6 Python Flashback

Learning Outcomes

- Python Environment
- How to use Python shell ?
- Print(), Input(), & Type() Functions
- Variable and their type
- Keywords
- Operator and Operand in Python
- Expression & Statement
- Comments
- Control Statements of Python
- Conditional Statements

As you know that Python was created by Guido Van Rossum when he was working at CWI (Centrum Wiskunde & Informatica) which is a National Research Institute for Mathematics and Computer Science in Netherlands. The language was released in 1991. Python got its name from BBC comedy series from seventies- "Monty Python's Flying Circus".

Python can be used to follow both Procedural approach and Object-Oriented approach of programming. It is a free to use language.

Some of the features which make Python so popular are

- It is a general-purpose programming language which can be used for both scientific and nonscientific programming.
- It is excellent for beginners as the language is interpreted, hence gives immediate results.
- It is easy to learn and use.
- It is a very simple high-level language with vast library of add-on modules.
- It is a platform independent programming language. A Python program written for one OS can run in others without any modifications.
- The programs written in Python are easily readable and understandable.
- It is suitable as an extension language for customizable applications.
- The language is used by companies in real revenue generating products, such as:
  - In operations of Google search engine, YouTube, etc.
  - Maya provides a Python scripting API.
  - Intel, Cisco, HP, IBM, etc use Python for hardware testing.
  - i-Robot uses Python to develop commercial Robot.
  - NASA and others use Python for their scientific programming task.
  - U-Torrent peer to peer file sharing program is also written using Python.

What is a Program?

A program is a sequence of instructions that specifies how to perform a particular task. It may be as simple as finding sum of two numbers to a complex program for launching a satellite. You have learnt that a computer follows IPO cycle, i.e. Input, Process and Output cycle. For example, your mother takes the raw material like floor, sugar, coco powder etc. as ingredients for making a cake. She mixes all of them together in the process to make a proper dough of all the ingredients. The process ends with backing the dough. Once the dough is packed properly, delicious cake comes out as output. Similarly, a computer takes the inputs (ingredients) from the user, processes it as per the instructions (program) and gives meaningful results (output).

Python Environment

To write and run Python program, we need to have Python interpreter or compiler installed in our computer. Integrated Development Environment (IDE GUI integrated) is the standard, most popular Python development environment. It lets edit, compile, run, browse and debug Python Programs from a single interface.

How to use Python Shell

Python shell can be used in two ways:

- Interactive mode
- Script mode

Interactive Mode

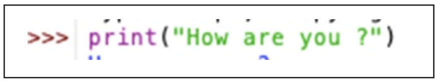
Interactive Mode allows us to interact with OS directly,

For working in the interactive mode, click on the **Python Icon** available on the desktop. Python window will appear.

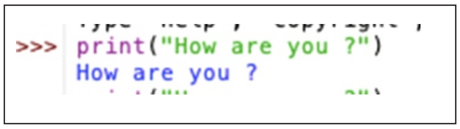
The **Python prompt >>>** will appear which means the interactive mode is ready and we can enter commands to get the output instantly. There is secondary prompt also which is **...** indicating that interpreter is waiting for additional input(s) to complete the current statement.

Let us start with typing print "How are you" after the prompt.

>>>print ("How are you?")



```
>>> print("How are you ?")
```

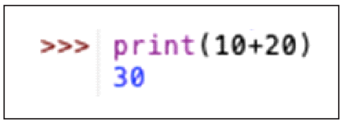


```
>>> print("How are you ?")
How are you ?
```

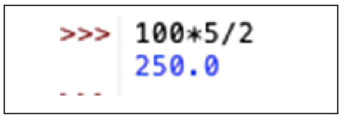
OUTPUT

The output will come instantly **How are you?**

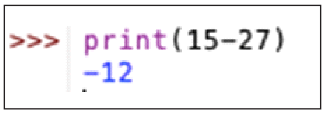
We may now try some simple statements and check the response:



```
>>> print(10+20)
30
```

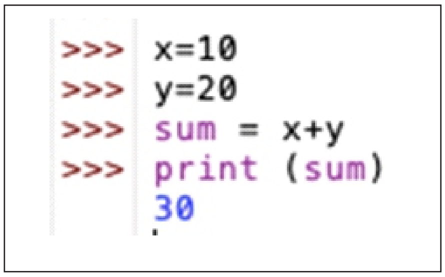


```
>>> 100*5/2
250.0
```



```
>>> print(15-27)
-12
```

we can also enter script at the interpreter level. For example:



```
>>> x=10
>>> y=20
>>> sum = x+y
>>> print (sum)
30
```

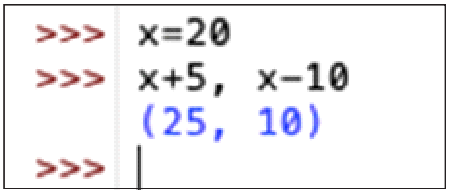
As the value of x=10, y=20 and sum= x+y which means sum=10+20=30. So, the Output will be **30**

Another example is given below:

Input : **x=20, x+5, x-20**

So the result will be x+5=20+5=25 and x-5=20-10=10.

So, the Output will be **(25, 10)**



```
>>> x=20
>>> x+5, x-10
(25, 10)
>>> |
```

Script Mode

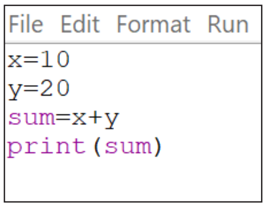
In the script mode you can create and edit Python source file as a complete program.

To create and run a Python script, we write the script in IDE (Integrated Development Environment).

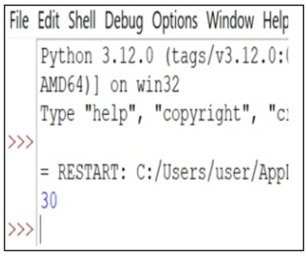
- Click **File -> New** option.
- A new window will open.
- Now write the code.
- To save the code, click on **File -> Save**. The file will be saved with **.py** extension.
- To run from the same window, press **F5** key or select **RUN -> Run Module**.
- To open an existing Python file, click on **File -> open**.

For example, type the following in the Script mode:

Output will be displayed in Python shell as **30**.



```
x=10
y=20
sum=x+y
print(sum)
```



```
Python 3.12.0 (tags/v3.12.0:(
AMD64)] on win32
Type "help", "copyright", "c:
>>>
= RESTART: C:/Users/user/Appl
30
>>>|
```

OUTPUT

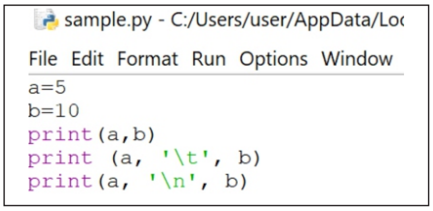
Print(), Input() and Type() Functions

Print() Function

print() function is used to print and display user messages and values of variables and expression as enclosed in parenthesis (). It can be used both in Interactive as well as in Script mode.

