THE NEW COLLEGE (AUTONOMOUS), CHENNAI-14.
DEPARTMENT OF COMPUTER APPLICATIONS – SHIFT - II

STUDY MATERIAL

SUBJECT: PYTHON PROGRAMMING (20BHM512) CLASS: III BCA - A

----

1. INTRODUCTION TO PYTHON

Python is a programming language that lets you work more quickly and integrate your systems more effectively. The mission of the Python Software Foundation is to promote, protect, and advance the Python programming language to support and facilitate the growth of a diverse and international community of Python programmers.

Python

It is an **Interpreted**, **High Level** and **General purpose programming** language. Which is created by **Guido Van Rossum** from **CWI** (Centrum Wiskunde & Informatica) which is a National Research Institute for Mathematics and Computer Science in Netherlands. The language was **released in I991**. Python supports both **Procedura**l and **OOP**'s (Object Oriented programming) approaches.

F**eatures of Python**

- ✓ It is a general purpose programming language which can be used for both scientific and non-scientific programming.
- ✓ It is a platform independent programming language.
- ✓ The programs written in Python are easily readable and understandable.

Programming in Python

Python programs can be written in two ways namely **interactive mode** and **Script mode**. The Interactive mode allows us to write codes in Python command prompt (>>>) whereas in script mode programs can be written and stored as separate file with the extension **.py** and executed. Script mode is used to create and edit python source file.

2. VARIABLES

Python Variable is containers which store values. We do not need to declare variables before using them or declare their type. A variable is created the moment we first assign a value to it. A Python variable is a name given to a memory location. It is the basic unit of storage in a program.

Example of Python Variables

```
Var = "Geeksforgeeks"
print(Var)
```

Output:

```
Geeksforgeeks
```

- The value stored in a variable can be changed during program execution.
- A Python Variables is only a name given to a memory location, all the operations done on the variable effects that memory location.

RULES FOR CREATING VARIABLES IN PYTHON

- Variable name must start with a letter or the underscore character.
- Variable name cannot start with a number.
- Variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _).
- Variable names are case-sensitive (name, Name and NAME are three different variables).
- The reserved words (keywords) cannot be used naming the variable.

Simple variable creation:

```
age = 45                # An integer assignment
salary = 1456.8         # A floating point
name = "John"           # A string
print(age)
print(salary)
print(name)
```

Output:

```
45
1456.8
John
```

DECLARE THE VARIABLE

Let’s see how to declare the variable and print the variable.

```
Number = 100       # declaring the var
 print( Number)      # display
```

Output:

```
100
```

ASSIGNING A SINGLE VALUE TO MULTIPLE VARIABLES

Also, Python allows assigning a single value to several variables simultaneously with "=" operators.

```
a = b = c = 10
print(a)
print(b)
print(c)
```

Output:

```
10
10
10
```

ASSIGNING DIFFERENT VALUES TO MULTIPLE VARIABLES

Python allows adding different values in a single line with ","operators.

```
a, b, c = 1, 20.2, "GeeksforGeeks"
print(a)
print(b)
print(c)
```

Output:

```
1
20.2
GeeksforGeeks
```

GLOBAL AND LOCAL PYTHON VARIABLES

Local variables are the ones that are defined and declared inside a function. We can not call this variable outside the function.

```
def f():        # This function uses global variable s
s = "Welcome geeks"
print(s)
f()
```

Output: Welcome geeks

Global variables are the ones that are defined and declared outside a function, and we need to use them inside a function.

```
# This function has a variable with
# name same as s.
def f():
print(s)
s = "I love Geeksforgeeks"                    # Global scope
f()
```

Output:

```
I love Geeksforgeeks
```

VARIABLE TYPE IN PYTHON

Data types are the classification or categorization of data items. It represents the kind of value that tells what operations can be performed on a particular data. Since everything is an object in Python programming, data types are actually classes and variables are instance (object) of these classes.

