CLASS-XI

STUDY NOTES

GETTING STARTED WITH PYTHON

Period-01

Introduction:

Python programming language was developed by Guido Van Rossum in February 1991. Python is based on or influenced with two programming languages:

- ABC language, a teaching language created as a replacement of BASIC, and
- Modula-3

Python is an easy-to-learn yet powerful object oriented programming language. It is a very high level programming Language yet as powerful as many other middle-level not so high-level languages like C, C++, Java etc.

Though Python language came into being in early 1990's, yet it is competing with ever-popular languages such as C, C++, Java etc. in popularity index. Although, it is not perfect for every type of application, yet it has much strength that makes it a good choice for many situations. Let's see what these pluses of Python are.

- Pluses of Python:
- Easy to Use Object Oriented Language
- Expressive Language
- Interpreted Language
- Its Completeness
- Cross-platform Language
- Free and Open Source

1. Easy to Use:

Python is compact and very easy to use object oriented language with very simple syntax rules. It is a very high level language and thus very-very programmer-friendly.

2. Expressive Language:

Python's expressiveness means it is more capable to expressing the code's purpose than many other languages. Reason being- fewer lines of code, simpler syntax.

For example, consider following two sets of codes:

```
# In  C++ :  Swap Values
int  a = 2,  b = 3,  tmp  ;
tmp  = a ;
a=  b ;
b = tmp;
```

```
# In Python :  Swap values
a,  b = 2,  3
a, b = b, a
```

3. Interpreted Language:

Python is an interpreted language, not a compiled language. This means that the Python installation interprets and executes the code line by line at a time. It makes Python an easy-to-debug language and thus suitable for beginners to advanced user.

4. Its Completeness:

When you install Python, you get everything you need to do real work. You do not need download and install additional libraries ; all types of required functionality is available through various modules of Python standard library. For example, for diverse functionality such as emails, web-pages, databases, GUI development network connections and many more,

everything is available in Python standard library. Thus, it is also called - Python follows - "Batteries Included" philosophy.

5. Cross-platform Language:

Python can run equally well on variety of platforms - Windows, Linux/UNIX, Macintosh, supercomputers, smart phones etc.2 Isn't that amazing ? And that makes Python a cross-platform language. Or in other words, Python is a portable language.

6. Free and Open Source:

Python language is freely available i.e., without any cost (from www.python.org). And not only it free, its source-code (i.e., complete program instructions) is also available, i.e., it is open-source also. Do you know, you can modify, improve/extend open-source software.

7. Variety of Usage/Applications:

Python has evolved into a powerful, complete and useful language over these years. These days Python is being used in many diverse fields/applications, some of which are:

- Scripting
- Web Applications
- Game development
- System Administrations
- Rapid Prototyping
- GUI Programs
- Database Applications

PYTHON - SOME MINUSES (SO HUMAN LIKE):

Although Python is very powerful yet simple language with so many advantages, it is not the Perfect Programming language. There are some areas where Python does not offer much or is not that capable. Let's see what these are:

1. Not the Fastest Language: Python is an interpreted language not a fully compiled one. Python is first semi-compiled into an internal byte-code, which is then exerted by a Python interpreter. Fully compiled languages are faster than their interpreted counterparts. So, here Python is little weaker though it offers faster development times but execution-times are not that fast compared to some compiled languages.

2. Lesser Libraries than C, Java, Perl:

Python offers library support for almost all computing programs, but its library is still competent with languages like C, Java, and Perl as they have larger collections available. Some· in some cases, these languages offer better and multiple solutions than Python.

3. Not Strong on Type-binding:

Python interpreter is not very strong on catching 'Type-mismatch' issues. For example, if you declare a variable as integer but later store a string value in it, Python won't complain or pin-point it.

4. Not Easily Convertible:

Because of its lack of syntax, Python is an easy language to program in. But this advantage has a flip-side too: it becomes a disadvantage

when it comes to translating a program into another programming language. This is because most other languages have structured defined syntax.

Since most other programming languages have strong-syntax, the translation from Python to another language would require the user to carefully examine the Python code and its structure and then implement the same structure into other programming language's syntax.

So, now you are familiar with what all Python offers. As a free and open-source language, its users are growing by leaps and bounds.

Period-02

Introduction:

Familiarization with the basics of Python Programming:

(Interactive & Script mode)

WORKING IN PYTHON:

Before you start working in Python, you need to install Python on your computers. There are multiple Python distributions available today.

