

WRITING OUR FIRST PYTHON PROGRAM

CHAPTER

2

In this chapter, we will learn to write a simple Python program and execute it using Python software. For this purpose, it is necessary to first install the Python software in our computer system. So, we will first learn how to install Python on Windows Operating system and then we will see how to execute a Python program using Python compiler and PVM.

Installing Python for Windows

The latest version of Python (at the beginning of 2018) is Python 3.6.4. This version has again got two variations, a 32-bit version and a 64-bit version. Depending upon our operating system, we can choose a version. Nowadays most people use 64-bit operating system and hence we can use the 64-bit version of Python by visiting the following link:

<https://www.python.org/downloads/release/python-364/>

At the bottom of this page, click the 'Windows x86-64 executable installer' link. The file by the name '*python-3.6.4-amd64.exe*' will be downloaded in our computer. By double clicking and following the instructions, we can easily install the Python software latest version in our system.

Let's perform the following steps to install Python:

1. Double click the '*python-3.6.4-amd64.exe*' file. The Setup dialog box appears (Figure 2.1).
2. Select the Install launcher for all users and Add Python 3.6 to PATH checkboxes.

3. Click the 'Install Now' link, as shown in Figure 2.1:

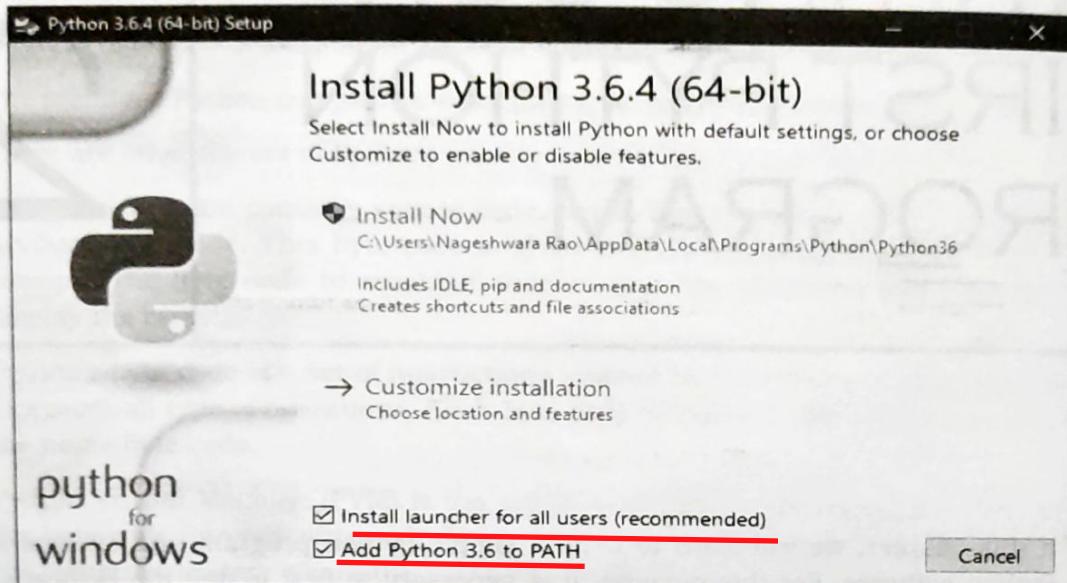


Figure 2.1: Running the Python Setup file

The Setup Progress bar will appear as shown in Figure 2.2:

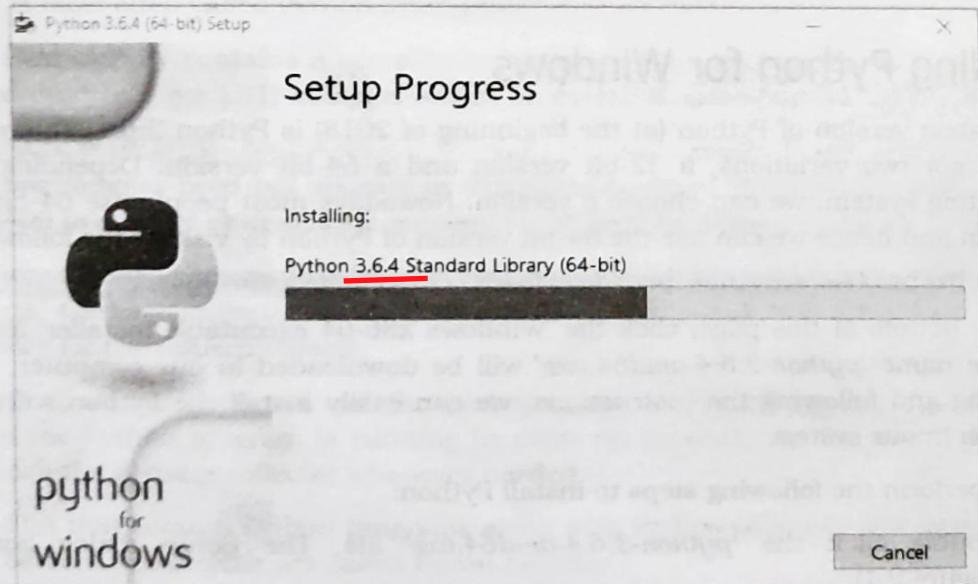


Figure 2.2: Python installation progress

When Python installation is complete, we can see 'Setup was successful' message, as shown in Figure 2.3.

4. Click the 'Close' button, as shown in Figure 2.3:

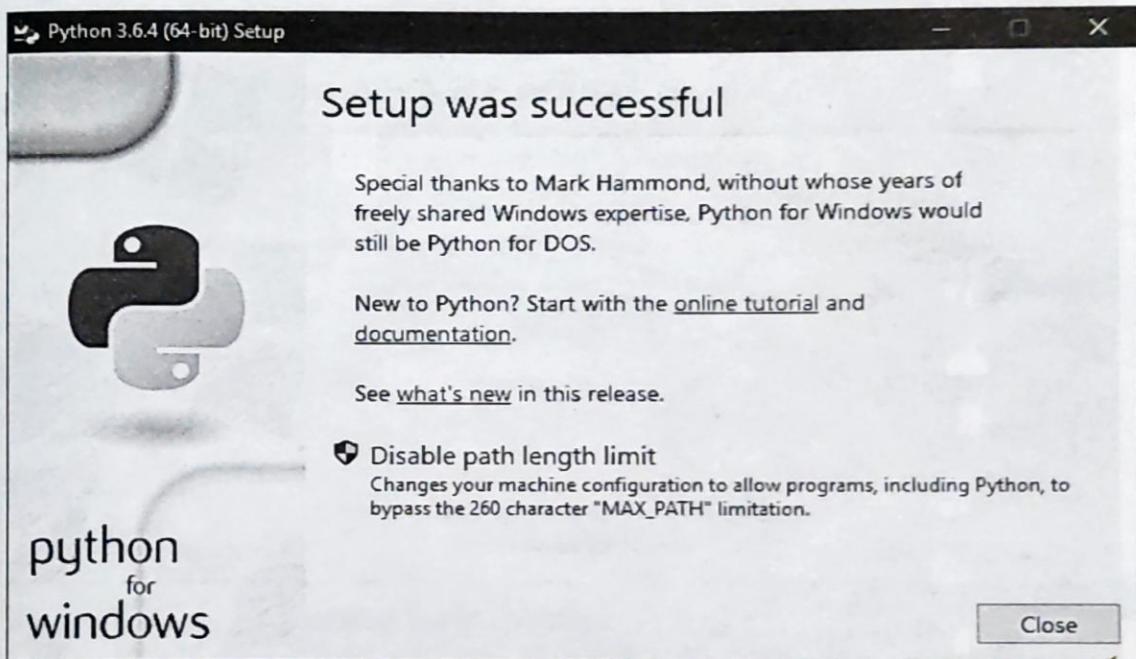


Figure 2.3: Python installation complete window

Testing the Installation in Windows 10

1. Click the 'Start' button on the task bar of the Windows 10 operation system. It displays all applications available in your system in alphabetical order. Go to 'P' and view the 'Python 3.6' folder. In this folder, you can see the following icons:
- IDLE (Python 3.6 64-bit)
 - Python 3.6 (64-bit)
 - Python 3.6 anuals (64-bit)
 - Python 3.6 Module Docs (64-bit)

This is shown in Figure 2.4:

