Computer Science

Class XII (As per CBSE Board)

New syllabus 2023-24

Chapter 1

Revision of the Basics of Python

Introduction

It is widely used general purpose,high level programming language.Developed by Guido van Rossum in 1991.

It is used for:

software development,
web development (server-side),
system scripting,
Mathematics.

Features of Python

1. **Easy to use** – Due to simple syntax rule
2. **Interpreted language** – Code execution & interpretation line by line
3. **Cross-platform language** – It can run on windows,linux,macinetosh etc. equally
4. **Expressive language** – Less code to be written as it itself express the purpose of the code.
5. **Completeness** – Support wide rage of library
6. **Free & Open Source** – Can be downloaded freely and source code can be modify for improvement

Shortcomings of Python

1. Lesser libraries – as compared to other programming languages like c++,java,.net
2. Slow language – as it is interpreted languages,it executes the program slowly.
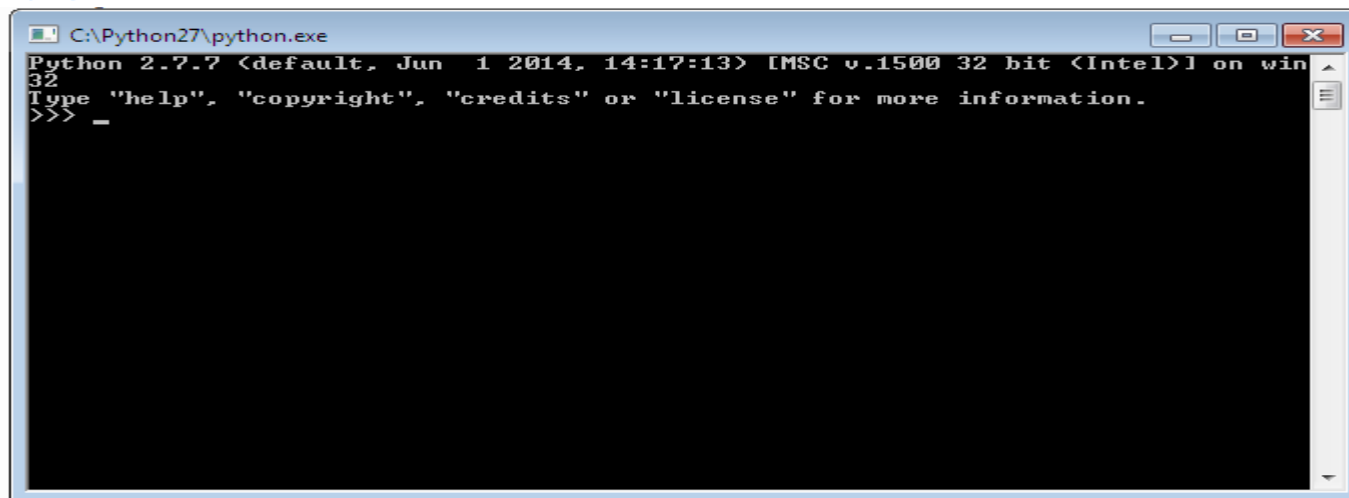3. Weak on Type-binding – It not pin point on use of a single variable for different data type.

