A Text Book Of

PYTHON PROGRAMMING

(COURSE CODE : 314004)

Semester - IV

SECOND YEAR DIPLOMA IN COMPUTER ENGINEERING GROUP
(BD / CM / CO / CW / HA / IF / IH / SE)

As Per MSBTE's 'K' Scheme Syllabus w.e.f. Academic Year 2024

Dr. Mrs. Meenakshi A. Thalor
Ph.D. (Computer Engg.), M.E. (IT), B.Tech. (IT), LMISTE
Associate Professor, Head Dept. of Information Technology
A.I.S.S.M.S. IOIT, Pune

Vijay T. Patil
M.E. (Computer Engg.)
Head, Dept. of Computer Engineering
Vidyalankar Polytechnic
Wadala (E), Mumbai

Prashant D. Somwanshi
M.E. (Comp. Engg.)
Sr. Software Engineer, Pune
Former, Asst. Professor, Dept. of Computer Engineering
JSPM & TSSM Group of Institutes, Pune

Price ₹ 210.00

NIRALI PRAKASHAN
ADVANCEMENT OF KNOWLEDGE

N8201

PYTHON PROGRAMMING **ISBN 978-93-6109-408-8**

First Edition : January 2025
© : Authors

The text of this publication, or any part thereof, should not be reproduced or transmitted in any form or stored in any computer storage system or device for distribution including photocopy, recording, taping or information retrieval system or reproduced on any disc, tape, perforated media or other information storage device etc., without the written permission of Authors with whom the rights are reserved. Breach of this condition is liable for legal action.

Every effort has been made to avoid errors or omissions in this publication. In spite of this, errors may have crept in. Any mistake, error or discrepancy so noted and shall be brought to our notice shall be taken care of in the next edition. It is notified that neither the publisher nor the authors or seller shall be responsible for any damage or loss of action to any one, of any kind, in any manner, therefrom. The reader must cross check all the facts and contents with original Government notification or publications.

Published By :
NIRALI PRAKASHAN
Abhyudaya Pragati, 1312, Shivaji Nagar,
Off J.M. Road, Pune – 411005
Tel - (020) 25512336/37/39
Email : niralipune@pragationline.com

Polyplate

Printed By :
YOGIRAJ PRINTERS AND BINDERS
Survey No. 10/1A, Ghule Industrial Estate
Nanded Gaon Road
Nanded, Pune - 411041

DISTRIBUTION CENTRES

PUNE

Nirali Prakashan
(For orders outside Pune)
S. No. 28/27, Dhayari Narhe Road, Near Asian College
Pune 411041, Maharashtra
Tel : (020) 24690204; Mobile : 9049672225
Email : bookorder@pragationline.com

Nirali Prakashan
(For orders within Pune)
119, Budhwar Peth, Jogeshwari Mandir Lane
Pune 411002, Maharashtra
Mobile : 9657703145, 9890997937
Email : niralilocal@pragationline.com

MUMBAI

Nirali Prakashan
Rasdhara Co-op. Hsg. Society Ltd., 'D' Wing Ground Floor, 385 S.V.P. Road
Girgaum, Mumbai 400004, Maharashtra
Mobile : 7045821020, Tel : (022) 2385 6339 / 2386 9976
Email : niralimumbai@pragationline.com

DISTRIBUTION BRANCHES

DELHI

Nirali Prakashan
Room No. 2 Ground Floor
4575/15 Omkar Tower, Agarwal Road Darya Ganj, New Delhi 110002
Mobile : 9555778814/9818561840
Email : delhi@niralibooks.com

BENGALURU

Nirali Prakashan
Maitri Ground Floor, Jaya Apartments, No. 99, 6th Cross, 6th Main, Malleswaram, Bengaluru 560003 Karnataka; Mob : 9686821074
Email : bengaluru@niralibooks.com

NAGPUR

Nirali Prakashan
Above Maratha Mandir, Shop No. 3, First Floor, Rani Jhanshi Square, Sitabuldi Nagpur 440012 (MAH)
Tel : (0712) 254 7129
Email : nagpur@niralibooks.com

KOLHAPUR

Nirali Prakashan
438/2, Bhosale Plaza, Ground Floor
Khasbag, Opp. Balgopal Talim
Kolhapur 416 012, Maharashtra
Mob : 9850046155
Email : kolhapur@niralibooks.com

JALGAON

Nirali Prakashan
34, V. V. Golani Market, Navi Peth, Jalgaon 425001, Maharashtra
Tel : (0257) 222 0395
Mob : 94234 91860
Email : jalgaon@niralibooks.com

SOLAPUR

Nirali Prakashan
R-158/2, Avanti Nagar, Near Golden Gate, Pune Naka Chowk
Solapur 413001, Maharashtra
Mobile 9890918687
Email : solapur@niralibooks.com

marketing@pragationline.com | www.niralibooks.com

Also find us on www.facebook.com/niralibooks

Preface ...