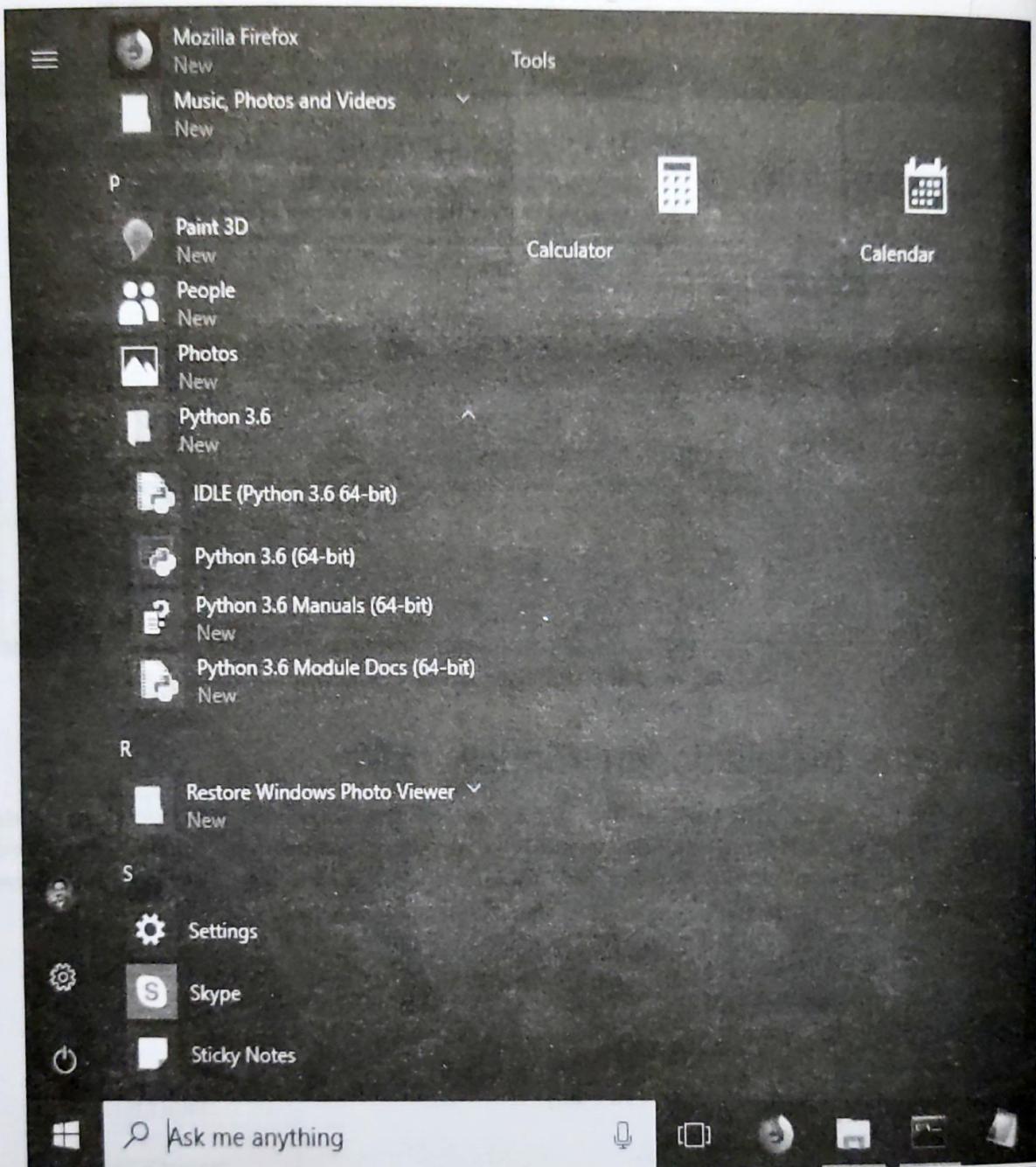


Figure 2.4: Windows start button displaying Python installation

2. Click the 'IDLE (Python 3.6 64-bit)' option. The Python's IDLE (Integrated Development Environment)'s Graphical user interface window opens (Figure 2.5).

At the bottom of the screen, we can see a white icon displayed on the taskbar.

3. Right click on the icon and click the 'Pin to taskbar' option. Clicking on this taskbar icon will be sufficient to open Python IDLE whenever we want, as shown in Figure 2.5:

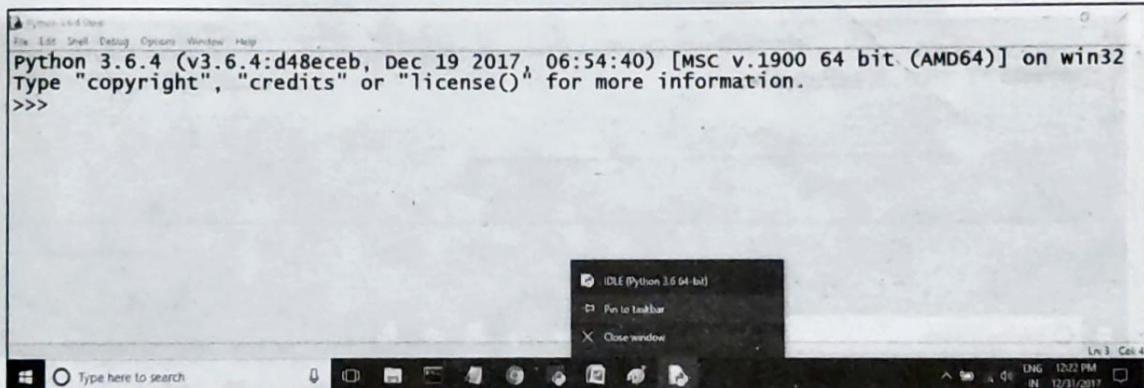


Figure 2.5: The Python IDLE Window

4. Click the Python prompt, i.e., triple greater than symbol and type quit() to close the Python window (Figure 2.6). A prompt asking 'Do you want to kill it?' appears.
5. Click the 'OK' button to quit from the IDLE window, as shown in Figure 2.6:

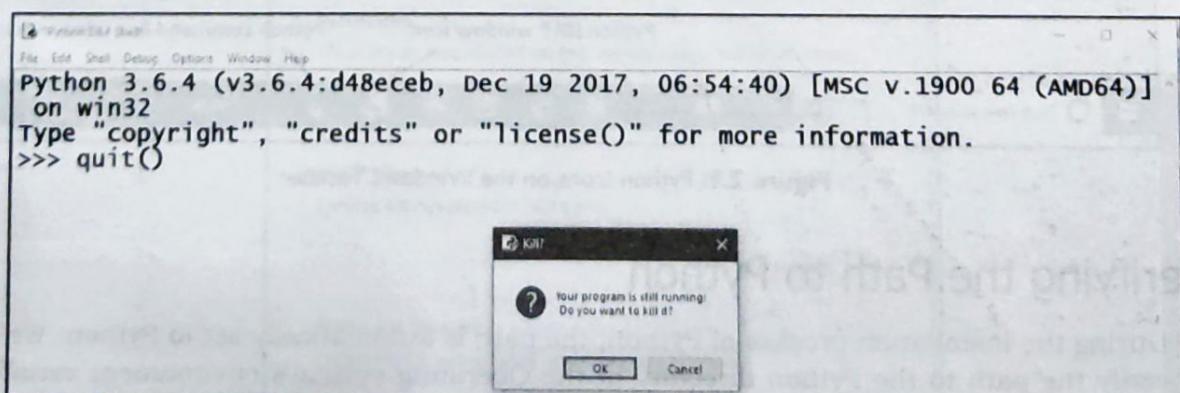
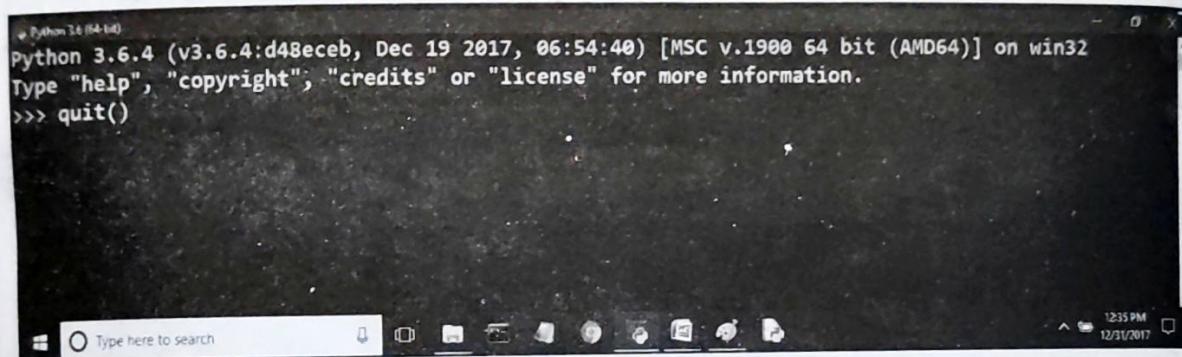


Figure 2.6: Closing the Python IDLE Window

6. Click the windows 'Start' button and then click 'Python 3.6 (64 - bit)' to open the 'Python command line window' which displays a black screen (Figure 2.7).

The Python command line window is also useful in the same manner as the Python's IDLE window to type the Python code and run it. Right click the Python command line window icon displayed on the taskbar at the bottom of the monitor and pin it to the taskbar so that we can simply click on this icon to open Python command line window at any time.

To quit from this command line window, we can simply type `quit()` at the Python prompt (the triple greater than symbol), as shown in Figure 2.7:



A screenshot of a Windows command line window titled "Python 3.6.4 (v3.6.4:d48ebeb, Dec 19 2017, 06:54:40) [MSC v.1900 64 bit (AMD64)] on win32". The window contains the text "Type 'help', 'copyright', 'credits' or 'license' for more information." followed by the command ">>> quit()". The window has a standard Windows title bar and a taskbar at the bottom with various icons.

Figure 2.7: Python command line window

- As discussed in the previous steps, we would have two icons added to our windows taskbar. They are: the Python IDLE window icon (white in color) and the Python command line window icon (black in color). We can click any of these icons to open these windows, type Python programs and run them in these windows. Figure 2.8 shows the Python icons:

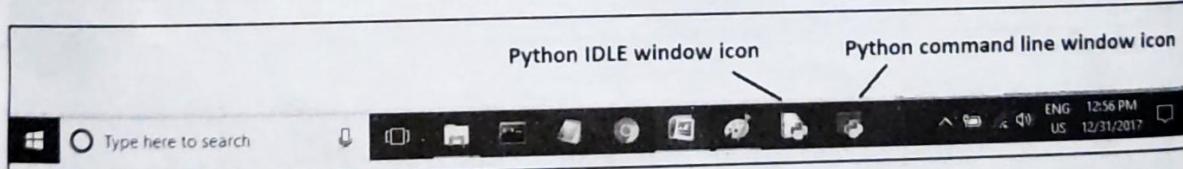


Figure 2.8: Python Icons on the Windows Taskbar

Verifying the Path to Python

During the installation process of Python, the path is automatically set to Python. We can verify the path to the Python directory in the Operating system's environment variables. Let's perform the following steps to view the path to the Python directory:

- Right click the 'This PC' icon in Windows 10. Then click the 'Properties' option as shown in Figure 2.9:

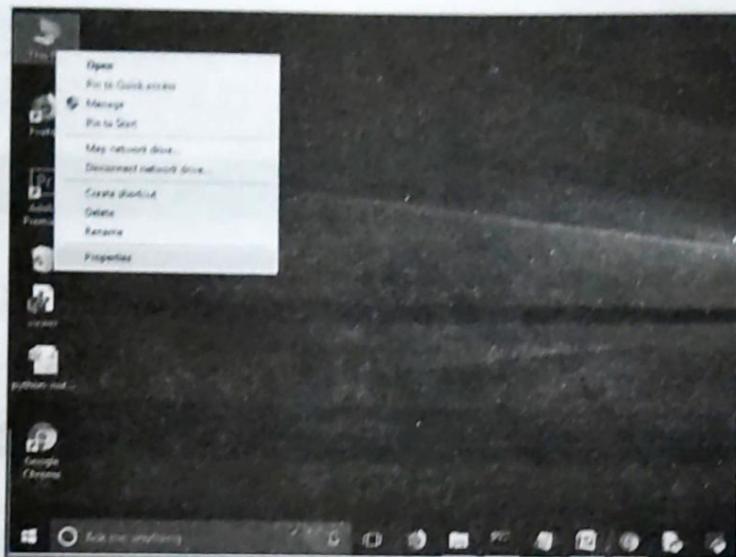


Figure 2.9: Going to properties of Computer system

It will display a page where we should click the 'Advanced system settings' option. It will display a System Properties dialog box.