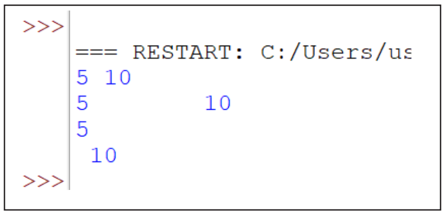
To separate values in the print statement we use many separators like:

- **Comma (,)** - to print the next value on the same line after the space.
- **'\t' (tab)** - to print the next value on the same line separated with a tab (5 spaces) space.
- **‘\n' (New line)** - to print the next value on the next line.


sample.py - C:/Users/user/AppData/Loc
File Edit Format Run Options Window

```
a=5
b=10
print(a,b)
print (a, '\t', b)
print(a, '\n', b)
```

INPUT



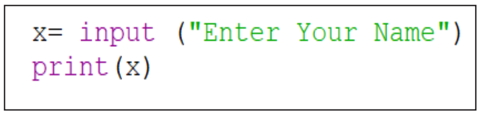
```
>>>
=== RESTART: C:/Users/us
5 10
5         10
5
 10
>>>
```

OUTPUT

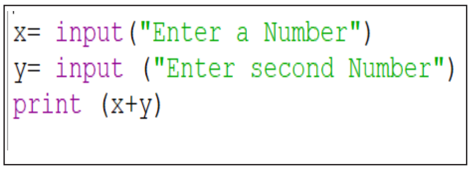
Input() Function

Input() function is used to take input from the user during the program run. It increases the interactivity with user when the program is being run. The value returned by the input() function is always of str (string) type. Refer the following:

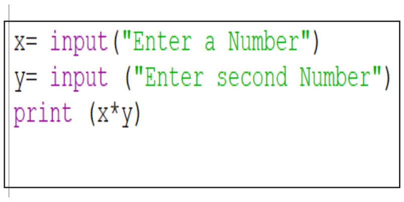
INPUT



```
x= input ("Enter Your Name")
print(x)
```

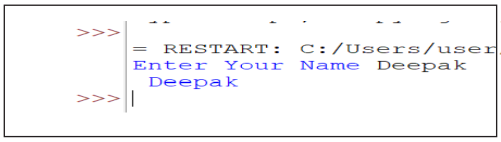


```
x= input("Enter a Number")
y= input ("Enter second Number")
print (x+y)
```



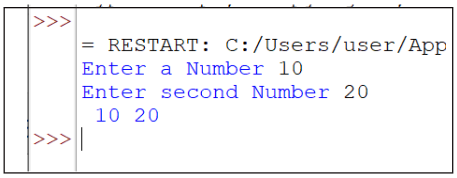
```
x= input("Enter a Number")
y= input ("Enter second Number")
print (x*y)
```

OUTPUT



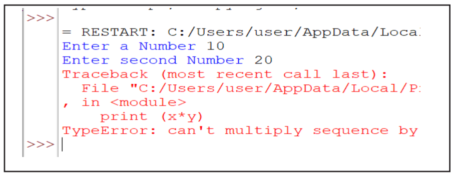
```
>>>
    = RESTART: C:/Users/user
    Enter Your Name Deepak
     Deepak
>>> |
```

The input given is stored in variable x and same is printed by the print() function.



```
>>>
    = RESTART: C:/Users/user/App
    Enter a Number 10
    Enter second Number 20
     10 20
>>> |
```

Two input statements assign the values in x and y respectively and x+y simply concatenates these values and the output is 1020.

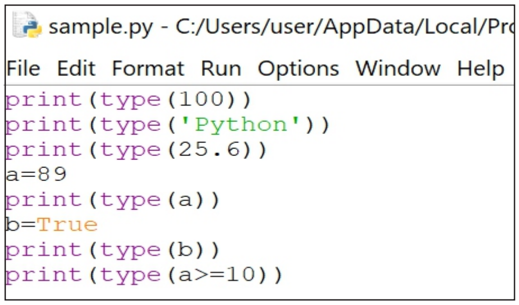


```
>>>
    = RESTART: C:/Users/user/AppData/Loca
    Enter a Number 10
    Enter second Number 20
    Traceback (most recent call last):
      File "C:/Users/user/AppData/Local/P
    , in <module>
        print (x*y)
    TypeError: can't multiply sequence by
>>> |
```

print(x*y) will prompt for error as * operator can only be used with numbers and not with strings. So, there is a need to convert these values to numbers by using either int() or float().

type() Function

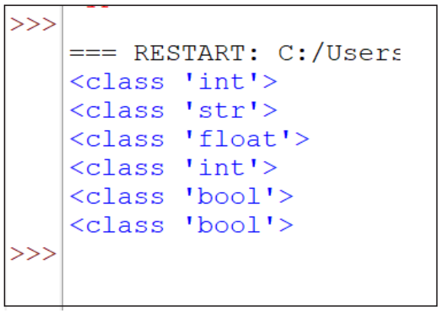
type() function returns type of value written between the parenthesis (). So, if you want to know the value you are using, you can use type() function. Look at the following example:



sample.py - C:/Users/user/AppData/Local/Pr
File Edit Format Run Options Window Help

```
print(type(100))
print(type('Python'))
print(type(25.6))
a=89
print(type(a))
b=True
print(type(b))
print(type(a>=10))
```

INPUT



```
>>>
    === RESTART: C:/Users
    <class 'int'>
    <class 'str'>
    <class 'float'>
    <class 'int'>
    <class 'bool'>
    <class 'bool'>
>>>
```

OUTPUT

Variable and Their Types

In Python programs you often like to store values during the execution of the program so that these values can be used or manipulated later in the program. So, a variable is a name which refers to a value in the memory of the computer.

Rules for Writing Variables

When we write variables, following rules are applicable:

- It must start with an alphabet (upper or lower case or an underscore(_).
- It can consist of only alphabets, digits and underscore without violating the first rule.
- No special character is allowed in a variable name except underscore.Reserve keywords of Python cannot be used as variable name.
- It can be of any length.
- Variables in python are case sensitive (like Sum, sum, SUM, SuM are different variable names).

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

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

Some valid Variable names are**: Sum_XY, var1, obj_3, age_student**, etc.

Some Invalid examples are:

Invalid Variable Names	Reason for Invalidity
Book Name	Spaces in a Variable name is not allowed.
20Names	Not starting with an alphabet or underscore
Employees@XYZ	Sp ecial chara cter @ not permitted
for	Is a reserve word in Python

Every Variable has

- An Identity, i.e. a name
- A data type
- A value of that data type.

Identity of a Variable : It is an address in memory and does not change once it has been created.

Type (i.e. Data type) : It is the set of values and operations that can be performed on those values. It can be one of the following types:

Number

Number data type stores Numerical Values. This data type is immutable i.e. value of these type of variable cannot be changed. These are of three different types:

- Integer & Long
- Float/floating point
- Complex

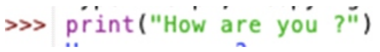
Range of an integer in Python can be from -2147483648 to 2147483647, and long integer has unlimited range subject to available memory.

Data Types	Values
Int	6, -10, 65, 3245
float	78.54, -67.43, 3.147
str	“Python” , ‘Python’
bool	5>2

Sequence : A sequence is an ordered collection of items, indexed by positive integers (in other languages it is called an array). Three types of sequence datatype available in Python are Strings, Lists & Tuples.

