First Python Program

In this Section, we will discuss the basic syntax of Python, we will run a simple program to print **Hello World** on the console.

Python provides us the two ways to run a program:

- Using Interactive interpreter prompt
- Using a script file

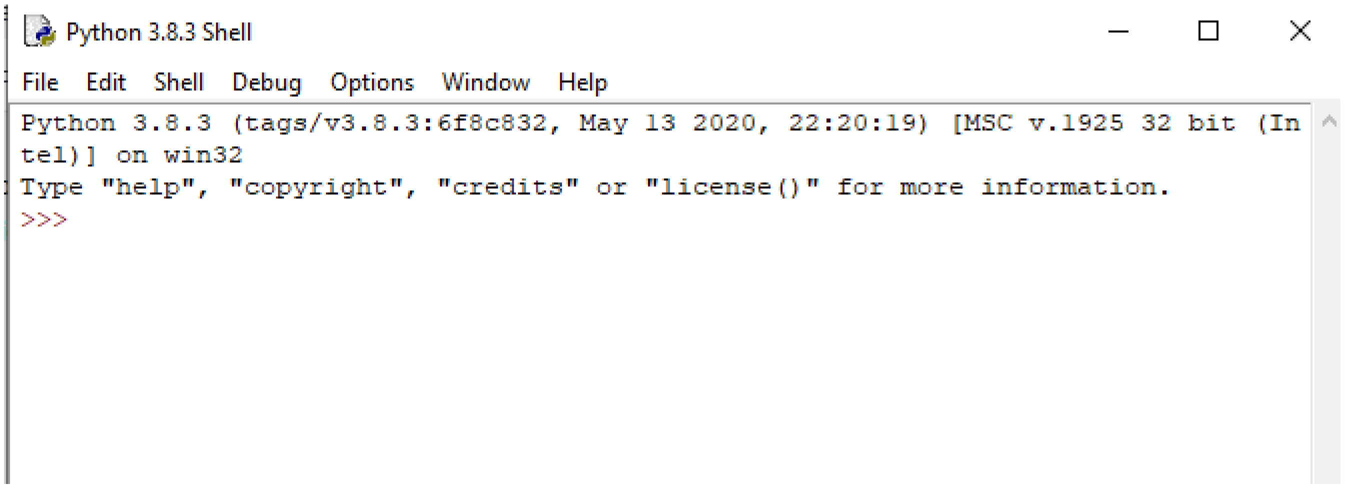
Let's discuss each one of them in detail.

Interactive interpreter prompt

Python provides us the feature to execute the Python statement one by one at the interactive prompt. It is preferable in the case where we are concerned about the output of each line of our Python program.

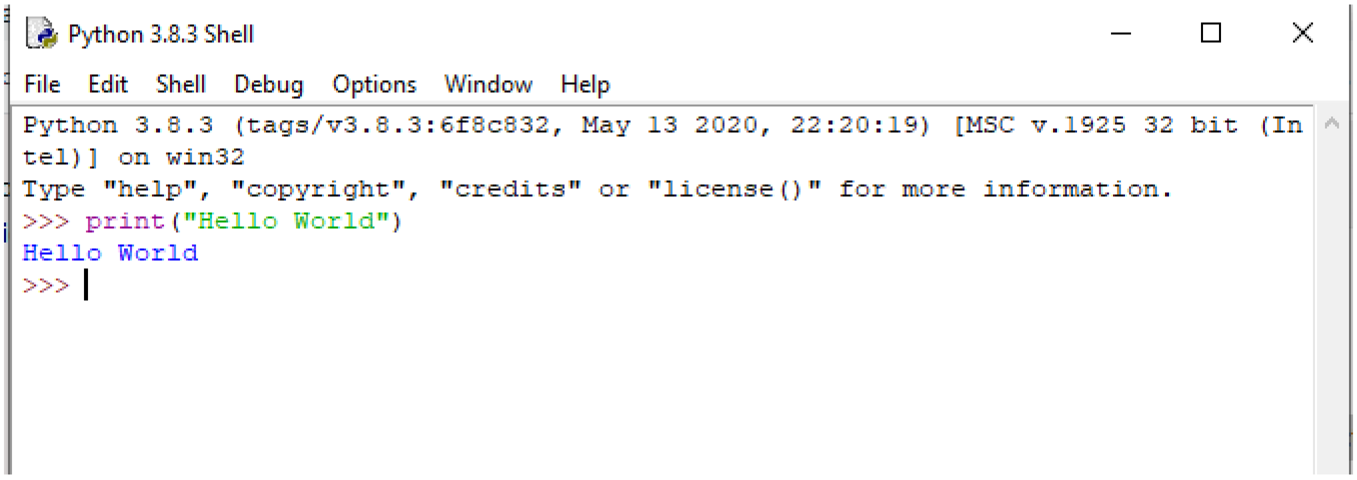
To open the interactive mode, open the terminal (or command prompt) and type python (python3 in case if you have Python2 and Python3 both installed on your system).

It will open the following prompt where we can execute the Python statement and check their impact on the console.



```
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:20:19) [MSC v.1925 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
```

After writing the print statement, press the **Enter** key.



```
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:20:19) [MSC v.1925 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print("Hello World")
Hello World
>>> 
```

Here, we get the message **"Hello World !"** printed on the console.

Using a script file (Script Mode Programming)

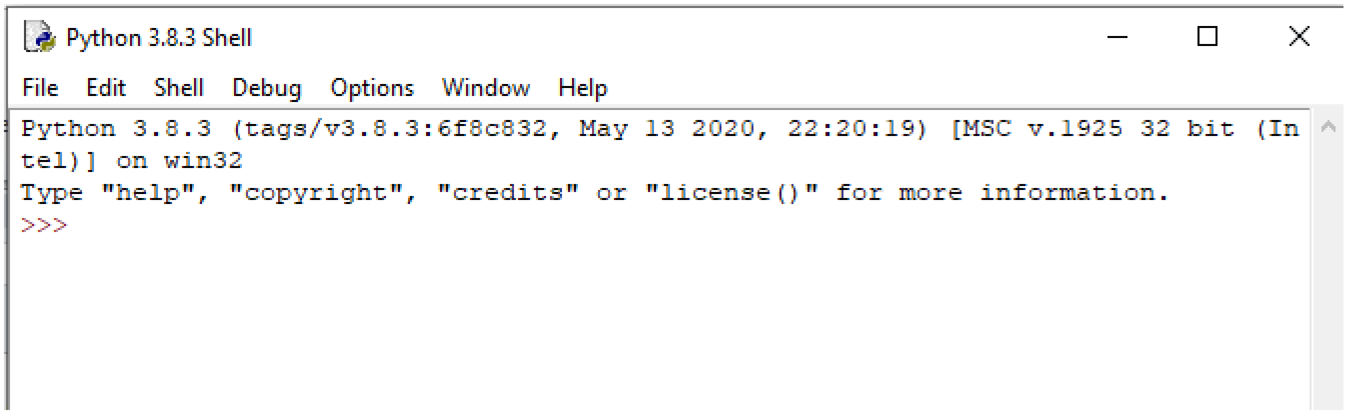
The interpreter prompt is best to run the single-line statements of the code. However, we cannot write the code every-time on the terminal. It is not suitable to write multiple lines of code.

