

1...

Introduction and Syntax of Python Program

Chapter Outcomes...

- Identify the given variables, keywords and constants in Python.
- Use indentation, comments in the given program.
- Install the given Python IDE and editor.
- Develop the python program to display the given text.

Learning Objectives...

- To understand Basic Concepts in Python Programming
- To learn Features and Environment for Python Programming
- To know Python Programming Building Blocks like Keywords, Variables, Identifiers etc.
- To learn Data Types in Python Programming.

1.0 INTRODUCTION

- Python is a high-level, interpreted, interactive and object-oriented programming language. Today, Python is the trendiest programming language programming.
- There are several reasons for why Python programming language is the preferable choice of the programmers/developers over other popular programming languages like C++, Java and so on.
- Python is popular programming language because of it provides more reliability of code, clean syntax of code, advanced language features, scalability of code, portability of code, support object oriented programming, broad standard library, easy to learn and read, support GUI mode, interactive, versatile and interpreted, interfaces to all major commercial databases, and so on.

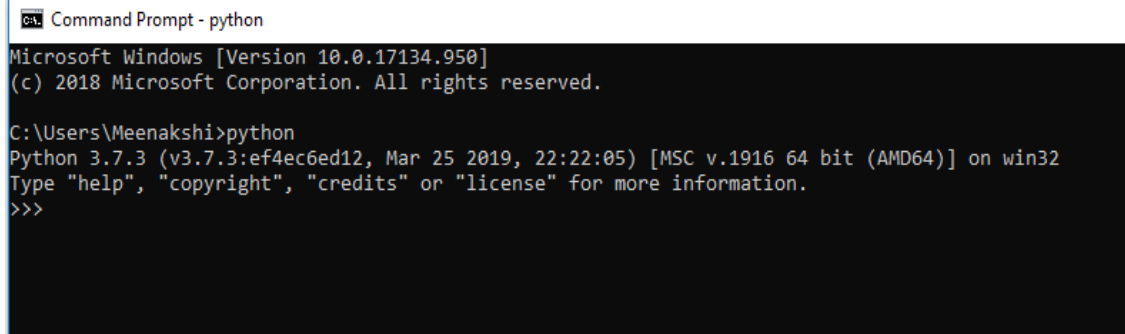
History of Python Programming Language:

- Python laid its foundation in the late 1980s. Python was developed by Guido Van Rossum at National Research Institute for Mathematics and Computer Science in Netherlands in 1990.
- Inspired by Monty Python's Flying Circus, a BBC comedy series, he named the language Python.
- Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.
- ABC programming language is said to be the predecessor of Python language which was capable of Exception Handling and interfacing with Amoeba Operating System.
- Like Perl, Python source code is now available under the GNU General Public License (GPL).
- In February 1991, Guido Van Rossum published Python 0.9.0 (first release) to alt.sources. In addition to exception handling, Python included classes, lists and strings.

- In 1994, Python 1.0 was released with new features like lambda, map, filter, and reduce which aligned it heavily in relation to functional programming.
- Python 2.0 added new features like list comprehensions, garbage collection system and it supported Unicode.
- On December 3, 2008, Python 3.0 (also called "Py3K") was released. It was designed to rectify fundamental flaw of the language. In Python 3.0 the print statement has been replaced with a print() function.
- Python widely used in both industry and academia because of its simple, concise and extensive support of libraries.
- Python is available for almost all operating systems such as Windows, Mac, Linux/Unix etc. Python can be downloading from <http://www.python.org/downloads>.
- Some common **applications of Python Programming** are:
 1. Google's App Engine web development framework uses Python as an application language.
 2. Maya, a powerful integrated 3D modeling and animation system, provides a Python scripting API.
 3. Linux Weekly News, published by using a web application written in Python.
 4. Google makes extensive use of Python in its Web Search Systems.
 5. The popular YouTube video sharing service is largely written in Python programming.
 6. The NSA uses Python for cryptography and intelligence analysis.
 7. iRobot uses Python programming to develop commercial and military robotic devices.
 8. The Raspberry Pi single-board computer promotes Python programming as its educational language.
 9. Netflix and Yelp have both documented the role of Python in their software infrastructures.
 10. Industrial Light and Magic, Pixar and others uses Python in the production of animated movies.

1.1 FEATURES OF PYTHON

- Python's features include:
1. **Easy to Learn and Use:**
 - Python is easy to learn and use. It is developer-friendly and high level programming language.
 - Python has few keywords, simple structure, and a clearly defined syntax that makes it easily understandable for beginners.
 - Python language is more expressive means that it is more understandable and readable for programmers.
 - In Python programming programs are easy to write and execute as it omits some cumbersome, poorly understandable and confusing features of other programming language such as C++ and Java.
 2. **Interpreted Language:**
 - Python is an interpreted language i.e., interpreter executes the code line by line at a time. This makes debugging easy and thus suitable for beginners. There are excellent, straightforward tools to work with python code, is interactive interpreter.
 - In python, we need not to learn a build system, IDE, special text editor, or anything else to start using python. All we need only a command prompt and the interactive editor.
 - Python provides a Python Shell (also known as Python Interactive Shell) which is used to execute a single Python command and get the result as shown below. If python is installed on the PC then to open the Python Shell on Windows, open the command prompt, write python and press enter.
 - As we can see, a Python Prompt comprising of three Greater Than symbols (>>>) appears, (See Fig. 1.1).



```

C:\Users\Meenakshi>python
Microsoft Windows [Version 10.0.17134.950]
(c) 2018 Microsoft Corporation. All rights reserved.






C:\Users\Meenakshi>python
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 22:22:05) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>

```

Fig. 1.1

3. Interactive Mode:

- Python programming language has support for interactive mode, which allows interactive testing and debugging of code. Graphical User Interfaces (GUIs) can be developed using Python.
- Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh and the X Window system of Unix.
- There are many free and commercial editors available for Python. Following table lists Python editors:

Sr. No.	Editor	Description	Icon/Logo
1.	IDLE	<ul style="list-style-type: none"> • IDLE is a popular Integrated Development Environment written in Python and it has been integrated with the default language. • Mainly used by the beginner level developers who want to practice on Python development. 	
2.	PyCharm	<ul style="list-style-type: none"> • PyCharm is one of the widely used Python IDE which was created by Jet Brains. • With PyCharm, the developers can write a neat and maintainable code. It helps to be more productive and gives smart assistance to the developers. • It takes care of the routine tasks by saving time and thereby increasing profit accordingly. 	
3.	Spyder	<ul style="list-style-type: none"> • It was mainly developed for scientists and engineers to provide a powerful scientific environment for Python. • It offers an advanced level of edit, debug, and data exploration feature. • It is very extensible and has a good plugin system and API. 	
4.	PyDev	<ul style="list-style-type: none"> • PyDev is an outside plugin for Eclipse. It is basically an IDE that is used for Python development. • It is linear in size. It mainly focuses on the refactoring of python code, debugging in the graphical pattern, analysis of code etc. It is a strong python inter-preter. 	
5.	Jupyter Notebook	<ul style="list-style-type: none"> • The Jupyter Notebook is a browser-based graphical interface to the IPython shell. • Allows us to create and share documents that contain live code, equations, visualizations and narrative text. 	

- In this text book, IDLE is used for Python programming. IDLE (Integrated Development and Learning Environment) is an Integrated Development Environment (IDE) for Python.
- To start IDLE interactive shell, search for the IDLE icon in the start menu and double click on it and we will get the following window (See Fig. 1.2).
- In Python IDLE shell not only we can execute commands one by one like in Python Command Prompt but also can create .py files and see execution of those files.

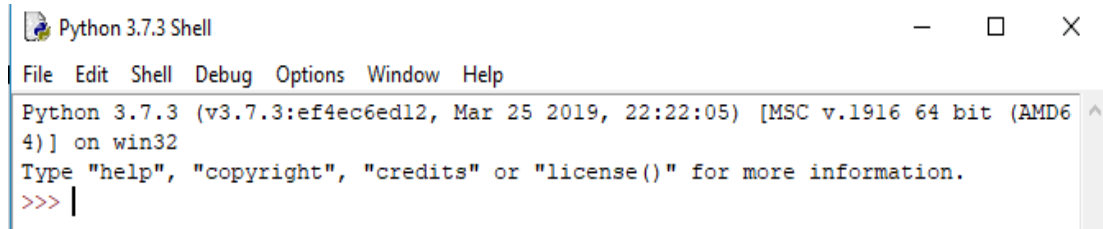


Fig. 1.2

4. Free and Open Source:

- Python programming language is developed under an OSI approved open source license making it freely available at official web address.
- The source-code is also available. Therefore, it is open source. The Python software can be freely distributed and any one can use and read its source code make changes/modifications to it, and use the pieces in new free programs.

5. Platform Independence/Cross-platform Language/Portable:

- Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- Python can run equally on different platforms such as Windows, Linux, Unix and Macintosh etc. So, we can say that Python is a portable language.

