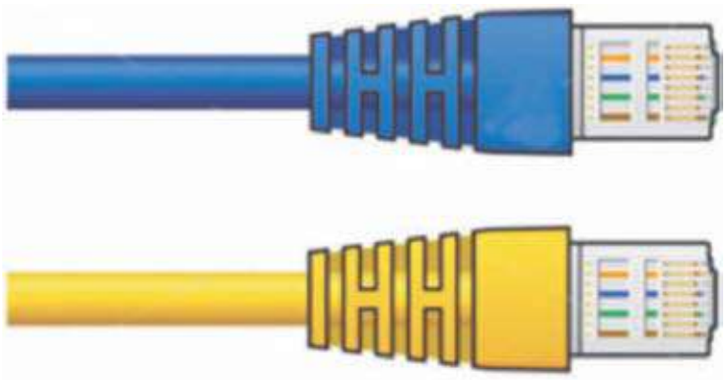


Fig: RJ-45

Ethernet Card

An Ethernet card is a kind of network adapter and is also known as Network Interface Card (NIC). These adapters support the Ethernet standard for high-speed network connections via cables. An Ethernet Card contains connections for either coaxial or twisted pair cables or even for fibre optic cable.

Newer Ethernet cards are installed usually by the manufacturer inside the desktop computers that resemble credit cards are readily available for laptop and other mobile computers. These insert conveniently into slots on the side or front of the device.

Fig: An Ethernet Card

Though they look more like small boxes than cards, external USB Ethernet adapters also exist. These are a convenient alternative to PCI cards for desktop computers and also commonly used with video game consoles and other consumer devices lacking expansion slots.

Ethernet cards may operate at different network speeds depending on the protocol standard they support. Old Ethernet cards were capable only of the 10 Mbps maximum speed offered by Ethernet originally. Modern Ethernet adapters can support the speed of upto100 Mbps. Fast Ethernet standards are also available now that offer speeds upto1 Gbps (Gigabit Ethernet).

Switch

A switch is a device that is used to break a network into different sub-networks called subnet or LAN segments. This prevents traffic overloading on the network. Switches are another fundamental part of many networks because they speed things up. They allow different nodes of a network to communicate directly with one another in a smooth and efficient manner. In simple terms, a network switch is a small hardware device that joins multiple computers together within one local area network (LAN).

Network switches appear nearly identical to network hubs, but a switch generally contains more intelligence than a hub. We can say that a switch is an intelligent hub and is obviously more expensive than a hub. Unlike hubs, network switches are capable of inspecting data packets as they are received, determining the source and destination device of each packet, and forwarding them appropriately. By delivering messages only to the connected device intended, a network switch conserves network bandwidth and offers generally better performance than a hub.

Mainstream Ethernet network switches support either 10/100Mbps fast Ethernet or Gigabit Ethernet (10/100/1000) standards.

Switches that provide a separate connection for each node in a company's internal network are called LAN switches. Essentially, a LAN switch creates a series of instant networks that contain only the two devices communicating with each other at that particular moment.

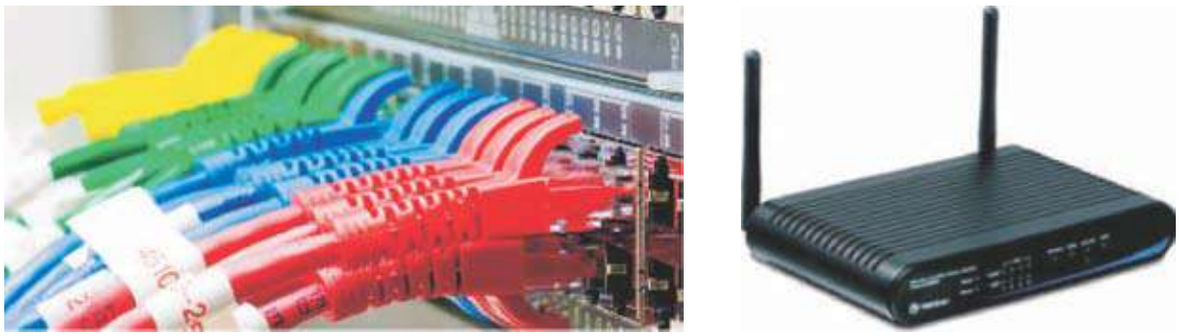
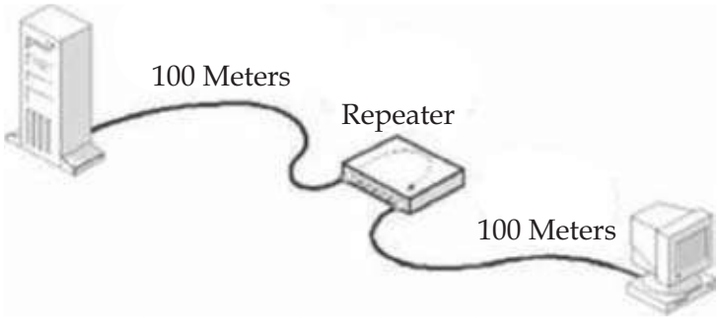


Fig: Switches

Switches are commonly used in home networks and in small businesses. They need not be monitored or configured using external software applications. They are easy to set up and require only cable connections.

Repeater

A repeater is an electronic device that receives a signal , amplifies it and then retransmits it on the network so that the signal can cover longer distances. Network repeaters regenerate incoming electrical, wireless or optical signals. An electrical signal in a cable gets weaker with the distance it travels, due to energy dissipated in conductor resistance and dielectric losses. Similarly a light signal travelling through an optical fibre suffers attenuation due to scattering and absorption. With physical media like Ethernet or WiFi, data transmissions can only span a limited distance before the quality of the signal degrades. Repeaters attempt to preserve signal integrity by periodically regenerating the signal and extend the distance over which data can safely travel.



http://jaringankomunikasi.wordpress.com/

Fig: A Repeater

Actual network devices that serve as repeaters usually have some other name. Active hubs, for example, are repeaters. Active hubs are sometimes also called "multiport repeaters," but more commonly they are just "hubs." In WiFi, access points may function as repeaters. A repeater cannot do the intelligent routing performed by bridges and routers. It cannot filter the traffic to ease congestion and also cannot work across multiple network architectures.

Routers

A Router is a network device that works like a bridge to establish connection between two networks but it can handle networks with different protocols. For example a router can link an Ethernet network to a mainframe or to internet. If the destination is unknown to the router, it sends the traffic to another router which knows the destination. The data is sent to the router which determines the destination address (using logical address) and then transmits the data accordingly. Hence routers are smarter than hubs and switches. Using a routing table that stores calculated paths, routers make sure that the data packets are travelling through the best possible paths to reach their destinations. If a link between two routers fails, the sending router can determine an alternate route to keep traffic moving. Routers provide connectivity inside enterprises, between enterprises and the Internet, and within an Internet Service Provider (ISP). Routers can be wireless or wired.