How to work in Python

(i) in Interactive mode

* Search the **python.exe** file in the drive in which it is installed.
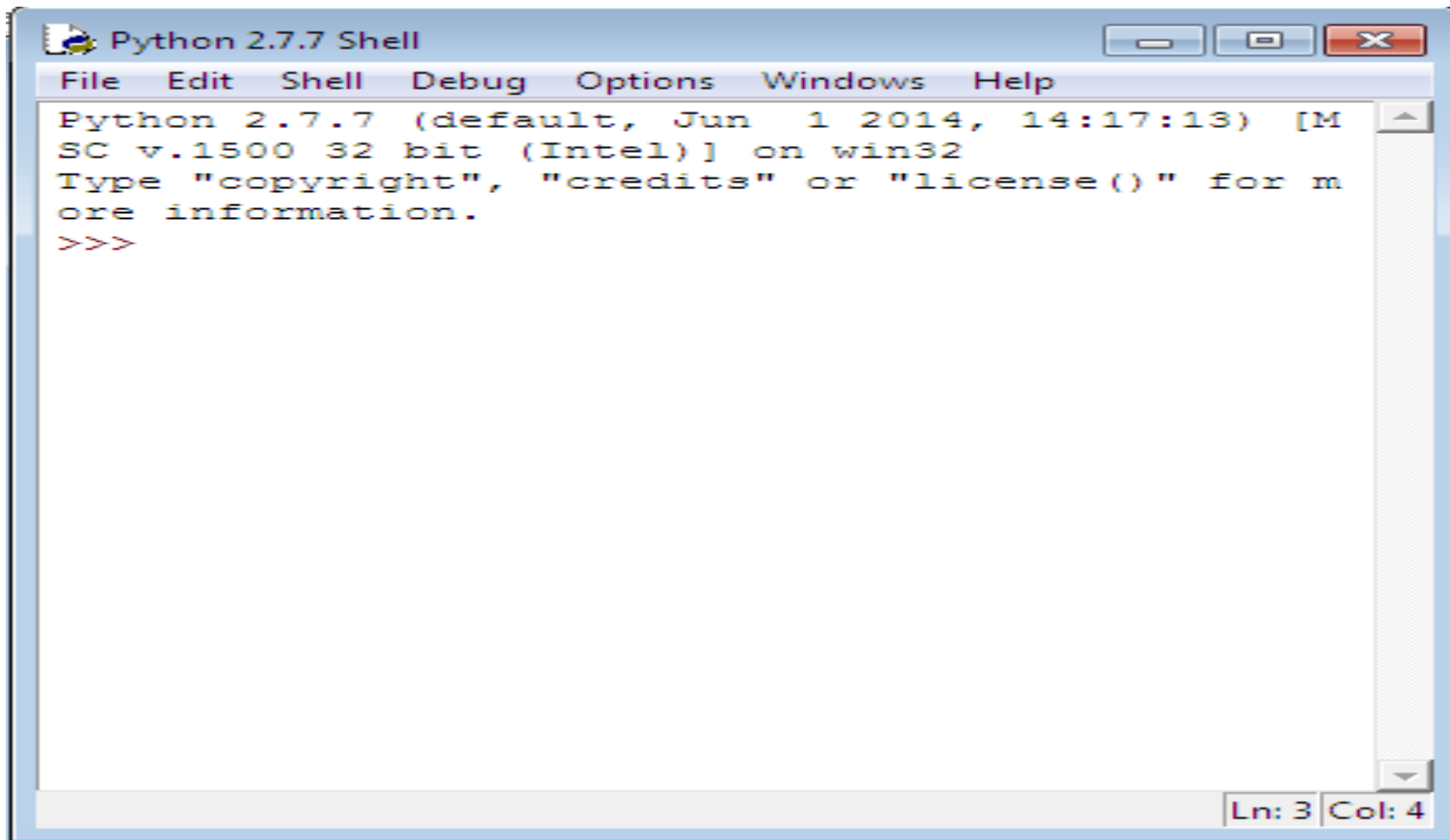If found double click it to start python in interactive mode

How to work in Python

* Click start button -> All programs -> python<version>->IDLE(Python GUI)

How to work in Python

Python command prompt >>>
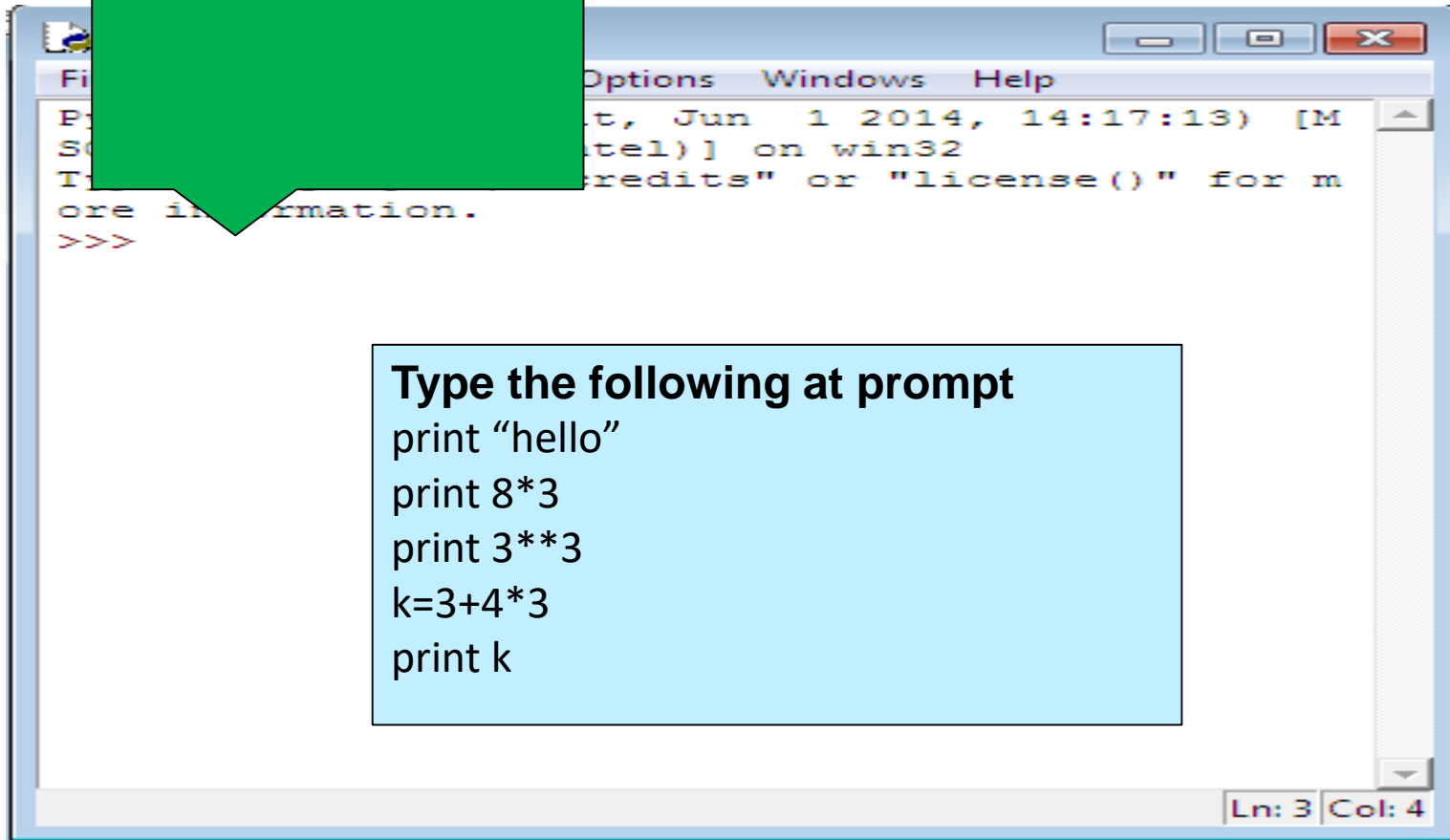
```
Fi          Options   Windows   Help
P          t,  Jun   1 2014,  14:17:13)  [M
S          tel)]  on  win32
T          redits"  or  "license()"  for  m
ore  i   rmation.
>>>
```