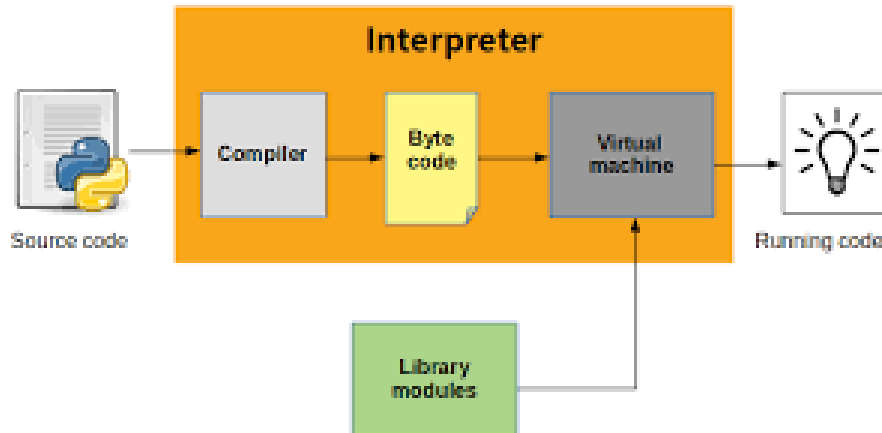


Fig. 1.3

- Python source code goes through Compiler which compiles the **source code** into a format known as **byte code**.
- Byte code is a lower level, platform independent, efficient and intermediate representation of the source code. As soon as **source code** gets converted to **byte code**, it is fed into **PVM (Python Virtual Machine)**.
- The PVM is the runtime engine of Python; it's always present as part of the Python system, and is the component that truly runs the scripts. Technically, it's just the last step of what is called the Python interpreter.

6. Object-Oriented Language:

- A programming language that can model the real world is said to be object-oriented. It focuses on objects, and combines data and functions. Contrarily, a procedure-oriented language revolves around functions, which are code that can be reused.

- Python supports both procedure-oriented and object-oriented programming which is one of the key python features. It also supports multiple inheritance, unlike Java.
- Python has a powerful but simplistic way also doing OOP, especially when compared to C++ and Java languages. Python supports object oriented language and concepts of classes and objects come into existence.

7. Extensible:

- Python programming implies that other languages such as C/C++ can be used to compile the code and thus it can be used further in the python code.
- Python has a large and broad library and provides rich set of module and functions for rapid application development.
- Python languages bulk library is portable and cross platform compatible with Unix, Windows

Limitations of Python:

1. Python is an interpreter based language. Therefore, it is bit slower than compiler based languages.
2. Python is a high level language like C/C++/Java, it also uses many layers to communicate with the operating system and the computer hardware.
3. Graphics intensive applications such as games make the program to run slower.
4. Due to the flexibility of the data types, Python's memory consumption is also high.

Structure of a Python Program:

- Fig. 1.4 shows a typical program structure Python programming.
- Python programming programs are structured as a sequence as statements. A Python statement is smallest program unit. Statements are the instructions that are written in a program to perform a specific task.
- A Python statement is a complete instruction executed by the Python interpreter. A Python program may consist as more than are statement.
- By default, the Python interpreter execute all statements sequentially, but we can change order of execution using control statements.

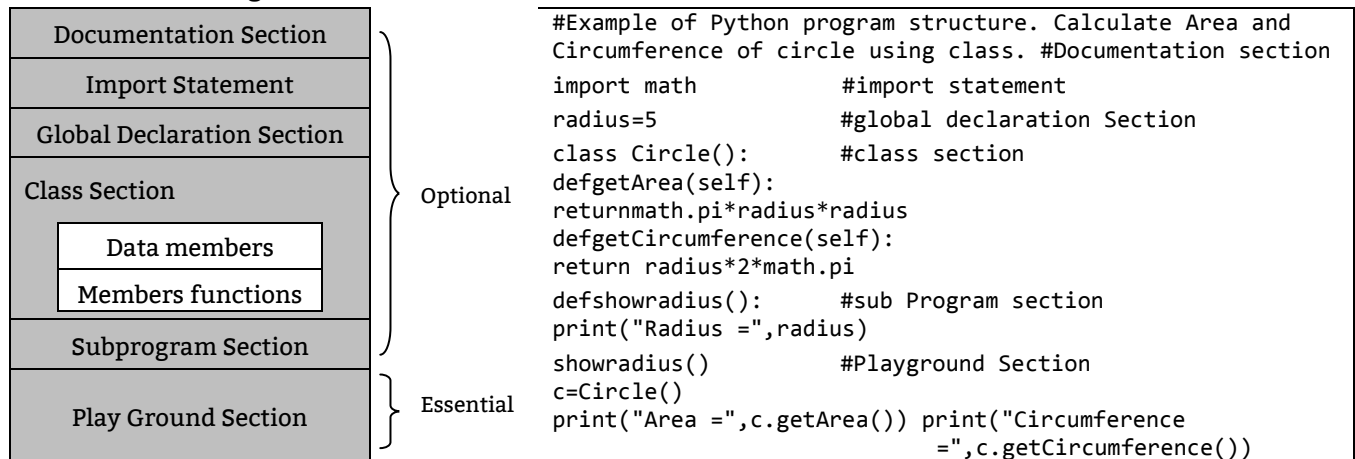


Fig. 1.4: Typical Program Structure of Python Programming with Example

- Program structure of Python programming contains following sections:
 1. **Documentation Section** includes the comments that specify the purpose of the program. A comments that is a non-executable statement which is ignored by the compiler while program execution. Python comments are written anywhere in the program.
 2. **Import Section** is used includes different built in or user defined modules.
 3. **Global Declaration Section** is used to define the global variables for the programs.
 4. **Classification** describe the information about the user defined classes in the program. A class is a collection of data members and member functions called method, that operation data members.

5. **Sub program Section** includes use defined functions. The functions include the set of statements that need to be executed when the function is called from anywhere.
6. **Pay Ground Section** is the main section of Python program and the main section starts where the function calling.

1.1.1 Running Python Scripts

- Python has two basic modes namely, normal and interactive.
- The **normal mode** is the mode where the scripted and finished .py files are run in the Python interpreter.
- The **interactive mode** is a command line shell which gives immediate feedback for each statement, while running previously fed statements in active memory.
- As new lines are fed into the interpreter, the fed program is evaluated both in part and in whole.

1. Interactive Mode:

- Interactive mode is great for quickly and conveniently running single lines or blocks of code. Here's an example using the python shell that comes with a basic python installation.
- The ">>>" indicates that the shell is ready to accept interactive commands. So for example if we want to print the statement "this is interactive mode", simply type the appropriate code and hit enter.

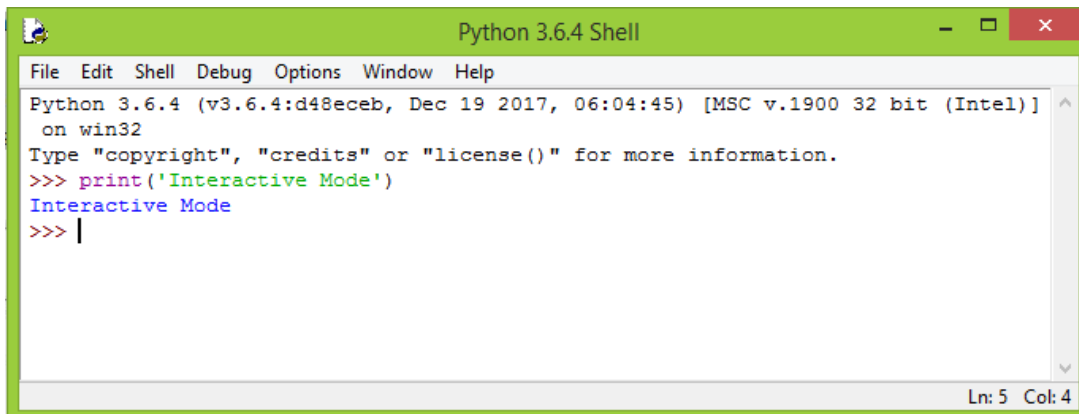


Fig. 1.5

2. Script Mode:

- In the standard Python shell we can go to "File" → "New File" (or just hit Ctrl + N) to pull up a blank script in which to put the code. Then save the script with a ".py" extension.
- We can save it anywhere we want for now, though we may want to make a folder somewhere to store the code as we test Python out. To run the script, either select "Run" → "Run Module" or press F5.
- We should see something like the following.