2. Click the 'Environment Variables' button, as shown in Figure 2.10:

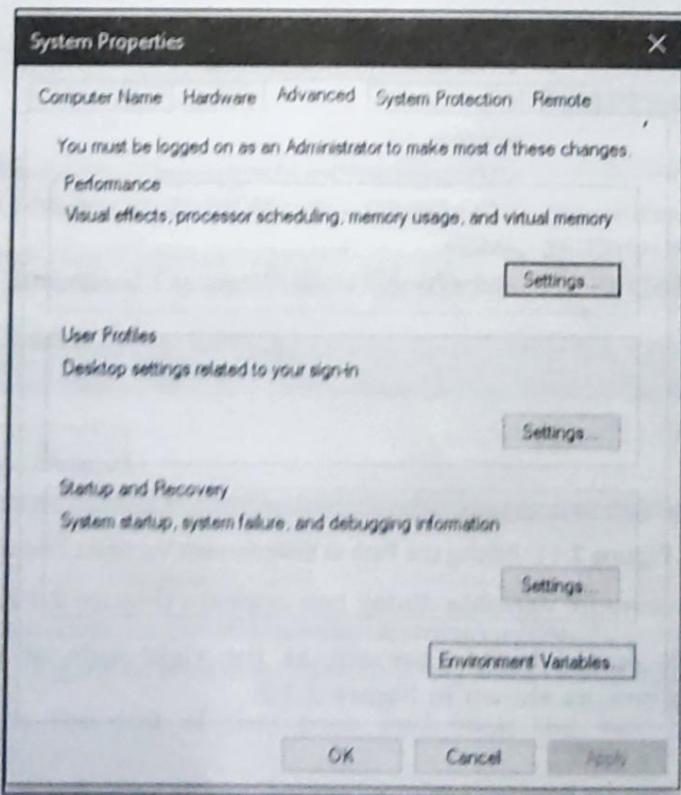


Figure 2.10: Going to Environment Variables in Advanced System Settings

It will display User variables and System variables (Figure 2.11).

3. Select 'Path' variable and click the 'Edit...' button in the User variables, as shown in Figure 2.11:

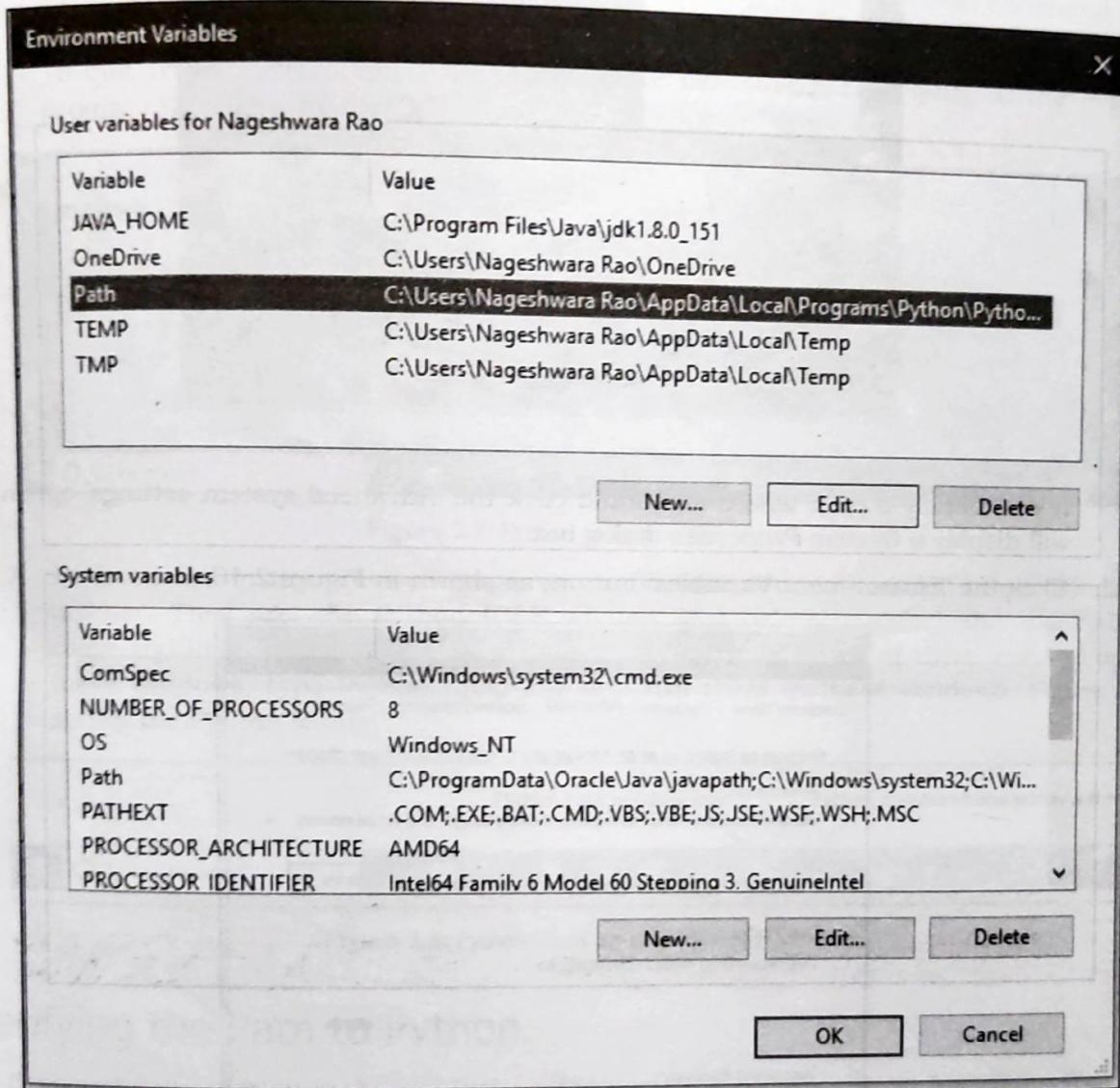


Figure 2.11: Editing the Path in Environment Variables Dialog Box

The Edit Environment Variable dialog box appears (Figure 2.12).

4. Click the 'Edit text...' button present at the right side of the Edit Environment Variable dialog box, as shown in Figure 2.12:

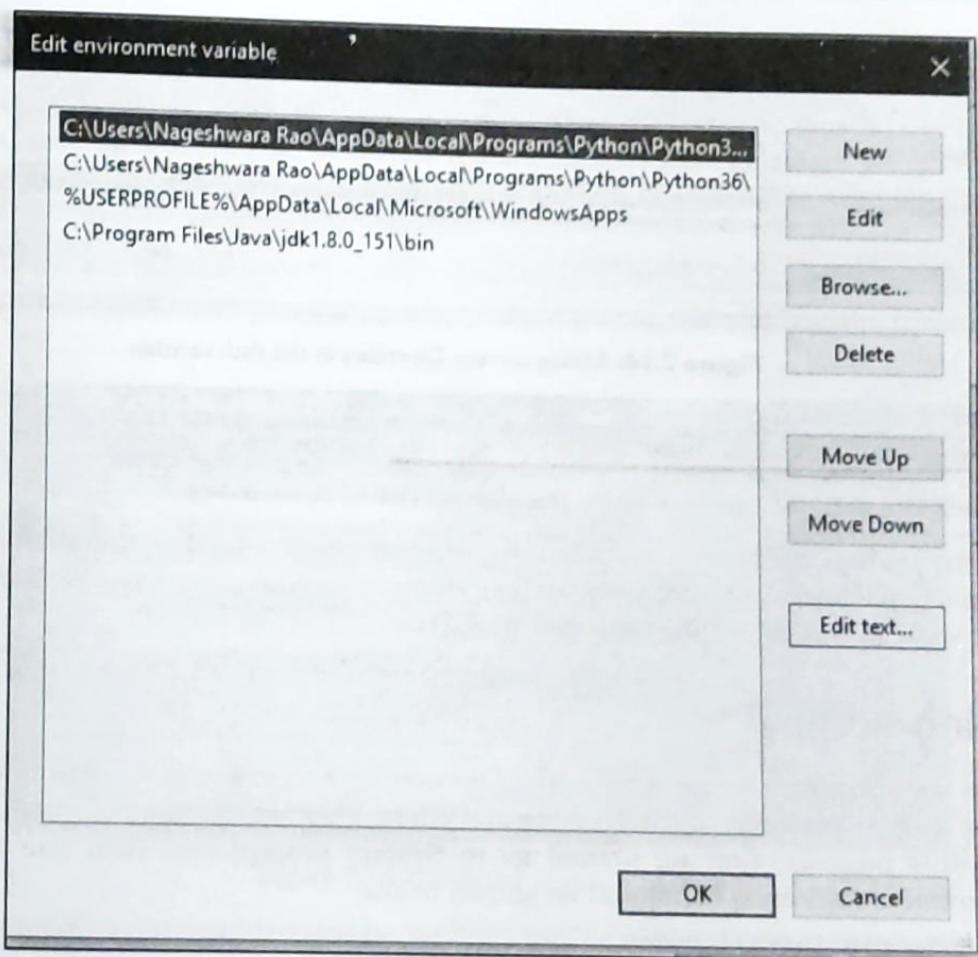


Figure 2.12: Viewing Python Directory in the Path variable

It will display the directories involved in the Path. We can observe that the directory by the name Python36\Scripts is added to the Path, as shown in Figure 2.13:

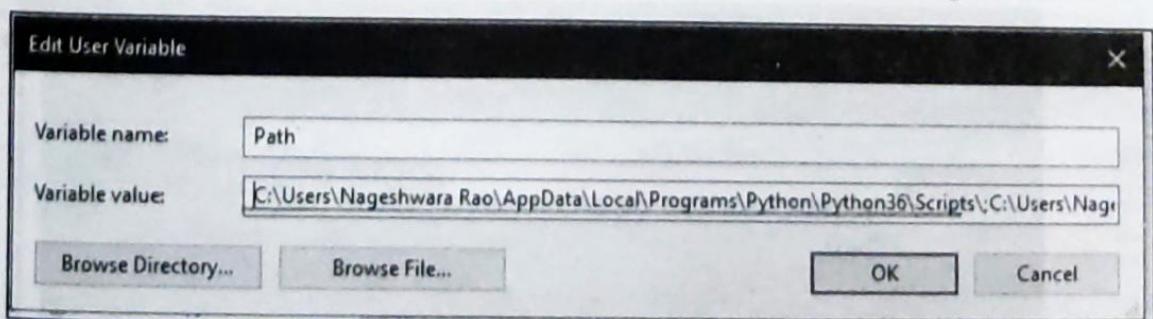


Figure 2.13: Viewing Python Directory in the Path variable

5. Add a dot (.) at the end of that path and click the 'OK' button, as shown in Figure 2.14;

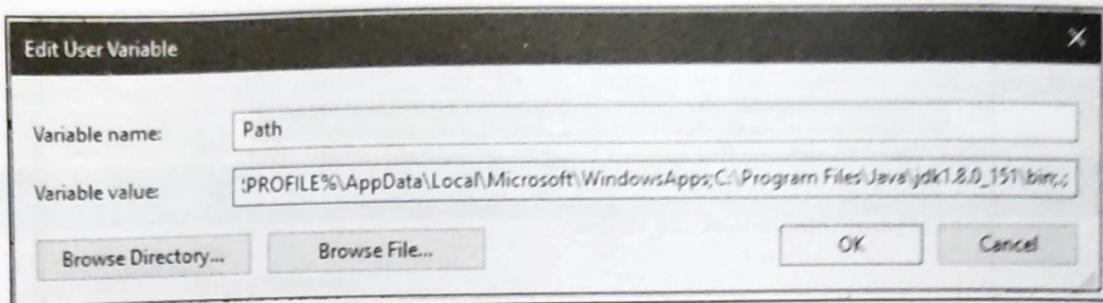


Figure 2.14: Adding current Directory in the Path variable

Dot represents the directory where Python is executed. By adding dot, we are making Python to run in any directory in our system.

- Click the 'OK' button two times to close all the opened boxes.

Once the Python software is installed, we have to proceed further to install other useful third party packages that are used along Python in various programs. We will see how to install some of the important packages now.

Installing numpy

Python supports only single dimensional arrays. To create multi-dimensional arrays, we need a special package called Numerical Python package or *numpy*. To download and install this package, first we should go to System prompt and then use 'pip' (Python Installation of Packages) command as shown below:

C:\> pip install numpy

python installer package

'pip' program comes with Python software by default. When this command is given, it searches the latest version of the numpy package on the Internet, downloads it and installs it, as shown in Figure 2.15:

```
C:\>pip install numpy
Collecting numpy
  Downloading numpy-1.13.3-cp36-none-win_amd64.whl (13.1MB)
    100% |████████████████████████████████| 13.1MB 104kB/s
Installing collected packages: numpy
Successfully installed numpy-1.13.3

C:\>
```

Figure 2.15: Installation of numpy package

Of course, our computer would have connected to the Internet while using this command.

Installing pandas

pandas is a package used in data analysis. This package is mostly used by data scientists and data analysts. To download and install this package, we should go to System prompt and then use 'pip' (Python Installation of Packages) command as shown below:

```
C:\> pip install pandas
```

This command downloads *pandas* package from the Internet and installs it, as shown in Figure 2.16:

```
C:\>pip install pandas
Collecting pandas
  Downloading pandas-0.21.1-cp36-cp36m-win_amd64.whl (9.1MB)
    100% |████████████████████████████████| 9.1MB 133kB/s
Collecting python-dateutil>=2 (from pandas)
  Downloading python_dateutil-2.6.1-py2.py3-none-any.whl (194kB)
    100% |████████████████████████████████| 194kB 1.3MB/s
Collecting pytz>=2011k (from pandas)
  Downloading pytz-2017.3-py2.py3-none-any.whl (511kB)
    100% |████████████████████████████████| 511kB 595kB/s
Requirement already satisfied: numpy>=1.9.0 in c:\users\nageshwarrao\appdata\local\programs\python\python36\lib\site-packages (from pandas)
Collecting six>=1.5 (from python-dateutil>=2->pandas)
  Downloading six-1.11.0-py2.py3-none-any.whl
Installing collected packages: six, python-dateutil, pytz, pandas
Successfully installed pandas-0.21.1 python-dateutil-2.6.1 pytz-2017.3 six-1.11.0
C:\>
```

Figure 2.16: Installation of *pandas* package

Installing xlrd

xlrd is a package that is useful to retrieve data from Microsoft Excel spreadsheet files. Hence, it is useful in data analysis. To download and install this package, we should go to System prompt and then use 'pip' (Python Installation of Packages) command as shown below:

```
C:\> pip install xlrd
```

This command downloads *xlrd* package from the Internet and installs it, as shown in Figure 2.17:

```
C:\>pip install xlrd
Collecting xlrd
  Downloading xlrd-1.1.0-py2.py3-none-any.whl (108kB)
    100% |████████████████████████████████| 112kB 468kB/s
Installing collected packages: xlrd
Successfully installed xlrd-1.1.0
C:\>
```

Figure 2.17: Installation of *xlrd* package

Installing matplotlib

matplotlib is another important package in Python that is useful to produce good quality 2D graphics. It is mostly used for the purpose of showing data in the form of graphs and also for designing electronic circuits, machinery parts, etc. To download and install this package, we should go to System prompt and then use ‘pip’ (Python Installation of Packages) command as shown below:

```
C:\> pip install matplotlib
```

This command downloads the *matplotlib* package from the Internet and installs it as shown in Figure 2.18:

```
C:\>pip install matplotlib
Collecting matplotlib
  Downloading matplotlib-2.1.1-cp36-cp36m-win_amd64.whl (8.7MB)
    100% |████████████████████████████████| 8.7MB 152kB/s
Requirement already satisfied: numpy>=1.7.1 in c:\users\nageshwara rao\appdata\local\python\python36\lib\site-packages (from matplotlib)
Collecting cycler>=0.10 (from matplotlib)
  Downloading cycler-0.10.0-py2.py3-none-any.whl
Requirement already satisfied: six>=1.10 in c:\users\nageshwara rao\appdata\local\python\python36\lib\site-packages (from matplotlib)
Requirement already satisfied: python-dateutil>=2.0 in c:\users\nageshwara rao\appdata\local\python36\lib\site-packages (from matplotlib)
Requirement already satisfied: pytz in c:\users\nageshwara rao\appdata\local\python36\lib\site-packages (from matplotlib)
Collecting pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 (from matplotlib)
  Downloading pyparsing-2.2.0-py2.py3-none-any.whl (56kB)
    100% |████████████████████████████████| 61kB 2.0MB/s
Installing collected packages: cycler, pyparsing, matplotlib
Successfully installed cycler-0.10.0 matplotlib-2.1.1 pyparsing-2.2.0
C:\>
```

Figure 2.18: Installation of *matplotlib* package

Verifying Installed Packages

We can verify whether the installed packages are added to our Python software properly or not by going into Python and typing the following command at Python prompt (triple greater than symbol) as:

```
>>> help('modules')
```

For this purpose, first click the Python IDLE Window pinned at the taskbar and then type the preceding command as shown in Figure 2.19:

```

Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:54:40) [MSC v.1900 64 bit win32]
Type "copyright", "credits" or "license()" for more information.
>>> help(modules)

```

Figure 2.19: To view the modules available in Python

It will display all the module names currently available in your Python software as shown in Figure 2.20:

```

Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:54:40) [MSC v.1900 64 bit win32]
Type "copyright", "credits" or "license()" for more information.
>>> help(modules)

```

_abc	_collections	marshal	sre_compile
_decimal	_colorsys	_math	sre_constants
_dummy_thread	_compileall	matplotlib	sre_parse
_elementtree	_concurrent	_mimetypes	stackviewer
_functools	_config	mmap	stat
_hashlib	_config_key	modulefinder	statistics
_heapq	_configdialog	msilib	statusbar
_imp	_configparser	msvcrt	string
_io	_contextlib	multicall	stringprep
_json	_copy	multiprocessing	struct
_locale	_copyreg	netrc	subprocess
_lsprof	_crypt	nntplib	sunau
_lzma	_csv	nt	symbol
_markupbase	_ctypes	ntpath	syntable
_md5	_curses	nturl2path	sys
_msi	_cycler	numbers	sysconfig
_multibytecodec	_datetime	numpy	tabnanny
_multiprocessing	_dateutil	opcode	tarfile
_opcode	_dbm	operator	telnetlib
_operator	_debugger	optparse	tempfile
_osx_support	_debugger_r	os	test
_overlapped	_debugobj	outwin	textview
_pickle	_debugobj_r	pandas	textwrap
_pycbr	_decimal	paragraph	this
_pydecimal	_delegator	parenmatch	threading
_pyio	_difflib	parser	time
_random		pathbrowser	timeit

Figure 2.20: To verify installed packages in Python

You can verify that your Python software now has the packages like: numpy, xlrd, pandas, and matplotlib available as modules.

Writing Our First Python Program

We want to write our first Python program and execute it. This will also help us to test whether our installation is correct or not. Let's write our first program to add two numbers. First have a look at the Python program code given here:

```

# first program to add two numbers - line 1
a = 10 # store 10 into variable a - line 2
b = 15 # store 15 into variable b - line 3
c = a + b # add 10 and 15 and store into variable c - line 4
print("Sum= ", c) # display the sum - line 5

```

The hash symbol '#' represents the comment line in Python. A comment is useful to describe the elements of a program. Comments are not executed by the Python compiler or PVM. Hence, it is not compulsory to write comments; however, comments improve

understanding of a program. We started the above program with a comment line that gives the aim of our program.

We all know that the data and results are always stored in computer's memory locations. These memory locations are called 'variables'. In the second line of our program, we stored the data 10 into a variable (or memory location) by the name 'a' as:

```
a = 10
```

Similarly, in the third line, we stored data 15 into another variable 'b' as:

```
b = 15
```

In the fourth line, we added them by writing `a + b`. The result of this sum should be stored into another memory location by the name 'c' as:

```
c = a + b
```

So, the result is in 'c'. We should now display the result on the screen for the user. This is done by the `print()` function. The purpose of a function is to perform a task. The `print()` function is already written by Python developers and we can use it for display purpose as:

```
print(c)
```

This will display the result 25. But, we want to display some message while displaying the result as:

```
Sum= 25
```

This will help the user to understand the results clearly. For this purpose, we used the `print()` function in fifth line as:

```
print("Sum= ", c)
```

Here, "Sum=" is called a string. A string contains a group of characters and is usually enclosed in double quotes or single quotes. Strings are displayed without any change, as they are. After this string, we put a comma and then wrote 'c'. This will display the result as:

```
Sum= 25
```

Executing a Python Program

There are three ways of executing a Python program.