We take an opportunity to present this book entitled as **"Python Programming"** to the students of Fourth Semester (Second Year Diploma in Computer Engineering Group (BD / CM / CO / CW / HA / IF / IH / SE). The book is written strictly according to the Revised Syllabus (K-Scheme) prepared by Maharashtra State Board of Technical Education (MSBTE).

The book has its own unique features. It brings out the subject in a very simple and lucid manner for easy and comprehensive understanding of the basic concepts. The book covers theory of Introduction to Python and Control Flow Statements, Data Structures in Python, Functions, Modules and Packages in Python, Object Oriented Programming in Python and Introduction to Built-in Packages in Python.

A special word of thank to Shri. Dineshbhai Furia, Mr. Jignesh Furia for showing full faith in us to write this text book. We also thank to Mr. Amar Salunkhe and Mr. Akbar Shaikh of M/s Nirali Prakashan for their excellent co-operation.

We also thank Ms. Chaitali Takale, Mr. Ravindra Walodare, Mr. Sachin Shinde, Mr. Nilesh Deshmukh, Mr. Ashok Bodke, Mr. Moshin Sayyed and Mr. Nitin Thorat.

We welcome constructive suggestions from our colleagues and students.

Authors

■■■

Syllabus ...

1. Introduction to Python and Control Flow Statements

1.1 Introduction - Features, History and Applications of Python, Python IDE's

1.2 Python Building Blocks - Indentation, Identifiers, Variable, Comments, Keywords

1.3 Basic Input Output Operations - input(), print()

1.4 Operators - Arithmetic, Relational, Assignment, Logical, Bitwise, Membership and Identity Operator

1.5 Control Flow Statements - Conditional Statements (if, if-else, if-elif-else, nested if), Loops in Python (while, for, Nested Loops), Loop Manipulation Statements (continue, pass, break, else)

2. Data Structures in Python

2.1 List:

(a) Defining Lists, Accessing Values from List, Deleting List Values, Updating Lists

(b) Basic List Operations

(c) Built-in List Functions/Methods

2.2 Tuple:

(a) Defining Tuple, Accessing Values from Tuple

(b) Basic Tuple Operations

(c) Built-in Tuple Functions/Methods

2.3 Set:

(a) Defining Sets, Accessing Values from Set, Deleting Set Values

(b) Basic Set Operations

(c) Built-in Set Functions/Methods

2.4 Dictionary:

(a) Defining Dictionary, Accessing Values from Dictionary, Deleting Dictionary Values, Updating Dictionary

(b) Basic Dictionary Operations

(c) Built-in Dictionary Functions/Methods

3. Functions, Modules and Packages in Python

3.1 Functions - Defining Function, Calling Function, Function Arguments, Return Statement, Scope of Variable, Lambda Functions

3.2 Modules - Create User Defined Module, Importing a Module, Using Python Built-in Modules, Namespace and Scoping

3.3 Python Packages - Create User Defined Package, Importing a Package, Using Python Built-in Packages, Installing Packages using PIP

4. Object Oriented Programming in Python

4.1 Object Oriented Concepts - Creating Class, Creating Object

4.2 Constructors in Python (Parameterized and Non-Parameterized), the Self Parameter

4.3 Polymorphism - Method Overloading and Overriding

4.4 Data Hiding / Abstraction

4.5 Inheritance - Single Inheritance, Multiple Inheritance, Multilevel Inheritance

5. Introduction to Built-in Packages in Python

5.1 Pandas - Use of Pandas, Pandas Series, Pandas DataFrames, Pandas Read CSV

5.2 Creating GUI using Tkinter - Introduction to Tkinter, Widgets (Entry, Label, Button, RadioButton, Checkbutton), Creating a Simple GUI Application

5.3 Connecting to Database using MySQL - Installing MySQL - Connector, cursor( ) Object, execute( ) Method, fetchall( ) Method, Creating Simple Program to Connect Database

■■■

Contents ...

1. Introduction to Python and Control Flow Statements	**1.1 – 1.40**
2. Data Structures in Python	**2.1 – 2.34**
3. Functions, Modules and Packages in Python	**3.1 – 3.40**
4. Object Oriented Programming in Python	**4.1 – 4.16**
5. Introduction to Built-in Packages in Python	**5.1 – 5.38**

■■■

...1...