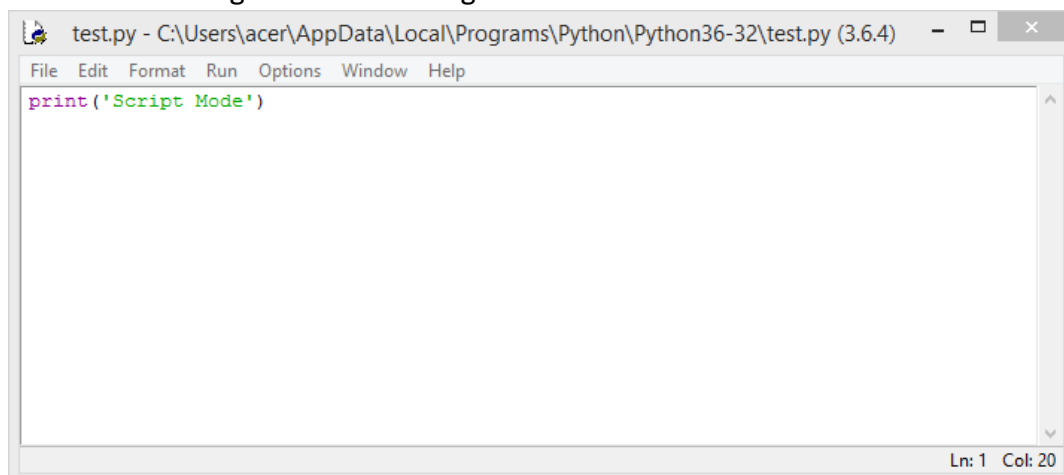


Fig. 1.6(a)

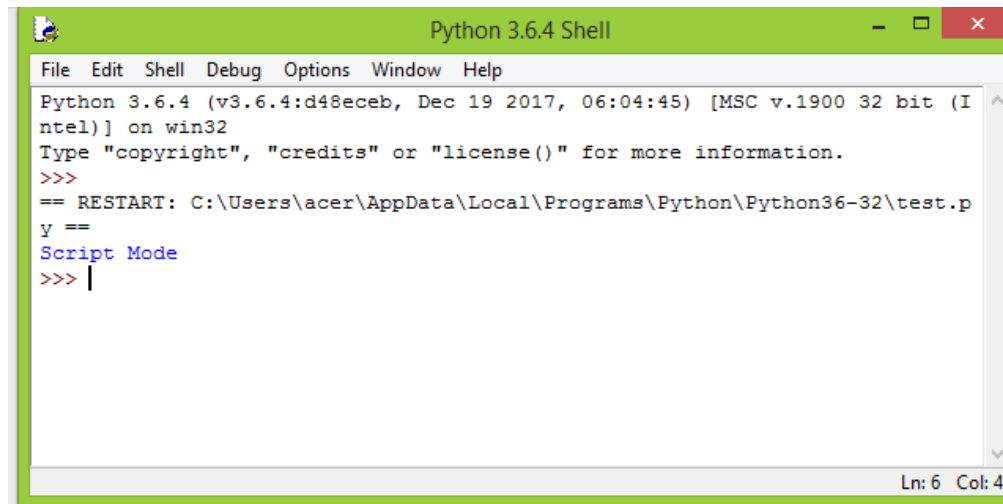


Fig. 1.6(b)

1.1.2 Internal Working of Python

- Python is an object oriented programming language like C++ and Java. Python is called an interpreted language means Python programs are executed by the interpreter. Python uses code modules that are interchangeable instead of a single long list of instructions that was standard for functional programming languages.
- The standard implementation of python is called “CPython”. It is the default and widely used implementation of the Python.
- Python does not convert its code into machine code. It actually converts code into byte code. So within python, compilation happens, but it’s just not into a machine language.
- It is into byte code and this byte code cannot be understood by CPU. So we need actually an interpreter called the Python Virtual Machine (PVM). The python virtual machine executes the byte codes.
- When a programmer tries to run a Python code as instructions in an interactive manner in a Python shell, then Python performs various operations internally.
- All such internal operations can be broken down into a series of steps as shown in Fig. 1.7.

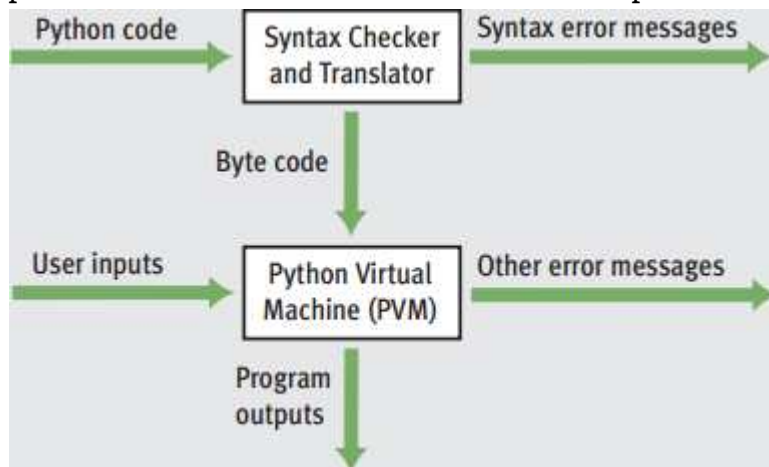


Fig. 1.7

- The Python interpreter performs following tasks to execute a Python program:
 1. The interpreter reads a Python expression or statement, also called the source code, and verifies that it is well formed. In this step, as soon as the interpreter encounters such an error, it halts translation with an error message.

2. If a Python expression is well formed, the interpreter then translates it to an equivalent form in a low-level language called byte code. When the interpreter runs a script, it completely translates it to byte code.
3. This byte code is next sent to another software component, called the Python Virtual Machine (PVM), where it is executed. If another error occurs during this step, execution also halts with an error message.

1.2 PYTHON BUILDING BLOCKS

- In order to write any Python program, we must be aware of its structure, available keywords and data types also have some knowledge of variables, constants, identification and so on.
- Keywords, identifiers, variables etc., are the basic building blocks of Python programming. Python uses the character sets as the building block to form the basic program elements such as variables, keywords, constants, etc.

1.2.1 Character Set

- The character set is a set of alphabets, letters, symbols and some special characters that are valid in Python programming language.
- Python uses the following character sets. These characters are submitted to the Python interpreter; they are interpreted or uniquely identified in various contexts, such as characters, identifiers, names or constants.
 1. **Lowercase English Letters:** a to z.
 2. **Uppercase English Letters:** A to Z.
 3. **Punctuation and Symbols:** "\$" and "!", to name a couple.
 4. **Whitespace Characters:** An actual space (" "), as well as a newline, carriage return, horizontal tab, vertical tab, and a few others.
 5. **Non-Printable Characters:** Characters such as backspace, "\b", that cannot be printed literally in the way that the letter A can be printed.
 6. **Delimiter:** Delimiters are symbols that perform a special role in Python like grouping, punctuation and assignment. Following symbols and symbol combination uses as a delimiter in python:

() [] { } , : . ` = ; += -= *= /= %= **= &= |= ^= >>= <<=
- A program in Python contains a sequence of instructions. Python breaks each statement into sequence of lexical components elements which are identify by the interpreter, known as tokens.
- A token is a smallest unit of the program. Python contains various types of tokens, such as keywords, variables, operators, literals, identifiers etc.

1.2.2 Identifiers

- A Python identifier is a name given to a function, class, variable, module or other objects that is used in Python program.
- All identifiers must obey the following rules:
 1. An identifier can be a combination of uppercase letters, lowercase letters, underscores, and digits (0-9). Examples include, Name, myClass, Emp_Salary, var_1, _Address and print_hello_world.
 2. We can use underscores to separate multiple words in the identifier.
 3. An identifier starts with a letter which can be alphabet (either lowercase or uppercase), underscore (_).
 4. Identifiers can be of any length.
 5. Identifiers cannot start with digit and must not contain any space or tabs. Example include, 2variable, 10ID.
 6. We cannot use Python keywords as identifiers.

- 7. Special characters such as %, @, and \$ are not allowed within identifiers. Example include, \$Money, @salary.
- 8. Python is a case-sensitive language and this behavior extends to identifiers. Thus, identifier Age and age are two distinct identifiers in Python.
- Example of valid identifiers includes: Circle_Area, EmpName, Student, Sum, Salary10, _PhoneNo.
- Example of invalid identifiers includes: !count, 4marks, % Loan.

1.2.3 Keywords

- Python keywords are reserved words with that have special meaning and functions. The keywords are predeclared words with specific meaning in the Python programs.
- Keywords should not be used as variable name, constant, function name, or identifier in the program code.
- In Python keywords are case sensitive. Keywords are used to define the syntax and structure of the programming language.
- Following table lists keywords in Python programming:

and	as	assert	break	class	continue
def	del	else	elif	except	exec
false	finally	for	from	global	if
import	in	is	lambda	none	not
or	pass	print	raise	return	true
try	while	with	yield		

1.2.4 Variables

- A variable is like a container that stores values that we can access or change. It is a way of pointing to a memory location used by a program. We can use variables to instruct the computer to save or retrieve data to and from this memory location.
- A variable is a name given to a location in the computer's memory location, where the value can be stored that can be used in the program.
- When we create a variable, some space in the memory is reserved for that variable to store a data value in it. The size of the memory reserved by the variable depends on the type of data it is going to hold.
- The variable is so called because its value may vary during the time of execution, but at a given instance only one value can be stored in it.

Variable Declaration:

- A variable is an identifier, that holds a value. In programming, we say that we assign a value to a variable. Technically speaking, a variable is a reference to a computer memory, where the value is stored.
- Basic rules to declare variables in python programming language:
 1. Variables in Python can be created from alphanumeric characters and underscore(_) character.
 2. A variable cannot begin with a number.
 3. The variables are case sensitive. Means Amar is differ the 'AMAR' are two separate variables.
 4. Variable names should not be reserved word or keyword.
 5. No special characters are used except underscore (_) in variable declaration.
 6. Variables can be of unlimited length.
- Python variables do not have to be explicitly declared to reserve memory space. The variable is declared automatically when the variable is initialized, i.e., when we assign a value to the variable first time it is declared with the data type of the value assigned to it.

- This means we do not need to declare the variables. This is handled automatically according to the type of value assigned to the variable. The equal sign (=) i.e., the assignment operator is used to assign values to variables.
- The operand to the left of the = operator is the name of the variable and the operand to the right of the = operator is the literal value or any other variable value that is stored in the variable.

Syntax: variable=value

Example: For variable.

```
>>> a=10
>>> a
10
>>>
```

-
- Python language allows assigning a single value to several variables simultaneously.