Using the script mode, we can write multiple lines code into a file which can be executed later. For this purpose, we need to open an editor like notepad, create a file named and save it with **.**

extension, which stands for **"Python".** Now, we will implement the above example using the script mode.

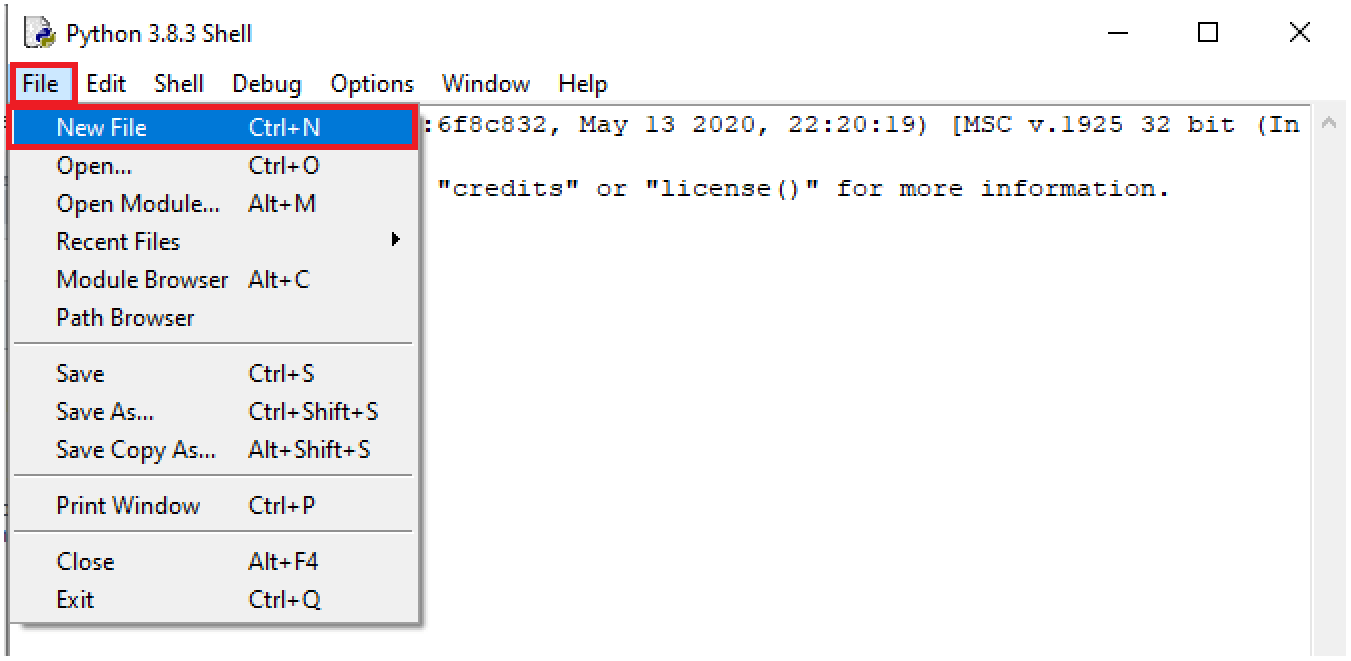
```
1. print ("hello world"); #here, we have used print() function to print the message on the
   console.
```

To run this file named as first.py, we need to run the following command on the terminal.



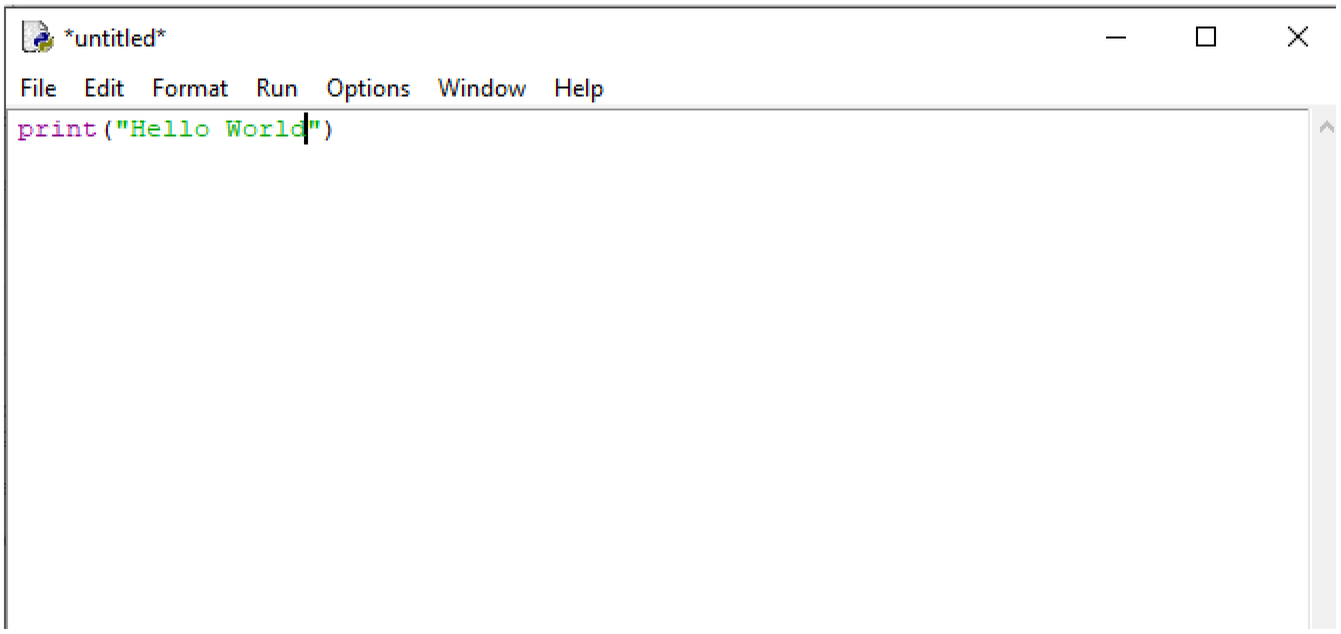
```
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:20:19) [MSC v.1925 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
```

Step - 1: Open the Python interactive shell, and click **"File"** then choose **"New",** it will open a new blank script in which we can write our code.