String : A string is a group of ordered characters/letters. They are enclosed in single quotes '' or double quotes "". The quotes are not part of string. They only mark the beginning and end of the string. Any character can be enclosed in the string including spaces, if any. Example :

>>> a = 'Priya':

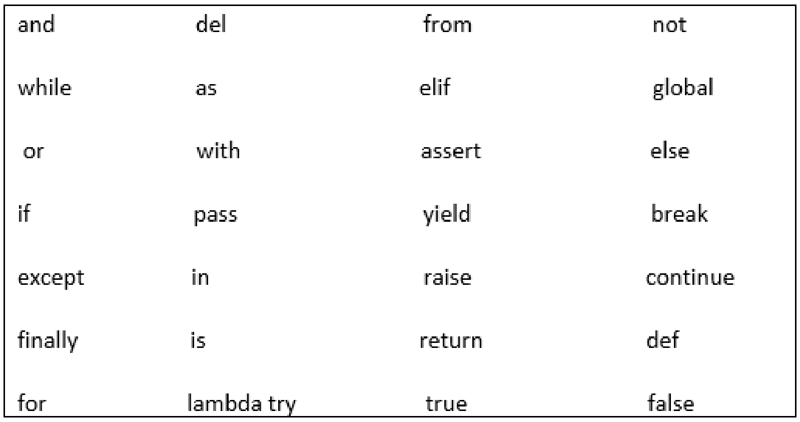


```
>>> print("How are you ?")
```

Keywords

These are the words which has a special meaning for the Python interpreter which recognizes the Structure of a Python program. As these words have specific meaning for interpreter, they cannot be used for any other purpose.

Few keyword in Python are listed below:



and	del	from	not
while	as	elif	global
or	with	assert	else
if	pass	yield	break
except	in	raise	continue
finally	is	return	def
for	lambda try	true	false

Operators and Operands in Python

Operators are special symbols which performs an operation. They are applied on operand(s), which can be values or another variable. Same operator can behave differently on different data types.

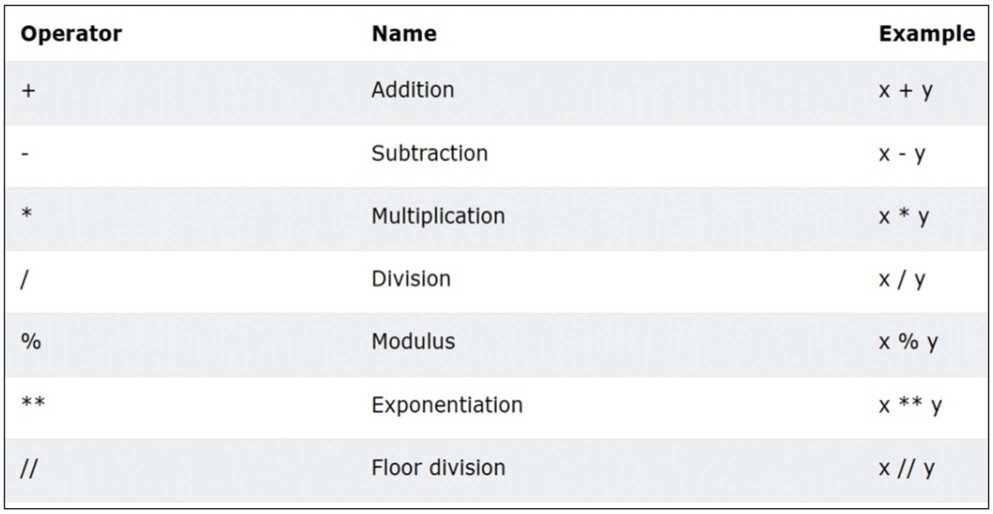
Expressions are formed when operand with suitable operators are joined together.

Operators are categorized as:

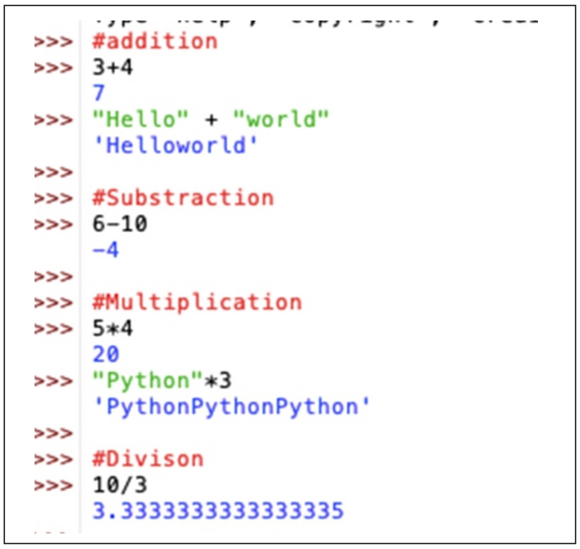
- Arithmetic operators
- Relational operators
- Logical operators
- Assignment operators.

Mathematical/ Arithmetic Operators

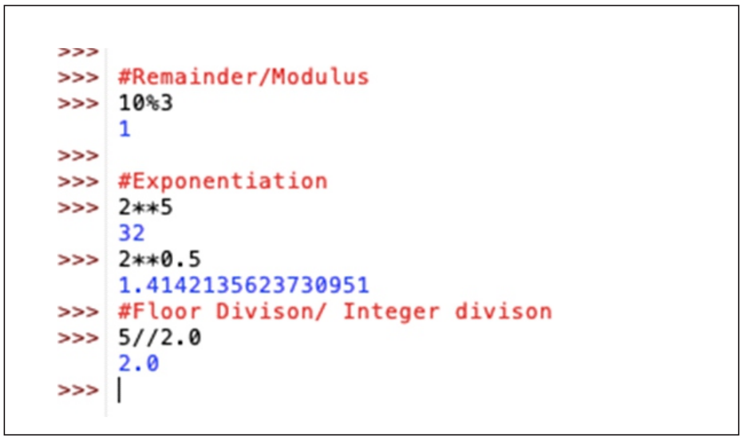
Following table describes Mathematical/Arithmetic Operators available in Python:



Operator	Name	Example
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y
**	Exponentiation	x ** y
//	Floor division	x // y



```
>>> #addition
>>> 3+4
    7
>>> "Hello" + "world"
    'Helloworld'
>>>
>>> #Substraction
>>> 6-10
    -4
>>>
>>> #Multiplication
>>> 5*4
    20
>>> "Python"*3
    'PythonPythonPython'
>>>
>>> #Divison
>>> 10/3
    3.3333333333333335
```



```
>>>
>>> #Remainder/Modulus
>>> 10%3
    1
>>>
>>> #Exponentiation
>>> 2**5
    32
>>> 2**0.5
    1.4142135623730951
>>> #Floor Divison/ Integer divison
>>> 5//2.0
    2.0
>>> |
```