Example: a=b=c=1

All above three variables are assigned to same memory location, when integer object is created with value 1.

- Multiple objects can also have assigned to multiple variables:

Example: a, b, c = 10,5.4,"hello"

In above example Integer object a assigned with value 10, float object b assigned with value 5.4 and string object c assigned with value "hello".

Example: If x, y, z are defined as three variable in a program, then x = 10 will store the value 10 in the memory location named as x, y = 5 will store the value 5 in the memory location named as y and x + y will store the value 15 in the memory location named as z (as a result after computation of x + y).

```
>>> x = 10
>>> y = 5
>>> name="Yogesh"
>>> z = x + y
>>> print(x); print(y); print(name) ; Print (z)
10
5
Yogesh
15
>>>
```

1.2.5 Literals

- A literal refers to the fixed value that directly appears in the program. Literals can be defined as, a data that is given in a variable or constant.
- Literals are numbers or strings or characters that appear directly in a program.
- Python support the following literals:
 - **string Literals:** "hello" , '12345'
 - **int Literals:** 0, 1, 2, -1, -2
 - **long Literals:** 89675L
 - **float Literals:** 3.14
 - **complex Literals:** 12j
 - **boolean Literals:** True or False
 - **special Literals:** None
 - **unicode Literals:** u"hello"
 - **list Literals:** [], [5, 6, 7]

- **tuple Literals:** (), (9), (8, 9, 0)
- **dict Literals:** {}, {'x':1}
- **set Literals:** {8, 9, 10}

1. String Literals:

- String literals can be formed by enclosing a text in the quotes. We can use both single quote ('...') as well as double quotes for ("...") a String.
- In simple words, a string literal is a collection of consecutive characters enclosed within a pair of single or double quotes.

Example: For string literal.

```
Fname='Hello'
Lname="Python"
print(Fname)
print(Lname)
```

Output:

```
Hello
Python
```

2. Numeric Literals:

- Numeric literals are immutable. Numeric literals comprise number or digits from 0 to 9.
- Numeric literals can belong to following four different numerical types.

int (signed integers)	long (long integers)	float (floating point)	complex (complex)
Numbers (can be both positive (+) and negative (-)) with no fractional part. Example: 100	Integers of unlimited size followed by lowercase or uppercase Example: L. 87032845L	Real numbers with both integer and fractional part. Example: 26.2	In the form of a+bj where a forms the real part and b forms the imaginary part of complex number. Example: 3.14j

3. Boolean Literals:

- A Boolean literal can have any of the two values True or False.

Example:

```
>>>5--2
false
>>>3<9
True
>>>
```

4. Special Literals:

- Python contains one special literal i.e., None. It is special constant in Python programming that represent the absence of a value or null value.
- None is used to specify to that field that is not created. It is also used for end of lists in Python.

Example: For special literal.

```
>>> val1=10
>>> val2=None
>>> val1
10
>>> val2
>>> print val2
None
```

```
>>>
```

5. Literal Collections:

- Collections such as tuples, lists and dictionary are used in Python.

(i) List:

- List contain items of different data types. Lists are mutable i.e., modifiable. The values stored in list are separated by commas(,) and enclosed within a square brackets ([]). We can store different type of data in a list.
- Value stored in a list can be retrieved using the slice operator([] and [:]). The plus sign (+) is the list concatenation and asterisk(*) is the repetition operator.

(ii) Tuple:

- Tuple is used to store the sequence of immutable python objects.
- A tuple can be created by using () brackets and separated by commas (,).

(iii) Directory:

- The directory in Python is a collection of key value pairs created using { }.
- The key and value are separated by a colon (:) and the elements/items are separated by commas (,).

Example: For literal collections.

```
#create list
>>>numbers=[1,2,3,4,5,6,7]
>>>print(numbers)
#create tuples
>>>list=('a','b','c')
>>>print(list1)
#create dictionary
>>>list2={'fname':'vijay', 'lname':'patil'}
>>>print(list2)
```

Output:

```
[1,2,3,4,5,6,7]
('a','b','c')
{'fname':'vijay', 'lname':'patil'}
```

6. Value and Type on Literals:

- Programming languages contain data in terms of input and output and any kind of data can be presented in terms of value.
- Value can be of numbers, strings or characters. To know the exact type of any value, python provides in-built method called type.

Syntax: type(value)

Example: For value and type literals.

```
>>> type('hello python')
<class 'str'>
>>> type('a')
<class 'str'>
>>> type(123)
<class 'int'>
>>> type(11.22)
<class 'float'>
```

- Most of the programming languages like C, C++, Java use braces { } to define a block of code. Python uses indentation.
- A code block (body of a function, loop etc.) starts with indentation and ends with the first un-indented line. The amount of indentation is up to we, but it must be consistent throughout that block.
- Generally, four whitespaces are used for indentation and is preferred over tabs, (See Fig. 1.8).

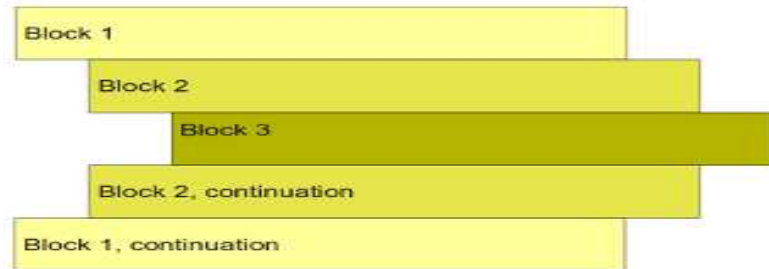


Fig. 1.8

- Indentation helps to convey a better structure of a program to the readers. It is used to clarify the link between control flow constructs such as conditions or loops, and code contained within and outside of them.

Example 1: For indentation.

```
>>> for i in range (1,11):
    print(i)
```

Output:

```
1
2
3
4
5
6
7
8
9
10
```

Example 2: for indentation in python.

```
>>> for i in range(1,11):
    print(i)
    if i==5:
        break
```

Output:

```
1
2
3
4
5
```

1.2.7 Commenting in Python

- Comments are meant for computer programmers for better understanding a program. Python interpreter ignores the comment in the program.
- 1. Single Line Comment (#):**
- Single line comments are created simply by beginning a line with the hash (#) character, and they are automatically terminated by the end of line.

Example: For single line comment.

```
# print is a statement      OR      print('Hello Python') # print is a statement
print('Hello Python')
```

- When the python interpreter sees #, it ignores all the text after # on the same line.
- 2. Multiple Line Comments ("):**
- In some situations, multiline documentation is required for a program. If we have comments that extend multiple lines, one way of doing it is to use hash (#) in the beginning of each line. Another way of doing this is to use quotation marks, either ''' or """".

- Similarly, when it sees the triple quotation marks `'''` it scans for the next `'''` and ignores any text in between the triple quotation marks.

Example: For multiline comment.

```
'''This is first python program
Print is a statement'''
```

1.3 PYTHON ENVIRONMENT SETUP (INSTALLATION AND WORKING OF IDE)

- The most up-to-date and current source code, binaries, documentation, news, etc. is available on the official website of Python <https://www.python.org/>.
- Python distribution is available for a wide variety of platforms such as Unix, Linux, Macintosh and Windows. We need to download only the binary code applicable for the platform and install Python.

Installing Python in Windows:

Step 1 : Open any internet browser then type <http://www.python.org/downloads/> in address bar and Enter. The Home page will appear, (See Fig. 1.9).

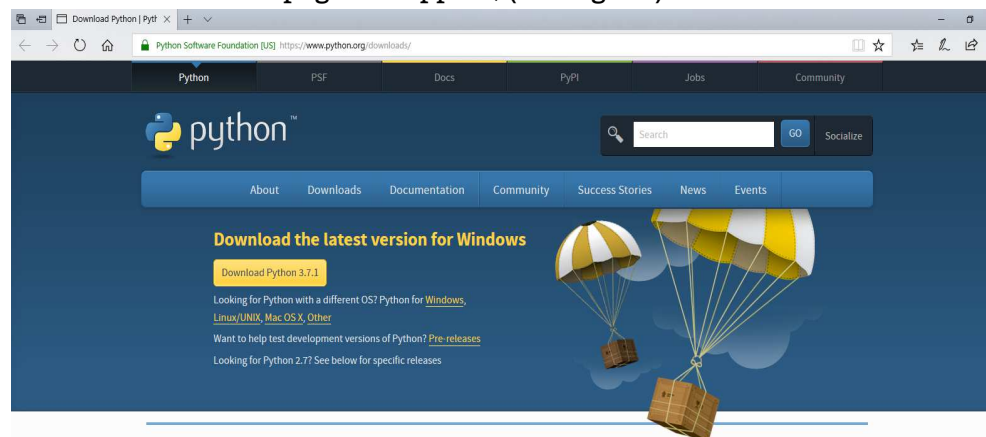


Fig. 1.9: Home Page

Step 2 : Click on download the latest version for windows:

Looking for a specific release?
Python releases by version number:

Release version	Release date		Click for more
Python 3.7.1	2018-10-20	Download	Release Notes
Python 3.6.7	2018-10-20	Download	Release Notes
Python 3.5.6	2018-08-02	Download	Release Notes
Python 3.4.9	2018-08-02	Download	Release Notes
Python 3.7.0	2018-06-27	Download	Release Notes
Python 3.6.6	2018-06-27	Download	Release Notes
Python 2.7.15	2018-05-01	Download	Release Notes