- Default installation available from **www.python.org** is called CPython installation and comes with Python interpreter, Python IDLE (Python GUI) and Pip (package installer).
- There are many other Python distributions available these days. Anaconda Python distribution is one such highly recommended distribution that comes preloaded with many packages and libraries (e.g., NumPy, SciPy, Panda libraries etc),

- Many popular IDEs are also available e.g., Spyder IDE, PyCharm IDE etc. Of these, Spyder IDE is already available as a part of Anaconda Python distribution.

Once you have Python installed on your computers, you are ready to work on it. You can work in Python in following different ways:

(i) In Interactive mode (also called Immediate Mode)

(ii) In Script mode

Working in Default CPython Distribution:

The default distribution, CPython, comes with Python interpreter, Python IDLE (GUI based) and pip (package installer). To work in interactive as well as script mode, you need to open Python IDLE.

Working in Interactive Mode (Python IDLE):

Interactive mode of working means you type the command - one command at a time, and the Python executes the given command there and then and gives you output. In interactive mode, you type the command in front of Python command prompt >>>.

For example, if you type 2 + 5 in front of Python prompt, it will give you result as 7 :

```
>>> 2 + 5   <-- command/expression given here
7
```

Result returned by Python

To work in interactive mode, follow the process given below :

(i) Click **Start button -> All Programs -> Python 3.6.x -> IDLE (Python GUI)** [see Fig. 5.1(a)]

Or

Click **Start button -> All Programs -> Python 3.6.x -> Python (**command line)

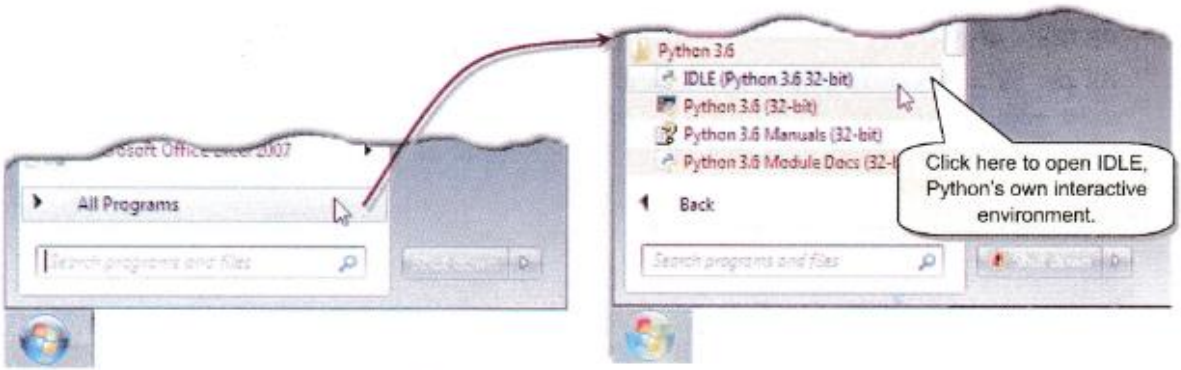


Figure 5.1 (*a*) Starting Python Shell.

(ii) It will open Python Shell [see Fig. 5.1(b)] where you'll see the Python prompt (three '>' signs i.e, >>>). The interactive interpreter of Python is also called Python Shell.

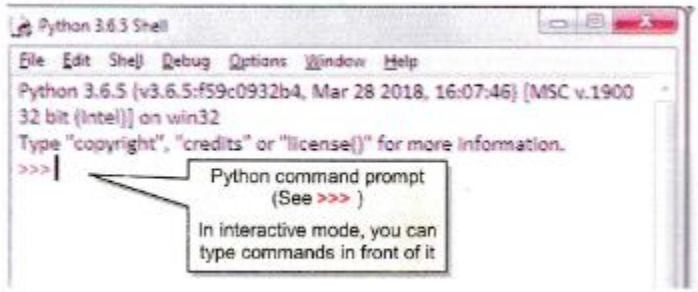


Figure 5.1 (*b*) Python's interactive interpreter – Python Shell.