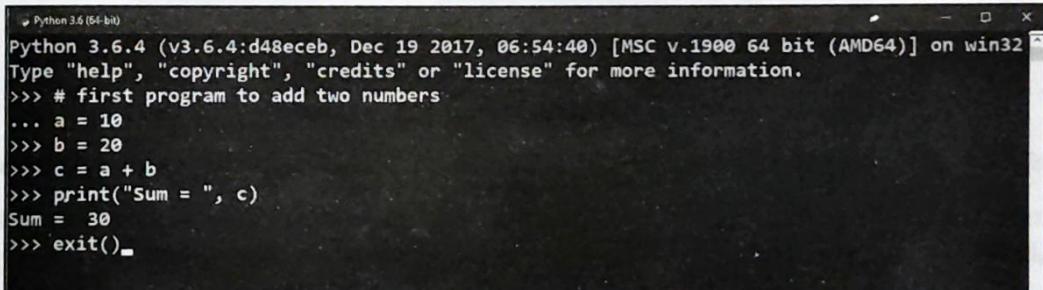
- ❑ Using Python's command line window
- ❑ Using Python's IDLE graphics window
- ❑ Directly from System prompt

The first two are called *interactive modes* where we can type the program one line at a time and the PVM executes it immediately. The last one is called *non-interactive mode* where we ask the PVM to execute our program after typing the entire program.

Using Python's Command Line Window

Click the Python's command line icon which is already added to our taskbar present at the bottom of the desktop as shown in Figure 2.8.

This will open Python's command line window. We can see >>> symbol which is called Python prompt. Type our first Python program at the >>> prompt, as shown in Figure 2.21:



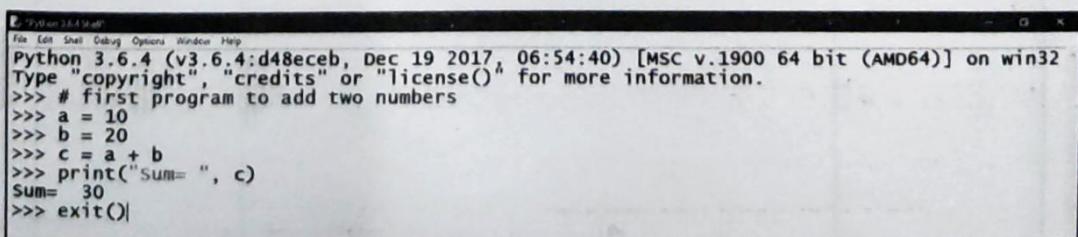
```
Python 3.6.4 (v3.6.4:d48ebeb, Dec 19 2017, 06:54:40) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> # first program to add two numbers
... a = 10
... b = 20
... c = a + b
>>> print("Sum = ", c)
Sum = 30
>>> exit()
```

Figure 2.21: The Python Command Line Window

After typing the last line and pressing the Enter button, we can see the result of our program. After that, type exit() or quit() at the Python prompt. This will terminate the PVM and close the window. Note that it is possible to select / modify the font style and font size in the command line window by right clicking on the top title bar and then selecting 'Properties' option.

Using Python's IDLE Graphics Window

It is possible to execute a Python program by going to Integrated Development Environment (IDLE) which provides graphical interface to the user. Click the Python's IDLE icon (Figure 2.8). This will open the Python's IDLE window. Type the program. After entering the last line, we can see the result, as shown in Figure 2.22:



```
File Edit Shell Debug Options Window Help
Python 3.6.4 (v3.6.4:d48ebeb, Dec 19 2017, 06:54:40) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> # first program to add two numbers
... a = 10
... b = 20
... c = a + b
>>> print("Sum= ", c)
Sum= 30
>>> exit()
```

Figure 2.22: Python's IDLE Shell Window

To terminate the IDLE window, type exit() or quit() at the Python prompt. It will display a message as to kill the program or not, as shown in Figure 2.23:

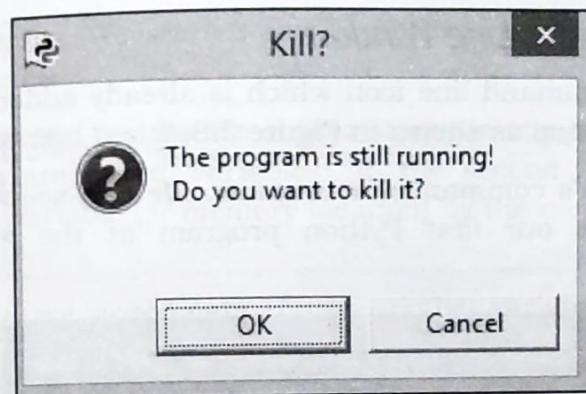


Figure 2.23: Closing IDLE window

Click the 'OK' button. The Python compiler will terminate and the window will close.

Note that we can select a particular font and also increase or decrease the font size in the IDLE window by choosing the 'Options' menu → Configure IDLE... option.

Running the Python programs as discussed in the preceding sections may not be convenient as we can type only one statement at a time. When the programmer wants to type a big program, then the best way is to use the File → New File option in the IDLE window. Then it opens a new window where we can type the entire program, as shown in Figure 2.24:

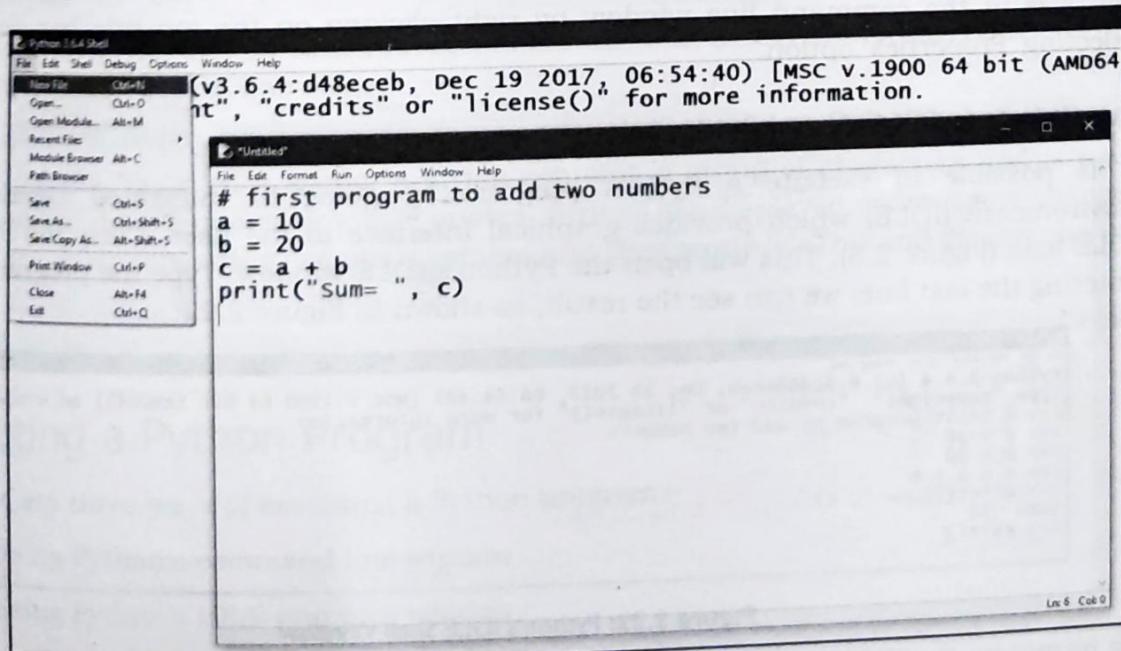


Figure 2.24: Typing a big Program in IDLE window

After typing the program, save it by clicking File → Save in the program window and then type a file name and .py as extension name in a directory. This is shown in Figure 2.25:

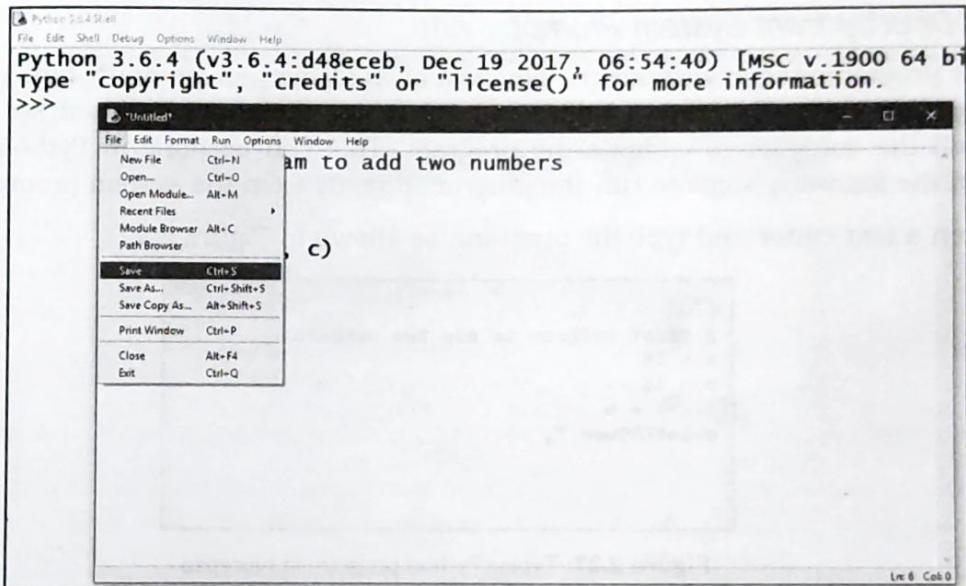


Figure 2.25: Saving a Program in IDLE window

We can compile and run the program by using the options File Run → Run Module or by simply pressing the F5 button on the keyboard. A new window will open where the results are displayed, as shown in Figure 2.26:

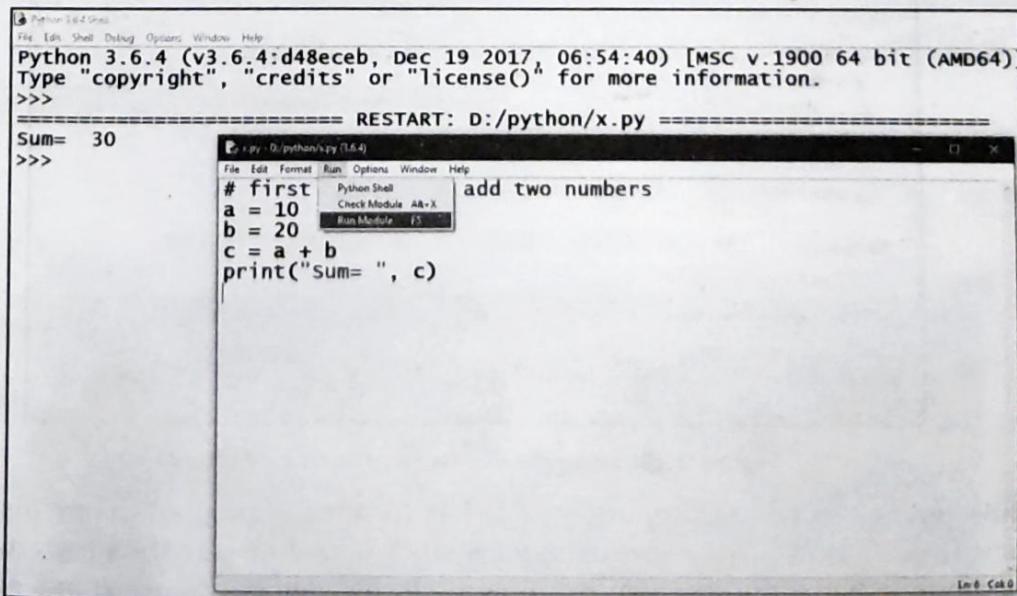


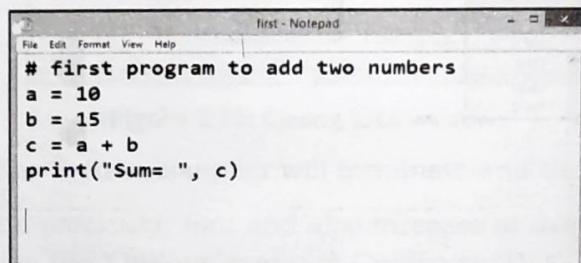
Figure 2.26: Running a Program in IDLE window

To reopen a Python program that is already saved in the Python IDLE window, we can use File → Open option and then go to the directory where the file is saved and click the filename.

Running Directly from System Prompt

In most programming environments like Java or .NET, we can generally type the program in a Text editor like Notepad or EditPlus. After typing the entire program, we save it and then call the compiler to compile the program. This can be done in Python also. Let's perform the following steps to run the program directly from the system prompt:

1. Open a text editor and type the program, as shown in Figure 2.27:



```
# first program to add two numbers
a = 10
b = 15
c = a + b
print("Sum= ", c)
```

Figure 2.27: Typing Python program in Notepad

2. Save the program by clicking File → Save As. Type the program name as "first.py", as shown in Figure 2.28:

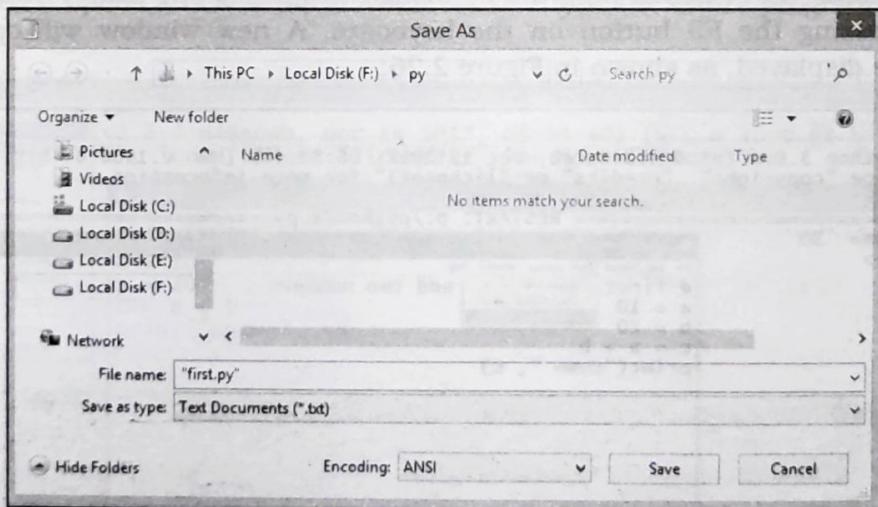


Figure 2.28: Saving the Python Program in a Directory

While typing the program name, it is better to put the program name inside double quotes as: "first.py". These double quotes tell Notepad to save the program exactly as the name is typed and not to save it as a text file. All Python programs should have .py as the file name extension.

3. Open the command prompt (or DOS prompt) window and go to that directory where the program is saved. This can be done by typing COMMAND or CMD commands at Start → Run.

4. In Windows 10 operating system, we should first right click on the ‘Start’ button and then click ‘Run’ to see the Run dialog box where we should type ‘command’, as shown in Figure 2.29:

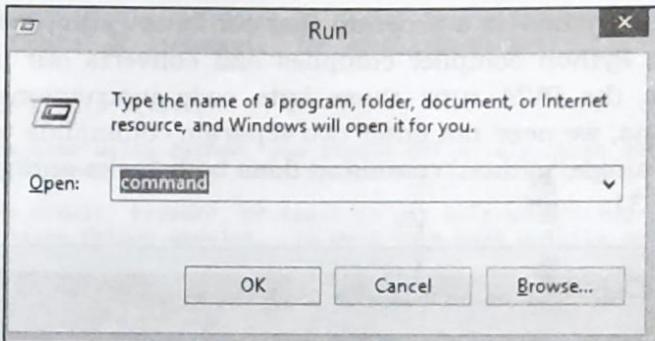


Figure 2.29: Going to DOS prompt

5. Click the ‘OK’ button. This opens the command prompt window.
6. Go to that directory where our Python program is saved. Suppose, we have saved our program in ‘F:’ drive and in the ‘py’ directory. Now, change the directory to F:\py using the CD (Change Directory to) command as:

```
C:>f:  
F:>cd py  
F:\PY>
```

To see the directory contents and verify whether our first.py program is already available in that directory or not, we can display the directory contents as:

```
F:\PY>dir
```

The above DOS command will display the contents of the directory, as shown in Figure 2.30:

A screenshot of a Windows Command Prompt window. The title bar says 'Microsoft (R) Windows DOS' and 'C:\Windows\system32\cmd.exe'. The command history shows: C:\USERS\RNR>f:, F:\>cd py, and F:\PY>dir. The output of the dir command is as follows:

Volume in drive F has no label.
Volume Serial Number is 8E96-7F03

Directory of F:\PY

02-02-2016 16:41 <DIR>
02-02-2016 16:41 <DIR>
02-02-2016 16:35 92 first.py
30-01-2016 15:23 107 test.py
30-01-2016 15:03 141 test1.pyc
3 File(s) 340 bytes
2 Dir(s) 77,201,829,888 bytes free

F:\PY>

Figure 2.30: Entering the Directory where Python Programs are Saved

Let's now execute the first.py program by calling the python command. For this purpose, type python along with the file name at the command prompt window as:

```
F:\PY>python first.py
```

Here, the command 'python' is a program that contains Python compiler as well as PVM. Hence, first of all, Python compiler compiles and converts our program into byte code instructions. Then the PVM runs those byte code instructions to produce the final results. That means, we need not enter two separate commands to compile and then run our program. The single 'python' command does both tasks and gives the final result, as shown in Figure 2.31:



Figure 2.31: Running the Python Program at System Prompt

We should understand that the Python compiler along with PVM is loaded into memory only when we type 'python' at the command prompt window. When the program execution is completed and the results are displayed, the Python compiler and the PVM will be terminated from memory. When we execute another program, then again the Python compiler is loaded into memory along with PVM. To close the command prompt window, we can type exit as:

```
F:\PY>exit
```

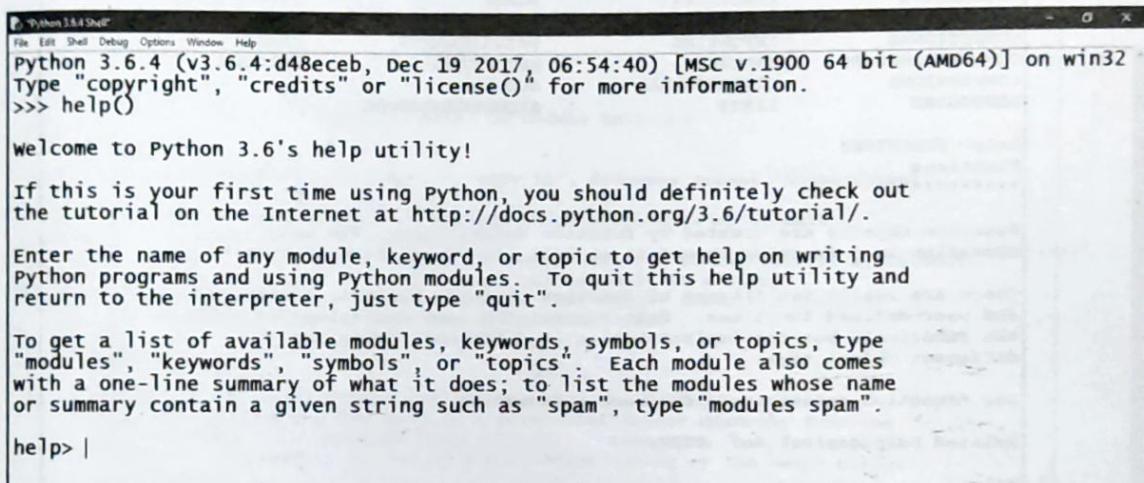
Note

All the programs in this book are executed from the command prompt window; however, readers can run these programs in any of the three modes discussed in the previous sections.