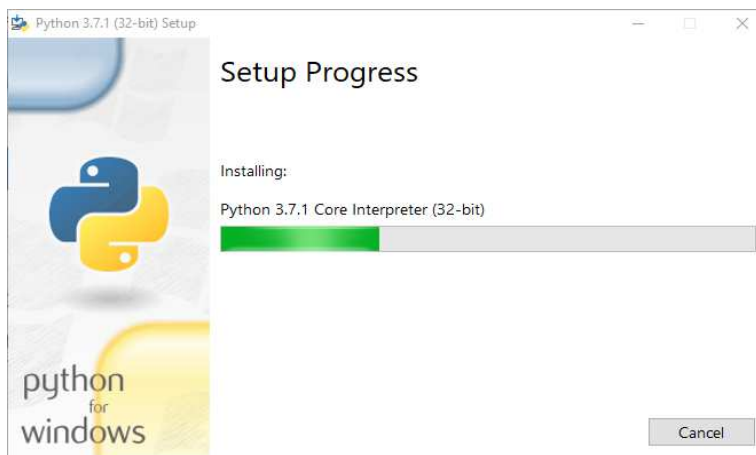
[View older releases](#)

Fig. 1.10: Python Release Versions

Step 3 : Open the python 3.7.1 version pack and double click on it to start installation.

**Fig. 1.11**

Step 4 : Click on next install now for installation.

**Fig. 1.12**

Step 5 : After complete the installation close the windows.

**Fig. 1.13**

Starting Python in different Modes:

1. Starting Python (Command Line):

- A Python script can be executed at command line also.
- This can be done by invoking the interpreter on the application. In command line mode, we type the Python programming program on the Python shell and the interpreter prints the result. The steps are:

Step 1 : Press Start button, (See Fig. 1.14).

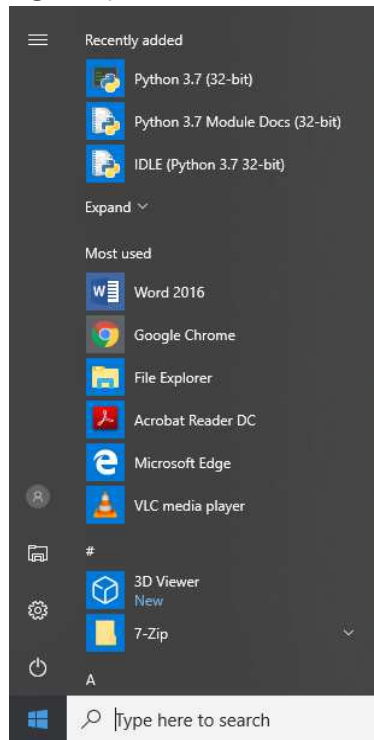


Fig. 1.14

Step 2 : Click on All Programs and then click on Python 3.7 (32 bit). We will see the Python interactive prompt in Python command line.

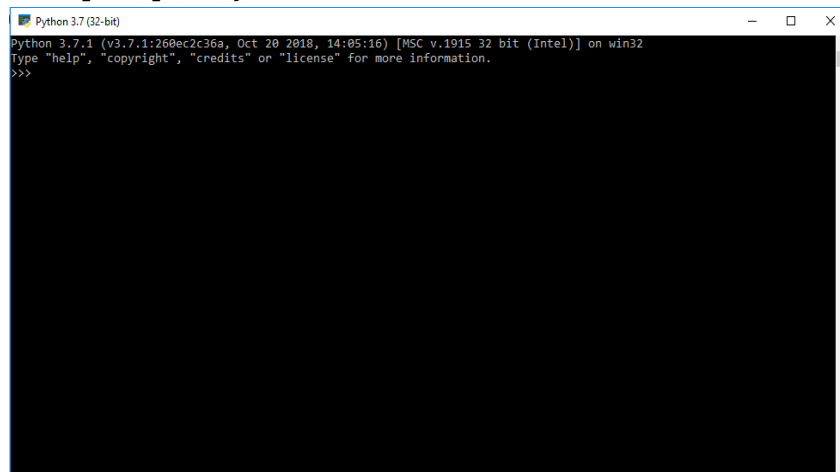
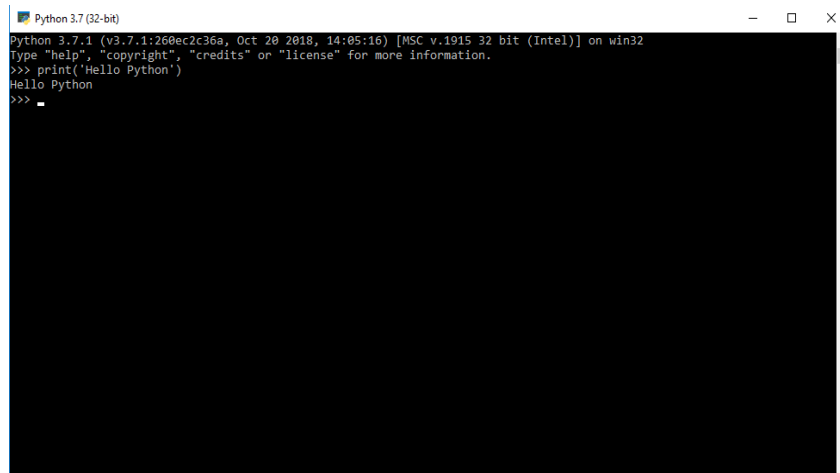


Fig. 1.15

Python command prompt contains an opening message >>> called command prompt. The cursor at command prompt waits for to enter Python command. A complete command is called a statement. For example check first command to print message.

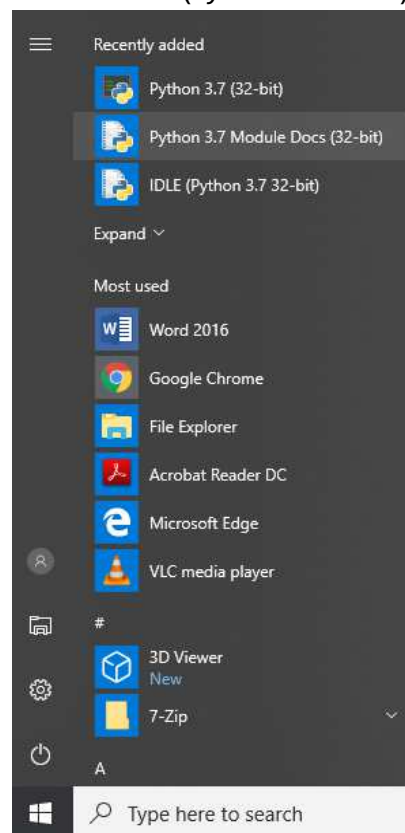
**Fig. 1.16**

Step 3 : To exit from the command line of Python, use Ctrl+z or quit() followed by Enter.

2. Starting Python IDLE:

- When we install Python 3, we also get IDLE (Integrated Development Environment). IDLE includes a color syntax-highlighting editor, a debugger, the Python Shell, and a complete copy of Python 3's online documentation set.
- The steps are:

Step 1 : Press Start button and click on IDLE (Python 3.7 32 bit) options.

**Fig. 1.17**

Step 2 : We will see the Python interactive prompt i.e. interactive shell.

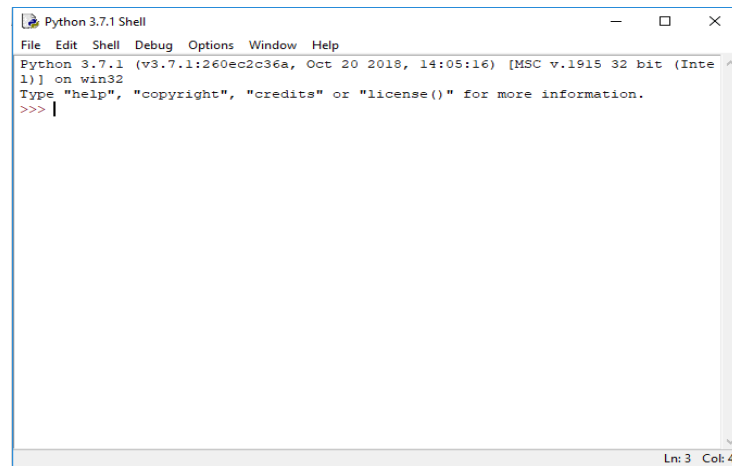


Fig. 1.18

Python interactive shell prompt contains opening message >>>, called shell prompt. A cursor is waiting for the command. A complete command is called a statement. When we write a command and press enter, the python interpreter will immediately display the result.