Following are the standard or built-in data type of Python:

- Numeric
- Sequence Type
- Boolean
- Set
- Dictionary

Example:

```
var = 123            # numberic
print("Numbric data : ", var)
 String1 = 'Welcome to the Geeks World'                # Sequence Type
print("String with the use of Single Quotes: ")
print(String1)
print(type(True))             # Boolean
print(type(False))
set1 = set("GeeksForGeeks")    # Creating a Set with
                               # the use of a String
print("\nSet with the use of String: ")
print(set1)
Dict = {1: 'Geeks', 2: 'For', 3: 'Geeks'}            # Creating a Dictionary
                                                      # with Integer Keys
print("\n Dictionary with the use of Integer Keys: ")
print(Dict)
```

Output:

```
Numbric data :  123

String with the use of Single Quotes:  Welcome to the Geeks World

<class 'bool'>

<class 'bool'>

Set with the use of String:      {'r', 'G', 'e', 'k', 'o', 's', 'F'}

Dictionary with the use of Integer Keys: {1: 'Geeks', 2: 'For', 3: 'Geeks'}
```

3. EXECUTING PYTHON FROM THE COMMAND LINE

Python is a well-known high-level programming language. The Python script is basically a file containing code written in Python. The file containing python script has the extension '.py' or can also have the extension '.pyw' if it is being run on a windows machine. To run a python script, we need a python interpreter that needs to be downloaded and installed.

Here is a simple python script to print 'Hello World!'

print('Hello World!')

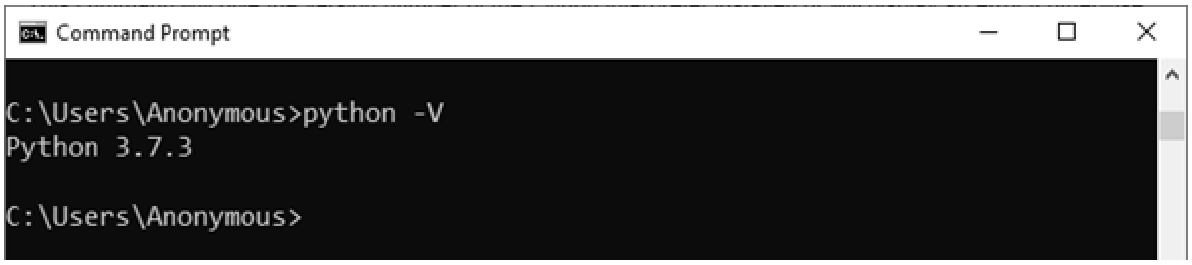
Here, the 'print()' function is to print out any text written within the parenthesis. We can write the text that we want to be printed using either a **single quote** as shown in the above script or a **double quote**.

If you are coming from any other language then you will also notice that there is **no semicolon** at the end of the statement as with Python, you no need to specify the end of the line. And also we don't need to include or import any files to run a simple python script.

There is more than one way to run a python script but before going toward the different ways to run a python script, we first have to check whether a python interpreter is installed on the system or not. So in windows, open **'cmd' (Command Prompt)** and type the following command.

```
python -V
```

This command will give the version number of the Python interpreter installed or will display an error if otherwise.


Command Prompt

```
C:\Users\Anonymous>python -V
Python 3.7.3

C:\Users\Anonymous>
```

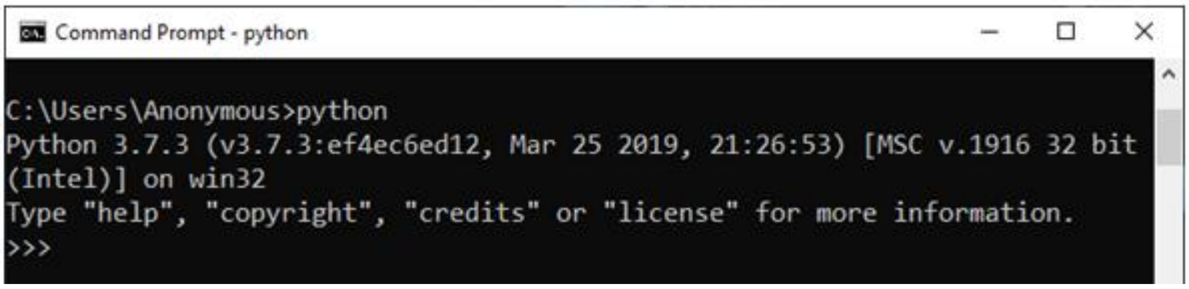
Different ways to run Python Script:

i. Interactive Mode
ii. Command Line
iii. Text Editor (VS Code)
iv. IDE (PyCharm)

- **Interactive Mode**:

  In Interactive Mode, you can run your script line by line in a sequence.

To enter in an interactive mode, you will have to open Command Prompt on your windows machine and type ‘python’ and press Enter.