Gateway

A gateway is a network device that establishes an intelligent connection between a local network and external networks with completely different structures i.e. it connects two dissimilar networks. In simple terms, it is a node on a network that serves as an entrance to another network.

The computers that control traffic within your company's network or at your local Internet Service Provider (ISP) are gateway nodes. A network gateway can be implemented completely in software, completely in hardware, or as a combination of both. In the network for an enterprise, a computer server acting as a gateway node is often also acting as a proxy server and a firewall server. Here a proxy server is a node that is not actually a server but just appears to be so and a firewall is a system designed to prevent unauthirised access to or from a private network. A gateway is often associated with both a router, which knows where to direct a given packet of data that arrives at the gateway, and a switch, which furnishes the actual path in and out of the gateway for a given packet. It expands the functionality of the router by performing data translation and protocol conversion.

You will sometimes see the term default gateway on network configuration screens in Microsoft Windows.

In computer networking, a default gateway is the device that passes traffic from the local subnet to devices on other subnets. The default gateway often connects a local network to the Internet, although internal gateways for local networks also exist.

Wi-Fi Card

Wi-Fi cards are small and portable cards that allow your desktop or laptop computer to connect to the internet through a wireless network. Wi-Fi transmission is through the use of radio waves. The antenna transmits the radio signals and these signals are picked up by Wi-Fi receivers such as computers and cell phones equipped with Wi-Fi cards. These devices have to be within the range of a Wi-Fi network to receive the signals. The Wi-Fi card then reads the signals and produces a wireless internet connection. Once a connection is established between user and the network, the user will be prompted with a login screen and password if the connection being established is a secure connection.

Wi-Fi cards can be external or internal. If a Wi-Fi card is not installed in your computer, you may purchase a USB antenna attachment and have it externally connected to your device. Many newer computers, mobile devices etc. are equipped with wireless networking capability and do not require a Wi-Fi card. However, it is important to understand that the Wi-Fi connection only exists between the device and the router. Most routers are further connected to a cable modem, which provides internet access to all connected devices.

LET'S REVISE

- **Transmission Medium:** One which carries a signal from one computer to another.
- **Wired Transmission Media:** Twisted Pair, Coaxial , Fibre Optic , Ethernet cable
- **Wireless Transmission Media:** Radio waves , Microwaves, Bluetooth , WiFi, Satellites, Infrared
- **Topology:** The pattern of interconnection of nodes in a LAN.
- **Network Topologies:** Bus , Star, Tree
- **Modem:** A device that enables a computer to transmit data over, telephone or cable lines.
- **RJ-45:** An eight wired connector used to connect computers on a LAN.
- **Ethernet card:** A kind of network adapter.
- **Switch:** A small hardware device that joins multiple computers together within a LAN.
- **Repeater:** An electronic device that amplifies the received signal and then retransmits it on the network
- **Router:** A network device that connects two networks with different protocols.
- **Gateway:** A network device that connects two dissimilar networks.
- **Wi-Fi card:** A small, portable card that allow your computer to connect to the internet through a wireless network.

EXERCISE

1. What do you mean by a transmission medium? Differentiate between guided and unguided transmission media.
2. Explain the structure of a coaxial cable and a fibre optic cable.
3. What are advantages of fibre optic cable?
4. Differentiate between a radio wave transmission and a microwave transmission.
5. Explain satellite communication. What are the advantages and disadvantages of using satellite communication?
6. Define the term topology.
7. List any two advantages and any two disadvantages of Star topology.
8. How is Tree Topology different from Bus topology?
9. Identify the type of topology from the following.
   a. Each node is connected with the help of single cable.
   b. Each node is connected with the help of independent cable with central switching.
10. What do you mean by a modem? Why is it used?
11. Explain the following devices:
    a. Switch
    b. Repeater
    c. Router
    d. Gateway
    e. Wi-Fi Card
12. Show a network layout of star topology and bus topology to connect 4 computers.
13. Ms. Anjali Singh, in charge of Knowledge centre in ABC school, recently discovered that the communication between her centre and the primary block of the school is extremely slow and signals drop quite frequently. The distance between these two blocks is 140 meters.
    a. Name the type of network.
    b. Name the device which may be used for smooth communication.
14. ABC International School is planning to connect all computers, each spread over distance of 50 meters. Suggest an economic cable type having high speed data transfer to connect these computers.

15. Sahil wants to transfer data across two continents at very high speed. Write the name of the transmission medium that can be used to do the same. Write the type of network also.
16. Mayank wants to transfer data within a city at very high speed. Write the name of the wired transmission medium that he should use. Write the type of network also.
17. Mr. Akash wants to send/receive email through internet. Which protocol will be used for this purpose?
18. Answer the following questions in the context of a computer lab with 100 computers.
    a. Which device is used to connect all computers inside the lab?
    b. Which device is used to connect all computers to the internet using telephone wire?
19. Name the device that establishes an intelligent connection between a local network and external network with completely different structures.
20. Name the network device that works like a bridge to establish connection between two networks but it can also handle networks with different protocols.

Chapter-3: Network Protocols

Learning Objectives:

At the end of this chapter the students will be able to:

- Network protocols
  - TCP/IP
  - FTP
  - HTTP
  - PPP
- E-mail protocols
  - SMTP
  - POP3
- Remote Access Protocol
  - Telnet
- Chat and VOIP Protocol

Network Protocols

In information technology, a protocol is the special set of rules that two or more machines on a network follow to communicate with each other. They are the standards that allow computers to communicate. A protocol defines how computers identify one another on a network, the form that the data should take in transit, and how this information is processed once it reaches its final destination. A protocol is needed every time we want to perform any task on a network. It may be transferring data or taking a printout on a network printer or accessing the central database.

Although each network protocol is different, they all share the same physical cabling. This common method of accessing the physical network allows multiple protocols to peacefully coexist over the network media, and allows the builder of a network to use common hardware for a variety of protocols. This concept is known as "Protocol Independence."

Some of the important protocols being used are as follows:

Transmission Control Protocol / Internet protocol(TCP/IP)

TCP/IP are the two protocols that are used together and together they form the backbone protocol of the internet. They can also be used for private networks i.e. intranets and extranets. When you are set up with direct access to the Internet, your computer is provided with a copy of the TCP/IP program

TCP/IP has two major components: TCP and IP.

