1 Getting Started with Python

In This Chapter

1.1 INTRODUCTION

The word *Python* – isn't it scary ? Does it bring the image of big reptile that we prefer to see either in jungles or zoo ? Well, it's time to change the image. Now on, you'll remember word *Python* for its playfulness and pleasant productivity. Confused ? Well, Don't be – because, now on you'll get introduced to a new programming language namely 'Python', which promises to make you a big programming fan :-).

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.

In this chapter, we shall introduce you to playful world of *Piquant Python* [Word 'Piquant' means pleasantly stimulating or exciting to the mind]. So, are we ready ? And... here we go.

NOTE

Do you know Python, the programming language, was named after famous *BBC* comedy show namely *Monty Python's Flying Circus.*

1.2 PYTHON – PLUSES

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 many strengths that make it a good choice for many situations. Let's see what are these *pluses of Python*.

Pluses of Python

- Easy to Use OO Language
- Expressive Language
- Interpreted Language
- Its Completeness
- Cross-platform Language
- Free and Open Source
- Variety of Usage / Applications

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
```

which one is compact and easier to understand ? Need I say more ? :). This is the simplest example, you'll find many more such examples of Python's expressiveness in the due course.

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 users.

4. Its Completeness

When you install Python, you get everything you need to do real work. You do not need to download and install additional libraries ; all types of required functionality is available through various modules of Python standard library[1]. 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 true cross-platform language. Or in other words, Python is a **portable language**.

