

The Python logo is a large blue shape with a white circle at the top left and a white horizontal bar below it. The word "PYTHON" is written in white, bold, uppercase letters across the middle, and "programming language" is written in a smaller, white, lowercase font below it.

PYTHON

programming language



www.tutorialspoint.com



<https://www.facebook.com/tutorialspointindia>



<https://twitter.com/tutorialspoint>

About the Tutorial

Today, Python is one of the most popular programming languages. Although it is a general-purpose language, it is used in various areas of applications such as Machine Learning, Artificial Intelligence, web development, IoT, and more. This **Python tutorial** is designed to be a self-learning guide for beginners, students looking for a career in software development and Data science. This tutorial shall also be useful for experienced software professionals to enhance their skills.

This Python tutorial is based on the latest Python 3.11.2 version.

What is Python?

Python is a very popular general-purpose interpreted, interactive, object-oriented, and high-level programming language. Python is dynamically-typed and garbage-collected programming language. It was created by Guido van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL).

Python supports multiple programming paradigms, including Procedural, Object Oriented and Functional programming language. Python design philosophy emphasizes code readability with the use of significant indentation.

This tutorial gives a complete understanding of Python programming language starting from basic concepts to advanced concepts. This tutorial will take you through simple and practical approaches while learning Python Programming language.

Python Jobs

Today, Python is very high in demand and all the major companies are looking for great Python Programmers to develop websites, software components, and applications or to work with Data Science, AI, and ML technologies. When we are developing this tutorial in 2023, there is a high shortage of Python Programmers whereas market demands more number of Python Programmers due to its application in Machine Learning, Artificial Intelligence etc.

Today a Python Programmer with 3-5 years of experience is asking for around \$150,000 annual package and this is the most demanding programming language in America. Though it can vary depending on the location of the Job. It's impossible to list all of the companies using Python, to name a few big companies are:

- Google
- Intel
- NASA
- PayPal
- Facebook
- IBM
- Amazon
- Netflix
- Pinterest
- Uber
- Many more...

So, you could be the next potential employee for any of these major companies. We have developed a great learning material for you to learn Python Programming which will help you prepare for the technical interviews and certification exams based on Python. So, start

learning Python using this simple and effective tutorial from anywhere and anytime absolutely at your pace.

Why Learn Python?

Python is consistently rated as one of the world's most popular programming languages. Python is fairly easy to learn, so if you are starting to learn any programming language then Python could be your great choice. Today various Schools, Colleges and Universities are teaching Python as their primary programming language. There are many other good reasons which makes Python as the top choice of any programmer:

- Python is Open Source which means its available free of cost.
- Python is simple and so easy to learn
- Python is versatile and can be used to create many different things.
- Python has powerful development libraries include AI, ML etc.
- Python is much in demand and ensures high salary

Python is a MUST for students and working professionals to become a great Software Engineer specially when they are working in Web Development Domain. I will list down some of the key advantages of learning Python:

- **Python is Interpreted** – Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive** – You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** – Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language** – Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

Python Online Compiler/Interpreter

We have provided **Python Online Compiler/Interpreter** which helps you to **Edit** and **Execute** the code directly from your browser.

You can run the following Python code online to print conventional "Hello, World!".

The following code box allows you to change the value of the code. Try to change the value inside **print()** and run it again to verify the result.

```
# This is my first Python program.  
# This will print 'Hello, World!' as the output  
  
print ("Hello, World!");
```

Careers in Python

If you know Python nicely, then you have a great career ahead. Here are just a few of the career options where Python is a key skill:

- Game developer
- Web designer
- Python developer
- Full-stack developer
- Machine learning engineer
- Data scientist
- Data analyst
- Data engineer
- DevOps engineer
- Software engineer
- Many more other roles

Characteristics of Python

Following are important characteristics of **Python Programming** –

- It supports functional and structured programming methods as well as OOP.
- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- It supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Applications of Python

The latest release of Python is 3.x. As mentioned before, Python is one of the most widely used language over the web. I'm going to list few of them here:

- **Easy-to-learn** – Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read** – Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** – Python's source code is fairly easy-to-maintain.
- **A broad standard library** – Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** – Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

- **Portable** – Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable** – You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases** – Python provides interfaces to all major commercial databases.
- **GUI Programming** – 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.
- **Scalable** – Python provides a better structure and support for large programs than shell scripting.

Audience

This tutorial has been prepared for the beginners to help them understand the basics to advanced concepts of Python programming language. After completing this tutorial, you will find yourself at a great level of expertise in Python programming, from where you can take yourself to the next levels.

Prerequisites

Although it is a beginners' tutorial, we assume that the readers have a reasonable exposure to any programming environment and knowledge of basic concepts such as variables, commands, syntax, etc.