The Transmission Control Protocol(TCP) breaks the data into packets that the network can handle efficiently. It manages the assembling of a message or file into smaller packets that are transmitted over the Internet. It verifies all the packets when they arrive at the destination computer and then reassembles them in proper order. Data can be lost in the intermediate network. So TCP adds support to detect errors or lost data and to trigger retransmission until the data is correctly and completely received.

The Internet Protocol(IP)handles the address part of each packet so that it reaches to the right destination. It gives distinct address (called IP address) to each data packet. Each gateway computer on the network checks this address to see where to forward the message. Even though some packets from the same message are routed differently than others, they'll be reassembled at the destination.

An IP address is a unique identifier for a node or host connection on an IP network. An IP address is a 32 bit binary number usually represented as 4 decimal values, each representing 8 bits, in the range 0 to 255 (known as octets) separated by decimal points. This is known as "dotted decimal" notation.

Example:

140.179.220.200

The Internet authorities assign ranges of numbers to different organizations. The organizations assign groups of their numbers to departments. IP operates on gateway machines that move data from department to organization to region and then around the world.

TCP/IP uses the client/server mode of communication in which a computer user (a client) makes a request and the server provides the requested service such as sending a Web page.Also TCP/IP communication is primarily point-to-point transmission of data which means each communication is from one computer in the network to another computer. TCP/IP and the higher-level applications that use it are collectively said to be "stateless" because each client request is considered a new request unrelated to any previous one.Till the time complete message or all packets in a message have been delivered, the connection between the two computers remains intact but after that the transmission path is available freely. So unlike ordinary phone conversations that require a dedicated connection for the entire call duration, no dedicated connection is required. This makes the network paths freely available for everyone to use.

File Transfer protocol (FTP)

This is the simplest and one of the oldest protocols designed for transferring files of any type(ASCII or binary) from one system to another on the internet. FTP is an application protocol that uses the Internet's TCP/IPprotocols.

FTP is based on Client/Server principle. By giving the ftp command with any remote address, the file transfer can be initiated. In any FTP interface, clients identify the FTP server either by its IP address (such as 192.168.0.1) or by its host name (such as ftp.about.com). It is an efficient means to send and receive files from a remote host. FTP establishes two connections between the hosts. One connection is used for data

transfer and the other for control information. The control connection remains connected during the entire interactive FTP session while the data connection is opened and closed for each file transfer. As a user, you can use FTP with a simple command from the Windows MS-DOS Prompt window or with a commercial program that offers a graphical user interface. You can even download programs by making FTP requests through your web browser.By logging on to an FTP server, you can delete, rename, move, or copy files at a server. However, publicly available files are easily accessed using anonymous FTP.In such a case you need not formally sign-in to the FTP server to make a file transfer, instead you may be simply asked to enter your email address. But if you are using a private FTP server, you must sign in with a user name and password to initiate the exchange of data.

Basic FTP support is usually provided as part of a suite of programs that come with TCP/IP. However, any FTP client program with a graphical user interface usually has to be downloaded from the company that makes it.

As mentioned before FTP can transfer both ASCII i.e. plain text and binary files but the mode has to be set in the FTP client.If you attempt to transfer a binary file (such as a program or music file) while in text mode, the transferred file becomesunusable.

HyperText Transfer Protocol (HTTP)

HTTP is the protocol that is used for transferring hypertext (i.e. text, graphic, image, sound, video etc.) between two computers and is particularly used on the World Wide Web. It is a TCP/IP based communication protocol and provides a standard for Web browsers and servers to communicate.

Hypertext is the text that is specially coded using a standard coding language called HyperTextMarkupLanguag(HTML) which basically creates hyperlinks and thereby controls how the World Wide Web works and how Web pages are formatted and displayed.

These hyperlinks can be in the form of text, graphic, image, sound or video and are used to "link "the user to some other file. HTTP defines how messages are formatted

and transmitted, and what actions Web servers and browsers should take in response to various commands. For example, when you enter a URL in your browser, this actually sends an HTTP command to the Web server directing it to fetch and transmit the requested Web page.

HTTP is based on Client/Server principle. Communication between the host and the client occurs through a request/response pair. A connection is established between two computers - out of which one is client (generally the browser)thatinitiates the request and the other is the server that responds to the request.Also HTTP identifies the resource that the client has requested for and informs the server about the action to be taken. When the user clicks on the hypertext link, the client program on their computer uses HTTP to contact the server, identify the resource and ask the server to respond with an action. The server accepts the request and then uses HTTP to respond to perform the action.

Although HTTP was designed for use in the web, it is being used in a much more general fashion because of increasing object oriented applications.

HTTP has three important features. Firstly, it is connectionless.After a request is made, the client disconnects from the server and waits for a response.To process the request, the server has to re-establish the connection with the client. Secondly, HTTP is media independent. This means any type of data(text , images , sound , video etc.) can be sent by HTTP as long as both the client and server know how to handle the data content. Thirdly HTTP is stateless.This is because the server and the client are aware of each other only during a request. Afterwards, they get disconnected.Hence neither the client nor the browser can retain information between different request across the web pages.

Point to Point Protocol (PPP)

PPP (Point-to-Point Protocol) is usedfor communication between two computers using a serial interface, mostly a personal computer connected by phone line to a server. For example, an Internet ServiceProvider(ISP) may provide you with a PPP connection so that the ISP'sserver can respond to your requests, pass them on to the Internet, and forward your requested Internet responses back to you. It was basically designed to help communication between two systems through telephone lines as it supports transmission of network packets over a serial point to point link.

PPP is sometimes considered a member of the TCP/IP suite of protocols. Essentially, it encapsulates and packages your computer's TCP/IP packets into PPP frames and then forwards them to the server over serial transmission lines such as telephone lines, ISDN etc. PPP defines the format of frame to be exchanged between devices on one or multiple links and also defines the authenticity of the two devices. It supports various authentication schemes such as Password Authentication Protocol(PAP) and Challenge Handshake Authentication protocol(CHAP).

E-Mail Protocols

Simple Mail transfer protocol (SMTP)

SMTP stands for Simple Mail Transfer Protocol that allows transmission of email over the Internet. Most email software is designed to use SMTP for communication purposes when sending email. It only works for outgoing messages. So when an email has to be sent, the address of their Internet Service Provider's SMTP server has to be given. The actual mail transfer is done through Message Transfer Agents(MTA). So the client computer must have a client MTA and the server must have a server MTA. SMTP actually defines the MTA client and the server on the internet.

