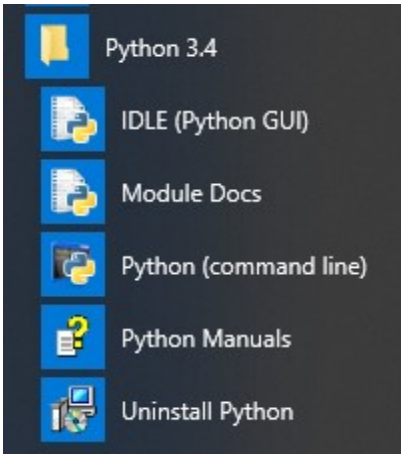
UNIT 3

Overview of Programming: Structure of a Python Program, Elements of Python
Introduction to Python: Python Interpreter, Using Python as calculator, Python shell, Indentation. Atoms, Identifiers and keywords, Literals, Strings, Operators (Arithmetic operator, Relational operator, Logical or Boolean operator, Assignment, Operator, Ternary operator, Bit wise operator, Increment or Decrement operator).
Creating Python Programs: Input and Output Statements, Control statements (Looping while Loop, for Loop , Loop Control, Conditional Statement- if...else, Difference between break, continue and pass).

Elements of Python

Python has 4 components



1. IDLE (Python GUI): It is a cross platform GUI Integrated Development Environmental that is provided with Python for editing and running a Python programs. It is a bundled set of software's tools such as Python Shell for Scripting programs and text editor for creating and modifying Python's source code, an integrated interactive debugger for finding error, help system etc.

2. Module Docs: This component allows us to access the python documents such as build in modules, DLLs, libraries, packages etc.

3. Python (Command line): Python can also be access from the command line. Command line mode provide very less features in comparison to IDLE but it is fast.

4. Python Manual : This component include various documents related to Python such as : installation cum setup guide, tutorials, Python API , FAQs etc.

Python Interpreter

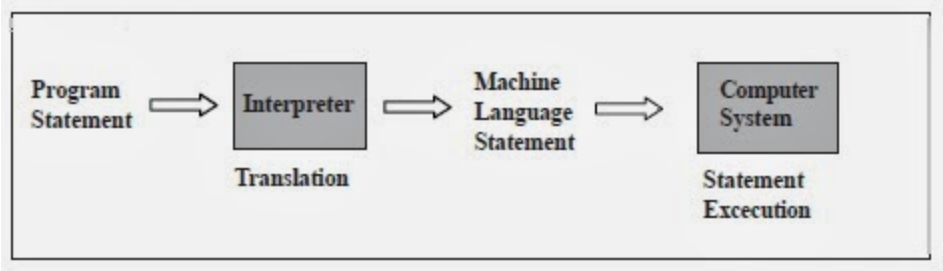
Python is a high level language. We know that computer understand machine language or binary language which is also known as low level language. The job to translate programming code written in high level language to low level language is done with the help of two type of softwares : compilers and interpreters.

Working of Compiler

Compiler is a special type of software it first check the source code of the complete program for the syntax errors, if no error found in the program code then the source code of the program is converted to the machine code and finally the executive file of the program is created. The executive file can be executed by the CPU and can produce desired result as per the program requirements.

Working of Interpreter

Interpreter check the code of program line by line if no error is found it is converted into the machine code and simultaneously gets executed by the CPU.



An interpreter is a computer program, which coverts each high-level program statement into the machine code. This includes source code, pre-compiled code, and scripts. Both compiler and interpreters do the same job which is converting higher level programming language to machine code. However, a compiler will convert the code into machine code (create an exe) before program run. Interpreters convert code into machine code when the program is run.

Atoms

Atoms are the most basic elements of expressions. The simplest atoms are identifiers or literals. Forms enclosed in reverse quotes or in parentheses, brackets or braces are also categorized syntactically as atoms. The syntax for atoms is:

```
atom:      identifier | literal | enclosure
enclosure: parenth_form|list_display|dict_display|string_conversion
```

Indentation

Indentation in Python refers to the (spaces and tabs) that are used at the beginning of a statement. The statements with the same indentation belong to the same group.