Command Prompt - python

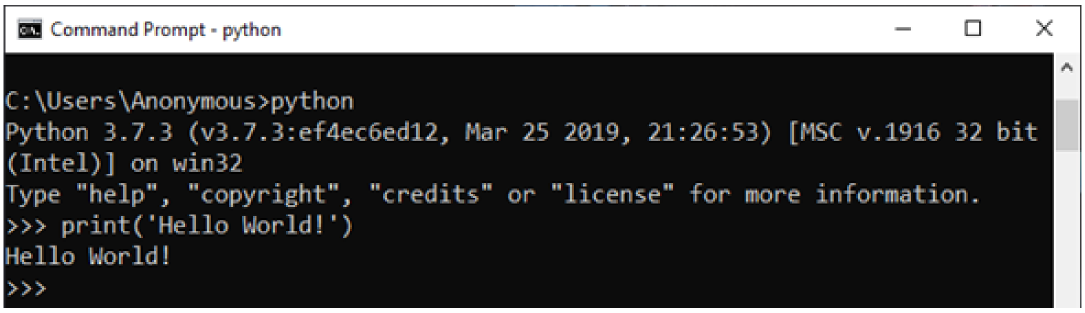
```
C:\Users\Anonymous>python
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Example 1:

Run the following line in the interactive mode:

print('Hello World !')

Output:


Command Prompt - python

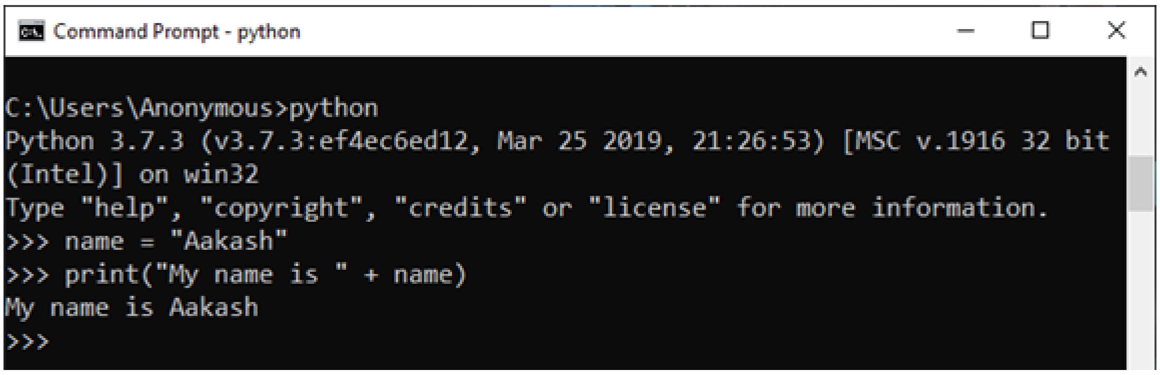
```
C:\Users\Anonymous>python
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print('Hello World!')
Hello World!
>>>
```

Example 2:

Run the following lines one by one in the interactive mode:

```
name = "Aakash"
print("My name is " + name)
```

Output:



Command Prompt - python

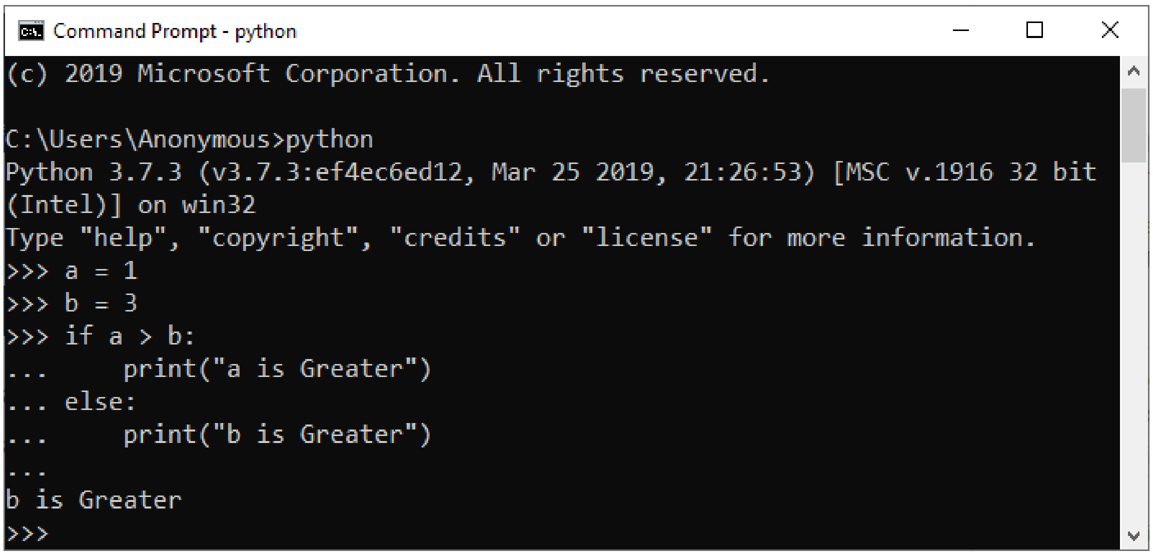
```
C:\Users\Anonymous>python
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> name = "Aakash"
>>> print("My name is " + name)
My name is Aakash
>>>
```

Example 3:

Run the following line one by one in the interactive mode:

```
a = 1
b = 3
if a > b:
   print("a is Greater")
else:
      print("b is Greater")
```

Output:



Command Prompt - python

```
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Anonymous>python
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32 bit
(Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> a = 1
>>> b = 3
>>> if a > b:
...     print("a is Greater")
... else:
...     print("b is Greater")
...
b is Greater
>>>
```

Note: To exit from this mode, press 'Ctrl+Z' and then press 'Enter' or type 'exit()' and then press Enter.

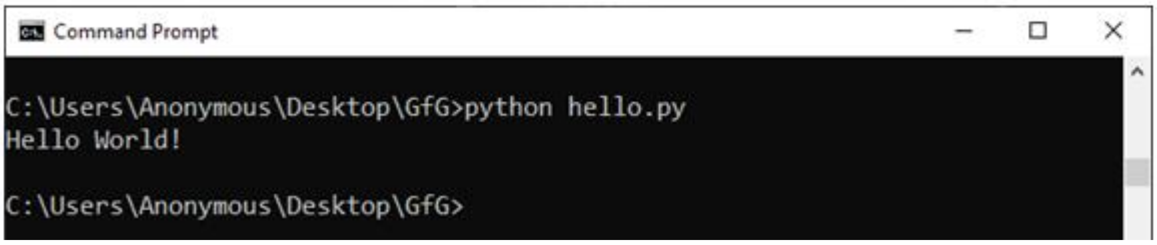
- **Command Line**

  To run a Python script store in a '.py' file in command line, we have to write 'python' keyword before the file name in the command prompt.

python hello.py

You can write your own file name in place of '**hello.py**'.

Output:



```
C:\Users\Anonymous\Desktop\GfG>python hello.py
Hello World!

C:\Users\Anonymous\Desktop\GfG>
```

- **Text Editor (VS Code)**

  To run Python script on a text editor like VS Code (Visual Studio Code) then you will have to do the following:

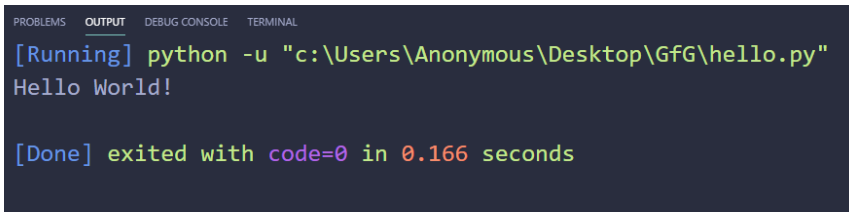
i. Go in the extension section or press 'Ctrl+Shift+X' on windows, then search and install the extension named 'Python' and 'Code Runner'. Restart your vs code after that.

ii. Now, create a new file with the name '**hello.py**' and write the below code in it:

iii. print('Hello World!')

iv. Then, right click anywhere in the text area and select the option that says 'Run Code' or press 'Ctrl+Alt+N' to run the code.

Output:



```
[Running] python -u "c:\Users\Anonymous\Desktop\GfG\hello.py"
Hello World!