Assignment Operators

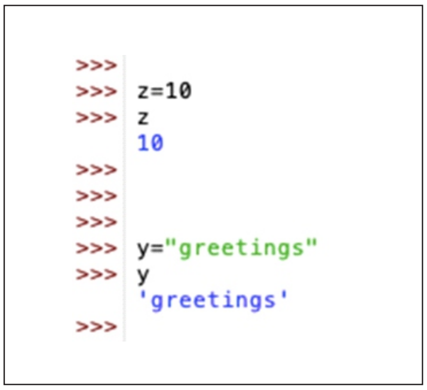
Assignment Operator assigns a value to a variable. Like

>>>z=10

z will be point to value 10

>>>y="greetings"

y will point to "greetings", a string

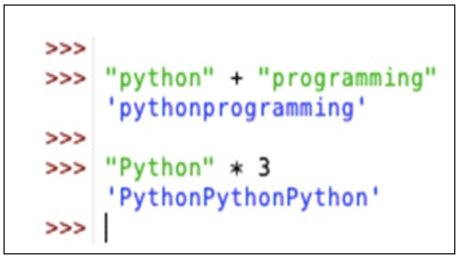


```
>>>
>>> z=10
>>> z
    10
>>>
>>>
>>>
>>> y="greetings"
>>> y
    'greetings'
>>>
```

String Operators

These operators can only be used with string data type.

Operator	Operation	Example
+	Concatenation	>>>"Pyhton"+"Programming" PythonProgramming
*	Replication	>>>"Python"*3 PythonPythonPython



```
>>>
>>> "python" + "programming"
    'pythonprogramming'
>>>
>>> "Python" * 3
    'PythonPythonPython'
>>> |
```

Relational Operators

These are the operators which are used to establish relationship between values/variables.

They always result in bool type of value - True or False.

Operator	Operation	Examples (let a=10 and b=15)
<	Less than	>>> a<b True
>	Greater Than	>>> a>b False
<=	Less than or Equals to	>>> a<=b true
>=	Greater Than or Equals to	>>> a>=b False
==	Equals to	>>>a==b False
!=	Not Equals to	>>>a!=b True

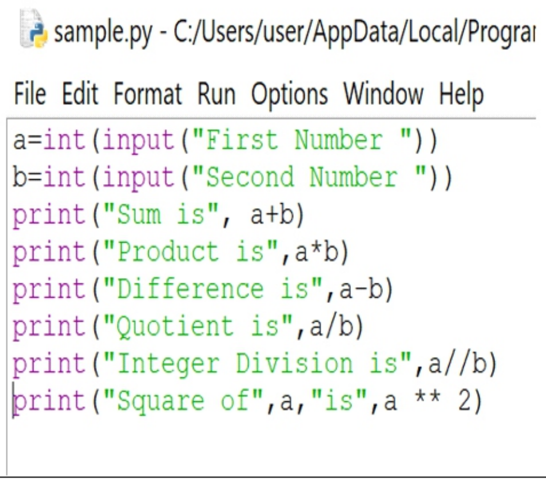
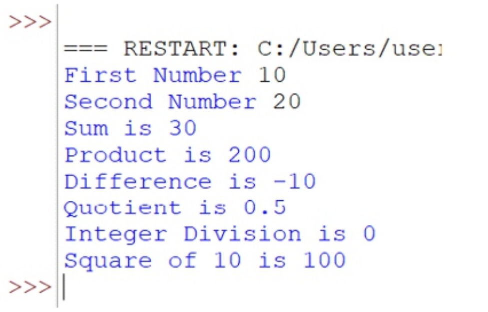
Logical Operators

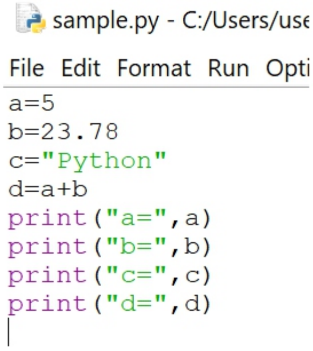
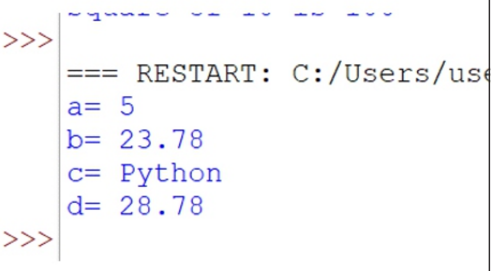
Logical operators are used to combine two or more conditions to give results in bool form - True or False.

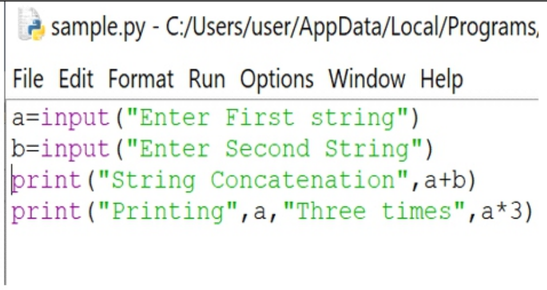
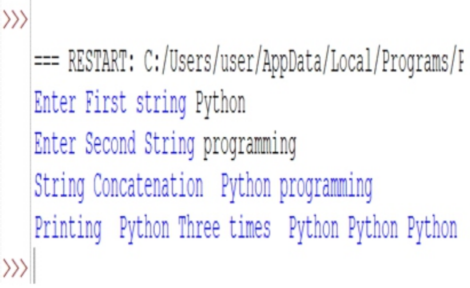
Operator	Operation	Examples (let a=10 and b=15)
and	Give result as True if both the operands are True	>>>a>5 and b<20 True
or	Give result as True if either of the operands are True	>>>a>5 or b>20 true
not	Give the result as True if the operand is False and vice versa	>>>not a>5 False

Programs related to the above operators:

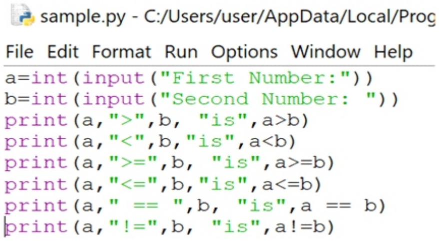
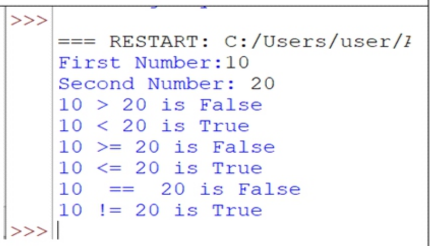
Operators	Program	Output

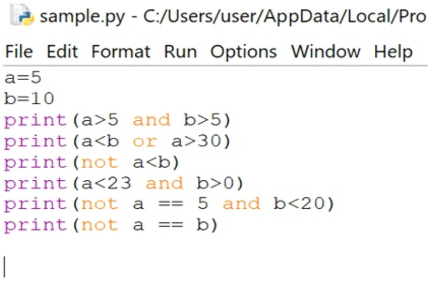
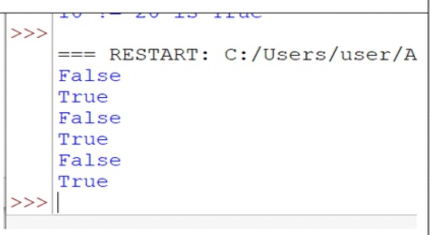
Airthmatic/ Mathmetical	sample.py - C:/Users/user/AppData/Local/Prograı
	File Edit Format Run Options Window Help
	a=int(input("First Number "))
	b=int(input("Second Number "))
	print("Sum is", a+b)
	print("Product is",a*b)
	print("Difference is",a-b)
	print("Quotient is",a/b)
	print("Integer Division is",a//b)
	print("Square of",a,"is",a ** 2)
		>>>
		=== RESTART: C:/Users/use:
		First Number 10
		Second Number 20
		Sum is 30
		Product is 200
		Difference is -10
		Quotient is 0.5
		Integer Division is 0
		Square of 10 is 100
		>>>