(iii) Type command in front of this Python prompt and Python will immediately give you the result. [See Fig. 5.1(c)]

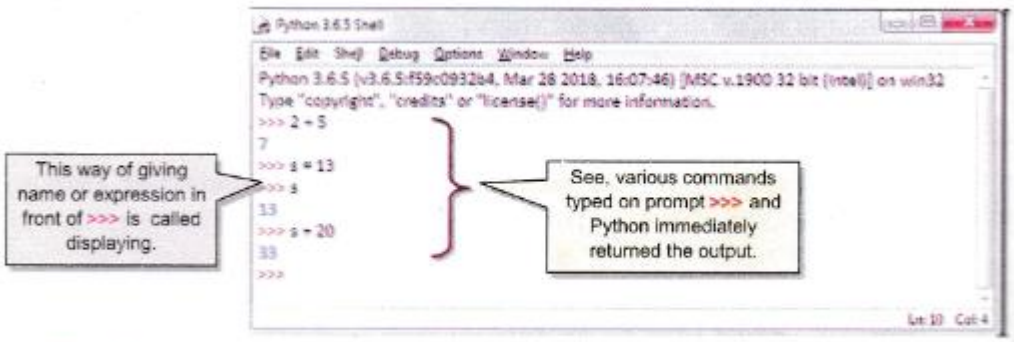


Figure 5.1 (*c*) Interactive commands and their output in Python Shell.

For example, to print string "Hello" on the screen, you need to type the following in front of Python prompt (>>>).

```
>>> print ("Hello")
```

And Python interpreter will immediately display string Hello below the command. To display, you just need to mention name or expression [Fig. 5.1(c)] in front of the prompt.

Figure 5.1(c) shows you some sample commands that we typed in Python shell and the output returned by Python interpreter.

Interactive mode proves very useful for testing code; you type the commands one by one and get the result or error one by one.

Working in Script Mode (Python IDLE):

What if you want to save all the commands in the form of program file and want to see all output lines together rather than sandwiched between successive commands? With interactive mode, you cannot do so, for:

- Interactive mode does not save the commands entered by you in the form of a program.
- The output is sandwiched between the command lines [see Fig. 5.1(c)].

The solution to above problems is the Script mode. To work in a script mode, you need to do the following.

Step 1: Create Module / Script / Program File:

Firstly, you have to create and save a module / Script / Program file. To do so, follow these instructions:

(i) Click Start button ->All Programs -> Python 3.6.x -> IDLE. [Fig. 5.2(a)]

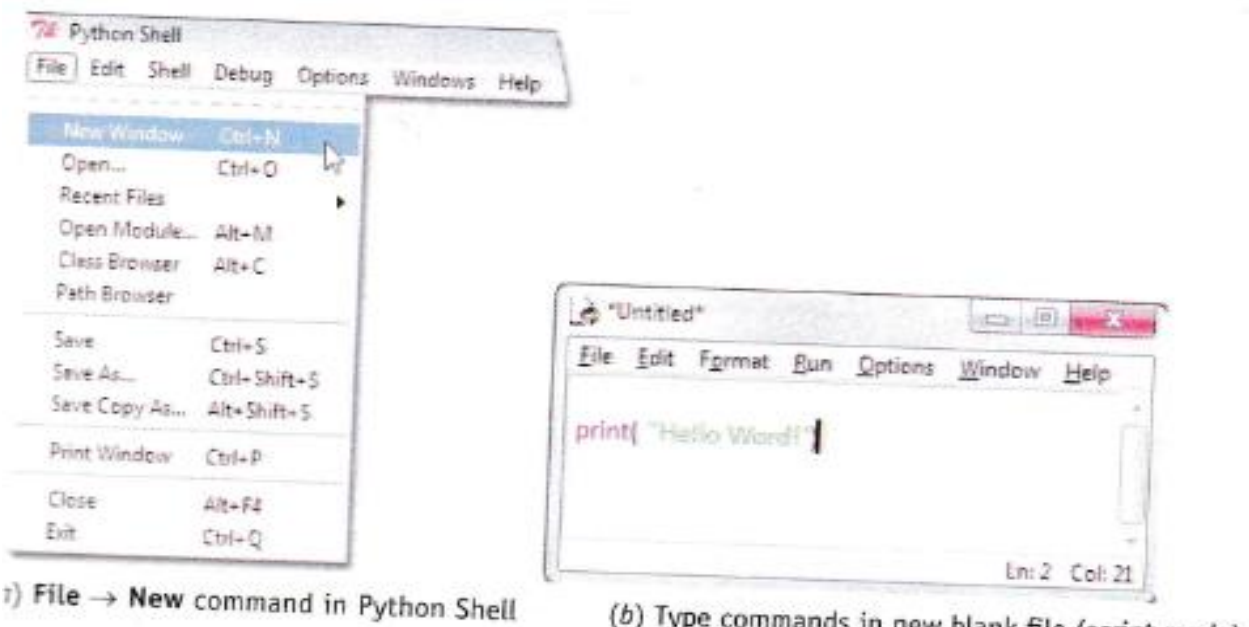
(ii) Click File -> **New** in IDLE Python Shell. [Fig. 5.2(a)]