Type the following at prompt
print "hello"
print 8*3
print 3**3
k=3+4*3
print k

Ln: 3 Col: 4

How to work in Python

(ii) in Script mode

Step 1 (Create program file)

Below steps are for simple hello world program

a. Click Start button->All Programs ->
Python<version>->IDLE

b. Now click File->New in IDLE Python Shell

Now type

print "hello"

print "world"

print "python is","object oriented programming lang."

c. Click File->Save and then save the file with filename
and .py extension

How to work in Python

(ii) in Script mode

Step 2 (Run program file)

a.	Click Open command from IDLE's File menu and select the file you have already saved

b.	Click Run-> Run Module

c.	It will execute all the commands of program file and display output in separate python shell window

Note :- Python comes in 2 flavours – python 2.x and python 3.x . Later one is Backward incompatible language as decide by Python Software foundation(PSF). Mean code written in 2.x will not execute on 3.x . Visit the below link for difference between 2.x & 3.x

https://www.geeksforgeeks.org/important-differences-between-python-2-x-and-python-3-x-with-examples/

Data Handling

Most of the computer programming language support data type, variables,operator and expression like fundamentals.Python also support these.

Data Types

Data Type specifies which type of value a variable can store. type() function is used to determine a variable's type in Python.

Data type continue

Data Types In Python

1. Number
2. String
3. Boolean
4. List
5. Tuple
6. Set
7. Dictionary

Data type continue

1. Number In Python

It is used to store numeric values

Python has three numeric types:
1. Integers
2. Floating point numbers
3. Complex numbers.

Data type continue

1. Integers

Integers or int are positive or negative numbers with no decimal point. Integers in Python 3 are of unlimited size.

e.g.
a= 100
b= -100
c= 1*20
print(a)
print(b)
print(c)
Output :-
100
-100
200

Data type continue

Type Conversion of Integer

int() function converts any data type to integer.

e.g.

a = "101" # string

b=int(a) # converts string data type to integer.

c=int(122.4) # converts float data type to integer.

print(b)

print(c)Run Code

Output :-

101

122

2. Floating point numbers

It is a positive or negative real numbers with a decimal point.

e.g.
```
a = 101.2
b = -101.4
c = 111.23
d = 2.3*3
print(a)
print(b)
print(c)
print(d)Run Code
Output :-
101.2
-101.4
111.23
6.8999999999999995
```

Data type continue

Type Conversion of Floating point numbers

float() function converts any data type to floating point number.

e.g.

a='301.4' #string
b=float(a) #converts string data type to floating point number.
c=float(121) #converts integer data type to floating point number.
print(b)
print(c)Run Code

Output :-
301.4
121.0

Data type continue

3. Complex numbers

Complex numbers are combination of a real and imaginary part.Complex numbers are in the form of X+Yj, where X is a real part and Y is imaginary part.

e.g.

a = complex(5) # convert 5 to a real part val and zero imaginary part
print(a)
b=complex(101,23) #convert 101 with real part and 23 as imaginary part
print(b)Run Code

Output :-
(5+0j)
(101+23j)

Data type continue

2. String In Python

A string is a sequence of characters. In python we can create string using single (' ') or double quotes (" ").Both are same in python. e.g.

```
str='computer science'
print('str-',   str) # print string
print('str[0]-',   str[0]) # print first char 'h'
print('str[1:3]-',   str[1:3]) # print string from postion 1 to 3 'ell'
print('str[3:]-',   str[3:]) # print string staring from 3rd char 'llo world'
print('str *2-',   str *2 ) # print string two times
print("str +'yes'-",   str +'yes') # concatenated string
Output
str- computer science
str[0]- c
str[1:3]- om
str[3:]- puter science
str *2- computer sciencecomputer science
str +'yes'- computer scienceyes
```

Iterating through string

e.g.
str='comp sc'
for i in str:
    print(i)
Output
c
o
m
p

s
c

Data type continue

3. Boolean In Python

It is used to store two possible values either true or false

e.g.

str="comp sc"

boo=str.isupper() # test if string contains upper case

print(boo)

Output

False

Data type continue

4.List In Python

List are collections of items and each item has its own index value.