For Example

```
a=2
print(a)
if a==3 :
    print(“hello world”)
    print(“HELLO WORLD”)
else :
    print(“ bye world”)
    print(“ BYE WORLD”)
```

Structure of Python Program

The typical structure of a python program include 3 parts

Import statements
// import statements are used to include library files to the python program
Function definitions
//This section include the definitions of various functions written in a Python Program
Program statements

// This section include the set of statements for solving the given problem.

Identifier

Identifier is a name given to various programming elements such as a variable, function, class, module or any other object. Following are the rules to create an identifier.

1. The allowed characters are a-z, A-Z, 0-9 and underscore (_)
2. It should begin with an alphabet or underscore
3. It should not be a keyword
4. It is case sensitive
5. No blank spaces are allowed.
6. It can be of any size

Valid identifiers examples : si, rate_of_interest, student1, ageStudent
Invalid identifier examples : rate of interest, 1student, @age

Keywords

Keywords are the identifiers which have a specific meaning in python, there are 33 keywords in python. These may vary from version to version.

False	None	True	and
as	assert	break	class
continue	def	del	elif
else	except	finally	for
from	global	if	import
in	is	lambda	nonlocal
not	or	pass	raise
return	try	while	with
yield			

Variables and Types

When we create a program, we often need store values so that it can be used in a program. We use variables to store data which can be manipulated by the computer program.

Every variable has:

A. Name and memory location where it is stored.

A variable name:

1. Can be of any size
2. Have allowed characters, which are a-z, A-Z, 0-9 and underscore (_)
3. should begin with an alphabet or underscore
4. should not be a keyword

It is a good practice to follow these identifier naming conventions:

1. Variable name should be meaningful and short
2. Generally, they are written in lower case letters

B. A type or datatype which specify the nature of variable (what type of values can be stored in that variable and what operations can be performed)

We can check the type of the variable by using type command

```
>>> type(variable_name)
```

C. A value

Various datatypes available in python are as follow :

1. Number

Number data type stores Numerical Values. These are of three different types:

a) Integer & Long
b) Float/floating point
c) Complex

1.1 Integers are the whole numbers consisting of + or – sign like 100000, -99, 0, 17. While writing a large integer value

e.g. age=19
salary=20000

1.2 **Floating Point:** Numbers with fractions or decimal point are called floating point numbers. A floating point number will consist of sign (+,-) and a decimal sign(.)

e.g. temperature= -21.9,
growth_rate= 0.98333328

The floating point numbers can be represented in scientific notation such as

$-2.0X\ 10^{5}$ will be represented as -2.0e5
$2.0X10^{-5}$ will be 2.0E-5

1.3 **Complex:** Complex number is made up of two floating point values, one each for real and imaginary part. For accessing different parts of a variable x we will use x.real and x.imag. Imaginary part of the number is represented by j instead of i, so 1+0j denotes zero imaginary part.

Example

```
>>> x = 1+0j
>>> print x.real,x.imag
1.0 0.0
```

2. None

This is special data type with 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. Three types of sequence data type available in Python are Strings, Lists & Tuples.

3.1 **String:** is an ordered sequence of letters/characters. They are enclosed in single quotes (‘’) or double (“”).

Example

```
>>> a = 'Ram'
>>>a=”Ram”
```

3.2 **Lists:** List is also a sequence of values of any type. Values in the list are called elements / items. The items of list are accessed with the help of index (index start from 0). List is enclosed in square brackets.

Example

Student = [“Ajay”, 567, “CS”]

3.3 **Tuples:** Tuples are a sequence of values of any type, and are indexed by integers. They are immutable i.e. we cannot change the value of items of tuples . Tuples are enclosed in ().
Student = (“Ajay”, 567, “CS”)

4. Sets
Set is an unordered collection of values, of any type, with no duplicate entry. Sets are immutable.
Example
s = set ([1,2,3,4])

5. Dictionaries: Dictionaries store a key – value pairs, which are accessed using key. Dictionary is enclosed in curly brackets.
Example
d = {1:'a', 2:'b', 3:'c'}

Literals
A literal is a constant that appear directly in a program and this value does not change during the program execution i.e. remain fixed.
Example:
Num1=5
Principle_amount= 5000.00

Here 5 and 5000.00 are literals. Python support the following literals

- Numeric literals
- String Literals
- Special Literals
- Collection Literals

Operators and Operands
Operators are special symbols which represents specific operations. They are applied on operand(s), which can be values or variables. Same operator can behave differently on different data types. Operators when applied on operands form an expression. Following is the partial list of operators:
Arithmetic Operators : Arithmetic operators are used to apply arithmetic operation.