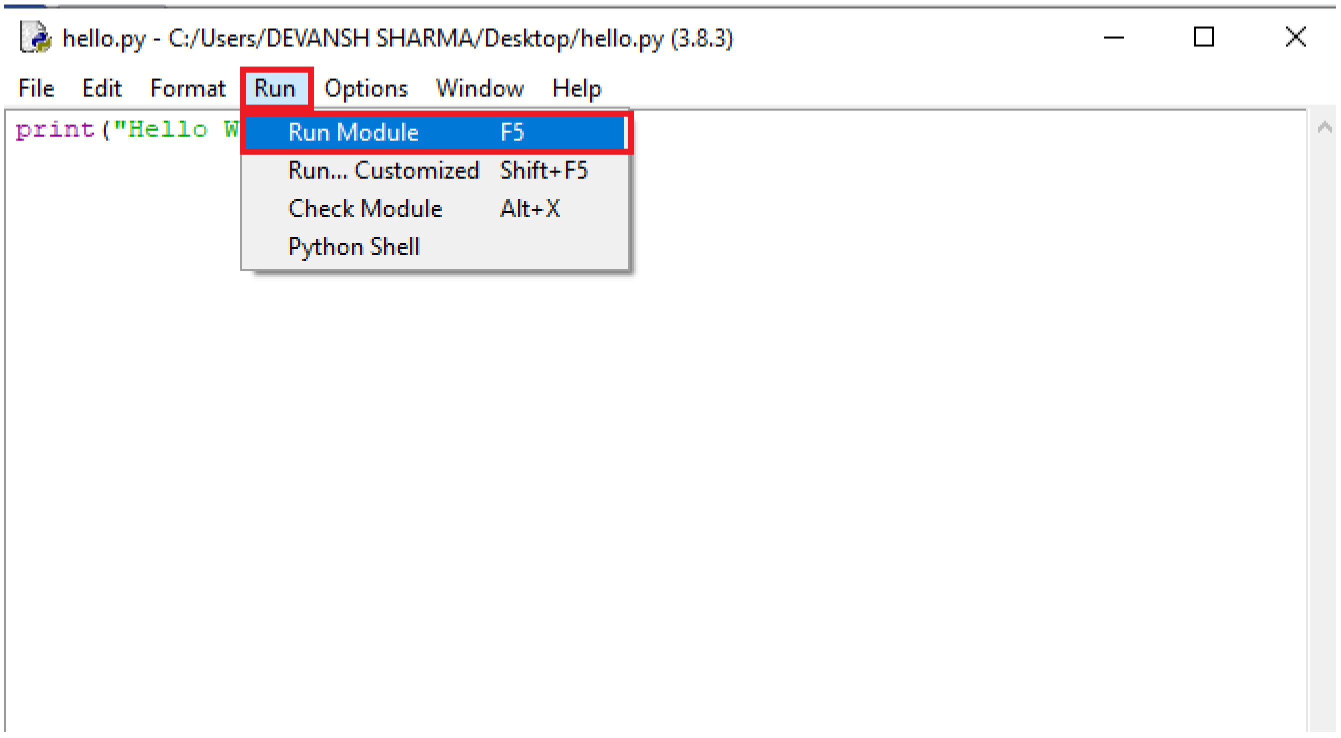
```
:6f8c832, May 13 2020, 22:20:19) [MSC v.1925 32 bit (In
 "credits" or "license()" for more information.
```

Step -2: Now, write the code and press **"Ctrl+S"** to save the file.



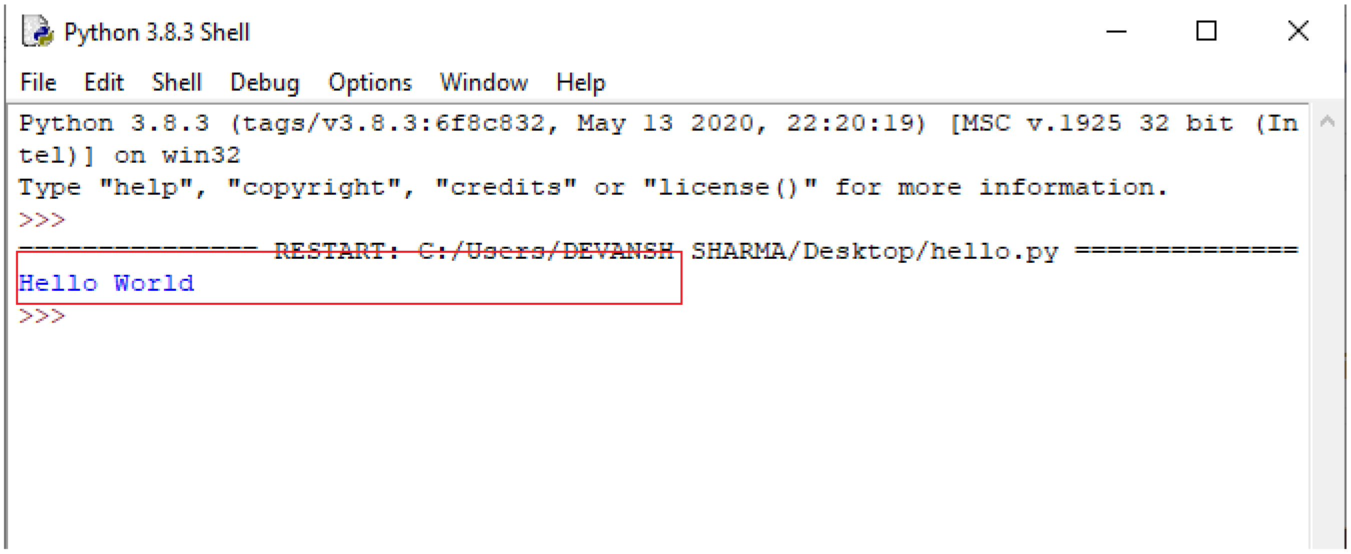
```
print("Hello World")
```

Step - 3: After saving the code, we can run it by clicking "Run" or "Run Module". It will display the output to the shell.


hello.py - C:/Users/DEVANSH SHARMA/Desktop/hello.py (3.8.3)
File Edit Format Run Options Window Help
Run Module F5
Run... Customized Shift+F5
Check Module Alt+X
Python Shell

```
print("Hello W
```