SMTP is a reliable and easy to set up protocol. Messages either get to a recipient, or there is an error message that explains why that wasn't possible. One of the purposes of an SMTP is that it simplifies the communication of email messages between servers. It allows the server to break up different parts of a message into categories the other server can understand. Any email message has a sender, a recipient or sometimes multiple recipients - a message body, and usually a title heading. Once a message goes out on

the internet, everything is turned into strings of text. This text is separated by code words or numbers that identify the purpose of each section of an email. SMTP provides those codes, and email server software is designed to interpret these codes.

The other purpose of SMTP is to set up communication rules between servers. Every server has its own way to identify itself, define the mode of communication that they will follow, check for errors and handle them.In a typical SMTP transaction, a server will identify itself, and announce the kind of operation it is trying to perform. The other server will authorize the operation, and the message will be sent. If the recipient address is wrong, or if there is some other problem, the receiving server may reply with some error message.

SMTP has a major disadvantage that it is relatively easy to send a message with a fake sender address. This results in spread of many email-based viruses. Someone may receive a message that they think is coming from a friend, when someone else is actually sending it. Although attempts are being made to overcome this disadvantage but it still causes some problems.

Most servers these days actually us a slightly updated version of the SMTP protocol called ESMTP (Extended Simple Mail Transfer Protocol). This was created to allow transmission of multimedia through email. When someone sends a picture or music file through their email program, ESMTP communication codes are used to identify the kind of data being transferred.Mutipurpose Internet Mail Extension(MIME) is a supplementary protocol that allows non ASCII data to be sent through SMTP. Please note that MIME is not a protocol and cannot replace SMTP.

Post Office Protocol Version 3 (POP3)

Post Office Protocol 3 or POP3 is the third version of a widespread method of receiving email which receives and holds email for an individual until they pick it up. SMTP has a disadvantage that if the destination computer is not online, mails cannot be received.So the SMTP server receives the mail on behalf of every host and the respective host then interacts with the SMTP server to retrieve messages by using a client server protocol called POP3.

POP3 makes it easy for anyone to check their email if their email program is configured properly to work with the protocol. It is extremely common among most mail servers because of its simplicity and high success rate and minimumerrors. Also it can work with virtually any email program, as long as the email program is configured to host the protocol. Many popular email programs, including Microsoft Outlook, are automatically designed to work with POP3. Each POP3 mail server has a different address, which is usually provided to an individual by their web hosting company. This address must be entered into the email program so that the program can connect effectively with the protocol.The individuals receiving POP3 email will have to input their username and password in order to successfully receive email.

Remote Access Protocol

Telnet

Telnet is the main internet protocol for creating a connection with a remote machine. It allows you to connect to remote computers (called remote hosts) over a TCP/IP network (such as the Internet). Once your telnet client establishes a connection to the remote host, your client becomes a virtual terminal, allowing you to communicate with the remote host from your computer with whatever privileges you may have been granted to the specific application and data on that host computer.

Telnet clients are available for all major operating systems viz. Mac OS X, Windows, Unix, and Linux. To use these clients, go to their respective command lines and then enter:telnet host wherehost isthe name of the remote computer to which you wish to connect.In most cases, you'll need to have an account on that system but canalsolog in as guest or public without having an account.

Telnet is most likely to be used by program developers and anyone who has a need to use specific applications or data located at a particular host computer. It gives the user the opportunity to be on one computer system and do work on another, which may be anywhere across the globe.Telnet provides an error free connection which is always faster than the latest conventional modems.

Chat Protocol and VOIP

Chatting

A real time informal communication over the Internet is chatting. A chat program is software which is required for chatting over the internet. AOL Instant Messenger, Campfire, Internet Messenger, MSN Messenger are some commonly used chat programs. In order to chat, the user should have an account on a chatting program. A phone call is a voice based chat while online chat is textual conversation.

Internet Relay Chat (IRC)

IRC protocol is used for chatting. It provides chatting between a group or between two individuals. It was developed by JarkkoOikarinen in Finland in the late 1980s. It is based on client/server model. The IRC client sends and receives messages to and from an IRC server. The IRC server transports the message from one client to another. The IRC server is linked to many other servers to form an IRC network. IRC server identifies every user through a unique nickname. Each user is assigned a unique channel in case multiple discussions are taking place.

VOIP

VOIP stands for voice over internet protocol. It enables the transfer of voice using packet switched network rather than using public switched telephone network. By using VOIP software, phone calls can be done using standard internet connection. This method of making phone calls is much cheaper than convectional way because the service of Telecommunication Company is not used.There are three different methods of VoIP service in common use today:

ATA - ATA stands for analog-to-digital converted. It is used to connect the telephone device to the computer. It takes the analog signals from the phone and converts them to digital signals. These digital signals can known be transmitted over the internet. Some providers also are bundling ATAs free with their service.

IP phones - IP phones appear much like an ordinary telephone or cordless phone. They are directly connected to the router or the LAN. They have all the hardware and software necessary right onboard to handle the IP call. IP Phones are sometimes calledVoIP telephones, SIP phones orSoft phones.

Computer-to-computer - It is the most easy and simplest way to use VoIP. The basic hardware requirements are as follows:

- Computer
- Internet
- Speakers
- Microphone

The only cost involved with computer - to- computer VoIP is the monthly ISP fee

LETS REVISE

- **Protocol:** A special set of rules that two or more machines on a network follow to communicate with each other.
- **Transmission Control Protocol(TCP):** It breaks the data into packets that the network can handle efficiently.
- **Internet protocol(IP):** It gives distinct address (called IP address) to each data packet.
- **File Transfer Protocol(FTP):** It is used for transferring files from one system to another on the internet.
- **HyperText Transfer Protocol(HTTP):** It is the protocol that is used for transferring hypertextfileson the World Wide Web.
- **Point-to-Point Protocol(PPP):** It is usedfor communication between two computers using a serial interface.
- **Simple Mail Transfer Protocol (SMTP):** It allows transmission of email over the Internet.
- **Post Office Protocol 3(POP3):** It receives and holds email for an individual until they pick it up.
- **Telnet:** A protocol for creating a connection with a remote machine.
- **IRC:** IRC protocol is used for chatting.It is based on client/server model.
- **VOIP:** VOIP stands for voice over internet protocol. It enables the transfer of voice using packet switched network rather than using public switched telephone network.