1. If you install Python through *Anaconda Python Distribution* it loads most libraries and packages with Python.
2. Python even has versions that run on different languages such as Java (Jython), .NET (IronPython) etc.

6. Free and Open Source

Python language is freely available *i.e.*, without any cost (from www.python.org). And not only is 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 an 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

1.3 PYTHON – SOME MINUSES (SO HUMAN LIKE)

Minuses of Python

- Not the Fastest Language
- Lesser Libraries than C, Java, Perl
- Not Strong on Type-binding
- Not Easily Convertible

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 not competent with languages like C, Java, Perl as they have larger collections available. Sometimes 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.

As per February 2013 popularity index, Python was 4th most popular programming language[3] after – *Java, PHP* and *C#*. That is the reason, it's part of your syllabus. Together we'll make it playful Python ;).

Since Python is an interpreted language and not a compiled language, it would be a good idea to know the difference between working of an interpreter and a compiler (as explained in following information box).

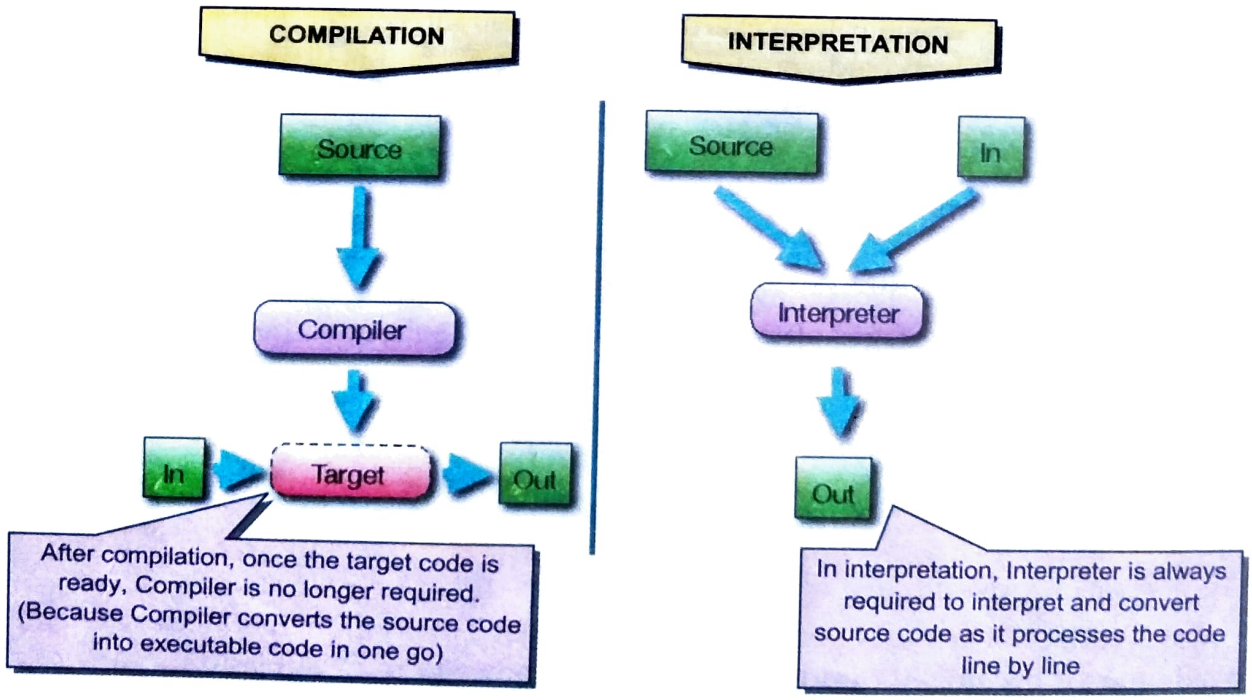
Interpretation vs Compilation

Interpreter

This language processor converts a HLL(High Level Language) program into machine language by *converting and executing it line by line*. If there is any error in any line, it reports it at the same time and program execution cannot resume until the error is rectified. Interpreter must always be present in the memory every time the program