5. Tuple In Python

List and tuple, both are same except ,a list is mutable python objects and tuple is immutable Python objects. Immutable Python objects mean you cannot modify the contents of a tuple once it is assigned.

e.g. of list
list =[6,9]
list[0]=55
print(list[0])
print(list[1])

OUTPUT
55
9

e.g. of tuple
tup=(66,99)
Tup[0]=3 # error message will be displayed
print(tup[0])
print(tup[1])

6. Set In Python

It is an unordered collection of unique and immutable (which cannot be modified)items.

e.g.
set1={11,22,33,22}
print(set1)

Output
{33, 11, 22}

Data type continue

7. Dictionary In Python

It is an unordered collection of items and each item consist of a key and a value.

e.g.

```
dict = {'Subject': 'comp sc', 'class': '11'}
print(dict)
print ("Subject : ", dict['Subject'])
print ("class : ", dict.get('class'))
```

Output

{'Subject': 'comp sc', 'class': '11'}

Subject : comp sc

class : 11

Operator

Operators are special symbols in Python that carry out arithmetic or logical computation. The value that the operator operates on is called the operand.

Arithmetic operators

Used for mathematical operation

Operator	Meaning	Example
+	Add two operands or unary plus	x + y +2
-	Subtract right operand from the left or unary minus	x - y -2
*	Multiply two operands	x * y
/	Divide left operand by the right one (always results into float)	x / y
%	Modulus - remainder of the division of left operand by the right	x % y (remainder of x/y)
//	Floor division - division that results into whole number adjusted to the left in the number line	x // y
**	Exponent - left operand raised to the power of right	x**y (x to the power y)

Arithmatic operator continue

e.g.

x = 5

y = 4

print('x + y =',x+y)

print('x - y =',x-y)

print('x * y =',x*y)

print('x / y =',x/y)

print('x // y =',x//y)

print('x ** y =',x**y)

OUTPUT

('x + y =', 9)

('x - y =', 1)

('x * y =', 20)

('x / y =', 1)

('x // y =', 1)

('x ** y =', 625)

- Write a program in python to calculate the simple interest based on entered amount ,rate and time

Operator continue

Arithmatic operator continue

EMI Calculator program in Python

```python
def emi_calculator(p, r, t):
    r = r / (12 * 100) # one month interest
    t = t * 12 # one month period
    emi = (p * r * pow(1 + r, t)) / (pow(1 + r, t) - 1)
    return emi

# driver code
principal = 10000;
rate = 10;
time = 2;
emi = emi_calculator(principal, rate, time);
print("Monthly EMI is= ", emi)
```

Operator continue

Arithmatic operator continue

How to calculate GST

        GST (Goods and Services Tax) which is included in netprice of product for get GST % first need to calculate GST Amount by subtract original cost from Netprice and then apply

GST % formula = (GST_Amount*100) / original_cost

```python
# Python3 Program to compute GST from original and net prices.
 def Calculate_GST(org_cost, N_price):
# return value after calculate GST%
    return (((N_price - org_cost) * 100) / org_cost);
# Driver program to test above functions
org_cost = 100
N_price = 120
print("GST = ",end='')
print(round(Calculate_GST(org_cost, N_price)),end='')
print("%")
```

* Write a Python program to calculate the standard deviation

Operator continue

Comparison operators -used to compare values

Operator	Meaning	Example
>	Greater that - True if left operand is greater than the right	x > y
<	Less that - True if left operand is less than the right	x < y
==	Equal to - True if both operands are equal	x == y
!=	Not equal to - True if operands are not equal	x != y
>=	Greater than or equal to - True if left operand is greater than or equal to the right	x >= y
<=	Less than or equal to - True if left operand is less than or equal to the right	x <= y

Operator continue

Comparison operators continue

e.g.
x = 101
y = 121
print('x > y is',x>y)
print('x < y is',x<y)
print('x == y is',x==y)
print('x != y is',x!=y)
print('x >= y is',x>=y)
print('x <= y is',x<=y)

Output
('x > y is', False)
('x < y is', True)
('x == y is', False)
('x != y is', True)
('x >= y is', False)
('x <= y is', True)

Operator continue

Logical operators

Operator	Meaning	Example
and	True if both the operands are true	x and y
or	True if either of the operands is true	x or y
not	True if operand is false (complements the operand)	not x

e.g.
x = True
y = False
print('x and y is',x and y)
print('x or y is',x or y)
print('not x is',not x)
Outpur
('x and y is', False)
('x or y is', True)
('not x is', False)

Operator continue

Bitwise operators

Used to manipulate bit values.

| Operator	Meaning	Example
&	Bitwise AND	x& y
|	Bitwise OR	x | y
~	Bitwise NOT	~x
^	Bitwise XOR	x ^ y
>>	Bitwise right shift	x>> 2
<<	Bitwise left shift	x<< 2