[Done] exited with code=0 in 0.166 seconds
```

- **IDE (PyCharm)**

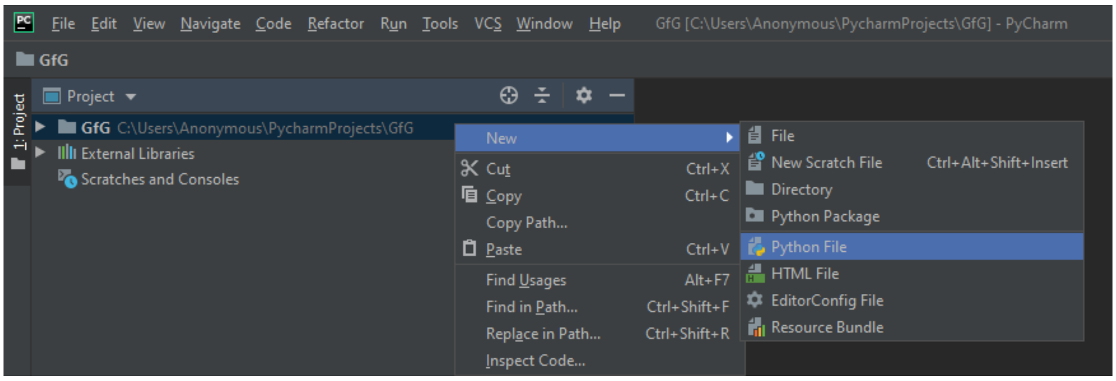
  To run Python script on a IDE (Integrated Development Environment) like PyCharm, you will have to do the following:

i. Create a new project.

ii. Give a name to that project as 'GfG' and click on Create.

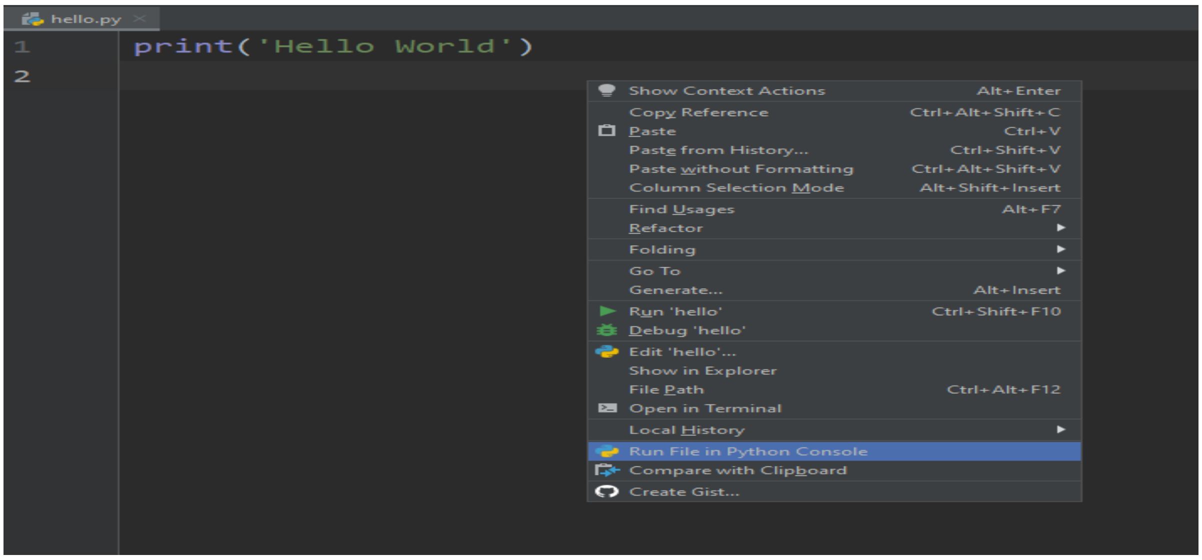
iii. Select the root directory with the project name we specified in the last step. **Right click** on it, go in **New** and click on '**Python file**' option. Then give the name of the file as '**hello**' (you can specify any name as per your project requirement). This will create a

‘**hello.py**’ file in the project root directory.
Note: You don’t have to specify the extension as it will take it automatically.


File Edit View Navigate Code Refactor Run Tools VCS Window Help GfG [C:\Users\Anonymous\PycharmProjects\GfG] - PyCharm
GfG
Project
1: Project
GfG C:\Users\Anonymous\PycharmProjects\GfG
External Libraries
Scratches and Consoles
New
Cut Ctrl+X
Copy Ctrl+C
Copy Path...
Paste Ctrl+V
Find Usages Alt+F7
Find in Path... Ctrl+Shift+F
Replace in Path... Ctrl+Shift+R
Inspect Code...
File
New Scratch File Ctrl+Alt+Shift+Insert
Directory
Python Package
Python File
HTML File
EditorConfig File
Resource Bundle

iv. Now write the below Python script to print the message:
print('Hello World !')

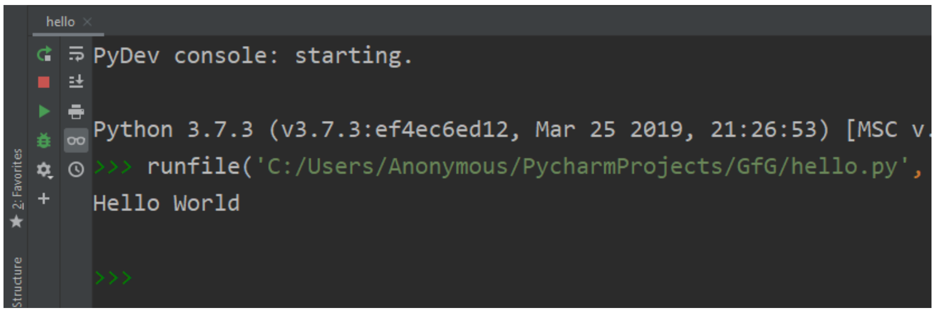
v. To run this python script, **Right click** and select ‘**Run File in Python Console**’ option. This will open a console box at the bottom and show the out put there. We can also run using the **Green Play Button** at the top right corner of the IDE.