is executed as every time the program is run, it is first interpreted and then executed. For error debugging, interpreter is very much useful as it reports the error(s) at the same time. But once errors are removed, unnecessary usage of memory takes place as it has to be present in the memory always.

Compiler

It also converts the HLL program into machine language but the conversion manner is different. It converts the *entire HLL program in one go*, and reports all the errors of the program along with the line numbers. After all the errors are removed, the program is recompiled, and after that the compiler is not needed in the memory as the object program is available.



Python is an interpreted language, that is, all the commands you write are interpreted and executed one by one.

3. Source : https://sites.google.com/site/pydatalog/pypl/PyPL-PopularitY-of-Programming-Language.

1.4 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, PytCharm IDE etc. Of these, Spyder IDE is already available as a part of Anaconda Python distribution.

To install any of these distributions, **PLEASE REFER TO APPENDIX A.** We shall learn to work with both these distribution types [but my personal favourite is *Anaconda* ;) – not the reptile, the 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

1.4.1 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.**

1.4.1A 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. 1.1(*a*)]

Or

Click **Start button → all Programs → Python 3.6.x → Python** (command line)

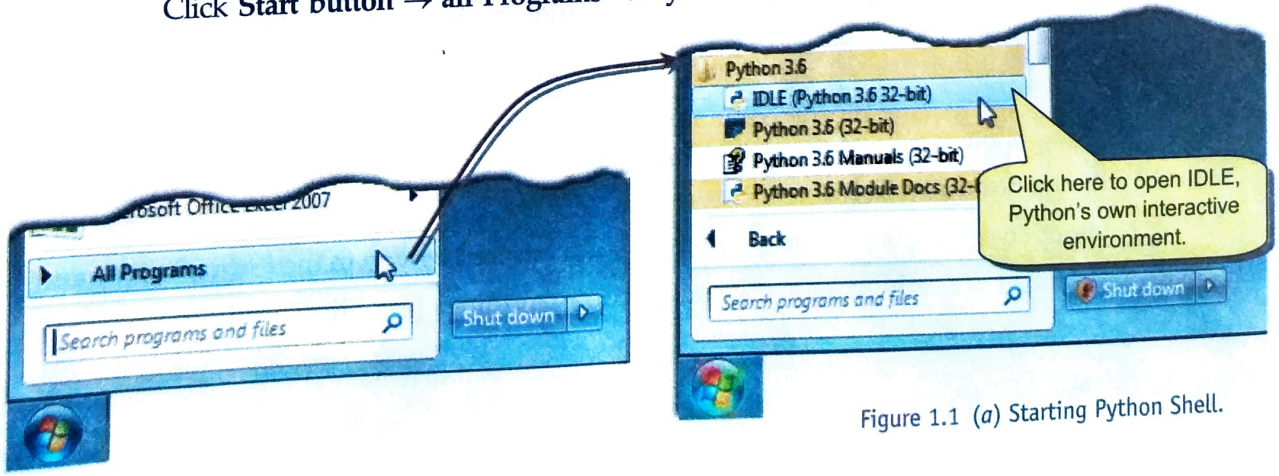


Figure 1.1 (*a*) Starting Python Shell.

(*ii*) It will open Python Shell [see Fig. 1.1(*b*)] where you'll see the Python prompt (three '>' signs *i.e.*, >>>).

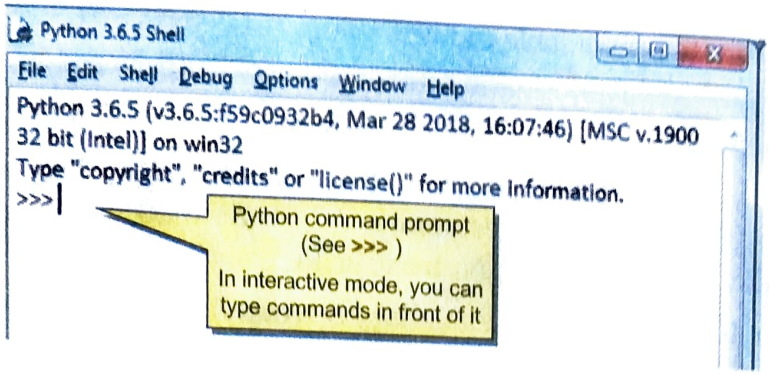


Figure 1.1 (*b*) Python's interactive interpreter – Python Shell.