Introduction to Python and Control Flow Statements

Learning Outcomes...

- Explain given feature of python.
- Write python program to perform basic input output operations.
- Write python program to solve given expression.
- Implement given decision-making statements and looping statements in python program.

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.

1.1 BASIC CONCEPTS IN PYTHON

- In this section we will study features, history, Python IDE's and applications of Python programming language.

1.1.1 History of Python

- 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.

1.1.2 Features of Python

- Features of Python's programming language 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).

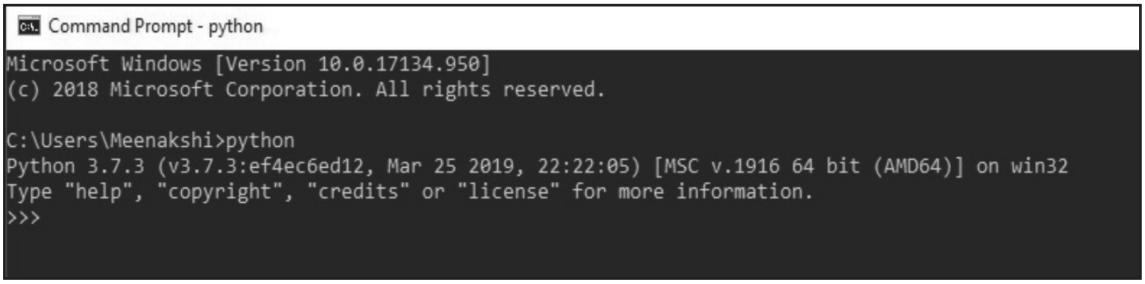


Fig. 1.1: Python Command Prompt

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.

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 for use.
- 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. 1.2 shows execution of Python code by interpreter.

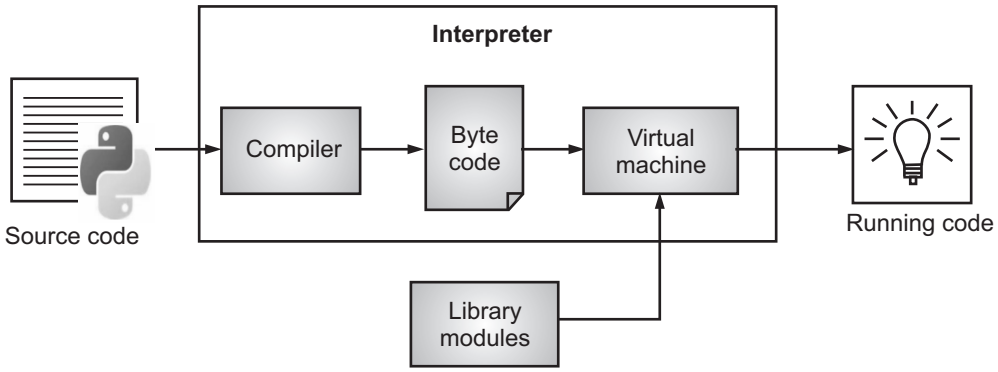


Fig. 1.2: Execution of Python Code

- 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 presents 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 of 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 etc.

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.

1.1.3 Python IDE's

- There are many free and commercial editors available for Python. Following table lists Python editors:

Sr. No.	Editor	Description	Icon/Logo
1.	IDLE	• 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	• 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	• 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	• 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	• 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. 1.3).
- 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.