EXERCISE

1. Expand the following abbreviations:
   FTP, TCP,SMTP,VoIP
2. What do you mean by the term Protocol Independence?
3. Write short notes on:
   a) TCP/IP
   b) HTTP
   c) SMTP
   d) FTP
   e) Telnet
4. List three important features of HTTP.
5. Explain VOIP.
6. Explain IRC
7. Neha wants to upload and download files from/to a remote internal server. Write the name of the relevant communication protocol, which will let her do the same.
8. Meha wants to upload hypertext document on the internet. Write the name of protocol, which will let her do the same.
9. This protocol is used for communication between two personal computers using a serial interface and connected by a phone line. Write the name of the protocol.
10. This protocol is used to transfer email over internet. What is the name of the protocol?
11. This protocol is used to implement remote login. What is the name of the protocol?
12. This protocol is used for chatting between two groups or between two individuals. Write the name of the protocol.
13. This protocol is used to transfer of voice using packet switched network. Write the name of the protocol.
14. Explain Remote Access Protocol.
15. Why we need VoIP protocol?
16. Differentiate between FTP and HTTP.
17. Differentiate between VoIP and IRC.
18. Write the basic hardware requirements for VoIP.
19. Why TCP/IP based applications are considered to be stateless?
20. FTP is based on Client/Server principle. Explain.

Chapter-4: Mobile Telecommunication Technologies, (Network Security and Internet Services)

Learning Objectives:

At the end of this chapter the students will be able to:

- Learn about Mobile Telecommunication Technologies: 1G, 2G, 3G and 4G
- Learn about Network Security Concept such as Threats and prevention from Viruses, Worms, Trojan horse, Spams
- Understand the use of Cookies
- Learn about Firewall
- Learn about India IT Act, Cyber Law, Cyber Crimes, IPR issues, Hacking
- Learn about Web services such as WWW, Hyper Text Markup Language (HTML), eXtensible Markup
- Language (XML); Hyper Text Transfer Protocol (HTTP); Domain Names; URL; Website, Web browser,
- Web Servers; Web Hosting, Web Scripting - Client side (VB Script, Java Script, PHP) and Server side (ASP,JSP, PHP), Web 2.0 (for social networking)

Mobile Telecommunication Technologies

Mobile is a device which is portable. Mobile communication is based on cellular networks. A cellular network is nothing but a radio network. In this network, land is divided into areas called cells. Every cell in the network has a transmitter and a receiver known as cell site or base station. Each cell in the network uses different frequency for the transmission of signals.When joined together these cells provide radio coverage over a large geographical area. The network of cells enables the mobile devices to communicate even if they are moving from one cell to another via base stations.

The last two decades has seen a remarkable growth in the mobile industry both in terms of mobile technology and subscribers.

The first systems offering mobile telephone service were introduced in the late 1940s in the US and in the early 1950s in Europe. These single cell systems were severely constrained by restricted mobility, low capacity, limited service, and poor speech quality. Also the equipment was heavy, bulky, expensive, and susceptible to interference

The use of semiconductor technology and microprocessors made mobile systems smaller, lighter, and more sophisticated.

1G Mobile Systems

The 1G Mobile System was introduced in late 1970s and early 1980s.The 1G mobile system was based on the analog cellular technology. They only had voice facility available and were based on circuit-switched technology. In 1G mobile systems voice was modulated to a frequency of about 150MHz and higher. They used radio towers for transmission. The major drawbacks of the 1G system were its low capacity, poor voice links and no security.

http://www.electronicsforu.com/EFYLinux/efyhome/cover/jun2003/Mobile-tech.pdf

Before discussing about the 2G mobile systems, let's discuss some related terms like

FDMA

It stands for Frequency Division Multiple Access. In this, each user utilizes a portion of the frequency bandwidth available. Each user has its own frequency domain.

CDMA

It stands for Code Division Multiple Access. In this, each user is allocated a unique code sequence. On the sender's end, the data signal is encoded using the given unique code. The receiver decodes the signal according the unique code and recovers the original data.

TDMA

It stands for Time Division Multiple Access. In this, each user is allowed to transmit only within specified time intervals. Different users transmit in different time slots. When users transmit, they occupy the whole frequency bandwidth.

2G Mobile System

The 2G mobile system was introduced in early 1990s. They used digital signals for transmissions of voice. 2G enabled the mobile systems to provide paging, SMS, voicemail and fax services. Both voice and data conversations were digitally encrypted. The 2G system was based on GSM technology. GSM standard was defined by ETSI in 1989. GSM stands for Global System for Mobile Communication. GSM technology is a combination of FDMA and TDMA. With GSM, all subscriber and wireless provider information is stored on interchangeable modules known as SIM (Subscriber Identification Module) cards. By swapping out the SIM card, users can painlessly switch phones or providers. They used circuit switching.

The mobile technology using packet switched domain instead of circuit switched domain were termed as 2.5G mobile systems. They used GPRS (General Packet Radio Service) in addition to GSM. With 2.5G services like MMS, sending pictures through e-mail became possible. GPRS technology was also a major step towards 3G mobile system.

3G Mobile System

The 3G technology adds multimedia facilities to 2G phones by allowing video, audio, and graphics applications. With the advent of 3G technology watching streaming video or video telephony became a reality. The idea behind 3G is to have a single network standard instead of the different types adopted in the US, Europe, and Asia. 3G mobile systems are also known as Universal Mobile Telecommunications System (UMTS) or IMT-2000. They can sustain higher data rates and open the door to many Internet style applications. The main characteristics of IMT-2000 3G systems are:

- A single family of compatible standards that can be used worldwide for all mobile applications.
- Support for both packet-switched and circuit-switched data transmission.
- Data rates up to 2 Mbps (depending on mobility).
- High bandwidth efficiency

4G Mobile System

4G networks will be based on packet switching only. It will be able to support faster transmission. They are projected to provide speeds up to 100 Mbps while moving and 1Gbps while stationary. It is a wireless access technology. 4G can provide better-than-TV quality images and video-links.

Network Security Concepts

Network security deals with policies adopted by network administrator to protect the network from unauthorized access and misuse of network resources. It also ensures that the authorized users have adequate access to all the network resources.

Virus

If you observe that your system

- takes longer time to load applications
- shows unpredictable program behaviour
- shows inexplicable changes in file sizes
- has inability to boot,
- has strange graphics appearing on your screen

This could be because of your computer being infected by a virus.

Virus is a malicious program that attaches itself to the host program. It is designed to infect the host program and gain control over the system without the owner's knowledge. The virus gets executed each

time the host program is executed. Also it has the tendency to replicate. They can spread through external media such as CDs, browsing infected internet sites and from email attachments.

Types of Viruses

- **File Virus:** These viruses infect and replicate when it gets attached to MS-DOS program files with EXE or COM extensions.
- **Boot sector virus:** These viruses infect the boot sector of floppy disks or hard drives. Boot sector of a drive contains program that participates in booting the system. A virus can infect the system by replacing or attaching itself to these programs
- **Macro virus:** These viruses infect and replicate using the MS Office program suite, mainly MS Word and MS Excel. The virus inserts unwanted words or phrases in the document.