NOTE

The interactive interpreter of Python is also called **Python Shell.**

(*iii*) Type commands in front of this Python prompt and Python will immediately give you the result. [*see* Fig. 1.1(*c*)]

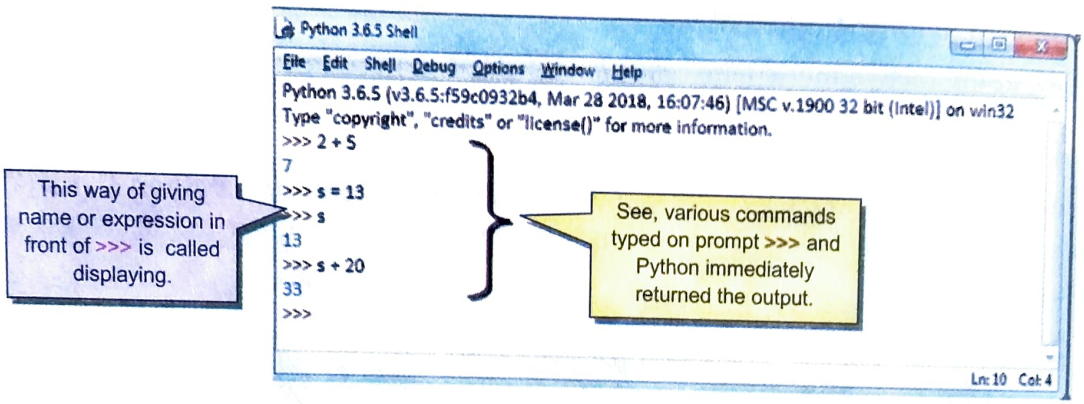


Figure 1.1 (*c*) Interactive commands and their output in Python Shell.

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. 1.1(*c*)] in front of the prompt.

Figure 1.1(*c*) shows you some sample commands that we typed in Python shell and the output returned by Python interpreter.

NOTE

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

1.4.1B 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[4].
- The output is sandwiched between the command lines [see Fig. 1.1(*c*)].

4. Python GUI Shell IDLE lets you save the entire session (commands followed by their results – as it appears on screen) but that is not the Python program/script containing only the instructions.

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. 1.2(*a*)]

(*ii*) Click **File → New** in IDLE Python Shell. [Fig. 1.2(*a*)]

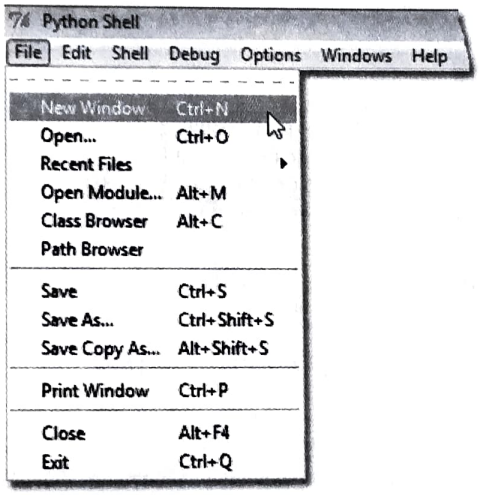
(*iii*) In the New window that opens, type the commands you want to save in the form of a program (or script). [Fig. 1.2(*b*)]

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

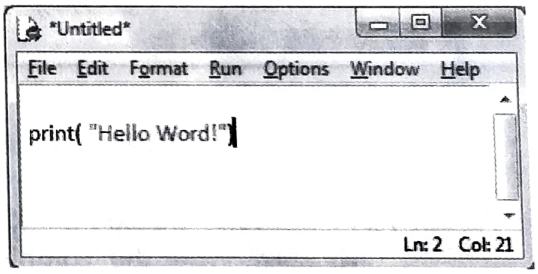
```
print ("Hello World !")
```

NOTE

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)

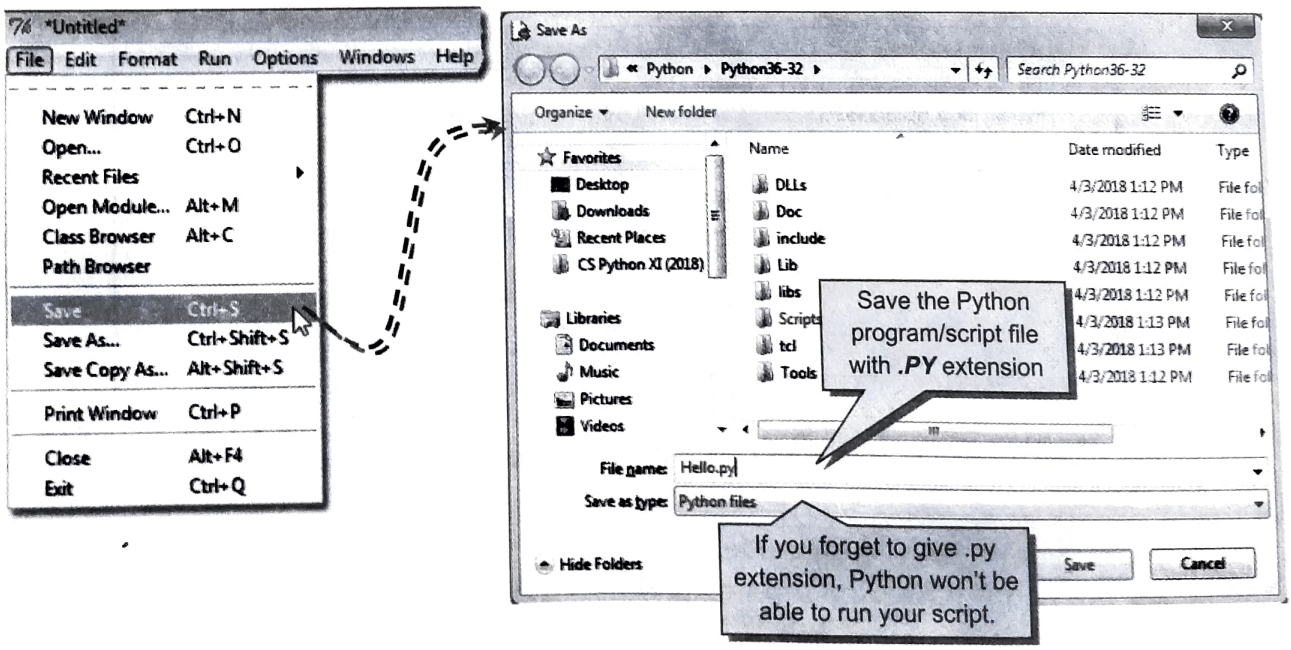


Figure 1.2 (*c*) Save file with .py extension with File → Save command (Script mode).

(*iv*) Click **File** → **Save** and then save the file with an extension **.py**. The Python programs has **.py** extension [Fig. 1.2(*c*)]. For instance, we gave the name to our program as **Hello.py**.