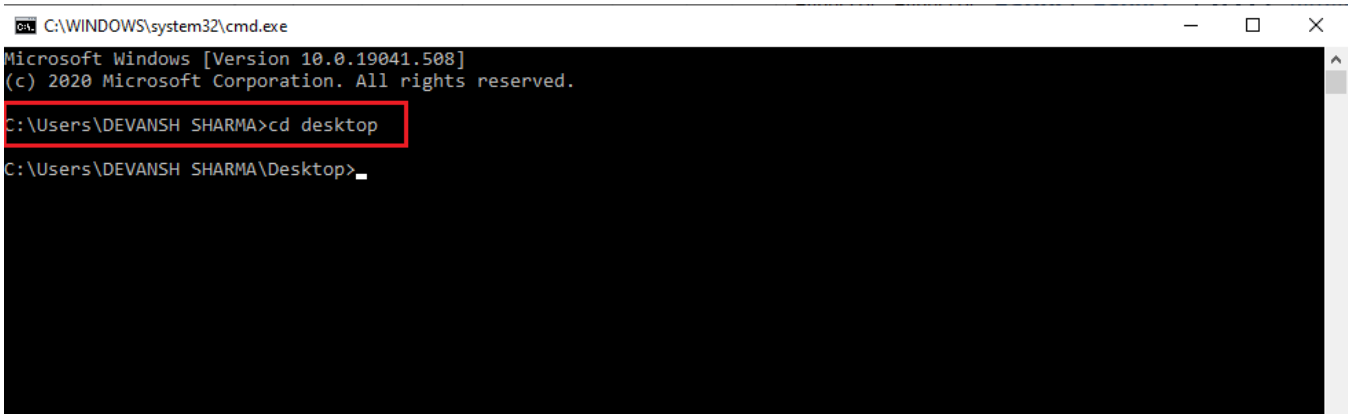
The output will be shown as follows.



```
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:20:19) [MSC v.1925 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
=============== RESTART: C:/Users/DEVANSH SHARMA/Desktop/hello.py ==============
Hello World
>>>
```

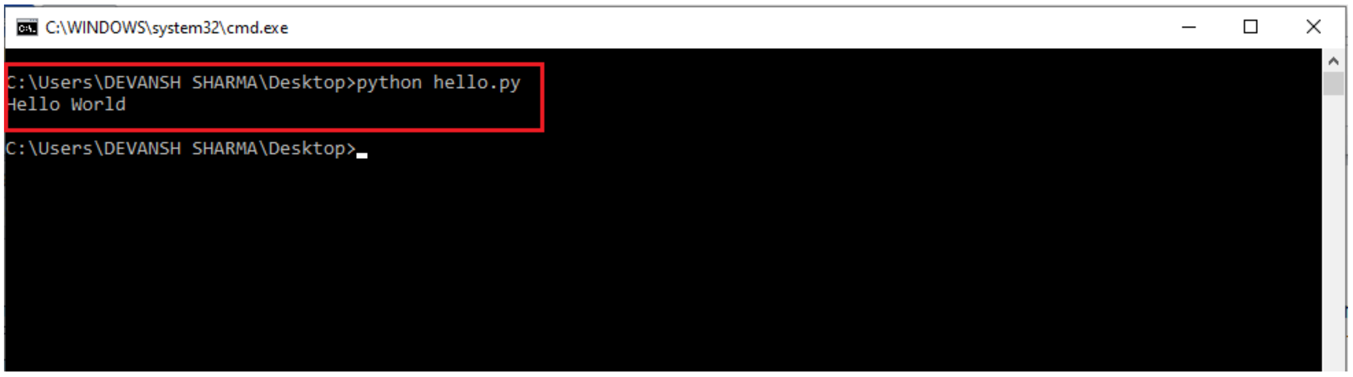
Step - 4: Apart from that, we can also run the file using the operating system terminal. But, we should be aware of the path of the directory where we have saved our file.

- Open the command line prompt and navigate to the directory.



```
Microsoft Windows [Version 10.0.19041.508]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\DEVANSH SHARMA>cd desktop

C:\Users\DEVANSH SHARMA\Desktop>_
```

- We need to type the **python** keyword, followed by the file name and hit enter to run the Python file.



```
C:\Users\DEVANSH SHARMA\Desktop>python hello.py
Hello World

C:\Users\DEVANSH SHARMA\Desktop>_
```

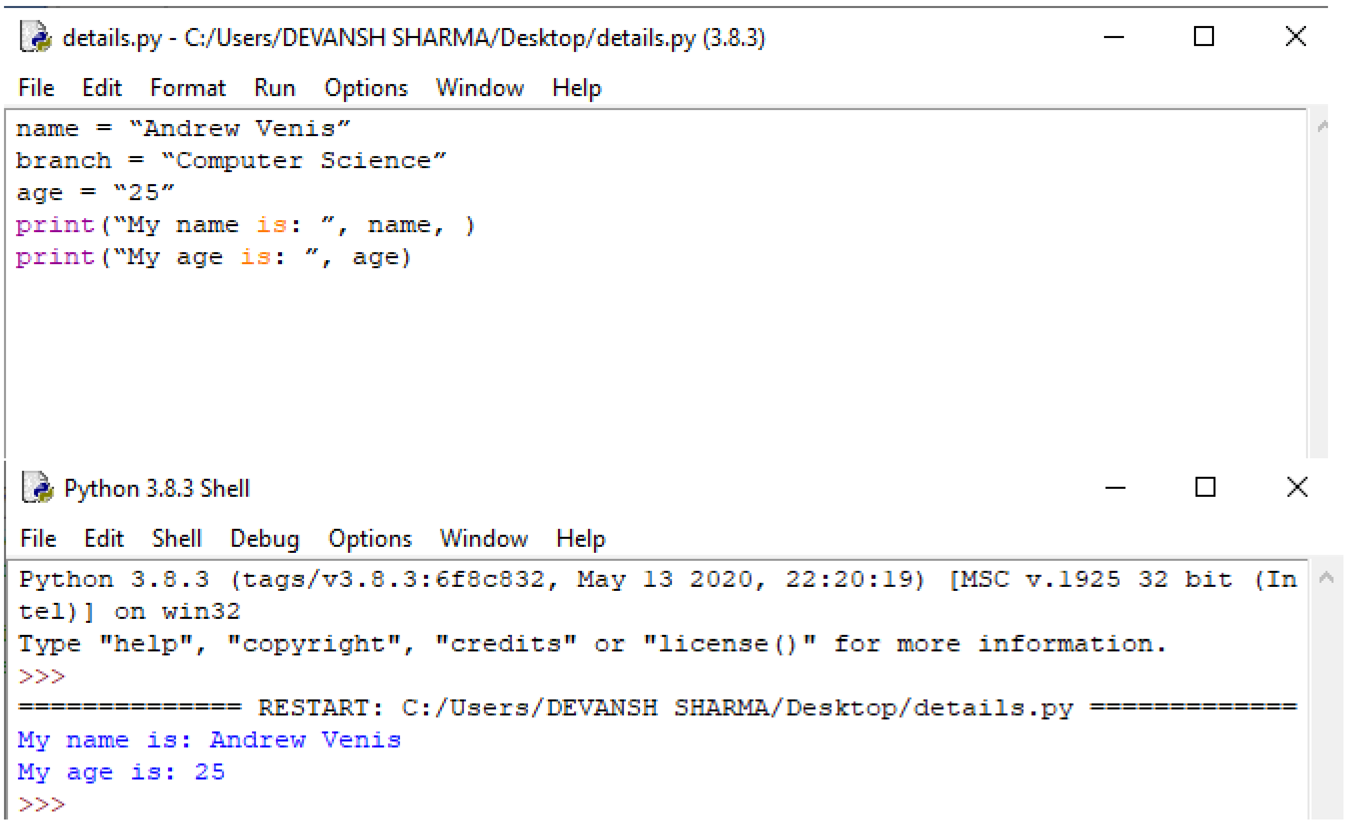
Multi-line Statements

Multi-line statements are written into the notepad like an editor and saved it with **.py** extension. In the following example, we have defined the execution of the multiple code lines using the Python script.