Assignment	sample.py - C:/Users/use
	File Edit Format Run Opti
	a=5
	b=23.78
	c="Python"
	d=a+b
	print("a=",a)
	print("b=",b)
	print("c=",c)
	print("d=",d)
		>>>
		=== RESTART: C:/Users/use
		a= 5
		b= 23.78
		c= Python
		d= 28.78
		>>>

String	sample.py - C:/Users/user/AppData/Local/Programs,
	File Edit Format Run Options Window Help
	a=input("Enter First string")
	b=input("Enter Second String")
	print("String Concatenation",a+b)
	print("Printing",a,"Three times",a*3)
		>>>
		=== RESTART: C:/Users/user/AppData/Local/Programs/P
		Enter First string Python
		Enter Second String programming
		String Concatenation Python programming
		Printing Python Three times Python Python Python
		>>>

Operators	Program	Output

Rational	
	sample.py - C:/Users/user/AppData/Local/Prog
	File Edit Format Run Options Window Help
	a=int(input("First Number:"))
	b=int(input("Second Number: "))
	print(a,">",b, "is",a>b)
	print(a,"<",b,"is",a<b)
	print(a,">=",b, "is",a>=b)
	print(a,"<=",b,"is",a<=b)
	print(a," == ",b, "is",a == b)
	print(a,"!=",b, "is",a!=b)
		
		>>>
		=== RESTART: C:/Users/user/A
		First Number:10
		Second Number: 20
		10 > 20 is False
		10 < 20 is True
		10 >= 20 is False
		10 <= 20 is True
		10 == 20 is False
		10 != 20 is True
		>>>

Logical	
	sample.py - C:/Users/user/AppData/Local/Pro
	File Edit Format Run Options Window Help
	a=5
	b=10
	print(a>5 and b>5)
	print(a<b or a>30)
	print(not a<b)
	print(a<23 and b>0)
	print(not a == 5 and b<20)
	print(not a == b)
		
		>>>
		=== RESTART: C:/Users/user/A
		False
		True
		False
		True
		False
		True
		>>>

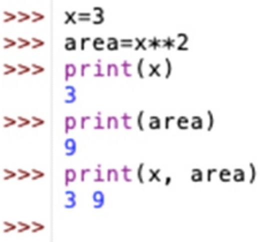
Precedence (Hierarchy) of Operators

Precedence of operators refers to the order in which the operator is executed in an expression comprising of many operators. In case there are same operators in the expression then the rule of Associativity is applicable which can be left to right (in most of the cases) or from right to left. The precedence is given in the following table.

Operators	Description	Associativity
()	Parentheses	Left to Right
**	Exponent	Right to Left
=x,-x	Unary plus, Unary mi	Left to Right
*,/,//,%	Multiplication, Divis Integer division, Modu	Left to Right
+,-	Addition, Subtraction	Left to Right
==,!=,>,>=,<,<=	Comparisons, operato	Left to Right
not	Logical NOT	Left to Right
and	Logical AND	Left to Right
or	Logical OR	Left to Right

Expression and Statement

An expression is a combination of constants, variable joined with suitable operator(s). An expression always generates a single value. A complete Python instruction is called the Statement. Example of statement with expression can be seen in the given image.



```
>>> x=3
>>> area=x**2
>>> print(x)
3
>>> print(area)
9
>>> print(x, area)
3 9
>>>
```

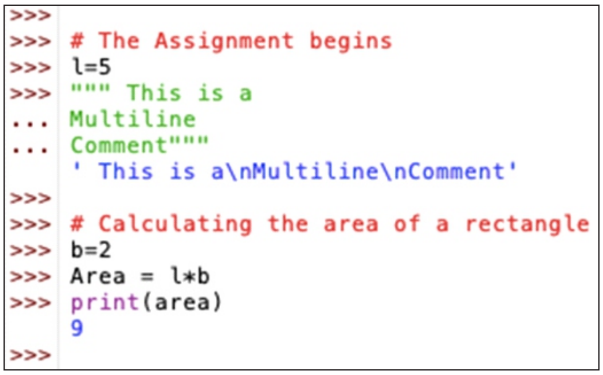
Comments

As the program size increases it becomes difficult to understand and read the program. It becomes difficult to make it as to what the program is doing by just looking at it. So, it is a good practice to add notes in the program. These notes are called comments and are not execute by the compiler/interpreter they are just ignored.

In Python, a comment can start with

- # symbol
- “”" symbol

Anything written after # in a line or between """ is ignored by interpreter and will not be executed. Look at the following example;



```
>>>
>>> # The Assignment begins
>>> l=5
>>> """ This is a
... Multiline
... Comment"""
    ' This is a\nMultiline\nComment'
>>>
>>> # Calculating the area of a rectangle
>>> b=2
>>> Area = l*b
>>> print(area)
    9
>>>
```

Control Statements of Python

You must be now familiar with that how a program is executed. A program, generally, executes from its first statement till the last statement in a sequential manner. That is why it is also called **sequential execution.**

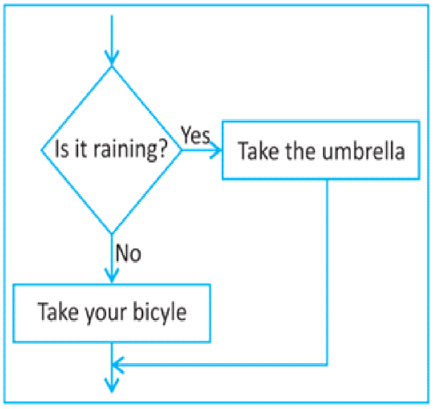
Sometimes we need to execute a few statements only and sometimes we need to execute a set of statements repeatedly. In this way the program itself decides which statement should to execute, which statements to leave and which statements to repeat. The statements which decide the program execution as stated above are known as program control statements. Python provides two such statements - Branching which is also known as **Conditional statements** and **Looping statements.**

Conditional Statements

In a conditional statement, execution of a Python statement depends on a condition. As you know that a condition will result in bool (True/False) type of data, so depending on the condition you can write the programming statements.