(iii) In the New window that opens, type the commands you want to save in the form of a program (or script). [Fig. 5.2(b)]

For instance, for the simple Hello World program, you'll need to type following line:

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

You can display as well as print values in interactive mode, but for script mode, print( ) command is preferably used to print results.



(a) File → New command in Python Shell (b) Type commands in new blank file (script mode)

Figure 5.2

(iv)Click **File-> Save** and then save the file with an extension .py. The Python program has .py extension [Fig. 5.2(c)]. For instance, we gave the name to our program as Hello.py

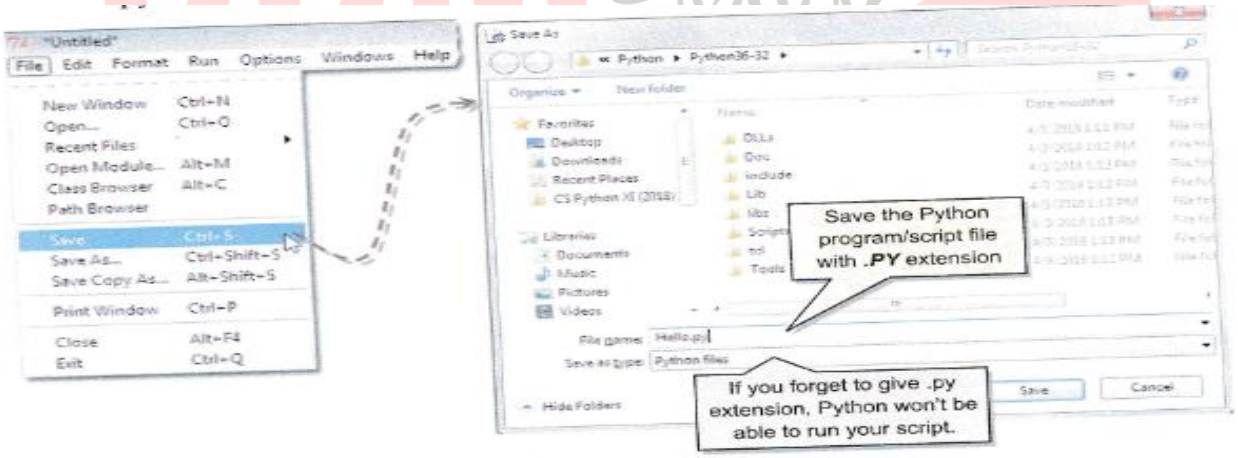


Figure 5.2 (c) Save file with .py extension with File → Save command (Script mode).

Now your program would be saved on the disk and the saved file will have .py extension.

Step 2: Run Module / Script / Program File:

After the program/script file is created, you can run it by following the given instructions:

(i) Open the desired program/script file that you created in previous Step 1 by using IDLE's File -> Open command.

If the program / script file is already open, you can directly move to next instruction

(ii) Click Run->Run Module command [Fig. 5.3(a)] in the open program / script file's window.

You may also press F5 key.

(iii) And it will execute all the commands stored in module I program I script that you had opened and show you the complete output in a separate Python Shell window. [Fig. 5.3(b)]

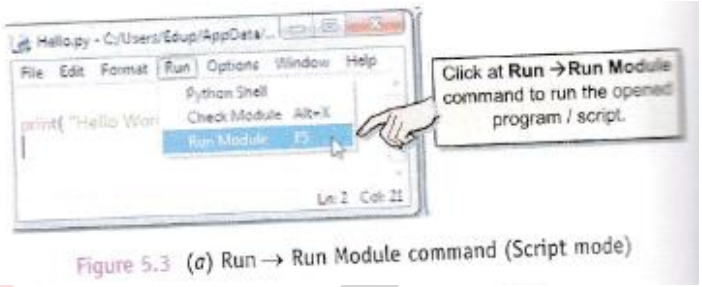


Figure 5.3 (a) Run → Run Module command (Script mode)