```
print('Hello World')
```

hello.py
Show Context Actions Alt+Enter
Copy Reference Ctrl+Alt+Shift+C
Paste Ctrl+V
Paste from History... Ctrl+Shift+V
Paste without Formatting Ctrl+Alt+Shift+V
Column Selection Mode Alt+Shift+Insert
Find Usages Alt+F7
Refactor
Folding
Go To
Generate... Alt+Insert
Run 'hello' Ctrl+Shift+F10
Debug 'hello'
Edit 'hello'...
Show in Explorer
File Path Ctrl+Alt+F12
Open in Terminal
Local History
Run File in Python Console
Compare with Clipboard
Create Gist...

Output:



```
PyDev console: starting.

Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v
>>> runfile('C:/Users/Anonymous/PycharmProjects/GfG/hello.py',
Hello World

>>>
```

5. PYTHON RESERVED WORDS

Reserved words (also called keywords) are defined with predefined meaning and syntax in the language. These keywords have to be used to develop programming instructions. Reserved words can’t be used as identifiers for other programming elements like name of variable, function etc.

Keyword	Description
and	A logical operator
as	To create an alias
assert	For debugging
break	To break out of a loop
class	To define a class
continue	To continue to the next iteration of a loop
def	To define a function
del	To delete an object
elif	Used in conditional statements, same as else if
else	Used in conditional statements
except	Used with exceptions, what to do when an exception occurs
False	Boolean value, result of comparison operations
finally	Used with exceptions, a block of code that will be executed no matter if there is an exception or not
for	To create a for loop
from	To import specific parts of a module
global	To declare a global variable

if	To make a conditional statement
import	To import a module
in	To check if a value is present in a list, tuple, etc.
is	To test if two variables are equal
lambda	To create an anonymous function
None	Represents a null value
nonlocal	To declare a non-local variable
not	A logical operator
or	A logical operator
pass	A null statement, a statement that will do nothing
raise	To raise an exception
return	To exit a function and return a value
True	Boolean value, result of comparison operations
try	To make a try...except statement
while	To create a while loop
with	Used to simplify exception handling
yield	To end a function, returns a generator

6. BASIC SYNTAX IN PYTHON (Execute Python Syntax)

As we learned in the previous page, Python syntax can be executed by writing directly in the Command Line:

```
>>> print("Hello, World!")
Hello, World!
```

Or by creating a python file on the server, using the .py file extension, and running it in the Command Line:

```
C:\Users\Your Name>python myfile.py
```

Example

```
if 5 > 2:
  print("Five is greater than two.")
```

Output

```
Five is greater than two.
```

7. COMMENTS IN PYTHON

It is the lines in the code that are ignored by the interpreter during the execution of the program. Comments enhance the readability of the code and help the programmers to understand the code very carefully. There are three types,

- Single line Comments
- Multiline Comments
- Docstring Comments

i).Single-Line Comments

Python single-line comment starts with the hashtag symbol (#) with no white spaces and lasts till the end of the line. If the comment exceeds one line then put a hashtag on the next line and continue the comment. Python’s single-line comments are proved useful for supplying short explanations for variables, function declarations, and expressions. See the following code snippet demonstrating single line comment:

Example:

```
# Print “GeeksforGeeks !” to console

print("GeeksforGeeks")
```

Output

```
GeeksforGeeks
```

ii).Multi-Line Comments

Python does not provide the option for multiline comments. However, there are different ways through which we can write multiline comments.

Example:

```
# Python program to demonstrate
# multiline comments
print("Multiline comments")
```

Output

```
Multiline comments
```

iii).Python Docstring

Python docstring is the string literals with triple quotes that are appeared right after the function. It is used to associate documentation that has been written with Python modules, functions, classes, and methods. It is added right below the functions, modules, or classes to describe what they do. In Python, the docstring is then made available via the **__doc__** attribute.

Example:

```
def multiply(a, b):
   """Multiplies the value of a and b"""
   return a*b
 # Print the docstring of multiply function
print(multiply.__doc__)
```

Output:

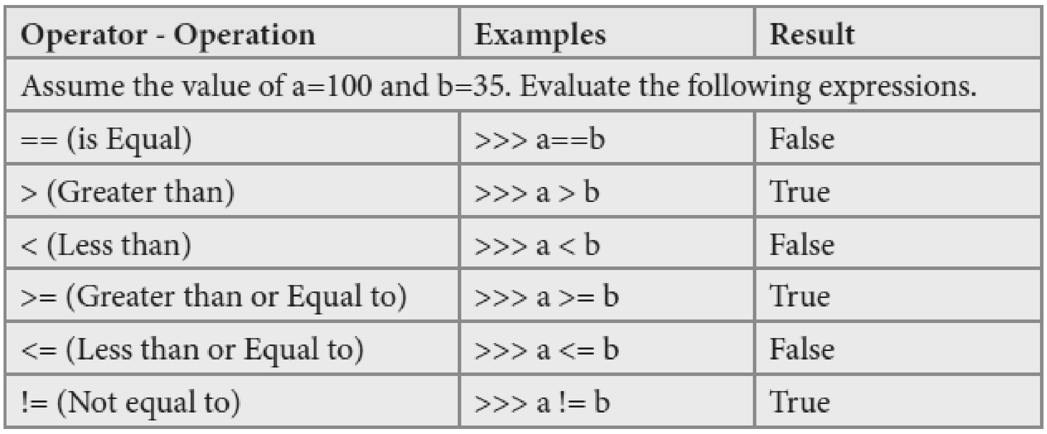
```
Multiplies the value of a and b
```

8. OPERATORS

In computer programming languages operators are special symbols which represent computations, conditional matching etc. The value of an operator used is called **operands**. Operators are categorized as Arithmetic, Relational, Logical, Assignment etc. Value and variables when used with operator are known as **operands**.

RELATIONAL OR COMPARATIVE OPERATORS

A Relational operator is also called as **Comparative** operator which checks the relationship between two operands. If the relation is true, it returns **True**; otherwise it returns **False**.



Operator - Operation	Examples	Result
Assume the value of a=100 and b=35. Evaluate the following expressions.		
== (is Equal)	>>> a==b	False
> (Greater than)	>>> a > b	True
< (Less than)	>>> a < b	False
>= (Greater than or Equal to)	>>> a >= b	True
<= (Less than or Equal to)	>>> a <= b	False
!= (Not equal to)	>>> a != b	True

Example

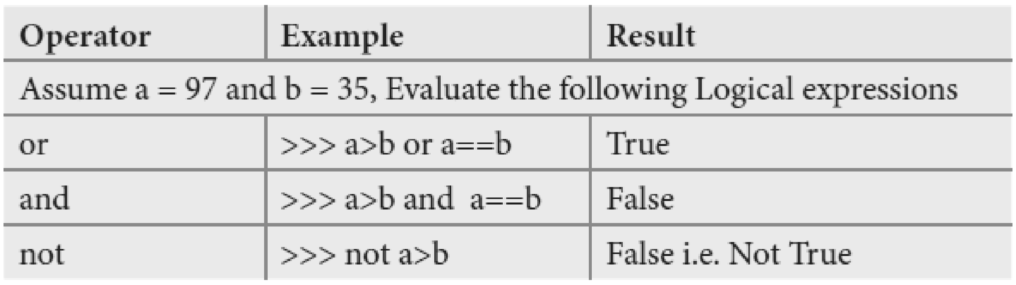
```
#Demo Program to test Relational Operators
a=int (input("Enter a Value for A:"))
b=int (input("Enter a Value for B:"))
print ("A = ",a," and B = ",b)
print ("The a==b = ",a==b)
print ("The a > b = ",a>b)
print ("The a < b = ",a<b)
print ("The a >= b = ",a>=b)
print ("The a <= b = ",a<=0)
print ("The a != b = ",a!=b)
#Program End
```

Output