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. 1.3(*a*)] in the open program / script file's window.

You may also press **F5 key**.

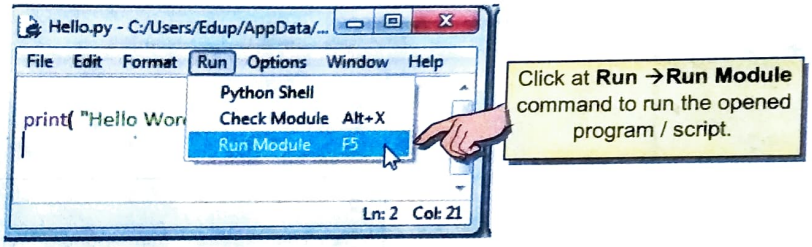


Figure 1.3 (*a*) Run → Run Module command (Script mode)

(*iii*) And it will execute all the commands stored in module / program / script that you had opened and show you the complete output in a separate Python Shell window. [Fig. 1.3(*b*)]

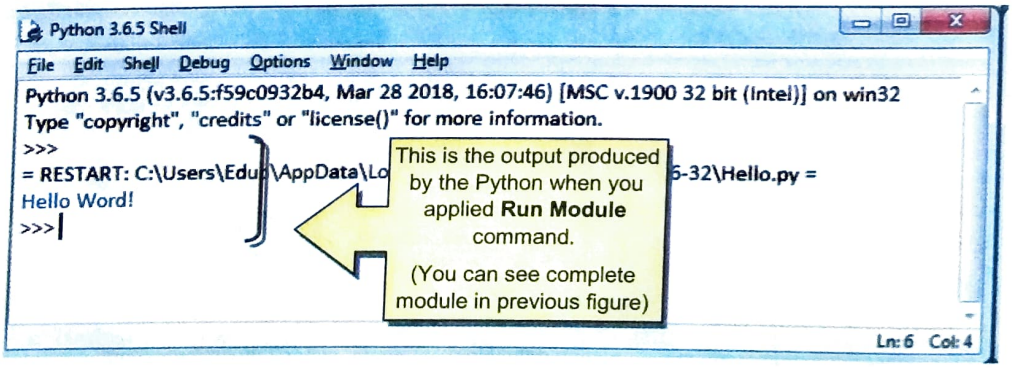


Figure 1.3 (*b*) Output of a module-run is shown in the shell window.

As you can see that with script mode, you can store all commands together in the form of a module / program / script and can get all output lines together. (No more command-output sandwiching :).

1.4.2 Working in Anaconda Distribution

Anaconda distribution comes with many preloaded packages and libraries. You can work in it in both interactive and script modes. Anaconda distribution provides the following tools that you can use to work in Python.

- **Jupyter notebook.** It is a web based, interactive computing environment.
- **Spyder.** It is a powerful Python IDE with many useful editing, interactive testing and debugging features.

Let us learn to work with both.

1.4.2A Working in Jupyter Notebook

In order to work in *jupyter notebook*, you need to first launch it using **Anaconda Navigator**[5] as it has come preloaded with Anaconda distribution.

1. Launch Anaconda Navigator

2. From the Navigator window, click on Launch below **jupyter notebook** tile.

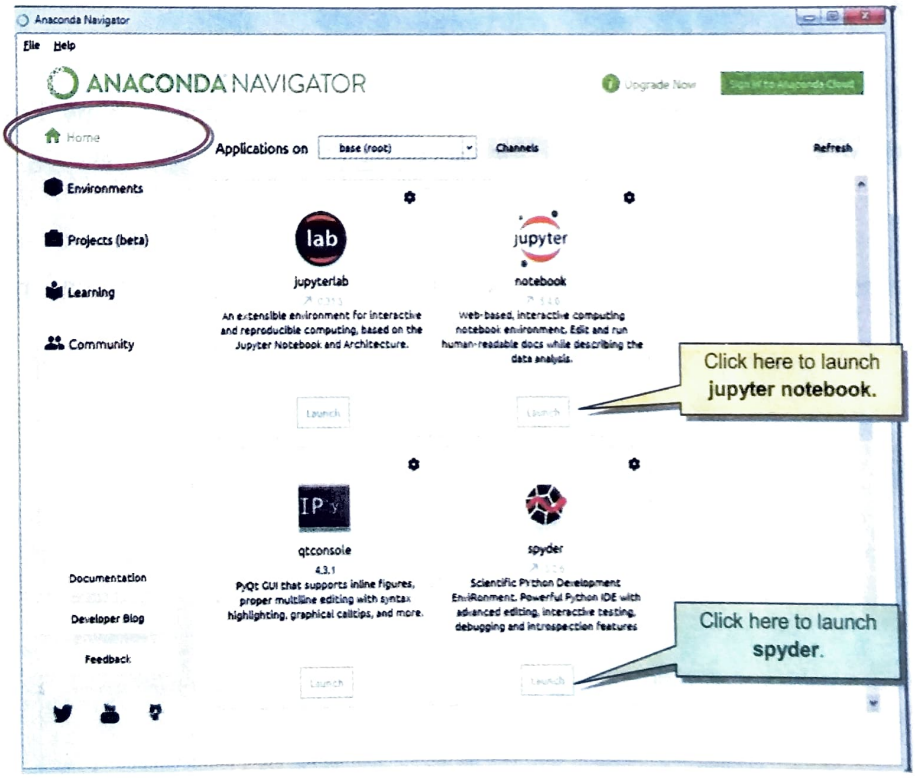


Figure 1.4 Launching applications from Anaconda Navigator.

5. Please note that jupyter notebook can also be installed separately, without Anaconda also.

Understanding print( )

To print or display output, Python 3.x provides **print( )** function[10]. You can use **print( )** as