Copyright & Disclaimer

© Copyright 2023 by Tutorials Point (I) Pvt. Ltd.

All the content and graphics published in this e-book are the property of Tutorials Point (I) Pvt. Ltd. The user of this e-book is prohibited to reuse, retain, copy, distribute or republish any contents or a part of contents of this e-book in any manner without written consent of the publisher.

We strive to update the contents of our website and tutorials as timely and as precisely as possible, however, the contents may contain inaccuracies or errors. Tutorials Point (I) Pvt. Ltd. provides no guarantee regarding the accuracy, timeliness or completeness of our website or its contents including this tutorial. If you discover any errors on our website or in this tutorial, please notify us at contact@tutorialspoint.com

Table of Contents

About the Tutorial	i
Audience	iv
Prerequisites	iv
Copyright & Disclaimer	iv
Table of Contents.....	v
PYTHON BASICS	1
1. Python – Overview	2
2. Python – History.....	4
3. Python – Features	6
4. Python vs C++.....	9
5. Python – Hello World	12
6. Python – Application Areas	15
7. Python – Interpreter	18
8. Python – Environment Setup.....	23
Install Python on MacOS.....	26
Install Python from Source Code	27
Python Command Line Options	30
Python Environment Variables	31
9. Python – Virtual Environment	34
10. Python – Basic Syntax	37
Python Keywords.....	37
Python Identifiers	39
Python Indents.....	40
Python Statements	41
11. Python – Variables	43
Naming Convention	44

Assignment Statement	45
Multiple Assignments	46
12. Python – Data Types	49
Number Type	49
Sequence Types	50
Strings in Python	50
List in Python	51
Tuples in Python	52
Dictionary Type.....	53
Set Type	54
13. Python – Type Casting	56
Implicit Casting in Python	56
int() Function	57
float() Function	59
str() Function	60
Conversion of Sequence types.....	62
14. Python - Unicode System	64
15. Python – Literals	66
Integer Literal	66
Float Literal	67
Complex Literal	67
String Literal.....	68
List Literal.....	69
Tuple Literal	69
Dictionary Literal	70
16. Python – Operators	71
Types of Operators	71
17. Python – Arithmetic Operators	72

Python – Addition Operator (+)	72
Python – Subtraction Operator (-)	73
Python – Multiplication Operator (*)	74
Python – Division Operator (/).....	75
Python – Modulus Operator (%).....	77
Python – Exponent Operator (**).....	78
Python – Floor Division Operator (//).....	80
Python – Complex Number Arithmetic.....	80
18. Python – Assignment Operators.....	83
19. Python – Augmented Addition Operator (+=).....	84
Python – Augmented Subtraction Operator (-=)	85
Python – Augmented Multiplication Operator (*=).....	85
Python – Augmented Division Operator (/=).....	86
Python – Augmented Modulus Operator (%=).....	87
Python – Augmented Exponent Operator (**=).....	88
Python – Augmented Floor division Operator (//=)	89
20. Python – Comparison Operators	90
21. Python – Logical Operators	96
22. Python – Bitwise Operators.....	100
Python – Bitwise AND Operator (&)	100
Python – Bitwise OR Operator ().....	101
Python – Binary XOR Operator (^).....	102
Python – Binary NOT Operator (~).....	102
Python – Left Shift Operator (<<)	103
Python – Right Shift Operator (>>)	103
23. Python – Membership Operators	105
24. Python – Identity Operators.....	108
25. Python – Comments.....	110

26. Python – User Input	111
input() Function	111
Numeric Input.....	113
27. Python – Numbers	117
Python – Integers.....	117
Python – Floating Point Numbers.....	120
Python – Complex Numbers	123
28. Python – Booleans	126
PYTHON CONTROL STATEMENTS	127
29. Python – Control Flow	128
30. Python – Decision Making	130
Python – The if Statement	130
31. Python – The if-else Statement	133
Python – elif Statement	135
Nested If Statements	137
32. Python – Match-Case Statement	140
Combined Cases	141
List as the Argument.....	142
Using "if" in "Case" Clause	142
33. Python – The for Loop	144
Using "for" with a String	144
Using "for" with a Tuple.....	145
Using "for" with a List	146
Using "for" with a Range Object	146
Using "for" Loop with Sequence Index	149
Using "for" with Dictionaries	149
34. Python – The for-else Loop.....	152
Nested Loops	153