Let us understand this with help of an example from our daily life: **If it is raining** then **Take your umbrella** otherwise **Take your bicycle**

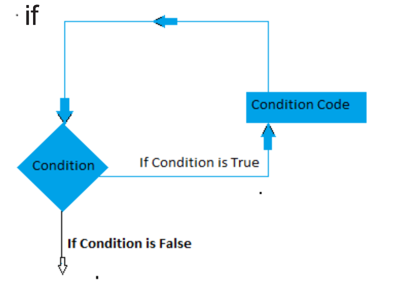
In the above statement it is clear that the condition is whether **it is raining or not**. And the statements are **Take your umbrella** and **take your bicycle**. Now if the condition is **True**, that is if it is raining, you will Take your umbrella and if it is not raining (False condition) then you will Take your bicycle. In no case both the statements will be executed and also it is not possible that either of them is not executed. This can easily be shown in the adjacent diagram.

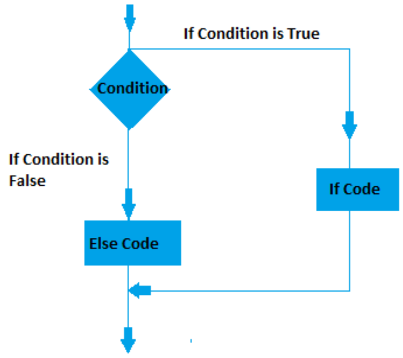


Python provides basically three conditional statements to be used in a program. They are:

- If statement
- if-else statement
- if-elif-else statement

Let us see in the table below their syntax and usage:

Conditional Statement	Syntax	Example
if  Condition Code Condition If Condition is True If Condition is False	**If condition:** **Statement 1** **Statement 2** Condition will be evaluated first and if the condition is true,**statement 1** will be executed and then**statement 2** If the condition is false program control will jump to **statement 2** in this case **statement 1** has been branched .	x=int(input("Enter 1st number" y=int(input("Enter 2nd number")) if x>y: print(x,"is greater than", y) **First Run** Enter 1st number 10 Enter 2nd number 20 **Second Run** Enter 1st t number 20 Enter 2nd number 10. 20 is greater than 10

If -else
 If Condition is True
Condition
If Condition is False
If Code
Else Code
	If condition:
	Statement 1
	else:
	Statement 2
	Statement 3
	Here **condition** will be evaluated first and if the condition is **True**, **Statement 1** will be executed and if the condition is **False**, then **Statement 2** will be executed. After executing either Statement 1 or Statement 2 the program control will move to **Statement 3.** In this case **Statement 1** and **Statement 2** both have been branched.
		x=int(input("Enter 1st Number "
		y=int(input("Enter 2nd Number "))
		if x>y:
		print(x, "is greater than",y)
		else:
		print(y, "is greater than",x)
		First Run
		Enter 1st Number 10
		Enter 2nd Number 20
		20 is greater than 10
		Second Run
		Enter 1st Number 20
		Enter 2nd Number 10
		20 is greater than 10

If-elif-else	**if condition 1:** **Statement 1** **elif condition 2:** **Statement 2** **elif condition3:** **Statement 3** : : **elif condition n:** **Statement n else:** **Statement 4** **:** **:** **Statement 5** Here we can write multiple conditions and corresponding statements to execute. Firstly, Condition 1 will be evaluated and if the condition is True Statement 1 will be executed and if the condition is False, then Condition 2 will be evaluated and if it is True Statement 2 will be executed and if Condition 2 is also False, Condition 3 will be evaluated. Similarly, other conditions will be evaluated executing the respective Statement. If none of the conditions is True, Statement 4 will be executed written under the last else. After executing one of the Statement in if else, Statement 5 will be executed.	x=int(input("Enter a number " y=int(input("Enter another Number " choice=input("Enter 1-To Add, 2-Difference, 3 Product ") if choice == '1': print("Addition of",x, "and",y, "is",x+y) e choice == '2': print("Difference of",x,"and",y,"is",x-y) eli choice == '3': print("Product of",x, "and",y, "is",x*y) else: print("Wrong Choice") **First Run** Enter a number 12 Enter another Number 14 Enter 1-To Add, 2-Difference, 3-Product 1 Addition of 12 and 14 is 26 **Second Run** Enter a number 12 Enter another Number 34 Enter 1-To Add, 2-Difference, 3-Product 5 Wrong Choice **Third Run** Enter a number 12 Enter another Number 14 Enter 1-To Add, 2-Difference, 3-Product 3 Product of 12 and 14 is 168

Few more programming examples are:

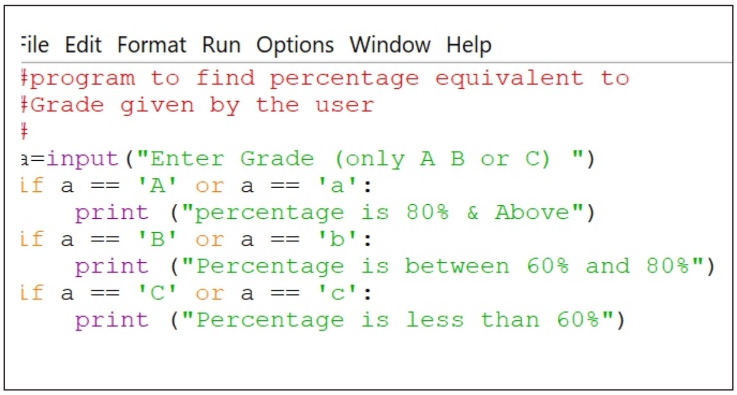
- To find percentage equivalent to Grade given by the user based on the following:

Grade Percentage

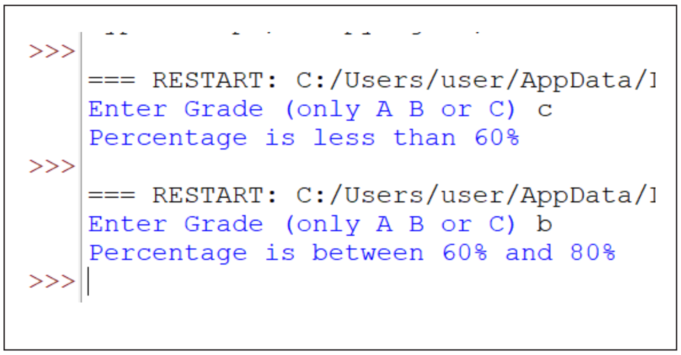
A for 80% & Above

B for 60% to 80%

C for Below 60%



```
#program to find percentage equivalent to
#Grade given by the user
#
a=input("Enter Grade (only A B or C)  ")
if a == 'A' or a == 'a':
    print ("percentage is 80% & Above")
if a == 'B' or a == 'b':
    print ("Percentage is between 60% and 80%")
if a == 'C' or a == 'c':
    print ("Percentage is less than 60%")
```