```
print(<objects to be printed>...)
```

e.g., when you wrote

print( ) statement 1

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

String "Hello World!" is the object to be printed

it printed :

```
Hello World!
```

Similarly, to print other strings you may give something like.

print( ) statement 2

```
print('My name is Misha")
```

String object 'My name is Misha' to be printed.

it will print

```
My name is Misha
```

Carefully look at both the **print( )** statements given above. Could you notice anything other than two different string values ? No ?

Carefully notice that the first string is enclosed in **double quotes** and the second string is enclosed in **single quotes**.

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

String enclosed in double quotes

```
print('My name is Misha')
```

String enclosed in single quotes

Both these strings are valid in Python. You can enclose your strings in either double quotes or in single quotes, but just ensure that opening and closing quotation mark should be of same type. You cannot have a string like

```
'Hey there"
```

Error! opening and closing Quotation marks do not match

Following are some valid **print( )** statements.

```
print('The Golden Ratio')
print("has same letters as")
print('The God Relation')
```

With this we have come to the end of this chapter. Let us quickly revise what we have learnt so far.

10. In Python 2.x, print is a statement, not a function.

PRIP PYTHON INTERACTIVE MODE — Progress In Python 1.1

Start Python (Anaconda Navigator and then Spyder IDE or any other IDE of your choice)

On the IPython console perform this session

Python prompt as calculator

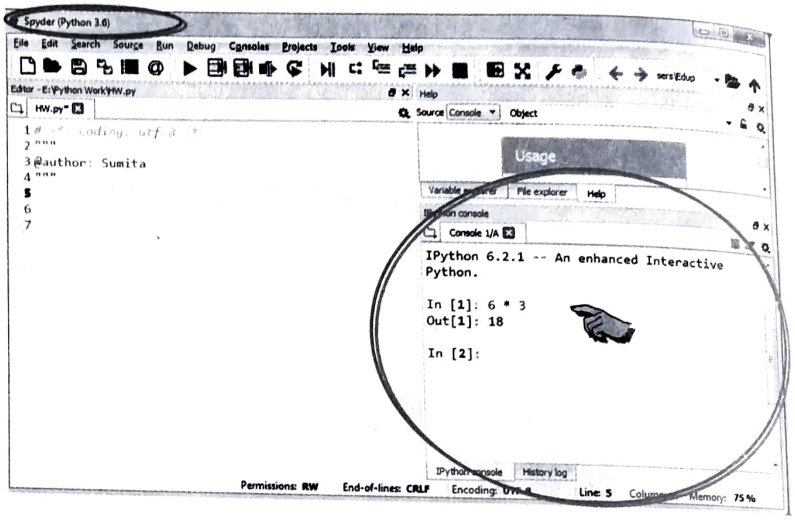
1. *In front of Python prompt, i.e.,* **In[ ]:**, *type following expressions one by one, by pressing return key after every expression, e.g.,*

```
In[1] : 6 * 3
```

It will show output lines starting with **Out[ ]:**, *e.g.* :