Symbol	Description	Example 1	Example 2
+	Addition	>>>55+45 100	>>> 'Good' + 'Morning' GoodMorning
-	Subtraction	>>>55-45 10	>>>30-80 -50
*	Multiplication	>>>55*45 2475	>>> 'Good'* 3 GoodGoodGood
/	Division	>>>17/5 3 >>>17/5.0 3.4 >>> 17.0/5 3.4	>>>28/3 9
%	Remainder/ Modulo	>>>17%5 2	>>> 23%2 1
**	Exponentiation	>>>2**3 8 >>>16**.5 4.0	>>>2**8 256
//	Integer Division	>>>7.0//2 3.0	>>>3/ / 2 1

Relational Operators : Relation operators are used to compare two items. The result of relational operator is True or False.

Symbol	Description	Example 1	Example 2

<	Less than	>>>7<10
		True
		>>> 7<5
		False
		>>> 7<10<15
		True
		>>>7<10 and 10<15
		True
			>>>'Hello'< 'Goodbye'
			False
			>>>'Goodbye'< 'Hello'
			True

>	Greater than	>>>7>5 True >>>10<10 False	>>>'Hello'> 'Goodbye' True >>>'Goodbye'> 'Hello' False

<=	less than equal to	>>> 2<=5
		True
		>>> 7<=4
		False
			>>>'Hello'<= 'Goodbye'
			False
			>>>'Goodbye' <= 'Hello'
			True

>=	greater than equal to	>>>10>=10 True >>>10>=12 False	>>>'Hello'>= 'Goodbye' True >>>'Goodbye' >= 'Hello' False
! =, <>	not equal to	>>>10!=11 True >>>10!=10 False	>>>'Hello'!= 'HELLO' True >>> 'Hello' != 'Hello' False
==	equal to	>>>10==10 True >>>10==11 False	>>>'Hello' == 'Hello' True >>>'Hello' == 'Good Bye' False

Note: Two values that are of different data type will never be equal to each other.

Logical Operators

Logical Operators give the logical relationship based upon the truth table.

AND			OR			NOT	
Value1	Value2	Result	Value1	Value2	Result	Value1	Result
False	False	False	False	False	False	True	False
False	True	False	False	True	True	False	True
True	False	False	True	False	True		
True	True	True	True	True	True		

Logical Operators

Symbol	Description
or	If any one of the operand is true, then the condition becomes true.
and	If both the operands are true, then the condition becomes true.
not	Reverses the state of operand/condition.

Bitwise Operators : Bitwise operators are applied upon the bits.

Operator	Description
& Binary AND	Operator copies a bit to the result if it exists in both operands
| Binary OR	It copies a bit if it exists in either operand.
^ Binary XOR	It copies the bit if it is set in one operand but not both.
~ Binary Ones Complement	It is unary and has the effect of 'flipping' bits.
<< Binary Left Shift	The left operands value is moved left by the number of bits specified by the right operand.
>> Binary Right Shift	The left operands value is moved right by the number of bits specified by the right operand.

```
a = 60            # 60 = 0011 1100
b = 13            # 13 = 0000 1101
c = 0

c = a & b;        # 12 = 0000 1100
print "Line 1 - Value of c is ", c

c = a | b;        # 61 = 0011 1101
print "Line 2 - Value of c is ", c

c = a ^ b;        # 49 = 0011 0001
print "Line 3 - Value of c is ", c

c = ~a;           # -61 = 1100 0011
print "Line 4 - Value of c is ", c

c = a << 2;       # 240 = 1111 0000
print "Line 5 - Value of c is ", c

c = a >> 2;       # 15 = 0000 1111
print "Line 6 - Value of c is ", c
```

Membership Operator

Python's membership operators test for membership in a sequence, such as strings, lists, or tuples. There are two membership operators as explained below −

Operator	Description	Example
in	Evaluates to true if it finds a variable in the specified sequence and false otherwise.	x in y, here in results in a 1 if x is a member of sequence y.
not in	Evaluates to true if it does not finds a variable in the specified sequence and false otherwise.	x not in y, here not in results in a 1 if x is not a member of sequence y.