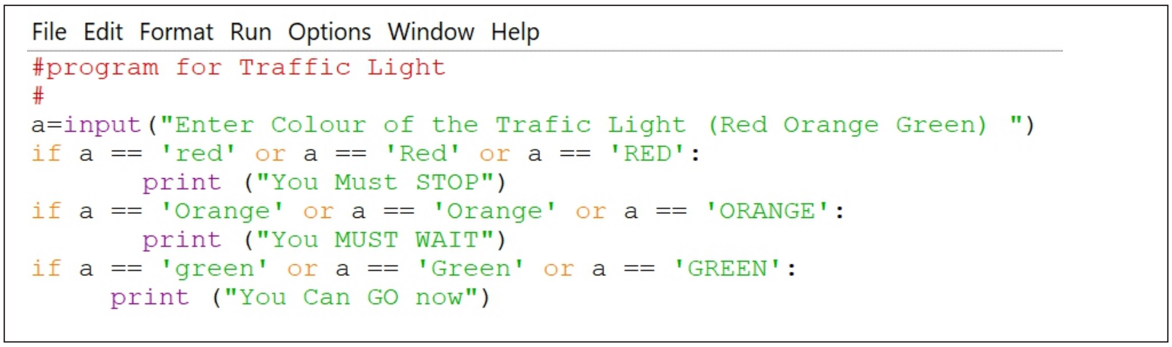
```
>>>
    === RESTART: C:/Users/user/AppData/l
    Enter Grade (only A B or C) c
    Percentage is less than 60%
>>>
    === RESTART: C:/Users/user/AppData/l
    Enter Grade (only A B or C) b
    Percentage is between 60% and 80%
>>>
```

- □ To print Stop, Look or Go depending on Traffic light Signal

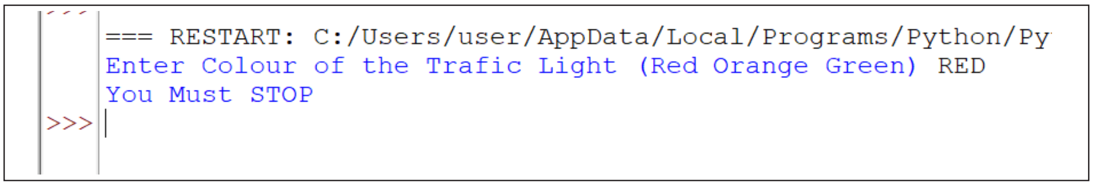
Red for Stop

Orange for Wait

Green for Go

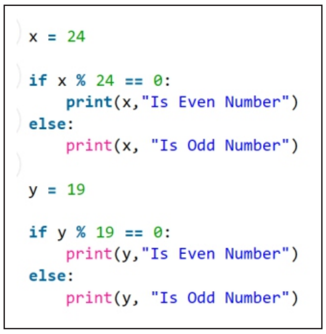


```
#program for Traffic Light
#
a=input("Enter Colour of the Trafic Light (Red Orange Green)  ")
if a == 'red' or a == 'Red' or a == 'RED':
       print ("You Must STOP")
if a == 'Orange' or a == 'Orange' or a == 'ORANGE':
       print ("You MUST WAIT")
if a == 'green' or a == 'Green' or a == 'GREEN':
     print ("You Can GO now")
```



```
    === RESTART: C:/Users/user/AppData/Local/Programs/Python/Py
    Enter Colour of the Trafic Light (Red Orange Green) RED
    You Must STOP
>>>
```

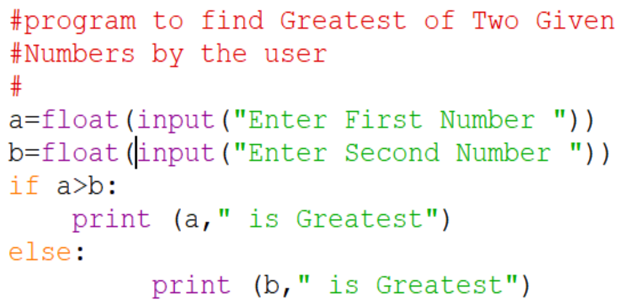
- Write a Program to Find a number in Odd or Even



```
x = 24

if x % 24 == 0:
    print(x,"Is Even Number")
else:
    print(x, "Is Odd Number")

y = 19

if y % 19 == 0:
    print(y,"Is Even Number")
else:
    print(y, "Is Odd Number")
```

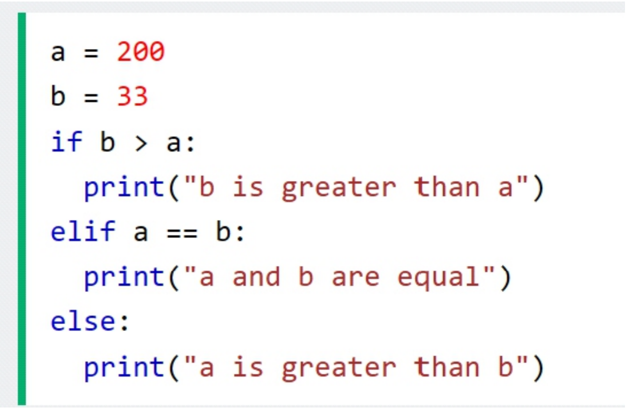
- To find greatest of Two Given numbers

Taking Input From the User



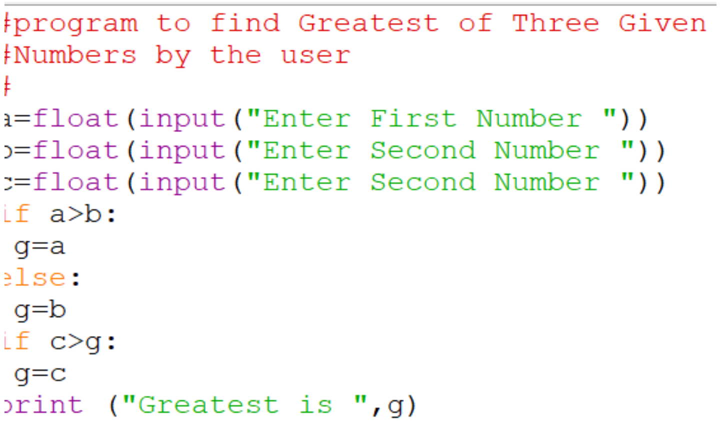
```
#program to find Greatest of Two Given
#Numbers by the user
#
a=float(input("Enter First Number "))
b=float(input("Enter Second Number "))
if a>b:
    print (a," is Greatest")
else:
         print (b," is Greatest")
```

Predefined User Input



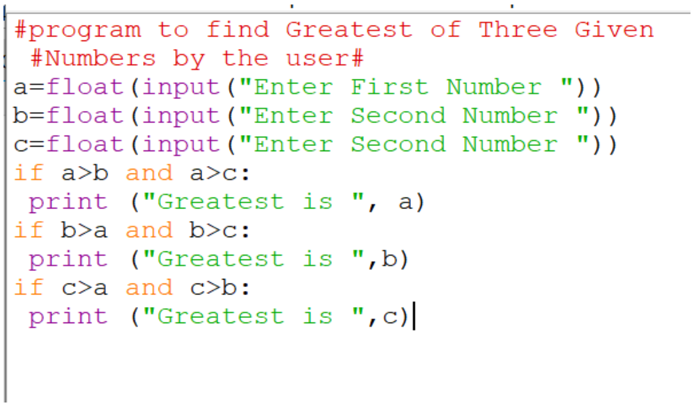
```
a = 200
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
else:
  print("a is greater than b")
```

- To find greatest of Three Given numbers:



```
#program to find Greatest of Three Given
#Numbers by the user
#
a=float(input("Enter First Number "))
b=float(input("Enter Second Number "))
c=float(input("Enter Second Number "))
if a>b:
 g=a
else:
 g=b
if c>g:
 g=c
print ("Greatest is ",g)
```