Code:

```python
name = "Andrew Venis"
branch = "Computer Science"
age = "25"
print("My name is: ", name, )
print("My age is: ", age)
```

Script File:



details.py - C:/Users/DEVANSH SHARMA/Desktop/details.py (3.8.3)
File Edit Format Run Options Window Help

```
name = "Andrew Venis"
branch = "Computer Science"
age = "25"
print("My name is: ", name, )
print("My age is: ", age)
```

Python 3.8.3 Shell
File Edit Shell Debug Options Window Help

```
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:20:19) [MSC v.1925 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
============== RESTART: C:/Users/DEVANSH SHARMA/Desktop/details.py =============
My name is: Andrew Venis
My age is: 25
>>>
```

Pros and Cons of Script Mode

The script mode has few advantages and disadvantages as well. Let's understand the following advantages of running code in script mode.

- We can run multiple lines of code.
- Debugging is easy in script mode.
- It is appropriate for beginners and also for experts.

Let's see the disadvantages of the script mode.

- We have to save the code every time if we make any change in the code.
- It can be tedious when we run a single or a few lines of code.

Get Started with PyCharm

In our first program, we have used gedit on our CentOS as an editor. On Windows, we have an alternative like notepad or notepad++ to edit the code. However, these editors are not used as IDE for python since they are unable to show the syntax related suggestions.

JetBrains provides the most popular and a widely used cross-platform IDE **PyCharm** to run the python programs.

PyCharm installation

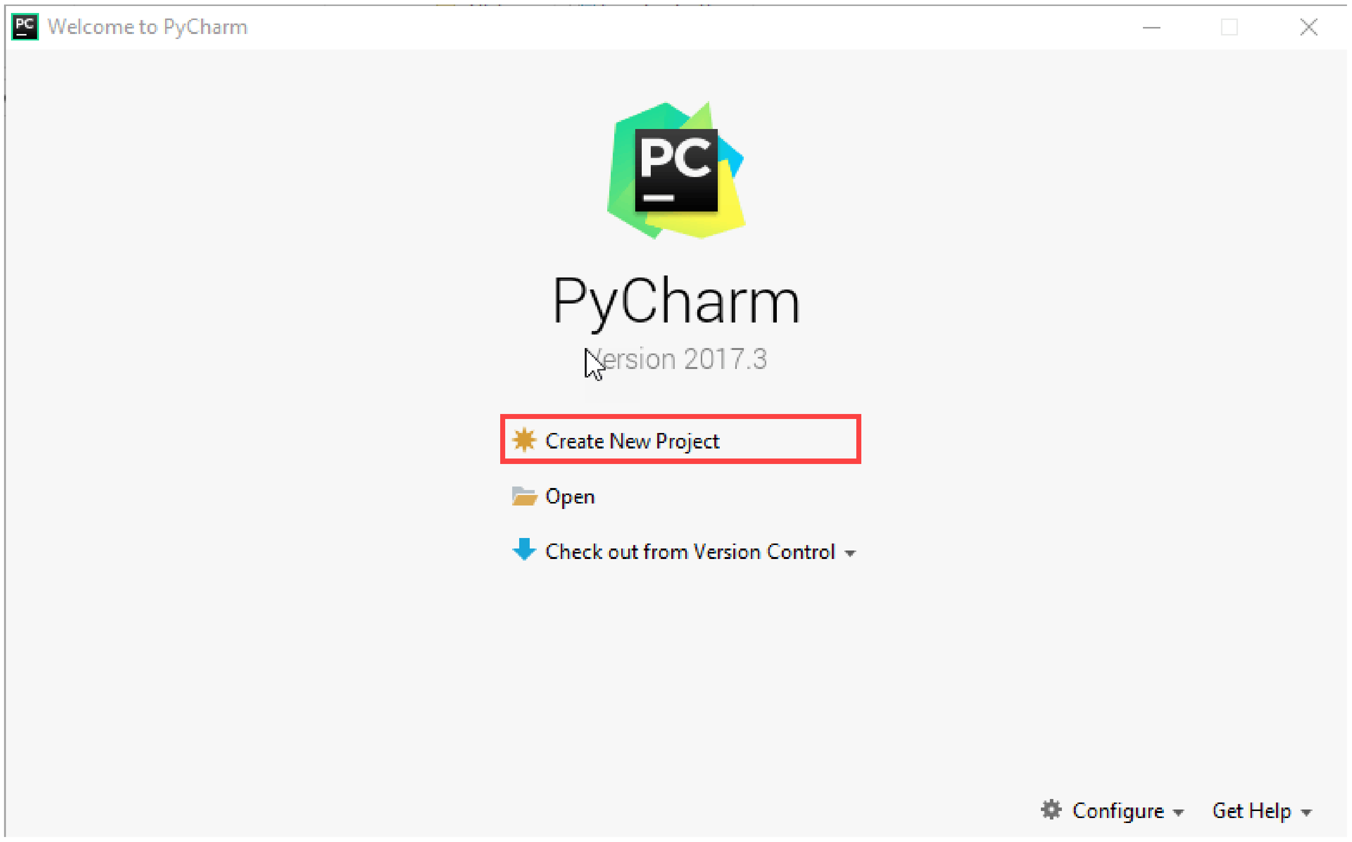
As we have already stated, PyCharm is a cross-platform IDE, and hence it can be installed on a variety of the operating systems. In this section of the tutorial, we will cover the installation process of PyCharm on Windows, MacOS, CentOS, and Ubuntu.

Windows

Installing PyCharm on Windows is very simple. To install PyCharm on Windows operating system, visit the link https://www.jetbrains.com/pycharm/download/download-thanks.html?platform=windows to download the executable installer. **Double click** the installer (.exe) file and install PyCharm by clicking next at each step.

To create a first program to Pycharm follows the following step.

Step - 1. Open Pycharm editor. Click on "Create New Project" option to create new project.