Assignment Operators

Assignment operators are used to assign a value to the variable.

=	Assigned values from right side operands to left variable	>>>x=12* >>>y='greetings'	

*(*we will use it as initial value of x for following examples)*

+=	added and assign back the result to left operand	>>>x+=2	The operand/ expression/ constant written on RHS of operator is will change the value of x to 14
-=	subtracted and assign back the result to left operand	x-=2	x will become 10
*=	multiplied and assign back the result to left operand	x*=2	x will become 24
/=	divided and assign back the result to left operand	x/=2	x will become 6
%=	taken modulus using two operands and assign the result to left operand	x%=2	x will become 0
**=	performed exponential (power) calculation on operators and assign value to the left operand	x**=2	x will become 144
//=	performed floor division on operators and assign value to the left operand	x / /= 2	x will become 6

Identity Operator

Identity operators compare the memory locations of two objects. There are two Identity operators as explained below –

Operator	Description	Example
is	Evaluates to true if the variables on either side of the operator point to the same object and false otherwise.	x is y, here is results in 1 if id(x) equals id(y).
is not	Evaluates to false if the variables on either side of the operator point to the same object and true otherwise.	x is not y, here is not results in 1 if id(x) is not equal to id(y).

Operator Precedence :

Input and Output Statements

How to print/display a statement in Python

A statement is print with the help of print() method.
The syntax to use print() is as follow :

```
>>>print("message to be printed")
      Or
>>>print('message to be printed')
      Or
>>>print(variable_name)
      Or
>>>print("message to be printed", variable_name)
      Or
>>>print('message to be printed', variable_name)
```

How to input a value for the user in Python

A value can be input from the user with the help of input() method the syntax is as follow :

```
variable_name = input("Enter any number")
```

input method return a string. It can be changed to other datatypes by using type.

```
e.g.  age = int(input("Enter your age"))
      percentage= float(input("Enter you percentage"))
```

Control statements

Control structureis used to control the sequence of execution of statements written in a program. Without using Control structure sequence of executionof statements in a program is line line by and every statement is executed once .

In python we can divide the control structureinto two parts

1. conditional control structure
2. looping/iterative control structure

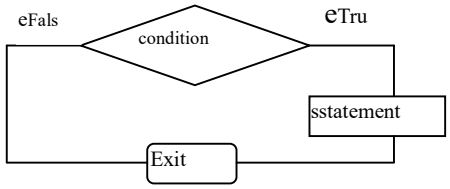
Conditional Control Structure

Conditional control structure is used in situations where the execution of statementsdepends upon the result of conditioni.e. trueor false

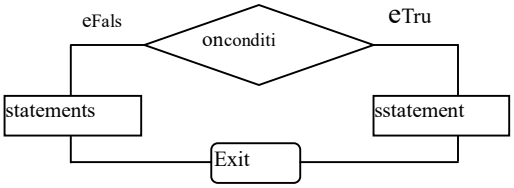
Conditional control structureis of 3 types

i) if ii) if....else iii) if.....elif..... else

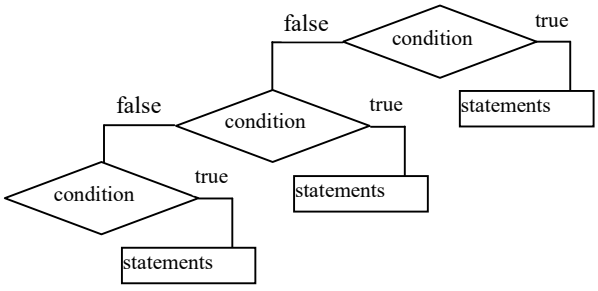
i) if: In this scontrol structure statement will execute only ifthe result of condition istrue .

flowchart	syntax
	if (condition) : Statement 1 Statement 2 …… Statement n

ii) if.... else: In this control structure for true and false result of condition different-different set of statements will execute.

flowchart	syntax
	if (condition) : Statement 1 Statement 2 …… Statement n else : Statement A Statement Z …… Statement N

iii) if.....elif..... else: This control structure is used in situations where on getting false result of condition we want to check if some other condition is meeting. We can increase the levels of elif according to the salutation of given problem. This control structure is also called laddered elif.