```
Out[2] : 18
```

⋮



Please check the practical component-book – Progress in Computer Science with Python and fill it there in PriP 1.1 under Chapter 1 after practically doing it on the computer.

>>>❖<<<

PRIP WORKING IN SCRIPT MODE — Progress In Python 1.2

Start ***Anaconda Navigator*** *and then* ***Spyder IDE*** *or any other* ***IDE*** *of your choice.*

Open new File window

Click **File → New File** in **Spyder.**

1. *Type following code (as it is) in the new window opened by IDLE*

```
print ("Hello Nathan", "Hello Lalit")
greet1 = "Hello Nathan"
greet2 = "Hello Lalit"
name1 = " Nathan "
name2 = " Lalit "
print (greet1, greet2)
print ("Hello ", name1, ",", name2)
```

⋮

Please check the practical component-book – Progress in Computer Science with Python and fill it there in PriP 1.2 under Chapter 1 after practically doing it on the computer.

>>>❖<<<

LET US REVISE

- *Python was developed by Guido Van Rossum in February 1991.*
- *Python offers following advantages*
  - easy to use
  - expressive
  - complete
  - cross-platform
  - free and open source
- *Python also has these limitations :*
  - Not the fastest language
  - Lesser libraries than C, Java, Perl
  - Not strong on Type-binding.
- *In Python, one can work in two different ways : (i) Interactive mode, (ii) Script mode.*
- *Interactive mode does not save commands in form of a program and also, output is sandwiched between commands.*
- *Interactive mode is suitable for testing code.*
- *Script mode is useful for creating programs and then run the programs later and get complete output*
- *Python is an interpreted language.*
- *Python's interactive interpreter is also called Python Shell.*

Solved Problems

1. *Who developed Python Programming Language* ?

   Solution. Guido Van Rossum in 1990s developed Python programming language.

2. *Is Python an Object Oriented language* ?

   Solution. Yes, Python is an Object Oriented language.

3. *'Python is an interpreted high level language'. What does it mean to you* ?

   Solution. **'Python is a high level language'** means it is programmer-friendly *i.e.*, easy to program and comprehend.

   'Python is an interpreted language' means it requires an interpreter (not compiler) to execute its code line by line – one statement at a time.

4. *Python programming language got its name from which show.*

   Solution. Python programming language was named after a British TV show namely 'Monty Python's Flying Circus'.

5. *What does a cross platform language mean* ?

   Solution. A cross platform language means it can run well on variety of platforms like Windows, Linux/Unix, Macintosh etc. etc.

6. *Python is a Free and Open Source language. What do you understand by this feature* ?

   Solution. It means – to download Python, one needs not pay anything, because it is **Free**. And its source-code is also available, which can be modified/improved etc., because it is **open-source**.

7. *What is the difference between interactive mode and script mode in Python ?*

Solution. In *interactive mode*, instructions are given in front of Python prompt (*e.g.*, **>>>** or **In[ ]:** prompts) in Python Shell. Python carries out the given instruction and shows the result there itself.

In *script mode*, Python instructions are stored in a file generally with .py extension and are executed together in one go as a unit. The saved instructions are known as *Python script* or *Python program.*

8. *What will be the output of following code :*

```
#This is a sample program
#to output simple statements
#print ("Such as")
print("Take every chance.")
print("Drop every fear.")
```

Pick the correct output from the following choices and give reason.

(*a*) *This is a sample program to output simple statements such as*

```
Take every chance.
Drop every fear.
```

(*b*) *Such as*

```
Take every chance.
Drop every fear.
```

(*c*)

```
Take every chance.
Drop every fear.
```

Solution. The correct output is (*c*).

Reason being : the code lines beginning with a # sign are comments. They are just for information and ignored by the Python interpreter. Hence, the third line **#print ("Such as")** will also be ignored by Python interpreter. Thus, the Python interpreter will give output of only **print( )** statements.

9. *Which of the following are not valid strings in Python ?*

(*a*) "Hello" (*b*) 'Hello' (*c*) "Hello' (*d*) 'Hello" (*e*) {Hello}

Solution. Strings (*c*), (*d*) and (*e*) are not valid strings in Python.

GLOSSARY

Cross-platform	Something that is compatible across various platforms.
Open -Source	Computer software for which source-code is freely available.
Python Shell	Interactive interpreter of Python.
Source Code	Complete program instructions.