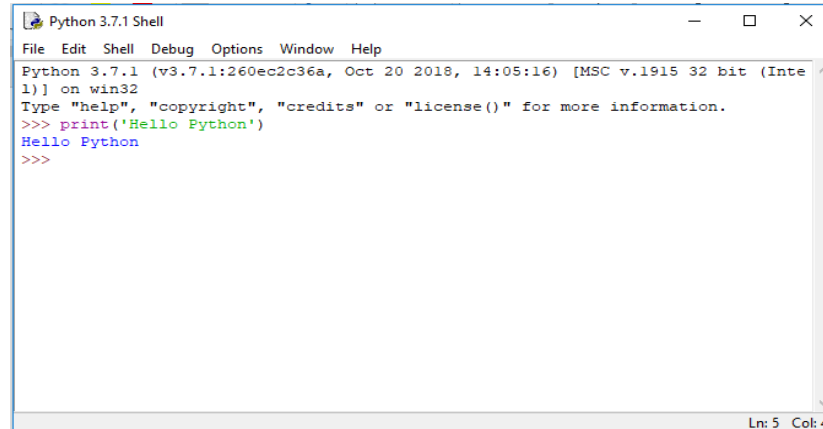


Fig. 1.19

Executing Python Programs:

- In Python IDLEs shell window, click on File, and select the New File or press Ctrl+N.

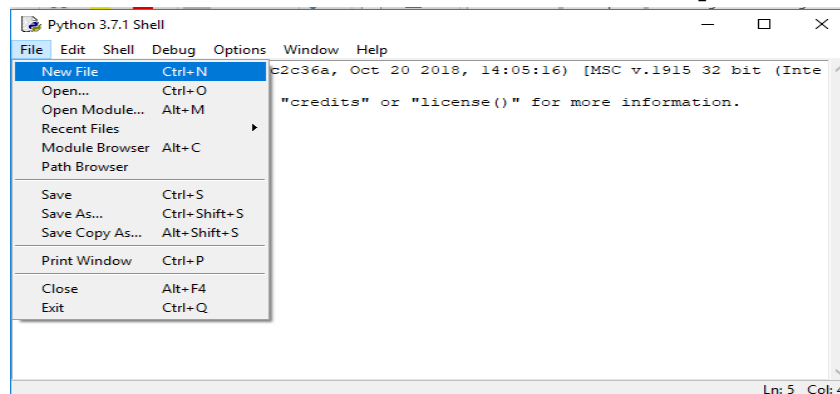
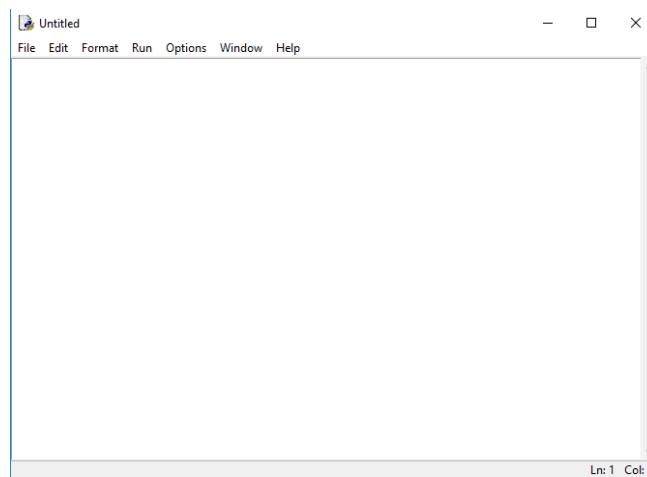
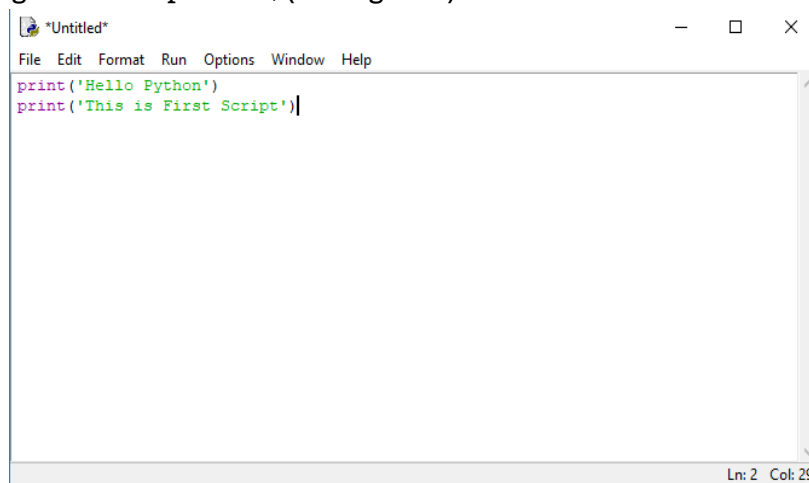


Fig. 1.20

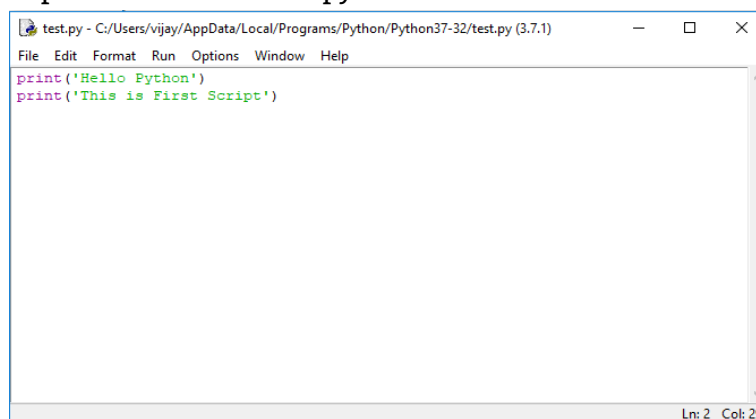
- As soon as you click on New File, the window appears as shown in fig. 1.21.

**Fig. 1.21**

- Write Python program in script mode, (See Fig. 1.22).

**Fig. 1.22**

- Save the above code with filename. By default, python interpreter will save it using the filename.py. Here, we save the script with file name test.py.

**Fig. 1.23**

- To run the python program, click on Run and then Run Module option or we can press Ctrl+F5.

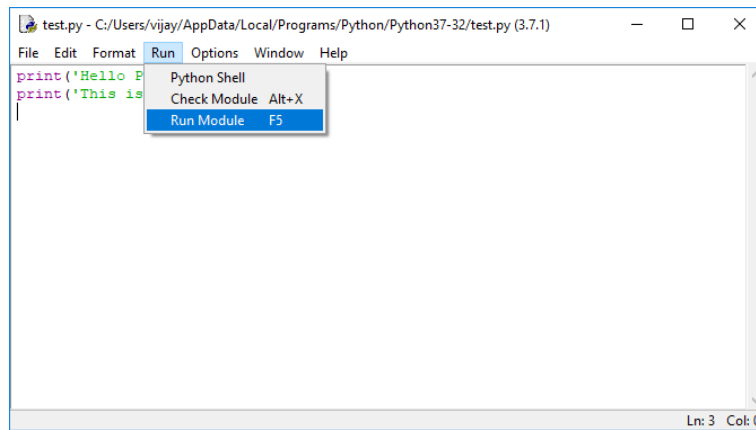


Fig. 1.24

- After clicking Run Module, we will get the output of program.

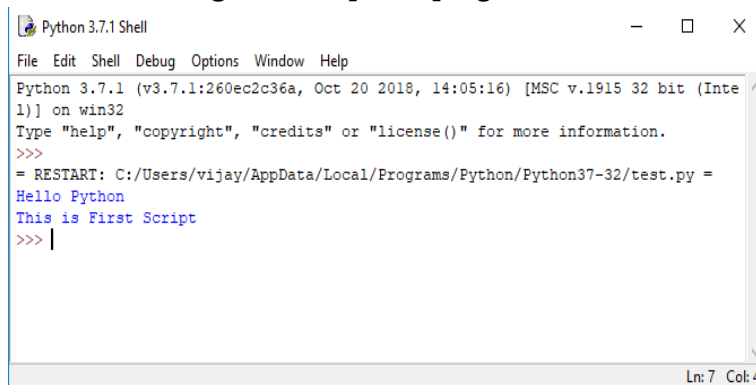


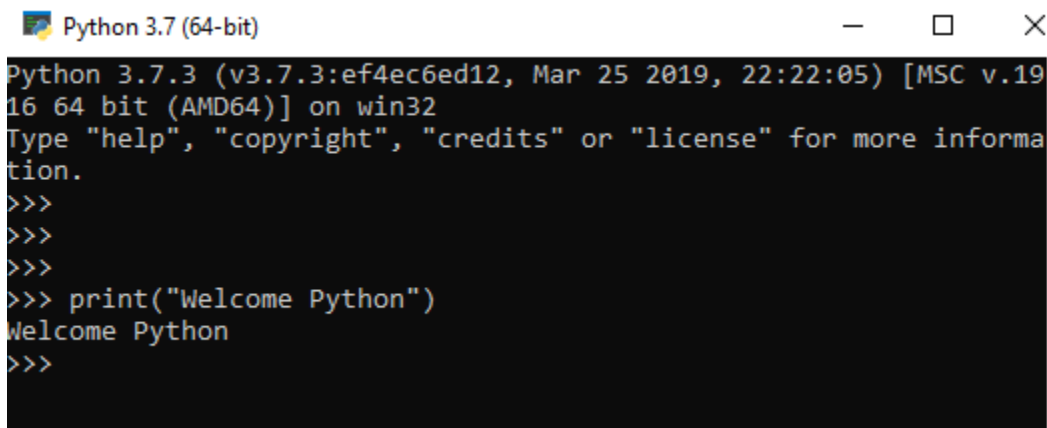
Fig. 1.25

1.4 RUNNING SIMPLE PYTHON SCRIPTS TO DISPLAY 'WELCOME' MESSAGE

- There are two modes for executing Python program namely Interactive mode programming and Script mode programming.
- In **interactive mode programming**, interpreter is invoked and the programmer can code statements directly to the interpreter without passing a script file as a parameter.
- In **script mode programming**, the complete script is written in an editor such as Notepad in Windows and then interpreter is invoked with a script parameter. It begins execution of the script and continues until the script is finished.