Bitwise operators continue

```
a = 6
b = 3
print ('a=',a,':',bin(a),'b=',b,':',bin(b))
c = 0
c = a & b;
print ("result of AND is ", c,':',bin(c))
c = a | b;
print ("result of OR is ", c,':',bin(c))
c = a ^ b;
print ("result of EXOR is ", c,':',bin(c))
c = ~a;
print ("result of COMPLEMENT is ", c,':',bin(c))
c = a << 2;
print ("result of LEFT SHIFT is ", c,':',bin(c))
c = a >> 2;
print ("result of RIGHT SHIFT is ", c,':',bin(c))
```

Output
('a=', 6, ':', '0b110', 'b=', 3, ':', '0b11')
('result of AND is ', 2, ':', '0b10')
('result of OR is ', 7, ':', '0b111')
('result of EXOR is ', 5, ':', '0b101')
('result of COMPLEMENT is ', -7, ':', '-0b111')
('result of LEFT SHIFT is ', 24, ':', '0b11000')
('result of RIGHT SHIFT is ', 1, ':', '0b1')
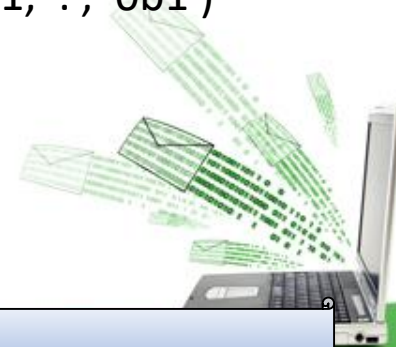
Operator continue

Python Membership Operators
Test for membership in a sequence

Operator	Description
in	Evaluates to true if it finds a variable in the specified sequence and false otherwise.
not in	Evaluates to true if it does not finds a variable in the specified sequence and false otherwise.

```
e.g.
a = 5
b = 10
list = [1, 2, 3, 4, 5 ]
if ( a in list ):
   print ("Line 1 - a is available in the given list")
else:
   print ("Line 1 - a is not available in the given list")
if ( b not in list ):
   print ("Line 2 - b is not available in the given list")
else:
   print ("Line 2 - b is available in the given list")
```

output
Line 1 - a is available in the given list
Line 2 - b is not available in the given list

Python Identity Operators

Operator	Description
is	Evaluates to true if the variables on either side of the operator point to the same object and false otherwise.
is not	Evaluates to false if the variables on either side of the operator point to the same object and true otherwise.

```
e.g.
a = 10
b = 10
print ('Line 1','a=',a,':',id(a), 'b=',b,':',id(b))
if ( a is b ):
   print ("Line 2 - a and b have same identity")
else:
   print ("Line 2 - a and b do not have same identity")
OUTPUT
('Line 1', 'a=', 10, ':', 20839436, 'b=', 10, ':', 20839436)
Line 2 - a and b have same identity
```

Operator continue

Operators Precedence : highest precedence to lowest precedence table

Operator	Description
**	Exponentiation (raise to the power)
~ + -	Complement, unary plus and minus (method names for the last two are +@ and -@)
* / % //	Multiply, divide, modulo and floor division
+ -	Addition and subtraction
>> <<	Right and left bitwise shift
&	Bitwise 'AND'td>
^ |	Bitwise exclusive `OR' and regular `OR'
<= < > >=	Comparison operators
<> == !=	Equality operators
= %= /= //= -= += *= **=	Assignment operators
is is not	Identity operators
in not in	Membership operators
not or and	Logical operators

Expression

It is a valid combination of operators,literals and variable.

1. Arithmatic expression :- e.g. c=a+b
2. Relational expression :- e.g. x>y
3. Logical expression :- a or b
4. String expression :- c="comp"+"sc"

Type conversion

The process of converting the value of one data type (integer, string, float, etc.) to another data type is called type conversion.

Python has two types of type conversion.

Implicit Type Conversion
Explicit Type Conversion

Implicit Type Conversion:

In Implicit type conversion, Python automatically converts one data type to another data type. This process doesn't need any user involvement.

e.g.
num_int = 12
num_flo = 10.23
num_new = num_int + num_flo
print("datatype of num_int:",type(num_int))
print("datatype of num_flo:",type(num_flo))
print("Value of num_new:",num_new)
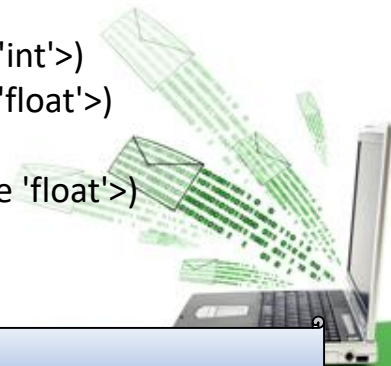print("datatype of num_new:",type(num_new))

OUTPUT
('datatype of num_int:', <type 'int'>)
('datatype of num_flo:', <type 'float'>)
('Value of num_new:', 22.23)
('datatype of num_new:', <type 'float'>)

Type conversion

Explicit Type Conversion:

In Explicit Type Conversion, users convert the data type of an object to required
data type. We use the predefined functions like int(),float(),str() etc.

e.g.

```
num_int = 12
num_str = "45"
print("Data type of num_int:",type(num_int))
print("Data type of num_str before Type Casting:",type(num_str))
num_str = int(num_str)
print("Data type of num_str after Type Casting:",type(num_str))
num_sum = num_int + num_str
print("Sum of num_int and num_str:",num_sum)
print("Data type of the sum:",type(num_sum))
OUTPUT
('Data type of num_int:', <type 'int'>)
('Data type of num_str before Type Casting:', <type 'str'>)
('Data type of num_str after Type Casting:', <type 'int'>)
('Sum of num_int and num_str:', 57)
('Data type of the sum:', <type 'int'>)
```

math module

It is a standard module in Python. To use mathematical functions of this module, we have to import the module using import math.