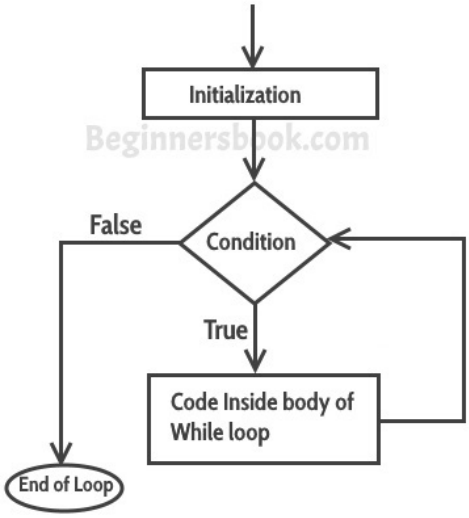
flowchart	syntax
	**if (condition) :** Statement 1 Statement 2 …… **elif (condition) :** Statement i Statement ii …… **elif (condition) :** Statement a Statement b …… **else** Statement A Statement B ……

Looping Control Structure : This control structure is used to execute a set of statements multiple times on the basis of condition. Loop is controlled by the loop counter. We need to initialize, update and use the counter in loop termination condition.

Python has two types of loops

1. While Loop : In this loop the body of the loop is executed again and again until the loop condition become false.

   Syntax 1 :
   While (condition):
   statements to be executed in while loop



In Python we can also use else in while loop, The statements written in this section gets executed when the loop condition become false.

Syntax 2 :
while (condition):
statements to be executed till the loop condition is true
else :
statements to be executed when the loop condition become false

FOR loop : For loops are used for sequential traversal. This loop is used when you know in advance that how many times the loop will execute.

Syntax:

for counter in sequence:

statements(s)

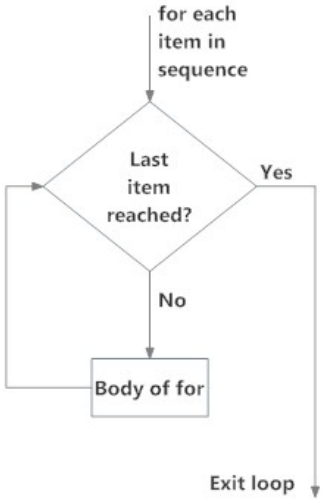


Fig: operation of for loop

Example
Program to find the sum of all numbers stored in a list

```
numbers = [6, 5, 3, 8, 4, 2, 5, 4, 11]
sum = 0
for val in numbers:
    sum = sum+val
print("The sum is", sum)
```

Break Statement

Break statement is used to force fully terminate a loop. The use of this statement can be easily understood with the help of following example.

Write a program to demonstrate the use of break

```
a=1
while a<=10:
    print(a)
    a=a+1
    if a==5:
        break
```

output: 1,2,3,4

Continue Statement

Continue statement is used to skip the execution of loop statements. The use of this statement can be easily understood with the help of following example.

Write a program to demonstrate the use of continue

```
a=1
while a<=10:
        if a==5:
            a=a+1
            continue
        print(a)
        a=a+1
```

output: 1,2,3,4,6,7,8,9,10

Pass Statement

Pass statement is used in situations where there are no statements for execution in a particular section of program but for following syntax rules the inclusion of section is must. The use of this statement can be easily understood with the help of following example.