1. Interactive Mode Programming:

- Click on All Programs and then click on Python 3.7 (32 bit). We will see the Python interactive prompt in Python command line.
- This method invokes the interpreter without passing a script file and brings up the following prompt, (See Fig. 1.26).



```
Python 3.7 (64-bit)
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 22:22:05) [MSC v.19
16 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more informa
tion.
>>>
>>>
>>>
>>> print("Welcome Python")
Welcome Python
>>>
```

Fig. 1.26

2. Script Mode Programming:

- Invoking the interpreter with a script parameter begins execution of the script and continues until the script is finished. When the script is finished, the interpreter is no longer active.
- Let us write a simple Python program in a script. Python files have extension .py. Type the following source code in a test.py file:

```
print"Welcome, Python!"
```

- We assume that you have Python interpreter set in PATH variable. Now, try to run this program as follows:

```
$ python test.py
```

Output:

```
Welcome, Python!
```

- On Linux OS to execute a Python script modified test.py file:

```
#!/usr/bin/python
print"Hello, Python!"
```

- We assume that we have Python interpreter available in /usr/bin directory. Now, try to run this program as follows:

```
$ chmod +x test.py      # This is to make file executable
$ ./test.py
Hello, Python!
```

1.5 PYTHON DATA TYPES

- The type of data value that can be stored in an identifier/variable such as variable is known as its data type.
- The data type determines how much memory is allocated to store data and what operations can be performed on it.
- The data stored in memory can be of many types and are used to define the operations possible on them and the storage method for each of them.
- Python handles several data types to facilitate the needs of programmers and application developers for workable data.

Declaration and Use of Data Types:

- One of the main differences between Python and strongly-typed languages like C, C++ or Java is the way it deals with types. In strongly-typed languages every variable must have a unique data type.
- For example, if a variable is of type integer, solely integers can be saved in the variable. In Java or C, every variable has to be declared before it can be used. Declaring a variable means binding it to a data type.
- Declaration of variables is not required in Python. If there is need of a variable, we think of a name and start using it as a variable.

- In the following line of code, we assign the value 42 to a variable:

```
i = 42
```

- The equal "=" sign in the assignment shouldn't be seen as "is equal to". It should be "read" or interpreted as "is set to", meaning in our example "the variable i is set to 42". Now we will increase the value of this variable by 1:

```
>>> i = i + 1
>>> print i
43
>>>
```

- Python has various standard data types that are used to define the operations possible on them and the storage method for each of them. Data types in Python programming includes:
 1. **Numbers:** Represents numeric data to perform mathematical operations.
 2. **String:** Represents text characters, special symbols or alphanumeric data.
 3. **List:** Represents sequential data that the programmer wishes to sort, merge etc.
 4. **Tuple:** Represents sequential data with a little difference from list.
 5. **Dictionary:** Represents a collection of data that associate a unique key with each value.
 6. **Boolean:** Represents truth values (true or false).

1.5.1 Numbers DataType

- Number data types store numeric values. Number objects are created when we assign a value to them.
- In Python, there are three distinct numeric types: Integers, floating point numbers and complex numbers falls under Python numbers category. They are defined as int, float and complex class in Python.

1. Integers:

- An int data type represents an integer number. An integer number is a number without any decimal or fractional point.
- For example, a = 57, here a is called the int type variable and stores integer value 57.
- These represent numbers in the range – 2147483648 to 2147483647.
- **Floating Point Numbers:** The float data type represents the floating point number. The floating point number is a number that contains a decimal point. For example, 0.5, –3.445, 330.44 and so on are called floating point numbers. For example, num = 2.345.

Complex Numbers:

- A complex number is a number that is written in the form of a+bi. Here, a represents the real part of the number and b represents the imaginary part of the number.
- The suffix J or j after b represents the square root value of –1. The part a and b may contain the integers or floats. For example, 3+5j, 0.2+10.5j are complex numbers.
- For example, in C=–1–5.5j, the complex number is –1–5.5j and is assigned to the variable C. Hence, the Python interpreter takes the data type of the variable C as a complex type.

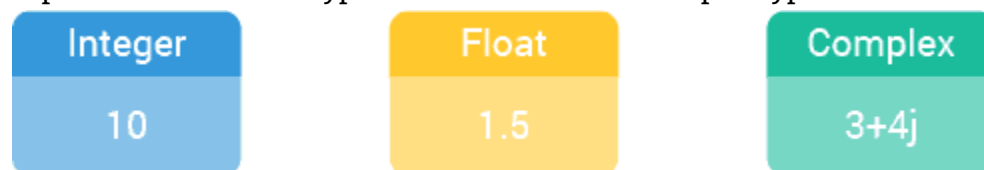


Fig. 1.27: Types of Numbers Data Type

- We can use the type() function to know which class a variable or a value belongs to and the isinstance() function to check if an object belongs to a particular class.

1. Integers (int Data Type):

- An integer is a whole number that can be positive (+) or negative (-). Integers can be of any length, it is only limited by the memory available.

Example: For number data types are integers.

```
>>>a=10
>>>a
10
>>>b=-10
>>>b
-10
```

To determine the type of a variable type() function is used.

```
>>>type(a)
<class 'int' >
```

- In Python 3, there is no limit to how long an integer value can be. It can grow to have as many digits as the computer's memory space allows.
- In Python programming one can write integers in Hexadecimal (base 16), Octal (base 8) and Binary (base 2) formats by using one of the following prefixes to the integer.

Sr. No.	Prefix	Interpretation	Base
1.	'0b' or '0B'	Binary	2
2.	'0o' or '0O'	Octal	8
3.	'0x' or '0X'	Hexadecimal	16

Example: Integers in binary, octal and hexadecimal formats.

```
>>>print(0b10111011) # binary number
187
>>>print(0o10)      # octal number
8
>>>print(0xFF)      # hexadecimal number
255
```

2. Boolean (Bool Data Type):

- In addition, Boolean is a sub-type of integers. The simplest build-in type in Python is the bool type, it represents the truth values False and True. Internally the true value is represented as 1 and false is 0.
- The Bool type can only take the two values True or False, which are 1 or 0 in numeric context.
- For example, a = 18 > 5 assign True value to a and creates a as a Boolean variable.

Example: For Bool data type.

```
>>> x=True
>>> type(x)
<class 'bool'>
>>> y=False
>>> type(y)
<class 'bool'>
>>> 3==4
False
>>> 2<3
True
>>>
```

3. Floating-Point/Float Numbers (Float Data Type):

- Floating-point number or Float is a positive or negative number with a fractional part.
- A floating point number is accurate up to 15 decimal places. Integer and floating points are separated by decimal points. 1 is integer, 1.0 is floating point number.
- One can append the character e or E followed by a positive or negative integer to specify scientific notation.

Example: Floating point number.

```
>>>x = 10.1
>>>x
10.1
y = -10.5
>>>y
-10.5
>>> print(72e3)
72000.0
print(7.2e-3)
0.0072
```

4. Complex Numbers (Complex Data Type):

- Complex numbers are written in the form, $x + yj$, where x is the real part and y is the imaginary part.

Example: Complex number.

```
>>>x = 3+4j
>>>print(x.real)
3.0
>>>print(x.imag)
4.0
```

1.5.2 String Data Type

- String is a collection of group of characters. Strings are identified as a contiguous set of characters enclosed in single quotes (') or double quotes (").
- Any letter, a number or a symbol could be a part of the string. Strings are unchangeable (immutable). Once a string is created, it cannot be modified.
- Strings in python supports Unicode characters. The default encoding for Python source code is UTF-8. So, we can also say that String is a sequence of Unicode characters.
- Strings are ordered. Strings preserved the order of characters inserted.

Example: For string data type.

```
>>> s1="Hello"           #string in double quotes
>>> s2='Hi'              #string in single quotes
>>> s3="Don't open the door"  #single quote string in double quotes
>>> s4='I said "yipee"'      #double quote string in single quotes
>>> s1
'Hello'
>>> s2
'Hi'
>>> s3
'Don't open the door'
>>> s4
'I said "yipee"'
>>>
```


- We can convert almost any object in Python to a string using a type constructor called `str()` function.
-

Example: For string with `str()`.

```
>>> S=str(42)
>>> S
'42'
>>> type(S)
<class 'str'>
>>>
```

Accessing the String:

- Individual characters in a string can be accessed using an index. Subsets of strings can be taken using the slice operation (`[]` and `[i]`) with index starting at 0 in the beginning of the string and -1 at the end.
-

Example: For accessing string.

```
>>> s="Hello Python"
>>> s[0]                # get element at index 0
'H'
>>> s[-1]              # get element at last index
'n'
>>> s[1:4]             # get element from m index to n-1 index.
'ell'
>>> s[6:]              # get element from m index to last index.
' Python'
>>> s[:5]              # get element from 0th index to n-1 index.
'Hello'
>>> s[1:12:2]          # get element from m index to n-1 index with i increments
'el Pto'
>>> s+" Programming"   # It will concatenate string
'Hello Python Programming'
>>> type(s)
<class 'str'>
>>> s*3                #It will repeat the string
'Hello PythonHello PythonHello Python'
>>>
```

Split and Join a String:

- `split()` function breaks a string into individual letters. Use `split()` method to chop up a string into a list of substrings, around a specified delimiter.
-

Example: For `split()` function.

```
>>> s="Python programming is easy"
>>> s
'Python programming is easy'
>>> l=s.split()         # split() without any argument
>>> l
['Python', 'programming', 'is', 'easy']
>>> s="Python,programming,is,easy"
>>> s
'Python,programming,is,easy'
```

```
>>> l=s.split(',')          # split() with any argument
>>> l
['Python', 'programming', 'is', 'easy']
>>>
>>> type(l)
<class 'list'>
```

- Use join() method to join the list back into a string, with a specified delimiter in between.