```
Enter a Value for A:35
Enter a Value for B:56
A = 35 and B = 56
The a==b = False
The a > b = False
The a < b = True
The a >= b = False
The a <= b = False
The a != b = True
```

9. LOGICAL OPERATORS

In python, Logical operators are used to perform logical operations on the given relational expressions. There are three logical operators they are **and, or** and **not**.



Operator	Example	Result
Assume a = 97 and b = 35, Evaluate the following Logical expressions		
or	>>> a>b or a==b	True
and	>>> a>b and a==b	False
not	>>> not a>b	False i.e. Not True

Example

```
#Demo Program to test Logical Operators
a=int (input("Enter a Value for A:"))
b=int (input("Enter a Value for B:"))
print ("A = ",a, " and b = ",b)
print ("The a > b or a == b = ",a>b or a==b)
print ("The a > b and a == b = ",a>b and a==b)
print ("The not a > b = ",not a>b)
#Program End
```

Output

```
Enter a Value for A:50
Enter a Value for B:40
A = 50 and b = 40
The a > b or a == b = True
The a > b and a == b = False
The not a > b = False
```

10. BITWISE OPERATORS

In Python, bitwise operators are used to performing bitwise calculations on integers. The integers are first converted into binary and then operations are performed on bit by bit, hence the name bitwise operators. Then the result is returned in decimal format.

Note: Python bitwise operators work only on integers.

OPERATOR	**DESCRIPTION**	**SYNTAX**

OPERATOR	DESCRIPTION	SYNTAX
&	Bitwise AND	x & y
|	Bitwise OR	x | y
~	Bitwise NOT	~x
^	Bitwise XOR	x ^ y
>>	Bitwise right shift	x>>
<<	Bitwise left shift	x<<

Bitwise AND operator: Returns 1 if both the bits are 1 else 0.

Example:

a = 10 = 1010 (Binary)

b = 4 = 0100 (Binary)

a & b = 1010 & 0100

= 0000

= 0 (Decimal)

Bitwise OR operator: Returns 1 if either of the bit is 1 else 0.

Example:

a = 10 = 1010 (Binary)

b = 4 = 0100 (Binary)

a | b = 1010 | 0100

= 1110

= 14 (Decimal)

Bitwise NOT operator: Returns one's complement of the number.

Example:

a = 10 = 1010 (Binary)

~a = ~1010

= -(1010 + 1)

= -(1011)

= -11 (Decimal)

Bitwise XOR operator: Returns 1 if one of the bits is 1 and the other is 0 else returns false.

Example:

a = 10 = 1010 (Binary)

b = 4 = 0100 (Binary)

a ^ b = 1010 ^ 0100

      = 1110

      = 14 (Decimal)

Example

```
# Bitwise operators
a = 10
b = 4
 # Print bitwise AND operation
print("a & b =", a & b)
# Print bitwise OR operation
print("a | b =", a | b)
# Print bitwise NOT operation
print("~a =", ~a)
 # print bitwise XOR operation
print("a ^ b =", a ^ b)
```

Output:

```
a & b = 0

a | b = 14

~a = -11

a ^ b = 14
```

11. SIMPLE INPUT AND OUTPUT IN PYTHON

Sometimes a developer might want to take user input at some point in the program. To do this Python provides an input() function.

Syntax:

```
input('prompt')
```

where prompt is an optional string that is displayed on the string at the time of taking input.

Example 1: Python get user input with a message

```
# Taking input from the user
name = input("Enter your name: ")
 # Output
```

```
print("Hello, " + name)
print(type(name))
```

Output:

```
Enter your name: GFG
Hello, GFG
<class 'str'>
```

~~~~~~ The End : Unit-I ~~~~~~~
~~~~~~