Function	Description	Example
ceil(n)	It returns the smallest integer greater than or equal to n.	math.ceil(4.2) returns 5
factorial(n)	It returns the factorial of value n	math.factorial(4) returns 24
floor(n)	It returns the largest integer less than or equal to n	math.floor(4.2) returns 4
fmod(x, y)	It returns the remainder when n is divided by y	math.fmod(10.5,2) returns 0.5
exp(n)	It returns e**n	math.exp(1) return 2.718281828459045
log2(n)	It returns the base-2 logarithm of n	math.log2(4) return 2.0
log10(n)	It returns the base-10 logarithm of n	math.log10(4) returns 0.6020599913279624
pow(n, y)	It returns n raised to the power y	math.pow(2,3) returns 8.0
sqrt(n)	It returns the square root of n	math.sqrt(100) returns 10.0
cos(n)	It returns the cosine of n	math.cos(100) returns 0.8623188722876839
sin(n)	It returns the sine of n	math.sin(100) returns -0.5063656411097588
tan(n)	It returns the tangent of n	math.tan(100) returns -0.5872139151569291
pi	It is pi value (3.14159...)	It is (3.14159...)
e	It is mathematical constant e (2.71828...)	It is (2.71828...)

Control Statements

Control statements are used to control the flow of execution depending upon the specified condition/logic.

There are three types of control statements.

1. Decision Making Statements
2. Iteration Statements (Loops)
3. Jump Statements (break, continue, pass)

Decision Making Statement

Decision making statement used to control the flow of execution of program depending upon condition.

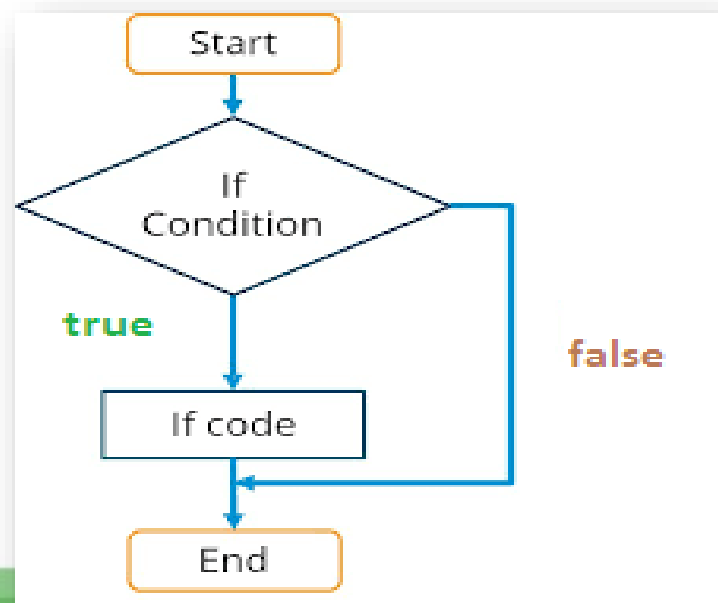There are three types of decision making statement.
1. if statements
2. if-else statements
3. Nested if-else statement

Decision Making Statement

1. if statements

An if statement is a programming conditional statement that, if proved true, performs a function or displays information.

Decision Making Statement

1. if statements

    Syntax:

    if(condition):

              statement

              [statements]

    e.g.

    noofbooks = 2

    if (noofbooks == 2):

              print('You have ')

              print('two books')

    print('outside of if statement')

    Output

    You have two books

Note:To indicate a block of code in Python, you must indent each line of the block by the same amount. In above e.g. both print statements are part of if condition because of both are at same level indented but not the third print statement.

Decision Making Statement

1. if statements

Using logical operator in if statement

```
x=1
y=2
if(x==1 and y==2):
  print('condition matcing the criteria')
```