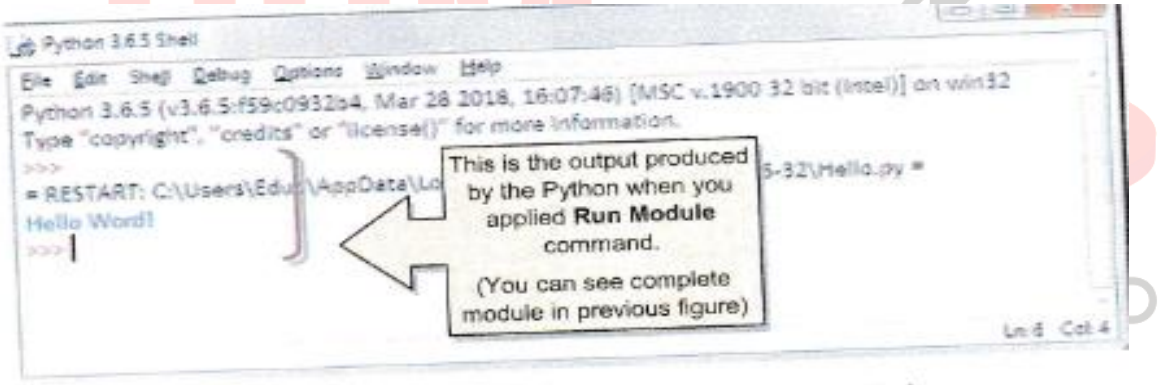


Figure 5.3 (b) Output of a module-run is shown in the shell window.

Period-03

Introduction:

Familiarization with the basics of Python Programming:

1. From the following, find out which assignment statement will produce an error. State reason(s) too.
   (a) $x = 55$ (b) $y = 037$ (c) $z = 0o98$ (d) 56thnumber = 3300
   (e) length = 450.17 (f) !Taylor = 'Instant' (g) this variable = 87.E02
   (h) float = .17E – 03 (i) FLOAT = 0.17E – 03

2. Find out the error(s) in following code fragments :

   (i)
   ```
   temperature = 90
   print temprature
   ```

   (ii)
   ```
   a = 30
   b = a + b
   print (a And b)
   ```

   (iii)
   ```
   a, b, c = 2, 8, 9
   print (a, b, c)
   c, b, a = a, b, c
   print (a ; b ; c)
   ```

   (iv)
   ```
   X = 24
   4 = X
   ```

   (v)
   ```
   print ("X =" X)
   ```

   (vi)
   ```
   else = 21 - 5
   ```

3. What will be the output produced by following code fragment (s) ?

   (i)
   ```
   X = 10
   X = X + 10
   X = X - 5
   print (X)
   X, Y = X - 2, 22
   print (X, Y)
   ```

   (ii)
   ```
   first = 2
   second = 3
   third = first * second
   print (first, second, third)
   first = first + second + third
   third = second * first
   print (first, second, third)
   ```

   (iii)
   ```
   side = int(input('side') )    #side given as 7
   area = side * side
   print (side, area)
   ```

4. What is the problem with the following code fragments ?

   (i)
   ```
   a = 3
   print (a)
   b = 4
   print (b)
   s = a + b
   print (s)
   ```

   (ii)
   ```
   name = "Prejith"
   age = 26
   print ("Your name & age are ", name + age)
   ```

   (iii)
   ```
   a = 3
   s = a + 10
   a = "New"
   q = a / 10
   ```

5. Predict the output :

   (a)
   ```
   x = 40
   y = x + 1
   x = 20, y + x
   print (x, y)
   ```

   (b)
   ```
   x, y = 20, 60
   y, x , y = x, y - 10, x + 10
   print (x, y)
   ```

   (c)
   ```
   a, b = 12, 13
   c, b = a*2, a/2
   print (a, b, c)
   ```

   (d)
   ```
   a, b = 12, 13
   print (print(a + b))
   ```

6. Predict the output
   ```
   a, b, c = 10, 20, 30
   p, q, r = c - 5, a + 3, b - 4
   print ('a, b , c :', a, b, c, end = ' ')
   print ('p, q, r :', p, q, r)
   ```

7. Find the errors in following code fragment

   (a)
   ```
   y = x + 5
   print (x, Y)
   ```

   (b)
   ```
   print (x = y = 5)
   ```

   (c)
   ```
   a = input("value")
   b = a/2
   print (a, b)
   ```

****