Worm

Worm is also a malicious program like a virus. But unlike viruses, it does not need to attach itself to a host program. A worm works by itself as an independent object.It uses security holes in a computer networks to replicate itself. A copy of the worm scans the network for another machine that has a specific security hole. It copies itself to the new machine using the security hole, and then starts replicating from there, as well.

Trojan horse

A Trojan horse is a program that contains hidden malicious functions. Trojan Horses trick users into installing them by appearing to be legitimate programs. Once installed on a system, they reveal their true nature and cause damage. Some Trojan horses will contact a central server and report back information such as passwords, user IDs, and captured keystrokes. Trojans lack a replication routine and thus are not viruses by definition.

Spam

The term spam means endless repetition of worthless text. In other words, unwanted messages or mails are known as Spam. At times internet is flooded with multiple copies of the same message, it is nothing but spam. Most spam is commercial advertising. In addition to wasting people's time, spam also eats up a lot of network bandwidth.

Cookies

When the user browses a website, the web server sends a text file to the web browser.This small text file is a cookie. Generally a cookie contains the name of the website from which it has come from and a unique ID tag.

Some cookies last only until the browser is closed. They are not stored on your hard drive. They are usually used to track the pages that you visit so that information can be customised for you for that visit. On the other hand, some cookies are stored on your hard drive until you delete them or they reach their expiry date. These may, for example, be used to remember your preferences when you use the website.

Firewall

A firewall is hardware or software based network security system. It prevents unauthorized access (hackers, viruses, worms etc.) to or from a network.

Firewalls are used to prevent unauthorized internet users to access private networks connected to the Internet. All data entering or leaving the Intranet pass through the firewall, which examines each packet and blocks those that do not meet the specified security criteria.

A firewall examines all traffic routed between the two networks to see if it meets certain criteria. If it does, it is routed between the networks, otherwise it is stopped. A firewall filters both inbound and outbound traffic. A firewall may allow all traffic through unless it meets certain criteria, or it may deny all traffic unless it meets certain criteria.

Cyber Crime

Cybercrime is defined as a crime in which a computer and internet is used in an illegitimate way to harm the user. Cyber criminals may use computer technology to access personal information, business trade secrets, or use the internet for exploitive or malicious purposes. Cybercrimes can be against persons or against property or against the government.

The list of Cyber Crimes includes

- harassment by computer (Cyber Stalking, defamation)
- pornography
- illegal downloads, plagiarism
- software piracy/counterfeiting, copyright violation of software, counterfeit hardware, black market sales of hardware and software, theft of equipment and new technologies
- fraud (credit card fraud, fraudulent use of ATM accounts, stock market transfers, telecommunications fraud), theft of (electronic) money

Cyber Law

Cyber law is an attempt to integrate the challenges presented by human activity on the internet with legal system of laws applicable to the physical world.

There was no statute in India for governing Cyber Laws involving privacy issues, jurisdiction issues, intellectual property rights issues and a number of other legal questions. With the tendency of misusing of technology, there has arisen a need of strict statutory laws to regulate the criminal activities in the cyber world and to protect the true sense of technology. "INFORMATION TECHNOLOGY ACT, 2000" [ITA-2000] was enacted by Parliament of India to protect the field of e-commerce, e-governance, e-banking as well as penalties and punishments in the field of Cyber Crimes. The above Act was further amended in the form of IT Amendment Act, 2008 [ITAA-2008].

In the IT Act the word 'computer' and 'computer system' have been so widely defined and interpreted to

mean any electronic device with data processing capability, performing computer functions like logical, arithmetic and memory functions with input, storage and output capabilities and therefore any high-end programmable gadgets like a washing machine or switches and routers used in a network can all be brought under the definition.

Some of the CYBER OFFENCES UNDER THE IT ACT

- Tampering with computer source documents - Section 65
- Hacking -Section 66
- Publishing of information which is obscene in electronic form -Section 67

Intellectual property rights (IPR) Issues

Intellectual property rights are the rights given to an individual over the invention of their own. They usually give the creator an exclusive right over the use of his/her creation for a certain period of time.

There are only three ways to protect intellectual property

1. Patents

A Patent is a term used for a specific product designed by an individual. The designer is given exclusive rights over the patent for a limited period of time. With help of the patent right, the owner can stop others from making, using or selling the product design. The owner can take a legal action if someone uses the patent without his/ her permission

In order to obtain a patent, the following conditions should be met:

- The product should be new
- It should be capable of being made or used in some kind of industry
- It should not be a scientific or mathematical discovery
- It should not be a dramatic, musical dramatic or artistic work

2. Trademarks

Trademark can be defined as a name or a different sign or a device identifying a product or a service. The product or the service is produced or provided by a specific person or a company. A Trademark is also known as brand name.It should be officially registered and legally restricted to the use of the specific person or the company.

3. Copyrights

Copyright is the term used for a written document. A legal action can be taken, if copyrights are violated. The following category of work can be considered for copyrights.

- literary works
- musical works, including any accompanying words

- dramatic works, including any accompanying music
- pantomimes and choreographic works
- pictorial, graphic and sculptural works
- motion pictures and other audio visual works
- sound recordings
- architectural works
- computer programs and websites

Hacking

The term hacking was first used at M.I.T during 1950s and 1960s. The term was used for people who engaged themselves in harmless technical experiments and fun learning activities.

A computer enthusiast, who uses his computer programming skills to intentionally access a computer without authorization is known as hacking. The computer enthusiast involved in this activity is known as a hacker. A hacker accesses the computer without the intention of destroying data or maliciously harming the computer.

Another term commonly used with hacking is cracking. Cracking can be defined as a method by which a person who gains unauthorized access to a computer with the intention of causing damage.

Introduction to Web Services

HTML(Hypertext Markup Language)

HTML is language the helps in creating and designing web content. It is a markup language. It has a variety of tags and attributes for defining the layout and structure of the web document. It is designed to display the data in formatted manner. A HTML document has the extension .htm or .html. Hypertext is a text which is linked to another document.

XML (EXtensible Markup Language)

XML is a markup language like HTML. It is designed to carry or store data. In contrast to HTML, it is not designed to display data. Unlike HTML, it does not have predefined tags. It is possible to define new tags in XML. It allows the programmer to use customized tags. XML is case sensitive. XML is deigned to be self-descriptive. XML is a W3C recommendation.

XML documents form a tree structure.

For Example

<root>

<child>

<subchild>.....</subchild>

</child>
</root>

WWW (World Wide Web):

WWW can be defined as a hypertext information retrieval system on the Internet. Tim Berners -Lee is the inventor of WWW. WWW is the universe of the information available on the internet.