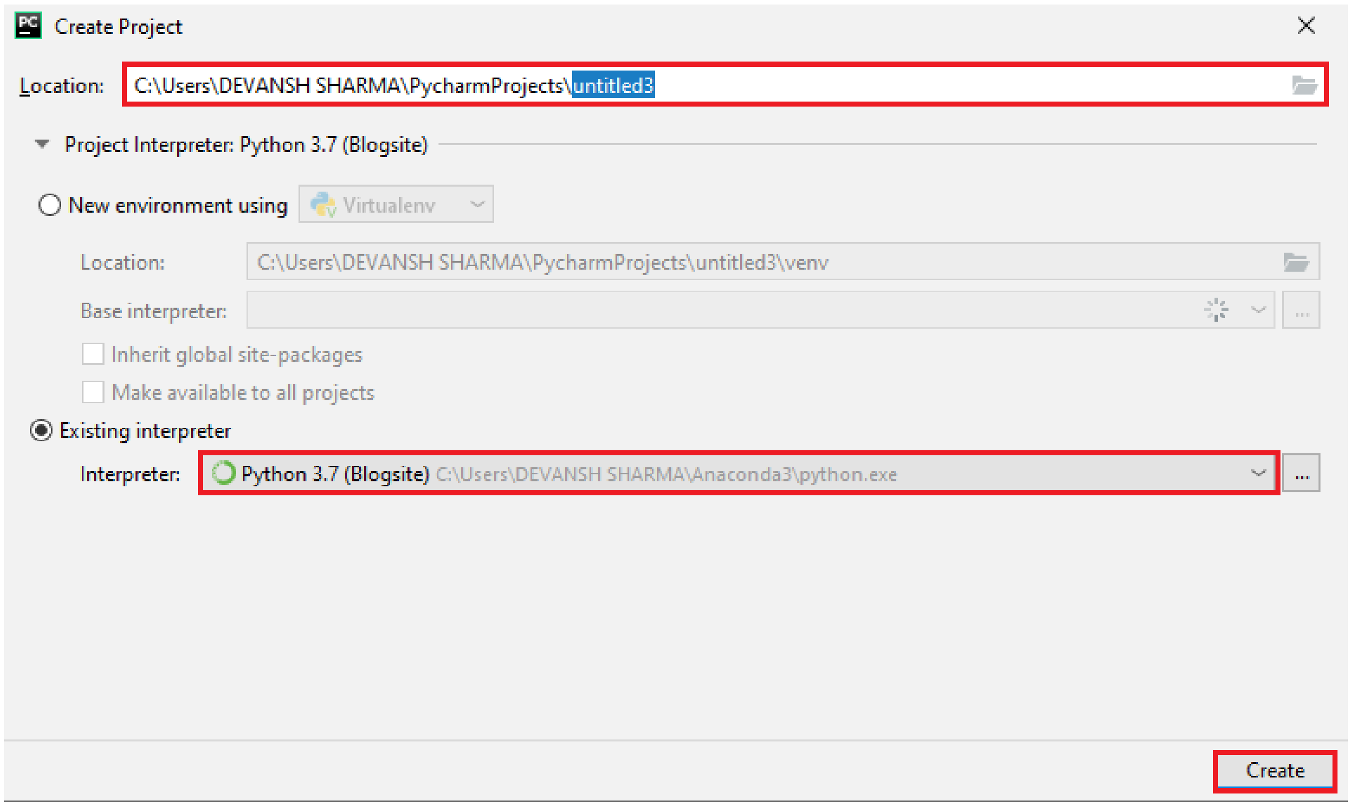


Step - 2. Select a location to save the project.

a. We can save the newly created project at desired memory location or can keep file location as it is but atleast change the project default name **untitled** to **"FirstProject"** or something meaningful.

b. Pycharm automatically found the installed Python interpreter.

c. After change the name click on the "Create" Button.



Create Project

Location: C:\Users\DEVANSH SHARMA\PycharmProjects\untitled3

Project Interpreter: Python 3.7 (Blogsite)

New environment using Virtualenv

Location: C:\Users\DEVANSH SHARMA\PycharmProjects\untitled3\venv

Base interpreter:

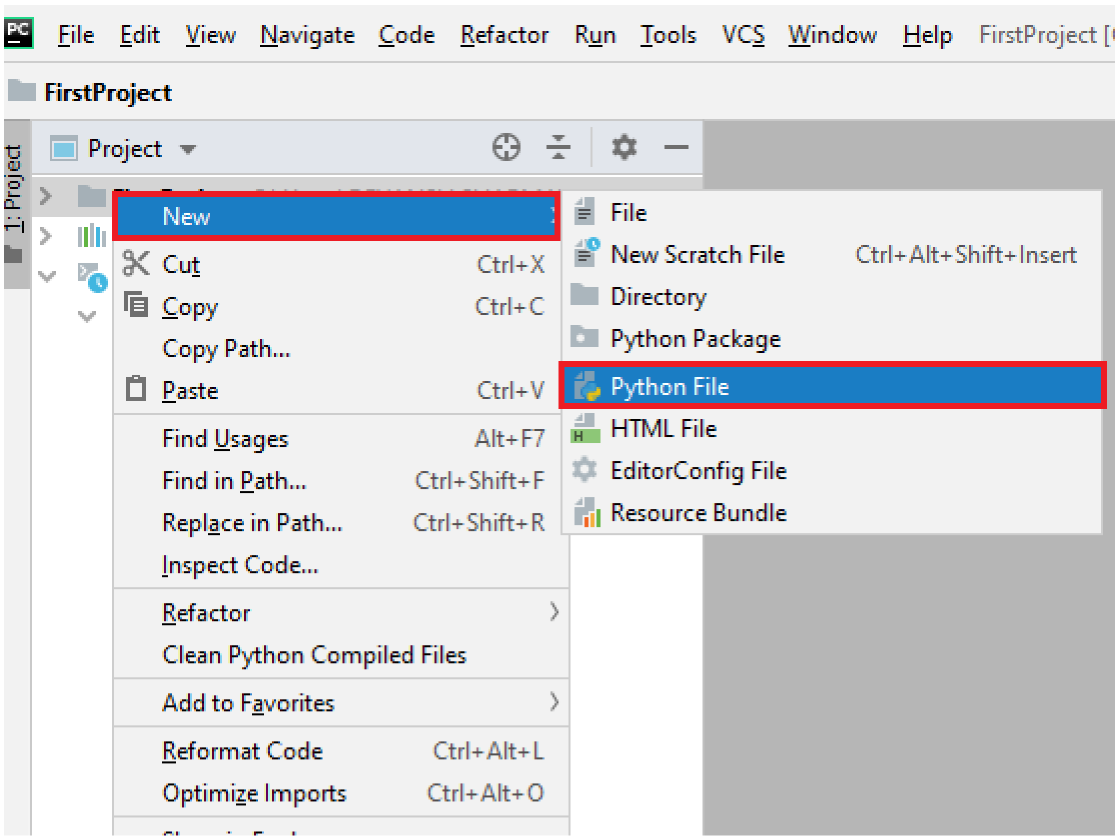
Inherit global site-packages

Make available to all projects

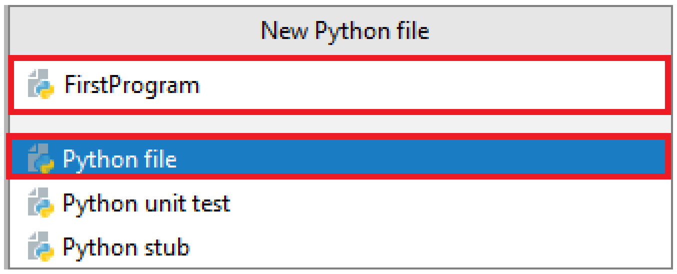
Existing interpreter

Interpreter: Python 3.7 (Blogsite) C:\Users\DEVANSH SHARMA\Anaconda3\python.exe

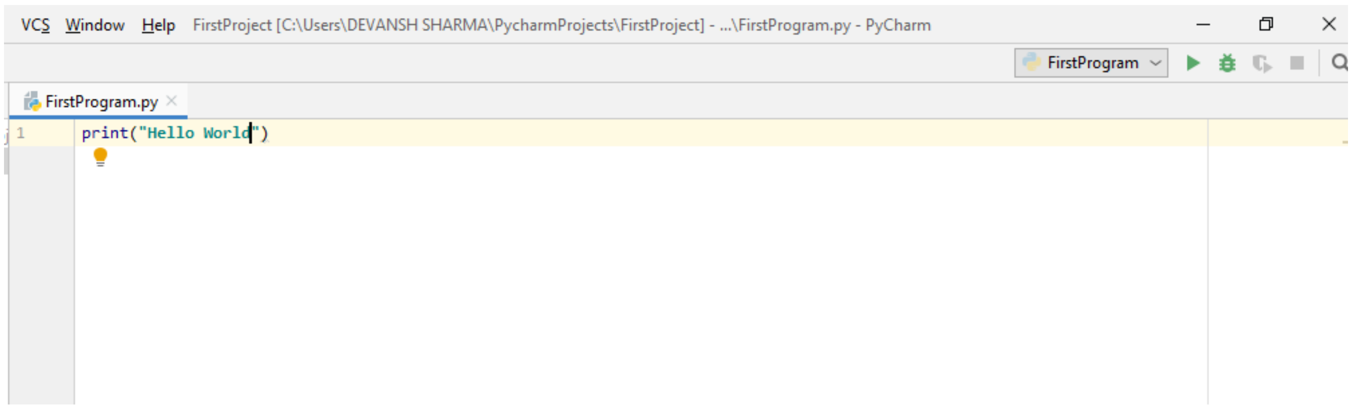
Step - 3. Click on "**File"** menu and select **"New"**. By clicking "New" option it will show various file formats. Select the "Python File".



Step - 4. Now type the name of the Python file and click on "OK". We have written the "FirstProgram".

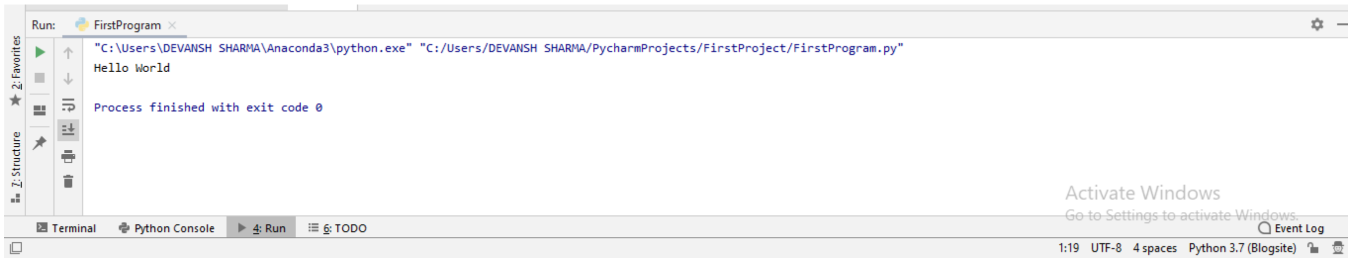


Step - 5. Now type the first program - print("Hello World") then click on the "Run" menu to run program.


VCS Window Help FirstProject [C:\Users\DEVANSH SHARMA\PycharmProjects\FirstProject] - ...\FirstProgram.py - PyCharm
FirstProgram
FirstProgram.py

```
print("Hello World")
```

Step - 6. The output will appear at the bottom of the screen.



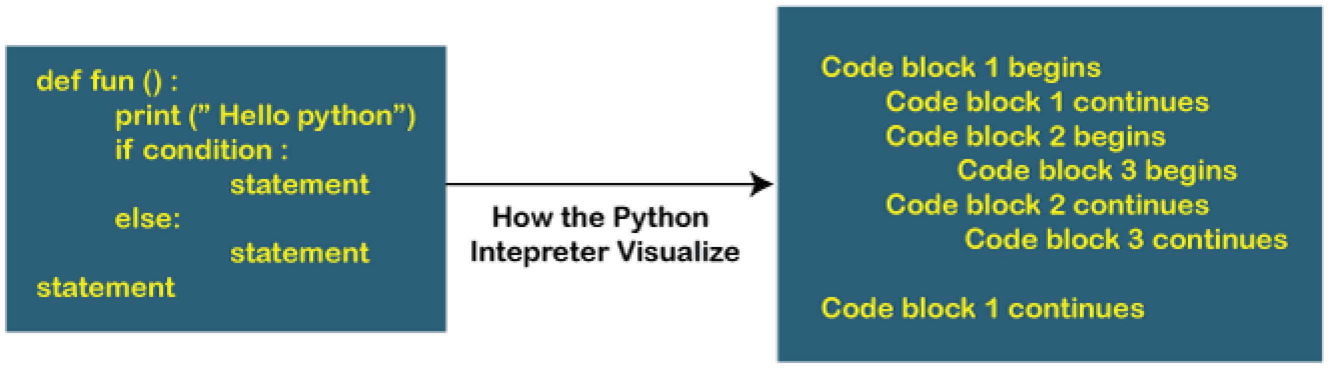
```
"C:\Users\DEVANSH SHARMA\Anaconda3\python.exe" "C:/Users/DEVANSH SHARMA/PycharmProjects/FirstProject/FirstProgram.py"
Hello World

Process finished with exit code 0
```

Basic Syntax of Python

Indentation and Comment in Python

Indentation is the most significant concept of the Python programming language. Improper use of indentation will end up **"IndentationError"** in our code.