Getting Help in Python

When a programmer faces some doubt about how to use a particular feature of the Python language, he can view the help. To get help, one can enter help mode by typing

`help()` at Python prompt (i.e. `>>>` prompt). We can see the help utility appearing as shown in Figure 2.32:



```

Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:54:40) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> help()

welcome to Python 3.6's help utility!

If this is your first time using Python, you should definitely check out
the tutorial on the Internet at http://docs.python.org/3.6/tutorial/.

Enter the name of any module, keyword, or topic to get help on writing
Python programs and using Python modules. To quit this help utility and
return to the interpreter, just type "quit".

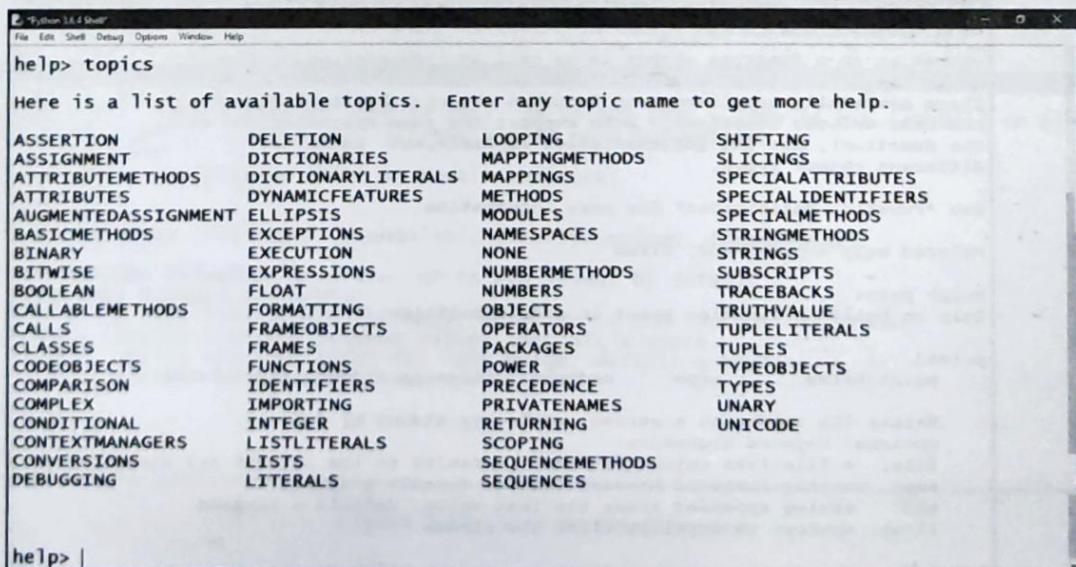
To get a list of available modules, keywords, symbols, or topics, type
"modules", "keywords", "symbols", or "topics". Each module also comes
with a one-line summary of what it does; to list the modules whose name
or summary contain a given string such as "spam", type "modules spam".

help> |

```

Figure 2.32: Entering Help Mode in Python

Now, we can type ‘modules’ to see which modules are available in Python. We can type ‘topics’ to know about topics in Python. Let’s enter topics at the help prompt, as shown in Figure 2.33:



```

Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:54:40) [MSC v.1900 64 bit (AMD64)] on win32
File Edit Shell Debug Options Windows Help
help> topics

Here is a list of available topics. Enter any topic name to get more help.

ASSERTION          DELETION          LOOPING           SHIFTING
ASSIGNMENT        DICTIONARIES      MAPPINGMETHODS  SLICINGS
ATTRIBUTEMETHODS  DICTIONARYLITERALS METHODS          SPECIALATTRIBUTES
ATTRIBUTES         DYNAMICFEATURES  MODULES          SPECIALIDENTIFIERS
AUGMENTEDASSIGNMENT ELLIPSIS        NAMESPACES      SPECIALMETHODS
BASICMETHODS       EXCEPTIONS       NONE             STRINGMETHODS
BINARY             EXECUTION        NUMBermETHODS   SUBSCRIPTS
BITWISE            EXPRESSIONS      NUMBERS         TRACEBACKS
BOOLEAN            FLOAT           OBJECTS         TRUTHVALUE
CALLABLEMETHODS    FORMATTING      OPERATORS      TUPLELITERALS
CALLS              FRAMEOBJECTS    PACKAGES       TUPLES
CLASSES            FRAMES          POWER           TYPEOBJECTS
CODEOBJECTS        FUNCTIONS       PRECEDENCE    TYPES
COMPARISON         IDENTIFIERS    PRIVATE NAMES  UNARY
COMPLEX            IMPORTING      RETURNING     UNICODE
CONDITIONAL       INTEGER         SCOPING        SEQUENCEMETHODS
CONTEXTMANAGERS   LISTLiterals   SEQUENCES
CONVERSIONS        LISTS          LITERALS
DEBUGGING          LITERALS

help> |

```

Figure 2.33: Getting Help on Topics

Among the topics, suppose we want to know about functions, we should enter ‘FUNCTIONS’ in capital letters since the FUNCTIONS topic is shown in capital letters in the help window, as shown in Figure 2.34:

```

Python 3.4.1 Shell
File Edit Shell Debug Options Windows Help
CODEOBJECTS      FRAMES          PACKAGES        TUPLES
COMPARISON       FUNCTIONS        POWER           TYPEOBJECTS
COMPLEX          IDENTIFIERS    PRECEDENCE     TYPES
CONDITIONAL     IMPORTING       PRIVATE NAMES  UNARY
CONTEXTMANAGERS INTEGER         RETURNING      UNICODE
CONVERSIONS      LISTLITERALS   SCOPING        SEQUENCEMETHODS
DEBUGGING        LISTS           help> FUNCTIONS
Functions
*****
Function objects are created by function definitions. The only
operation on a function object is to call it: "func(argument-list)".

There are really two flavors of function objects: built-in functions
and user-defined functions. Both support the same operation (to call
the function), but the implementation is different, hence the
different object types.

See *Function definitions* for more information.

Related help topics: def, TYPES
help>

```

Line 347 Col 6

Figure 2.34: Getting Help on Functions

To get help on the print function, we can type ‘print’ at the help prompt, as shown in Figure 2.35:

```

Python 3.4.1 Shell
File Edit Shell Debug Options Windows Help
operation on a function object is to call it: "func(argument-list)".

There are really two flavors of function objects: built-in functions
and user-defined functions. Both support the same operation (to call
the function), but the implementation is different, hence the
different object types.

See *Function definitions* for more information.

Related help topics: def, TYPES
help> print
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
    file: a file-like object (stream); defaults to the current sys.stdout.
    sep: string inserted between values, default a space.
    end: string appended after the last value, default a newline.
    flush: whether to forcibly flush the stream.

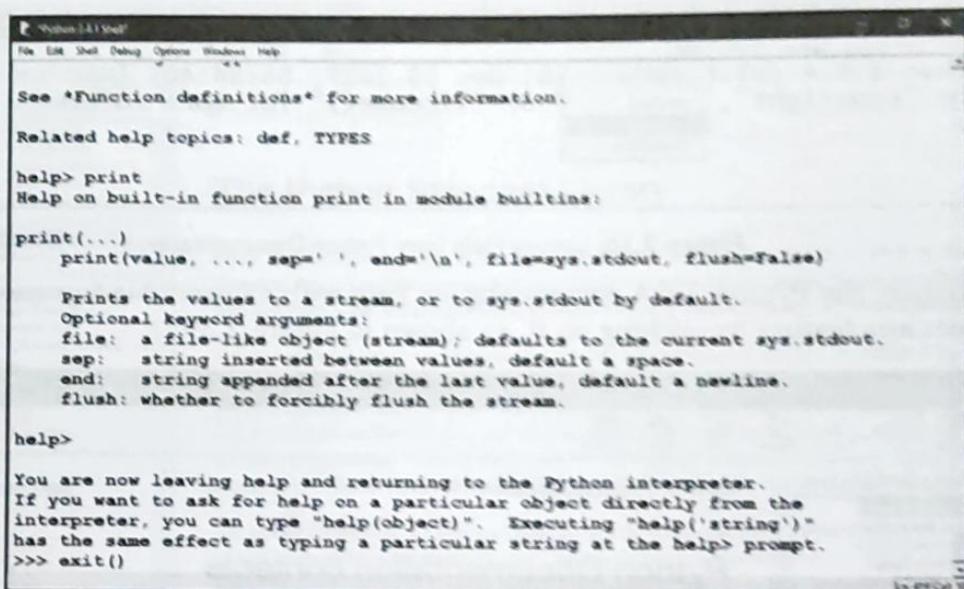
help> |

```

Line 413 Col 6

Figure 2.35: Getting Help on Print Function

To quit from the help mode, we should simply press the Enter button once again without typing anything. A message appears that we are leaving the help mode and then we arrive at the Python prompt, i.e. >>>. To exit the Python interpreter (or PVM), we should type either exit() or quit(), as shown in Figure 2.36:



The screenshot shows the Python 3.4 Shell window. The user has entered 'help> print' and is viewing the documentation for the built-in function 'print'. The output includes the function signature, optional keyword arguments (file, sep, end, flush), and a note about leaving help mode.

```

Python 3.4 Shell
File Edit Shell Debug Options Windows Help

See *Function definitions* for more information.

Related help topics: def, TYPES

help> print
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
        file: a file-like object (stream); defaults to the current sys.stdout.
        sep: string inserted between values, default a space.
        end: string appended after the last value, default a newline.
        flush: whether to forcibly flush the stream.

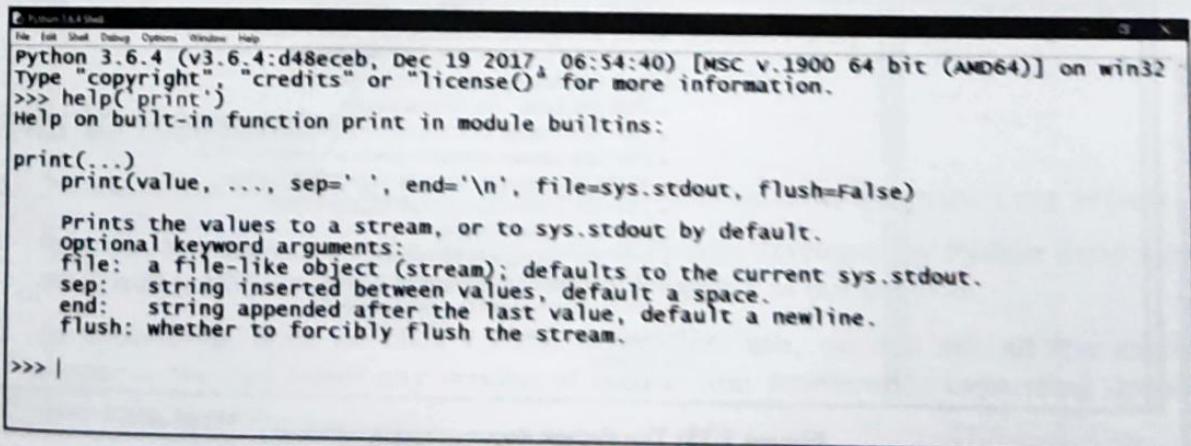
help>

You are now leaving help and returning to the Python interpreter.
If you want to ask for help on a particular object directly from the
interpreter, you can type "help(object)". Executing "help('string')"
has the same effect as typing a particular string at the help> prompt.
>>> exit()