Output :-
condition matcing the criteria

```
a=100
if not(a == 20):
    print('a is not equal to 20')
```
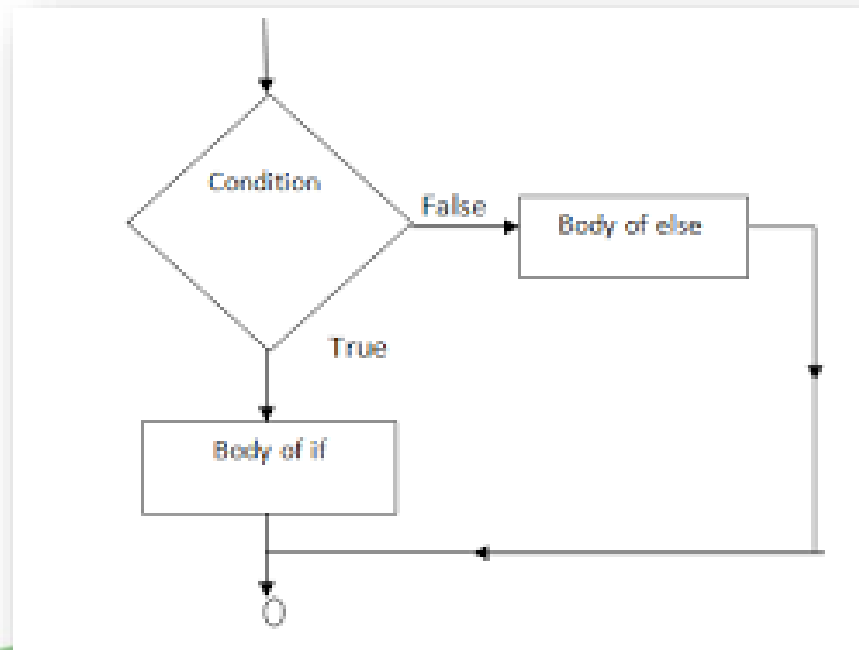
Output :-
a is not equal to 20

Decision Making Statement

2. if-else Statements

If-else statement executes some code if the test expression is true (nonzero) and some other code if the test expression is false.

Decision Making Statement

2. if-else Statements

Syntax:

        if(condition):

               statements

       else:

               statements

e.g.

a=10

if(a < 100):

  print('less than 100')

else:

  print('more than equal 100')
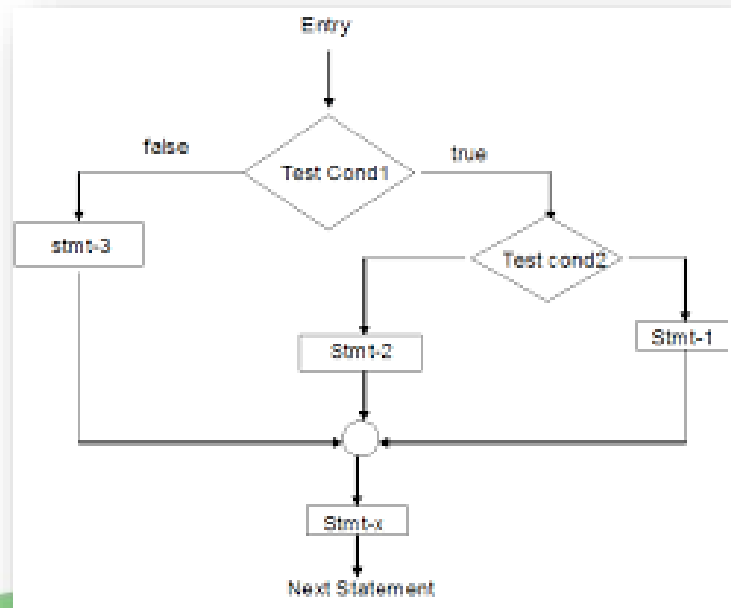
OUTPUT

less than 100

*Write a program in python to check that entered numer is even or odd

Decision Making Statement

3. Nested if-else statement
The nested if...else statement allows you to check for multiple test expressions and execute different codes for more than two conditions.

Decision Making Statement

3. Nested if-else statement

Syntax

If (condition):

     statements

elif (condition):

     statements

else:

     statements

<u>E.G.</u>

```python
num = float(input("Enter a number: "))
if num >= 0:
    if num == 0:
        print("Zero")
    else:
        print("Positive number")
else:
    print("Negative number")
```

OUTPUT

Enter a number: 5

Positive number

* Write python program to find out largest of 3 numbers.

Iteration Statements (Loops)

Iteration statements(loop) are used to execute a block of statements as long as the condition is true.
Loops statements are used when we need to run same code again and again.

Python Iteration (Loops) statements are of three type :-

1. While Loop

2. For Loop

3. Nested For Loops

Iteration Statements (Loops)

1. While Loop

It is used to execute a block of statement as long as a given condition is true. And when the condition become false, the control will come out of the loop. The condition is checked every time at the beginning of the loop.

Syntax
while (condition):
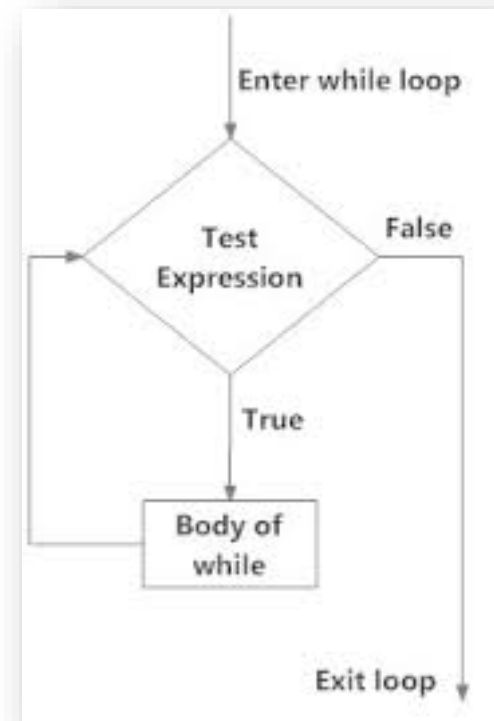   statement
        [statements]

e.g.