Alternate Code



```
#program to find Greatest of Three Given
 #Numbers by the user#
a=float(input("Enter First Number "))
b=float(input("Enter Second Number "))
c=float(input("Enter Second Number "))
if a>b and a>c:
 print ("Greatest is ", a)
if b>a and b>c:
 print ("Greatest is ",b)
if c>a and c>b:
 print ("Greatest is ",c)
```

Quick Look

- Python was created by Guido Van Rossum when he was working at CWI (Central Wiskunde & informatica).
- Python can be used to follow both Procedural approach and Object-Oriented approach of programming.
- Python shell can be used in two ways-Interactive mode and Script mode.
- The Python programs are stored with .py extension.
- Data types are categorized as Number(Integer & Long, Float/floating point, Complex) and sequence (string, Lists & Tuples).
- Keywords are the words which has a special meaning for the Python interpreter which recognizes the structure of a Python program.
- Operators are special symbols which performs an operation.
- Comments are special notes left in a Python program which are not execute by the compiler/interpreter they are just ignored.
- Python provides two control statements- Branching(conditional) statements and Looping Statement.

Brain Booster

Section - I

Objective Type Questions

A. Fill in the blanks with the correct words.

Looping	Conditional	Interactive	Script	1991	Sequential Execution	Sequential

1. Python was created by Guido Van Rossum when he was working at CWI (Centrum Wiskunde & Informatica) in __________________________.

2. We can use Python shell in__________________ mode and ____________________mode.

3. In____________________ type a set of statements are executed repeatedly one after the other.

4. In__________ all statements of the program are executed one after the other except comments.

5. In Python control statements can be _________________or____________________ statements.

B. Write T for the true statement and F for the false one:

1. Python is not an open source programming language. □

2. Python can be used both by beginners and expert programmers. Python contains three control statements. □

3. In Python multiple conditions can be specified in if-else-if statement. □

4. Asignment Operator assigns a value to a variable. □

5. In iterative control statements, all statements are not executed sequentially □

C. Choose the right option:

1. String operators in Python is/are:

   a. + ○ b. - ○

   c. * ○ d. Both (a) and © ○

2. Logical operators may result in ....

   a. False ○ b. String ○

   c. Complex Number ○ d. Float ○

3. Comments in Python are.................................. .

   a. Executed ○ b. Not Executed ○

   c. Printed ○ d. Checked & Corrected ○

4. In conditional statement if, the statement may be

   a. Executed ○ b. Not Executed ○

   c. both (a) & (b) ○ d. Printed ○

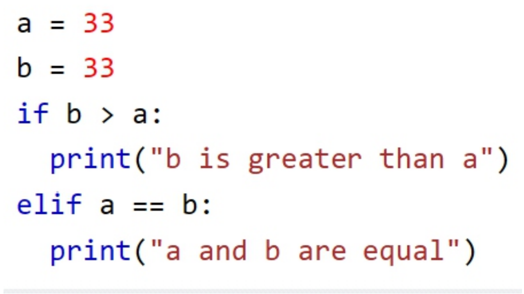
5. The extension of program files in Python is/are:

   a. Python ○ b. .pth ○

   c. . Ph ○ d. . Py ○

D. Application Based questions:

Drishti has written the following program in Python, but she is confused about the output.



```
a = 33
b = 33
if b > a:
  print("b is greater than a")
elif a == b:
  print("a and b are equal")
```

- Meethi wants to write her name 5 times on the screen with help of Python program. She is using an iterative statement for this purpose, but when Rig saw, he suggested her for a simpler way to do the same. Can you guess what?
- Ayaan has written the following program in Python. But it is prompting for some errors. Correct this program and rewrite it after removing all the errors.

```
A=10
B=20
If a>b print a
Else
print b
```

Section - II

Descriptive Type Questions:

E. Answer the following question.

1. Who created Python?

2. How can we leave programmers notes in a Python program?

3. Give any two relational operators.

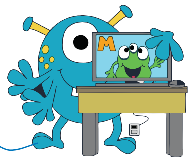
4. Give the main features of Python which makes it user friendly.

5. Differentiate between * operator when used with (a) Integers (b) Strings.

Activity TIME

LAB ACTIVITY

Practice Makes Perfect



Write Python codes to do the following:

- To find whether the number given by the user is Zero, Positive or Negative.
- To Find whether the number is positive or negative and if the number is positive then whether it is even or odd.
- To check how a student has performed in a competition based on the following criteria:

Grade	Message
If Grade >=80%	Excellent
If Grade >=60% but <80	% Very Good
If Grade >=40% but <60%	Good
If Grade <40%	You need to put more efforts

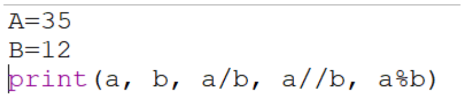
The program will ask for the Grade and print the massage as given above.

- To check whether a number N is completely divisible by another number P.
- To print remainder of two numbers if the choice of user is R and if the choice is Q the quotient of the number (without fractional part)
- To check whether a three-digit number in the form of ABC is Fibonacci number or not. A number is said to be Fibonacci number is the sum of the square of the individual digits is equals to the number, like A3+83+C3=ABC as in $371=3^3+7^3+1$?
- To check whether an alphabet or some other character has been entered by the user or not.

  To convert Fahrenheit to Celsius temperature by using the following formula:

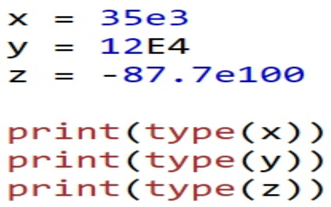
  C = (F - 32) ? 5/9
- **Give the output of the following Python programs:**

(a)



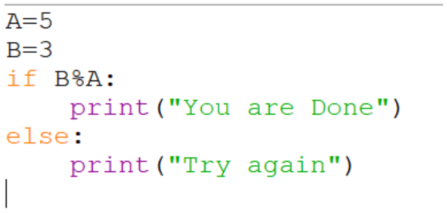
```
A=35
B=12
print(a, b, a/b, a//b, a%b)
```

©



```
x = 35e3
y = 12E4
z = -87.7e100

print(type(x))
print(type(y))
print(type(z))
```

(b)



```
A=5
B=3
if B%A:
    print("You are Done")
else:
    print("Try again")
```

(d)

```
a='AR'
b="QUE"
c=3
d=a+b
        print(d*c)
        print(c*c)
```

Activity TIME

GROUP DISCUSSION

For Better Clarity

Divide the class into groups and discuss about:

- Python a language for future
- Operators in Python
- Control Structures of Python

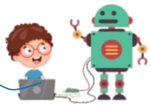
PROJECT WORK

For Practicing Ideas

- Design a calculator in Python to do simple arithmetic operation like addition, subtraction, multiplication and division on two floating point numbers given by the user. The operation will also be given by the user as an input. Design a Menu and do the operation as per the user's choice.

ONLINE LINKS

For Searching More



- https://www.geeksforgeeks.org/python-language-introduction/
- https://www.w3schools.com/python/python_intro.asp
- https://realpython.com/python-introduction/