```

Figure 2.36: Exiting the Python IDLE Window

We can also view help without entering the help mode. Viewing help is possible at the Python prompt. We can use the `help()` command and inside the parentheses, type the item name in single quotes. For example, to get help on topics, we can type `help('topics')` and to get help on the `print` function, we can type `help('print')`, as shown in Figure 2.37:



The screenshot shows the Python 3.6.4 shell. The user has entered '`>>> help('print')`' and is viewing the documentation for the built-in function 'print'. The output is identical to Figure 2.36, showing the function signature and optional keyword arguments.

```

Python 3.6.4 (v3.6.4:d48eceb, Dec 19 2017, 06:54:40) [MSC v.1900 64 bit (AMD64)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> help('print')
Help on built-in function print in module builtins:

print(...)
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)

    Prints the values to a stream, or to sys.stdout by default.
    Optional keyword arguments:
        file: a file-like object (stream); defaults to the current sys.stdout.
        sep: string inserted between values, default a space.
        end: string appended after the last value, default a newline.
        flush: whether to forcibly flush the stream.

>>> |

```

Figure 2.37: Using the `help()` function

Getting Python Documentation Help

Python developers have provided extensive description of all Python features in a document that is called 'Python documents'. This provides a great help for beginners and for professional programmers to understand all the features of Python. To see Python documentation help, we should open the IDLE window. Then select Help → Python Docs, as shown in Figure 2.38:

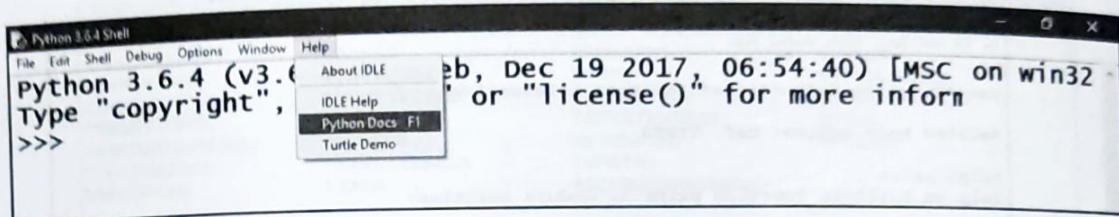


Figure 2.38: Getting Help from Python Documentation

It will display the Python 3.6.4 documentation help with all available features so that one can select any feature by clicking on it, as shown in Figure 2.39:

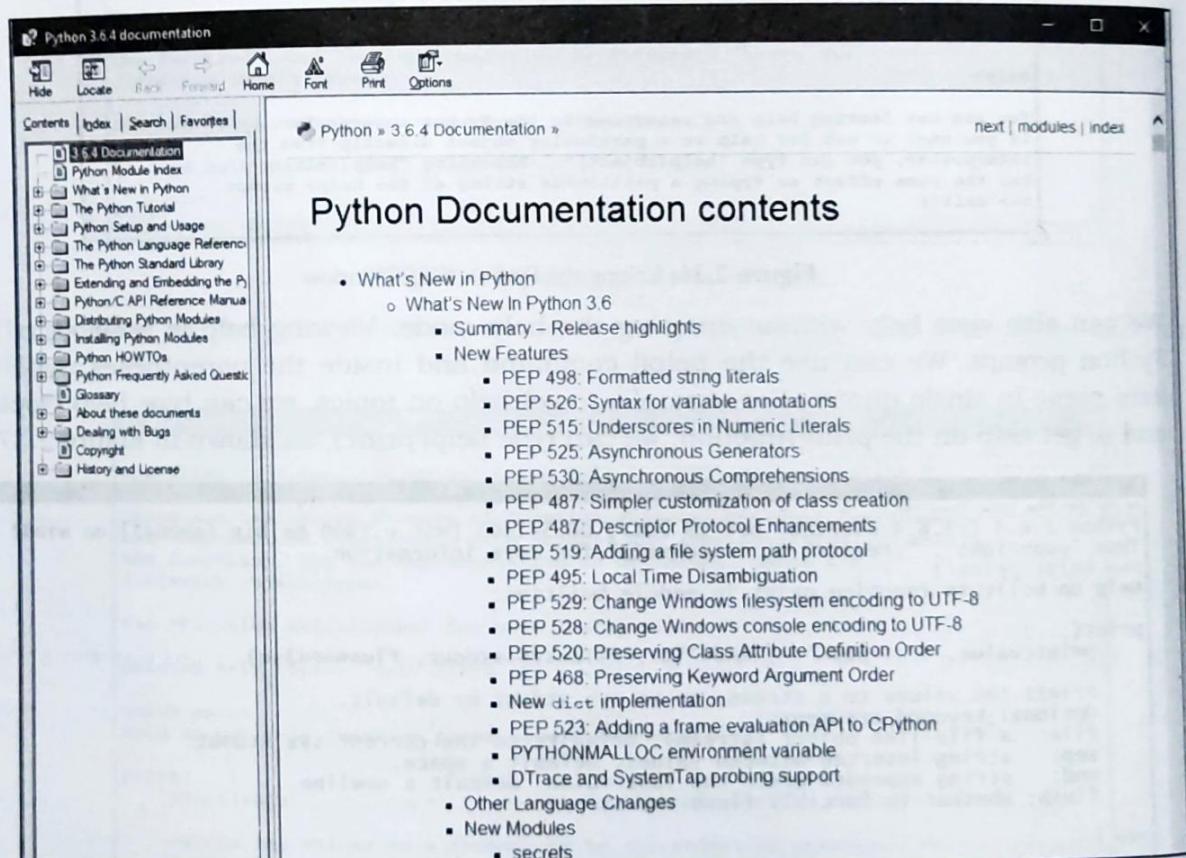


Figure 2.39: The Python documentation window

At the left side frame, we see contents out of which the following are very useful for us:

- ❑ The Python Tutorial
- ❑ The Python Language Reference
- ❑ The Python Standard Library

For example, click 'The Python Standard Library' to see the topic-wise comprehensive help on Python Standard Library. We can click on the close button (X) to close this help window, as shown in Figure 2.40:

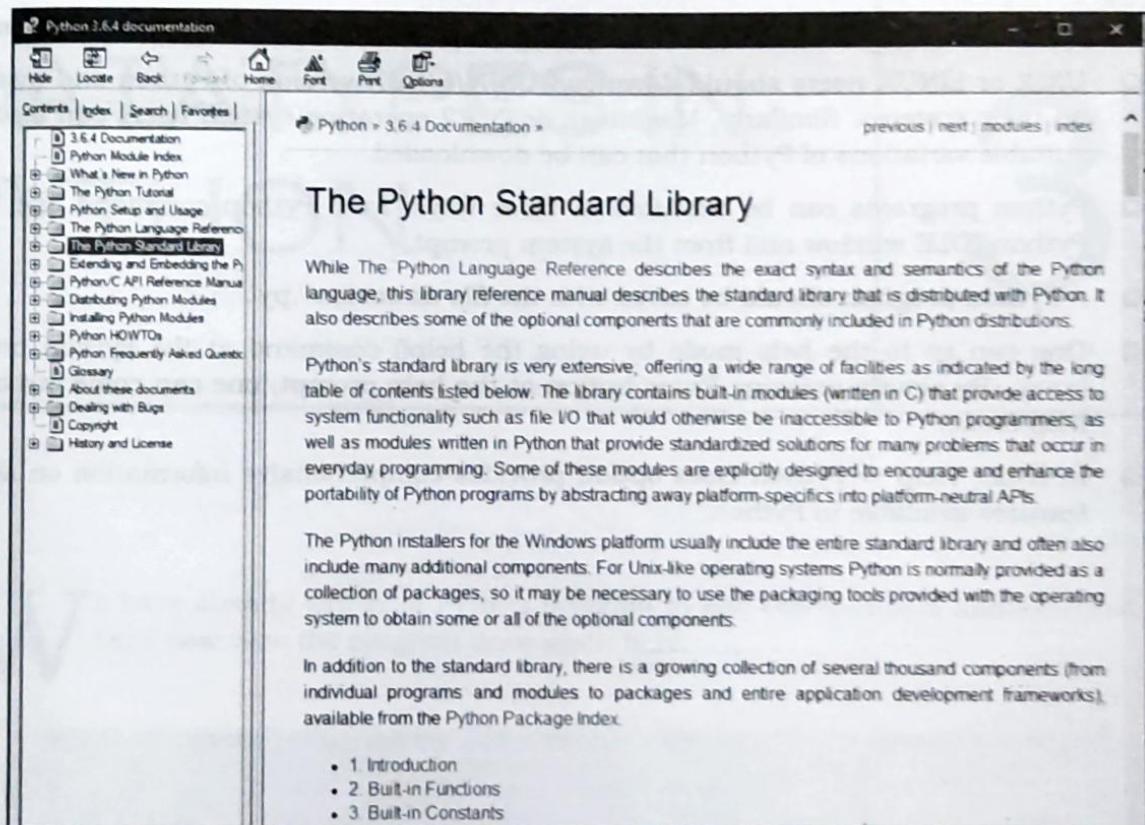


Figure 2.40: Going to Python Standard Library

Points to Remember

- ❑ Python is free software and can be downloaded freely from the python.org website.
- ❑ It is also possible to see the source code of Python developed by Python development team and modify and use it as per our requirements in our projects.
- ❑ In python.org, once we click on the Downloads tab, we can see all the available versions. We can select any version of Python and download it depending upon our operating system.
- ❑ We should select Python 32-bit version or 64-bit version depending whether our operating system is 32-bit or 64-bit.
- ❑ We should install Python 3.6.4 version along with the packages like numpy, pandas, xlrd and matplotlib.
- ❑ numpy is a package used to handle multidimensional arrays in Python.
- ❑ pandas is a package that is useful to analyze the data.
- ❑ xlrd is a package that is useful to extract data from Excel spreadsheet files.

- matplotlib is a package useful to represent data in the form of graphs and diagrams.
- UNIX or LINUX users should download UNIX/LINUX version of Python and install it on their systems. Similarly, Macintosh or OS/2 operating system users can also find suitable variations of Python that can be downloaded.
- Python programs can be executed in three ways: from Python command line, from Python IDLE window and from the system prompt.
- A Python program should be saved with the file extension '.py'.
- One can go to the help mode by using the help() command at the Python prompt (>>>). By simply pressing Enter button at the help prompt, one can come out of the help mode.
- In IDLE, Help → Python Docs option provides comprehensive information on all the features available in Python.