Example: For join() function/method.

```
>>> l
['Python', 'programming', 'is', 'easy']
>>>
>>> type(l)
<class 'list'>
>>> s=' '.join(l)
>>> s
'Python programming is easy'
>>> s='.'.join(l)
>>> s
'Python.programming.is.easy'
>>> type(s)
<class 'str'>
```

String Built-in Methods:

- String objects also have several useful methods to report various characteristics of the string, such as whether it consists of digits or alphabetic characters or is all uppercase or lowercase.

Sr. No.	String Operation	Explanation	Example
1.	+	Adds two strings together.	X = "hello"+"world"
2.	*	Replicates a string.	X = " "*20
3.	upper	Converts a string to uppercase	x.upper()
4.	lower	Converts a string to lowercase	x.lower()
5.	title	Capitalizes the first letter of each word in a string.	x.title()
6.	find, index	Searches for the target in a string.	x.find(y) x.index(y)
7.	rfind, rindex	Searches for the target in a string, from the end of the string.	x.rfind(y) x.rindex(y)
8.	startswith, endswith	Checks the beginning or end of a string for a match.	x.startswith(y) x.endswith(y)
9.	replace	Replaces the target with a new string.	x.replace(y,z)
10.	strip,rstrip,lstrip	Removes whitespace or other characters from the ends of a string.	x.strip()
11.	encode	Converts a Unicode string to a bytes object.	x.encode("utf_8")

1.5.3 List Data Type

- List is an ordered sequence of items. It is one of the most used datatype in Python and is very flexible.
- List can contain heterogeneous values such as integers, floats, strings, tuples, lists and dictionaries but they are commonly used to store collections of homogeneous objects.
- The list datatype in Python programming is just like an array that can store a group of elements and we can refer to these elements using a single name.
- Declaring a list is pretty straight forward. Items separated by commas (,) are enclosed within brackets [].

Example: For list.

```
>>> first=[10, 20, 30]                # homogenous values in list
>>> second=["One","Two","Three"]      # homogenous values in list
>>> first
[10, 20, 30]
>>> second
['One', 'Two', 'Three']
>>> third=[10,"one",20,"two"]          # heterogeneous values in list
>>> third
[10, 'one', 20, 'two']
>>> first + second                     # prints the concatenated lists
[10, 20, 30, 'One', 'Two', 'Three']
```

- Lists are mutable which means that value of elements of a list can be altered.

Example: For list with updation/alteration/modification.

```
>>> first=[10, 20, 30]
>>> first[2]
30
>>>first[2]=50
>>> first
[10, 20, 50]
>>> first[2]                          # print second value in the list
50
>>> print (first*2)                   # prints the list two times
[10, 20, 30, 10, 20, 30]
```

List and Strings:

- A string is a sequence of characters and list is a sequence of values, but a list of characters is not same as string. We can convert string to a list of characters.

Example: For conversion of string to a list.

```
>>> p="Python"
>>> p
'Python'
>>> l=list(p)
>>> l
['P', 'y', 't', 'h', 'o', 'n']
```

1.5.4 Tuple Data Type

- Tuple is an ordered sequence of items same as list. The only difference is that tuples are immutable. Tuples once created cannot be modified.
- Tuples are used to write-protect data and are usually faster than list as it cannot change dynamically. It is defined within parentheses () where items are separated by commas (,).

- A tuple data type in python programming is similar to a list data type, which also contains heterogeneous items/elements.

Example: For tuple.

```
>>> a=(10, 'abc', 1+3j)
>>> a
(10, 'abc', (1+3j))
>>> a[0]
10
>>> a[0]=20
Traceback (most recent call last):
  File "<pyshell#12>", line 1, in <module>
    a[0]=20
TypeError: 'tuple' object does not support item assignment
```

1.5.5 Dictionary

- Dictionary is an unordered collection of key-value pairs. It is the same as the hash table type.
- The order of elements in a dictionary is undefined, but we can iterate over the following:
 - The key
 - The value
 - The items (key-value pairs) in a dictionary.
- When we have the large amount of data, the dictionary data type is used. The dictionary data type is mutable in nature which means we can update modify update any value in the dictionary.
- Items in dictionaries are enclosed in curly braces { } and separated by the comma (.). A colon (:) is used to separate key from value. Values can be assigned and accessed using square braces ([]).

Example: For dictionary data type.

```
>>> dic1={1:"First", "Second":2}
>>> dic1
{1: 'First', 'Second': 2}
>>> type(dic1)
<class 'dict'>
>>> dic1[3]="Third"
>>> dic1
{1: 'First', 'Second': 2, 3: 'Third'}
>>> dic1.keys()
dict_keys([1, 'Second', 3])
>>> dic1.values()
dict_values(['First', 2, 'Third'])
>>>
```

Input and Output in Python Programming:

- Input means the data entered by the user of the program. In python, the input() function is used to accept an input from a user. The raw_input () function available for Input on older version.

Syntax: variable_name=input()

variable_name=input('String')

Example: For input in Python.

```
>>> input()
Hello python
'Hello python'
```

```
>>>x= input ("Enter data:")
Enter data:  11.22
>>>print(x)
11.22
```

- Output means the data comes from computer after processing. In Python programming the print() function display the input value on screen.

Syntax: print(expression/constant/variable)

Example: For output in python.

```
>>> print ("Hello")
Hello
>>> a="Hello"
>>> b="Python"
>>> print(a+b)
```

Output:

```
HelloPython
```

Additional Programs:

1. Program to find the square root of a number.

```
x=int(input("Enter an integer number:"))
ans=x**0.5
print("Square root= ", ans)
```

Output:

```
Enter an integer number: 144
Square root=  12.0
```

2. Program to find the area of Rectangle.

```
l=float(input("Enter length of the rectangle: "))
b=float(input("Enter breadth of the rectangle: "))
area=l*b
print("Area of Rectangle= ",area)
```

Output:

```
Enter length of the rectangle: 5
Enter breadth of the rectangle: 6
Area of Rectangle=  30.0
```

3. Program to calculate area and perimeter of the square.

```
int(input("Enter side length of square: "))
area=side*side
perimeter = 4*side
print("Area of Square =", area)
print("Perimeter of Square =", perimeter)
```

Output:

```
Enter side length of square: 5
Area of Square = 25
Perimeter of Square = 20
```

4. Program to calculate surface volume and area of a cylinder.

```
pi=22/7
height = float(input('Height of cylinder: '))
```

```
radian = float(input('Radius of cylinder: '))
volume = pi * radian * radian * height
sur_area = ((2*pi*radian) * height) + ((pi*radian**2)*2)
print("Volume is: ", volume)
print("Surface Area is: ", sur_area)
```

Output:

```
Height of cylinder: 4
Radius of cylinder: 6
Volume is: 452.57142857142856
Surface Area is: 377.1428571428571
```

5. Program to swap the value of two variables.

```
num1=input("Enter first value: ")
num2=input("Enter second value: ")
print("Numbers before swapping")
print("num1= ",num1)
print("num2= ",num2)
temp=num1
num1=num2
num2=temp
print("Numbers after swapping")
print("num1= ",num1)
print("num2= ",num2)
```

Output:

```
Enter first value: 10
Enter second value: 20
Numbers before swapping
num1= 10
num2= 20
Numbers after swapping
num1= 20
num2= 10
```

Practice Questions

1. What is Python programming language?
2. Give short history for Python.
3. Enlist applications for Python programming.
4. What are the features of Python?
5. List any four editors used for Python programming.
6. 'Python programming language is interpreted and interactive' comment this sentence.
7. How to run python scripts? Explain in detail.
8. What is interpreter? How it works?
9. Explain the following features of Python programming:
 - (i) Simple
 - (ii) Platform independent
 - (iii) Interactive
 - (iv) Object Oriented.
10. Explain about the need for learning Python programming and its importance.

11. Describe the internal working of Python diagrammatically.
12. Write in brief about characters set of Python.
13. Write in brief about any five keywords in Python.
14. Write the steps to install Python and to run Python code.
15. What is the role of indentation in Python?
16. How to comment specific line(s) in Python program?
17. What is variable? What are the rules and conventions for declaring a variables?
18. What are the various data types available in Python programming.
19. What are four built-in numeric data types in Python? Explain.
20. What is the difference between interactive mode and script mode of Python.
21. Python has developed as an open source project. Justify this statement.
22. Define the following terms:
 - (i) Identifier
 - (ii) Literal
 - (iii) Data type
 - (iv) Tuple
 - (v) List.
23. Explain dictionary data type in detail.