Write a program to demonstrate the use of pass

```
a=1
while a<=10:
        if a%5==0:
            pass
        else:
            print(a)
a=a+1
```

output: 1,2,3,4,6,7,8,9

Computer Programs

Formula Based Simple Programs

1. Write a program to find Simple Interest

```
p=int(input("Enter the principle amount"))
r=int(input("Enter the rate of interest"))
t=int(input("Enter the time"))
si=(p*r*t)/100
print("Simple Interest = ",si)
```

2. Write a program to find area of a circle

```
r= float(input("Enter the radius of circle"))
area_of_circle= 22/7*r*r
print("Area of circle = ", area_of_circle)
```

3. Write a program to convert temperature from Celsius to Fahrenheit

```
C= float(input("Enter the temperature in celsius"))
```

```
F=(C*9/5)+32
print("The temperature in Fahrenheit is = ", F)
```

Condition Based Simple Programs

1. Write a program to check whether a person is eligible for voting or note by checking his age.

```
age=int(input("Enter your age : "))
if  age>=18 :
        print("You can vote")
else :
        print("You cannot vote")
```

2. Write a program to find whether a year is leap year or not .

```
year1= int(input("Enter  any year"))
if year1%400 ==0 :
        print("It is a leap year")
elif  year1%4==0 and year1%100 !=0 :
        print("It is a leap year")
else :
        print("It is not a leap year")
```

3. Write a program to find the grade of a student based upon his percentage based upon following conditions.

Percentage >= 80 A+ grade
Percentage>= 60 A grade
Percentage >= 50 B grade
Percentage >=40 C grade
Percentage>=35 D grade
Percentage>=0 F grade
Other : invalid percentage entered

```
percentage= float(input("Enter your percentage : "))
if percentage>=80 and percentage<=100 :
        print("Your grade is A+")
elif  percentage>=60:
        print("Your grade is A")
elif  percentage>=50:
        print("Your grade is B")
elif  percentage>=40:
        print("Your grade is C")
elif  percentage>=35:
        print("Your grade is D")
elif  percentage>=0:
        print("Your grade is F")
else :
        print("You have entered a wrong percentage")
```

Q4. Write a program to find the largest number among 3 numbers.

```
num1=int(input("Enter 1st number"))
num2=int(input("Enter 2nd number"))
num3=int(input("Enter 3rd number"))
```

```
if num1>num2 :
        if num1>num3:
                print (num1 , “is greatest”)
        else
                print(num3, “is greatest”)
else :
        if num2>num3:
                print (num2 , “is greatest”)
        else
                print(num3, “is greatest”)
```

Loop Based Simple Programs

Q1. Write a program to print following series : 1,3,5……99

```
i=1
while i<=100:
        print(  i )
        i=i+2
```

Q2. Write a program to find the sum from 1 to 100

```
num=1
sum1=0
while num<=100 :
        sum1=sum1+num
        num=num+1
print(“Sum from 1 to 100 = ” sum1)
```

Q3. Write a program to find the product from 1 to 100

```
num=1
product=1
while num<=100 :
        product=product*num
        num=num+1
print(“ Product from 1 to 100 = ” product)
```

Q4. Write a program to find the factorial of a number

```
num=int(input(“Enter the number whose factorial you want to find”))
fact=1
i=num
while i>=1:
        fact=fact*i
        i=i-1
print(“Factorial of   ”, num ,” = ”, fact)
```

Q5. Write a program to print Fibonacci Series

```
a=0
b=1
num=int(input(“Enter the number of elements of Fibonacci series you want to print”))
i=1
while i<=num :
        if i==1:
                print (a)
```

```
    elif i==2:
        print(b)
    else
        c=a+b
        print (c)
        a=b
        b=c
```

Q6. Write a program to check whether a number is prime or not

```
num=int(input("Enter the number you want to check for prime number"))
factor=0
i=1
while i<=num :
    if num%i==0:
        factor=factor+1
    i=i+1
if factor==2:
    print("This number is a prime number")
else :
    print("This number is not a prime number")
```

Q7. Write a program to find the product of given 10 numbers

```
numbers=[23,2,34,45,21,4,5,6,12,9]
result=1
for s in numbers:
    result=result*number
print("The product of given 10 numbers = ",result)
```

Q8. Write a program to check whether a number is palindrome (same as its reverse) or not

```
num=int(input("Enter a number "))
num1=0
num2=num
while num2>0:
     num1=num1*10
     num1=num1+(num2%10)
     num2=num2//10
if num1==num:
     print(num ," is a palindrome")
else:
     print(num, " is not a palindrome")
```