WWW consists of web pages, which use HTML to interchange information on the internet. All the webpages on WWW use HTTP transfer protocol for any information with the capability for making hypertext jumps

Web page

Web page is an electronic document designed using HTML. It displays information in textual or graphical form. It may also contain downloadable data files, audio files or video files. Traversal from one webpage to another web page is possible through hyperlinks.

A web page can be classified into two types:

Static web page: A web page which displays same kind of information whenever a user visits it, is known as a static web page. A static web page generally has.htm or .html as extension

Dynamic web page: An interactive web page is a dynamic webpage. A dynamic web page uses scripting languages to display changing content on the web page. Such a page generally has php, .asp," or .jsp as extension.

A scripting language is a programming language which can be embedded or integrated with other languages. Some of the most widely used scripting languages are JavaScript, VBScript, PHP, Perl, Python, Ruby, and ASP. They have been used extensively to create dynamic web pages.

Dynamic web pages support two types of scripting:

- **Client-Side Scripting**

  On some web pages the contents change in response to an action done by the user, for example a click from the mouse or a key press from a keyboard action. Such pages use client-side scripting. In this technology, the content is generated on the user's local computer. VB Script and Java Script are examples of client-side scripting languages.

- **Server -Side Scripting**

  Some web pages use applications running on the server to generate the web content. Such pages use server-side scripting language. Web page display the current time and date, forums, submission forms, shopping carts etc., use server-side scripting. ASP,JSP, PHP are examples of server-side scripting languages.

Website: Related webpages from a single wen domain is termed as a website. A website has multiple

webpages providing information about a particular entity.

Web browser

Web browser is software program to navigate the web pages on the internet. A bowser interprets the coding language of the web page and displays it in graphic form. A web browser allows anyone to access the web without even knowing commands used in software languages to design a web page.

Internet works on client -server model. A web browser is a client which requests the information from the web server. The web server sends the information back to the client. The web address of the webpage written on the address bar tells the web browser which page to access.

Web Browser is of two types:

- Text based browsers
- Graphical browsers

URL (Uniform resource locator)

Web address of the web page written on the address bar of the browser is known as the uniform resource locator (URL). A URL is a formatted text string used to identify a network resource on the Internet. Network resources are files that can be plain Web pages, text documents, graphics, downloadable files, services or programs. Every network resource on the web has a unique URL.

The URL text string consists of three parts:

- network protocol
- host name or address
- file or resource location

The textstring of a URL has the following format:

protocol://server/path/resource

Network Protocol

The network protocol substring identifies the protocol to be used to access the network resource. These strings are short names followed by the three characters '://' . Other examples of protocols include http, gopher, wais, ftp and mailto.

URL Host/Server

The host name or address substring identifies the host/server that holds the resource. Hosts names are sometimes called domain names. For example: www. School.com is a domain name

Host names are mapped into numeric IP addresses. The domain name www.school.com may have IP address 192.2.100.1.An IP address is a binary number that uniquely identifies computers and other devices on a TCP/IP network. Services in the name of one host can be provided by many servers, which have

different IP addresses. One server, with one IP address, can provide services in the name of many hosts. So there is not a one-to-one relationship between host name and IP address. It is more convenient for the user to remember a numeric IP address than the domain name.

Host names are mapped to IP addresses by a server known as a DNS server, or domain nameserver. DNS stands for Domain Name Service. In a large network, many DNS servers may collaborate to provide the mapping between host names and IP addresses.

URL Resource Location

The file or resource location substring contains a path to one specific network resource on the host/server.Resources are normally located in a host directory or folder.

For example:www.school.com/syllabus/preprimary/nursery.htm is the location of this Web page including two subdirectories and the file name.

When the location element is omitted such as in http:// www.school.com/, the URL conventionally points to the root directory of the host and often a home page.

Web Server

A Web server is a computer or a group of computers that stores web pages on the internet.

It works on client/server model. It delivers the requested web page to web browser. Web servers use special programs such as Apache or IIS to deliver web pages over the http protocol.

Each server has a unique IP address and domain name. In order to access a webpage, the user writes the URL of the site on the address bar of the browser.The machine on which the browser is running sends a request to the IP address of the machine running the web server for that page. Once the web server receives that request, it sends the page content back to the IP address of the computer asking for it. The web browser then translates that content into all of the text, pictures, links, videos, etc.

A single web server may support multiple websites or a single website may be hosted on several linked servers.

Web hosting

Web hosting is the process of uploading/saving the web content on a web server to make it available on WWW. In case a individual or a company wants to make its website available on the internet, it should be hosted on a web server.

Web 2.0

The term web 2.0 was given by O'Reilly Media in 2004. Web 2.0 refers to new generation of dynamic and interactive websites. Web 2.0 websites uses a new programming language called AJAX (Asynchronous JavaScript and XML). AJAX helps a dynamic website connect to the web server and download small

amount of data based on the interaction with the user. In this technology only the part of the website which is updated is reloaded. The entire page does not get reloaded each time. This helps in making the website interactive.

Applications supported by web 2.0 are as followings:

- blogging
- social bookmarking
- RSS
- wikis and other collaborative applications
- interactive encyclopaedias and dictionaries
- Advanced Gaming

LETS REVISE

1G Mobile Systems: The 1G Mobile System was introduced in late 1970s and early 1980s.The 1G mobile system was based on the analog cellular technology. They only had voice facility available.

2G Mobile Systems: They used digital signals for transmissions of voice. 2G enabled the mobile systems to provide paging, SMS, voicemail and fax services.

3G Mobile Systems: The 3G technology adds multimedia facilities to 2G phones by allowing video, audio, and graphics applications.

4G Mobile Systems: 4G will provide better-than-TV quality images and video-links.

Virus: Virus is a malicious program that attaches itself to the host program. It is designed to infect the host program and gain control over the system without the owner's knowledge.

Worm: Worm is also a malicious program like a virus. But unlike viruses, it does not need to attach itself to a host program. A worm works by itself as an independent object.

Trojan horse: A Trojan horse is a program that contains hidden malicious functions. Trojan Horses trick users into installing them by appearing to be legitimate programs.

Spam: The term spam means endless repetition of worthless text. In other words, unwanted messages or mails are known as Spam.

Cookies: This small text file is a cookie. Generally a cookie contains the name of the website that it has come from and a unique ID tag.

Firewall: A firewall is hardware or software based network security system. It prevents unauthorized access (hackers, viruses, worms etc.) to or from a network.

Cyber Crime: Cybercrime is defined as a crime in which a computer and internet is used in an illegitimate way to harm the user.