Indentation is nothing but adding whitespaces before the statement when it is needed. Without indentation Python doesn't know which statement to be executed to next. Indentation also defines which statements belong to which block. If there is no indentation or improper indentation, it will display "**IndentationError"** and interrupt our code.



Python indentation defines the particular group of statements belongs to the particular block. The programming languages such as C, C++, java use the curly braces {} to define code blocks.

In Python, statements that are the same level to the right belong to the same block. We can use four whitespaces to define indentation. Let's see the following lines of code.

Example -

```
list1 = [1, 2, 3, 4, 5]
for i in list1:
    print(i)
    if i==4:
       break
print("End of for loop")
```

Output:

```
1
2
3
4
End of for loop
```

Explanation:

In the above code, for loop has a code blocks and if the statement has its code block inside for loop. Both indented with four whitespaces. The last **print()** statement is not indented; that's means it doesn't belong to for loop.

Comments in Python

Comments are essential for defining the code and help us and other to understand the code. By looking the comment, we can easily understand the intention of every line that we have written in code. We can also find the error very easily, fix them, and use in other applications.

In Python, we can apply comments using the # hash character. The Python interpreter entirely ignores the lines followed by a hash character. A good programmer always uses the comments to make code under stable. Let's see the following example of a comment.

```
name  = "Thomas"   # Assigning string value to the name variable
```

We can add comment in each line of the Python code.

```
Fees = 10000      # defining course fees is 10000
Fees = 20000      # defining course fees is 20000
```

It is good idea to add code in any line of the code section of code whose purpose is not obvious. This is a best practice to learn while doing the coding.

Types of Comment

Python provides the facility to write comments in two ways- single line comment and multi-line comment.

Single-Line Comment - Single-Line comment starts with the hash # character followed by text for further explanation.

```
# defining the marks of a student
Marks = 90
```

We can also write a comment next to a code statement. Consider the following example.

```
Name = "James"   # the name of a student is James
Marks = 90            # defining student's marks
Branch = "Computer Science"   # defining student branch
```

Multi-Line Comments - Python doesn't have explicit support for multi-line comments but we can use hash # character to the multiple lines. **For example -**

```
# we are defining for loop
# To iterate the given list.
# run this code.
```

We can also use another way.

```
" " "
This is an example
Of multi-line comment
Using triple-quotes
" " "
```

This is the basic introduction of the comments. Visit our **Python Comment** tutorial to learn it in detail.

Python First Program

Unlike the other programming languages, Python provides the facility to execute the code using few lines. **For example** - Suppose we want to print the **"Hello World"** program in Java; it will take three lines to print it.

```
public class HelloWorld {
 public static void main(String[] args){
// Prints "Hello, World" to the terminal window.
  System.out.println("Hello World");
 }
 }
```

On the other hand, we can do this using one statement in Python.

```
print("Hello World")
```

Both programs will print the same result, but it takes only one statement without using a semicolon or curly braces in Python.

Python print() Function

The **print()** function displays the given object to the standard output device (screen) or to the text stream file.

Unlike the other programming languages, Python **print()** function is most unique and versatile function.

The syntax of **print()** function is given below.

```
print(*objects, sep=' ', end='\n')
```

Let's explain its parameters one by one.

- **objects -** An object is nothing but a statement that to be printed. The * sign represents that there can be multiple statements.
- **sep -** The **sep** parameter separates the print values. Default values is ' '.
- **end -** The **end** is printed at last in the statement.

Let's understand the following example.

Example - 1: Return a value

```python
print("Welcome to Python Programming.")

a = 10
# Two objects are passed in print() function
print("a =", a)

b = a
# Three objects are passed in print function
print('a =', a, '= b')
```

Output:

```
Welcome to Python Programming.
a = 10
a = 10 = b
```

As we can see in the above output, the multiple objects can be printed in the single **print()** statement. We just need to use comma (,) to separate with each other.

Example - 2: Using sep and end argument

```python
a = 10
print("a =", a, sep='dddd', end='\n\n\n')
print("a =", a, sep='0', end='$$$$$')
```

Output:

```
a =dddd10
a =010$$$$$
```

In the first **print()** statement, we use the **sep** and **end** arguments. The given object is printed just after the **sep** values. The value of end parameter printed at the last of given object. As we can see that, the second **print()** function printed the result after the three black lines.

Taking Input from the User

Python provides the **input()** function which is used to take input from the user. Let's understand the following example.

Example -

```
name = input("Enter a name of student:")
print("The student name is: ", name)
```

Output:

```
Enter a name of student: Devansh
The student name is:   Devansh
```

By default, the **input()** function takes the string input but what if we want to take other data types as an input.

If we want to take input as an integer number, we need to typecast the **input()** function into an integer.

For example -

Example -

```
a  = int(input("Enter first number: "))
b = int(input("Enter second number: "))

print(a+b)
```

Output:

```
Enter first number: 50
Enter second number: 100
150
```

We can take any type of values using **input()** function.

Python Casting

There may be times when you want to specify a type on to a variable. This can be done with casting. Python is an object-orientated language, and as such it uses classes to define data types, including its primitive types.

Casting in python is therefore done using constructor functions:

- `int()` - constructs an integer number from an integer literal, a float literal (by removing all decimals), or a string literal (providing the string represents a whole number)

- `float()` - constructs a float number from an integer literal, a float literal or a string literal (providing the string represents a float or an integer)
- `str()` - constructs a string from a wide variety of data types, including strings, integer literals and float literals

Example

Integers:

```
x = int(1)   # x will be 1
y = int(2.8) # y will be 2
z = int("3") # z will be 3
```

Example

Floats:

```
x = float(1)     # x will be 1.0
y = float(2.8)   # y will be 2.8
z = float("3")   # z will be 3.0
w = float("4.2") # w will be 4.2
```

Example

Strings:

```
x = str("s1") # x will be 's1'
y = str(2)    # y will be '2'
z = str(3.0)  # z will be '3.0'
```

Escape sequence

```
print("Welcome to Python programming")
print("Welcome to\nPython programming")
print("Welcome to\tPython programming")
print("Welcome to \"Python programming\" ")
print("Welcome to \'Python programming\' ")
print("Python1 \
Python2 \
Python3")
print("5\\6")
print('\'')
print("\"")
print("Hello \n World!")
print("Hello \r World!")
print("Hello \t World!")

'''We can ignore the escape sequence from the given string
by using the raw string. We can do this by writing r or R in
front of the string. Consider the following example.'''

print(r"C:\\Users\\DEVANSH SHARMA\\Python32")
print("C:\\Users\\DEVANSH SHARMA\\Python32")
```