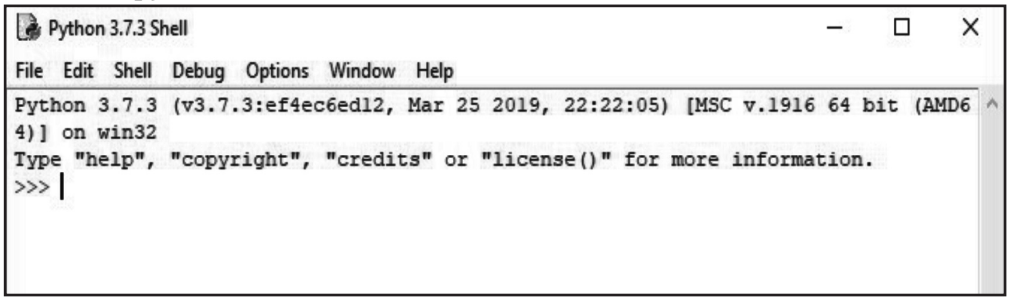


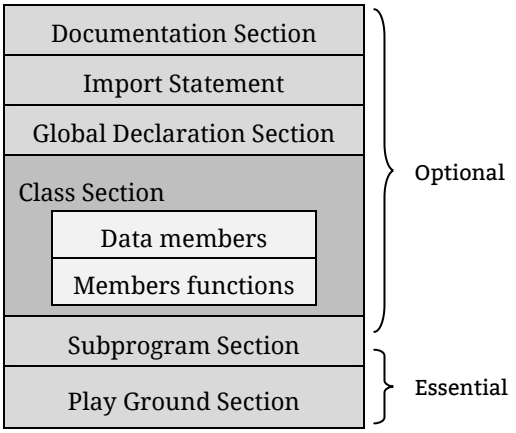
Fig. 1.3: Python Shell

1.1.4 Applications of Python

- Some common applications of Python Programming are listed below:
  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 programming.
  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 (National Security Agency) uses Python programming 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 programming in the production of animated movies.

1.1.5 Structure of a Python Program

- Fig. 1.4 shows a typical program structure of Python programming.
- Python programming programs are structured as a sequence of 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.
- By default, the Python interpreter executes all statements sequentially, but we can change order of execution using control statements.




```
#Python Program Structure. Calculate Area and Circumference
of circle using class.   # Documentation section
import math               # import statement
radius=5                  # global declaration Section
class Circle():           # class section
   def getArea(self):
      return math.pi*radius*radius
   def getCircumference(self):
      return radius*2*math.pi
   def showradius():      # sub Program section
      print("Radius =",radius)
showradius()              # Playground Section
c=Circle()
print("Area =",c.getArea()) print("Circumference
                                  =",c.getCircumference())
```

Output:

```
   Radius = 5
   Area = 78.53981633974483
   Circumference = 31.41592653589793
```

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 comment 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. **Class Section** describes the information about the user defined classes in the Python program. A class is a collection of data members and member functions called method, that operate on 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 form anywhere.
6. **Pay Ground Section** is the main section of Python program and the main section starts where the function calling.

Running Python Scripts:

- Python has two basic modes namely, normal and interactive.
  - The **normal script 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 used for quickly and conveniently running single line 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. For example, if we want to print the statement "Interactive Mode", simply type the appropriate code and hit enter.

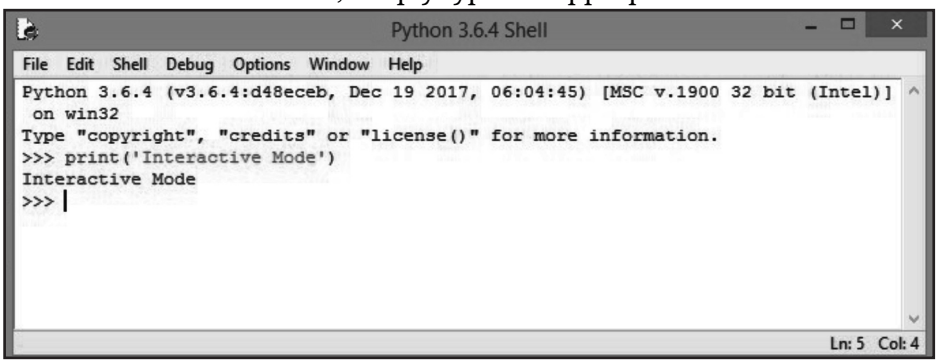


Fig. 1.5

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 to write 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, (See Fig. 1.6 (a) and 1.6 (b)).