Cyber Law: Cyber law is an attempt to integrate the challenges presented by human activity on the internet with legal system of laws applicable to the physical world.

Intellectual property rights are the rights given to an individual over the invention of their own. They usually give the creator an exclusive right over the use of his/her creation for a certain period of time

Intellectual property rights (IPR) Issues: Intellectual property rights are the rights given to an individual over the invention of their own. They usually give the creator an exclusive right over the use of his/her creation for a certain period of time.There are only three ways to protect intellectual property

- Patents
- Copyrights

2 Trademark

Hacking: The term was used for people who engaged themselves in harmless technical experiments and fun learning activities.

Cracking: Cracking can be defined as a method by which a person who gains unauthorized access to a computer with the intention of causing damage.

HyperText Transfer Protocol (HTTP): HTTP is the protocol that is used for transferring hypertext (i.e. text, graphic, image, sound, video etc.) between two computers and is particularly used on the World Wide Web. It is a TCP/IP based communication protocol and provides a standard for Web browsers and servers to communicate.

WWW (World Wide Web): WWW can be defined as a hypertext information retrieval system on the Internet. Tim Berners -Lee is the inventor of WWW. WWW is the universe of the information available on the internet.

Web page: Web page is an electronic document designed using HTML. It displays information in textual or graphical form. It may also contain downloadable data files, audio files or video files.

A web page can be classified into two types:

- Static web page
- Dynamic web page

Website: Related webpages from a single wen domain is termed as a website. A website has multiple webpages providing information about a particular entity.

Web browser: Web browser is software program to navigate the web pages on the internet. A bowser interprets the coding language of the web page and displays it in graphic form.

URL (Uniform resource locator): Web address of the web page written on the address bar of the browser is known as the uniform resource locator (URL).

Web hosting: Web hosting is the process of uploading/saving the web content on a web server to make it available on WWW.

Web 2.0: Web 2.0 refers to new generation of dynamic and interactive websites. Web 2.0 websites uses a new programming language called AJAX (Asynchronous JavaScript and XML).

EXERCISE

1. Differentiate between SMTP and POP3.
2. Give the full forms of the following terms:

2 CDMA

2 TDMA

2 FDMA

3. Briefly explain the generations in Mobile technologies.
4. Differentiate between Worm and Virus
5. Explain different types of viruses briefly.
6. Explain the following terms:
   - Spam
   - Cookies
   - Firewall
7. Explain the significance of IT Act.
8. Explain the following terms:
   - Patent
   - Copyright
   - Trademark
9. Differentiate between hacking and cracking
10. Mona is confused between the terms Domain name and URL. Explain the difference with the help of suitable example.
11. Identify the Domain name and URL from the following.

    http://www.ABCSchool.in/home.aboutus.hml

12. Mr. Rohan wants to prevent unauthorized access to/from his company's local area network. Write the name of the system, which he should install to do the same.
13. Define the following with reference to threats to network security.

    (i) Worm

    (ii) Trojan Horse

14. In this mode, each user has its own frequency domain. Write the name of this accessing mode.
15. In this mode, each user is allocated with a unique code sequence. Write the name of this accessing mode.
16. In this mode, each user is allowed to transmit data only within specified time intervals. Write the name of this accessing mode.
17. It means endless repetition of worthless text. In other words, it contains unwanted messages or mails. What is the name of this concept?
18. When the user browses a website, the web server sends a text file to the web browser. What is the name of this?
19. It is defined as a crime in which a computer and internet is used in an illegitimate way to harm the user. What is the name of this crime?
20. A person who gains unauthorized access to a computer with the intention of causing damage. What is the name of this crime?

Case Studies

Case Studies

Airline Reservation System

Fastest Fast Airlines wants to develop a software application to automate its reservation process to facilitate booking and cancellation process. The key points given by the CEO of the company are as follows:

a. The data of all the Fastest Fast flights includes details like flight no, departure city and time, arrival city and time, duration of flight, seat availability in business and economy class with their fares. The software should be able to add, modify and delete data from the permanent storage.

b. At the time of booking, the passenger details like name, age, sex, address, contact number, email id etc. have to be accepted and stored.

c. The booking amount is fixed at Rs. 2000/-.

d. Every cancellation to cost 50% of the booking amount.

e. At any time, one should be able to see the flight details and the seat availability.

f. If any flight is cancelled for any reason whatsoever, the entire booking amount should be refunded back to the passenger.

g. There should be an option to print the booking invoice/ticket.

Maths Tutorial cum Assessment Test

The maths department wants to make a tutorial- cum- assessment test for the students of class X, covering various mathematical concepts. The tutorial should cover the following topics:

a. Scientific Calculator

b. Linear equations

c. Quadratic equations

d. Arithmetic Progression

e. Triangles

f. Trignometric Applications

g. Mensuration

h. Statistics

i. Probability

The tutorial should explain the topic in brief with formulas and examples. The assessment test contain multiple choice questions(minimum 10) on each topic. The score of the students after completion of each

assessment should be displayed.

Binary search for solution of non-linear equation

Binary search is a technique used in the field of computer science to search for an element in a sorted list. It is very efficient algorithm. The algorithm can also be applied to find the root(s) of non-linear equations, as here we need to find a point where function

f(x) = 0

Assuming that f(x) is continues on [xi,xf], so f(xi)f(xf) <= 0. There will be x ? [xi,xf] such that f(x) = 0. So binary search can effectively find x lying between xi & xf satisfying our equation.

Write program(s) to find root(s) of ANY quadratic equation.

Minesweeper game

Implement the Minesweeper game in Python. Minesweeper is a popular computer game that comes free with Window's OS.

Before implanting game, play the game 5 times. This will help you in proper understanding of your project.

- To reduce the complexity of program you can fix the grid size to 6x6 and number of mines to 6.
- On the grid blank cell can be represented by 0 (if required)
- Program should have all the error checks.

Software Logging Module

Real World Software keeps a record of their activities while they are executing into a text file. This text file is called as log file.

Create a python module that provides functionality to record the following software activities to a log file

- When the function was called, i.e. Timestamp, function name.
- If any exception was thrown record the exception object along with data for later debugging,
- Categorized each activity as ERROR, WARNING, INFORMATION, DEBUG, FUNCTION_START, FUNCTION_END,
- Store all records to a single text file.

KNOW - AS YOU GROW

CENTRAL BOARD OF SECONDARY EDUCATION

Shiksha Kendra, 2, Community Centre, Preet Vihar, Delhi-110 092 India
Phone: 011 - 22509256 - 57 • Fax: 22515826 • Website: www.cbse.nic.in