x = 1
while (x <= 4):
  print(x)
  x = x + 1

Output
1
2
3
4

Iteration Statements (Loops)

While Loop With Else

e.g.

```
x = 1
while (x < 3):
   print('inside while loop value of x is ',x)
   x = x + 1
else:
   print('inside else value of x is ', x)
```

Output
inside while loop value of x is 1
inside while loop value of x is 2
inside else value of x is 5
*Write a program in python to find out the factorial of a given number

Iteration Statements (Loops)

While Loop continue

      Infinite While Loop

e.g.

x = 5

while (x == 5):

  print('inside loop')

Output

Inside loop

Inside loop

...

...

Iteration Statements (Loops)

2. For Loop

It is used to iterate over items of any sequence, such as a list or a string.

Syntax

for val in sequence:

      statements

e.g.

for i in range(3,5):

  print(i)

Output

3

4

Iteration Statements (Loops)

2. For Loop continue

Example programs

```
for i in range(5,3,-1):
    print(i)
```

Output
5

4

range() Function Parameters

start: Starting number of the sequence.

stop: Generate numbers up to, but not including this number.

step(Optional): Determines the increment between each numbers in the sequence.

Iteration Statements (Loops)

2. For Loop continue

For Loop With Else

e.g.

```
for i in range(1, 4):
    print(i)
else:  # Executed because no break in for
    print("No Break")
```

Output

1

2

3

4

Iteration Statements (Loops)

2. For Loop continue

Nested For Loop

e.g.
```
for i in range(1,3):
    for j in range(1,11):
        k=i*j
        print (k, end=' ')
    print()
```

Output
1 2 3 4 5 6 7 8 9 10
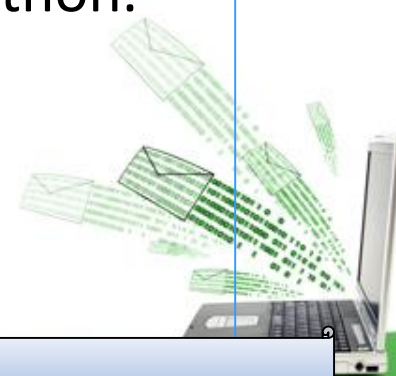2 4 6 8 10 12 14 16 18 20

Iteration Statements (Loops)

3. Jump Statements

      Jump statements are used to transfer the program's control from one location to another. Means these are used to alter the flow of a loop like - to skip a part of a loop or terminate a loop

There are three types of jump statements used in python.
        1.break
        2.continue
        3.pass

Iteration Statements (Loops)
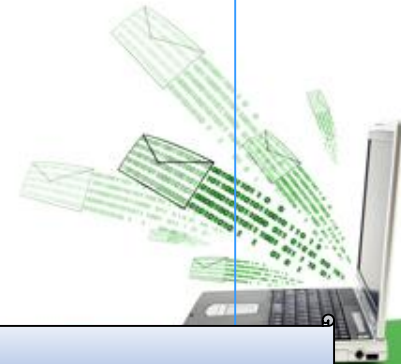
1.break

it is used to terminate the loop.

e.g.
```
for val in "string":
    if val == "i":
        break
    print(val)


print("The end")
```

Output
s
t
r
The end

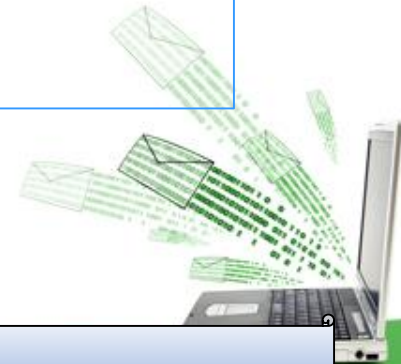# Iteration Statements (Loops)

2.continue

It is used to skip all the remaining statements in the loop and move controls back to the top of the loop.

e.g.

```
for val in "init":
    if val == "i":
        continue
    print(val)
print("The end")
```

Output
n
t
The end

Iteration Statements (Loops)

3. pass Statement

This statement does nothing. It can be used when a statement is required syntactically but the program requires no action.

Use in loop

while True:

       pass # Busy-wait for keyboard interrupt (Ctrl+C)

In function

It makes a controller to pass by without executing any code.

e.g.

def myfun():

   pass   #if we don't use pass here then error message will be shown

print('my program')

OUTPUT

My program

Iteration Statements (Loops)

3. pass Statement continue

e.g.
```
for i in 'initial':
    if(i == 'i'):
        pass
    else:
        print(i)
```

OUTPUT
n
t
a
L

NOTE : continue forces the loop to start at the next iteration while pass means "there is no code to execute here" and will continue through the remainder or the loop body.