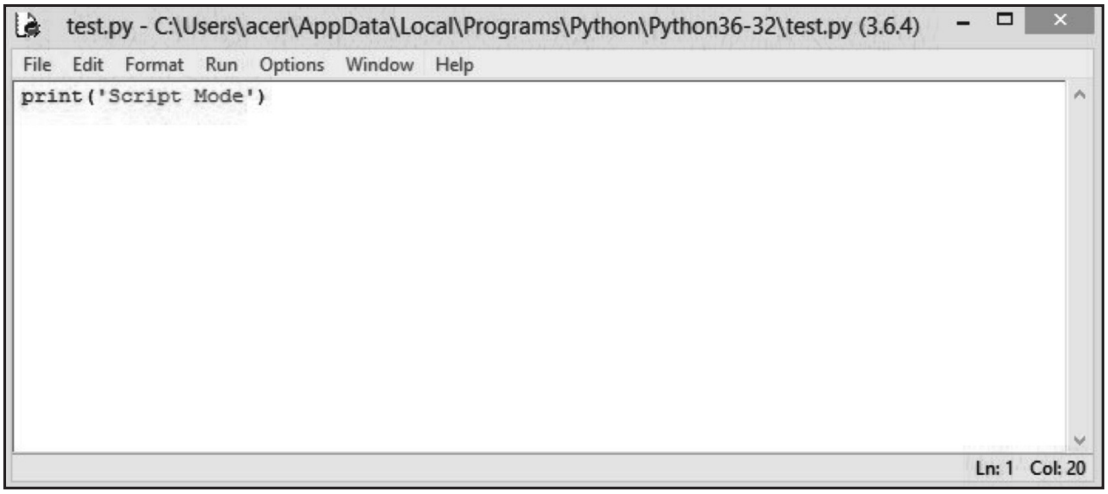


Fig. 1.6 (a)

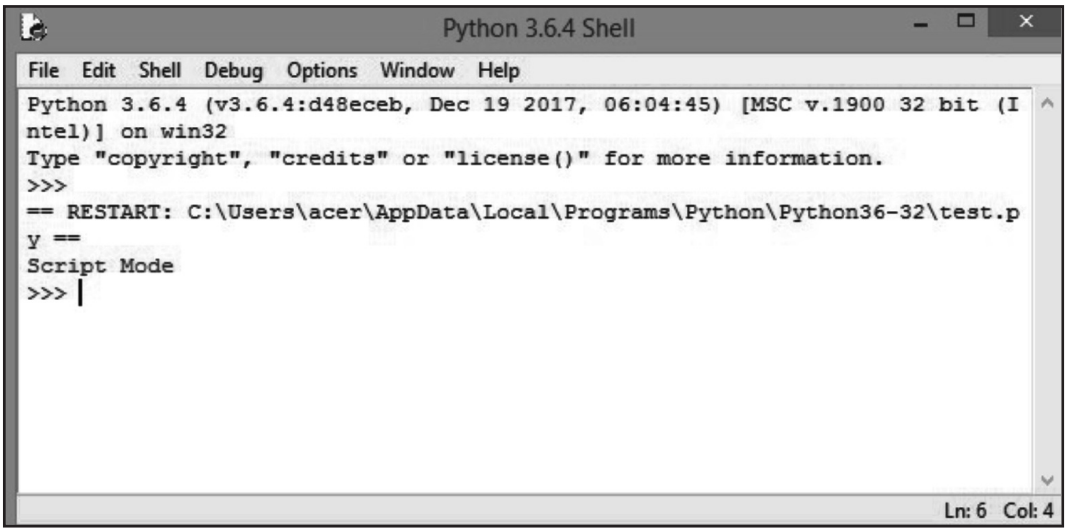


Fig. 1.6 (b)

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.
- 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.

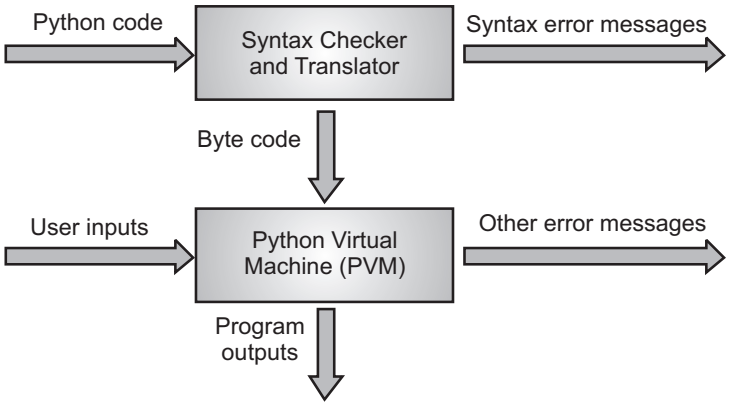


Fig. 1.7: Internal Operations

- 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, identifiers 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**: "$", "!", etc.
  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. For example, Emp_Salary.
  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 predefined 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 differing 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 is defined as three variables 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); Pring (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' # single quote
Lname="Python" #double quote
print(Fname)
print(Lname)
```

Output:

```
Hello
Python
```

2. Numeric Literals:

- Numeric literals are immutable. Numeric literals comprise number or digits form 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 lower-case or uppercase L. Example: 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 namely, True or False.

Example: For Boolean literal.

```
>>> 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     # N is in uppercase here
>>> val1
10
>>> val2
>>> print (val2)
None
>>>
```

5. Value and Type of 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)`