35. Python – The while Loop	155
The Infinite Loop.....	157
The while-else Loop	158
36. Python – The break Statement	160
Loop Control Statements.....	160
The break Statement	160
37. Python – The continue Statement	164
38. Python – The pass Statement	167
PYTHON FUNCTIONS & MODULES	168
39. Python – Functions.....	169
Types of Python Functions.....	169
Python – Defining a Function.....	169
Calling a Function.....	170
Pass by Reference vs Value.....	171
Function Arguments	173
Function with Return Value	174
Types of Function Arguments.....	174
Order of Arguments.....	175
40. Python – Default Arguments	176
41. Python – Keyword Arguments.....	178
42. Python – Keyword-Only Arguments	180
43. Python – Positional Arguments	181
44. Python – Positional-Only Arguments.....	183
45. Python – Arbitrary Arguments	185
46. Python – Variable Scope.....	188
globals() Function	188
locals() Function.....	189
Namespace Conflict	190

47. Python – Function Annotations	193
48. Python – Modules	196
Built-in Modules	196
User-Defined Modules.....	197
Create a Module	197
The import Statement	198
The from ... import Statement.....	199
The from...import * Statement.....	199
The import ... as Statement	199
Module Attributes	200
The __name__ Attribute	201
The reload() Function	202
49. Python – Built-in Functions	204
Built-in Mathematical Functions	206
abs() function.....	207
max() function	208
min() function	209
pow() function	210
round() Function.....	211
sum() function	212
PYTHON STRINGS.....	214
50. Python – Strings	215
51. Python – Slicing Strings	217
52. Python – Modify Strings	222
53. Python – String Concatenation	225
54. Python – String Formatting	227
String Formatting Operator	227
Using the format() Method.....	230

Using f-string Formatting.....	232
Using String Template Class.....	234
55. Python – Escape Characters	236
56. Python – String Methods.....	240
Case Conversion Methods	240
Alignment Methods	246
Find and Replace Methods	252
Boolean String Methods	260
Split and Join Methods	274
Translation Methods.....	285
57. Python – String Exercises.....	289
PYTHON LISTS	291
58. Python – Lists	292
Python – List Operations	292
59. Python – Access List Items.....	293
60. Python – Change List Items	295
61. Python – Add List Items.....	297
62. Python – Remove List Items	298
Using the remove() Method	298
Using the pop() Method	298
Using the "del" Keyword.....	298
63. Python – Loop Lists	300
64. Python – List Comprehension.....	301
List Comprehension Technique	301
Nested Loops in List Comprehension	302
Condition in List Comprehension.....	302
65. Python – Sort Lists.....	303
66. Python – Copy Lists	305

67. Python – Join Lists	307
68. Python – List Methods	309
Python – Clear a List	309
Python – Counting the List Items.....	310
Python – Extending a List.....	311
Python – Finding the Index of a List Item	311
Python – Reverse a List.....	312
69. Python – List Exercises	313
PYTHON TUPLES	315
70. Python – Tuples	316
Python – Tuple Operations	316
71. Python – Access Tuple Items	317
Extracting a Subtuple from a Tuple	317
72. Python – Update Tuples	319
How to Update a Python Tuple?.....	319
73. Python – Unpack Tuple Items	321
Using "*" to Unpack a Tuple	321
74. Python – Loop Tuples	323
75. Python – Join Tuples	324
76. Python – Tuple Methods	326
Finding the Index of a Tuple Item	326
Counting Tuple Items.....	326
77. Python – Tuple Exercises	328
PYTHON SETS	330
78. Python – Sets	331
set() Function.....	331
79. Python – Access Set Items	334

80. Python – Add Set Items	335
add() Method.....	335
update() Method	335
union() Method	336
81. Python – Remove Set Items	338
remove() Method	338
discard() Method	338
pop() Method	339
clear() Method.....	340
difference_update() Method.....	340
difference() Method	341
intersection_update() Method	341
intersection() Method	342
symmetric_difference_update() method	343
symmetric_difference() Method	343
82. Python – Loop Sets.....	345
83. Python – Join Sets	346
Using the " " Operator	346
Using the union() Method	346
Using the update() Method	346
Using the unpacking Operator.....	347
84. Python – Copy Sets.....	348
85. Python – Set Operators	349
Union Operator ()	349
Intersection Operator (&).....	349
Difference Operator (-)	350
Symmetric Difference Operator (^)	351
86. Python – Set Methods.....	353

87. Python – Set Exercises	354
PYTHON DICTIONARIES	356
88. Python – Dictionaries	357
Python – Dictionary Operators	358
89. Python – Access Dictionary Items	359
Using the "[" Operator.....	359
Using the get() Method	359
90. Python – Change Dictionary Items	361
91. Python – Add Dictionary Items	363
Using the "[" Operator.....	363
Using the update() Method	364
Using the Unpack Operator	366
Using the Union Operator ()	366
Using " =" Operator	367
92. Python – Remove Dictionary Items	368
Using del Keyword	368
Using pop() Method.....	369
Using popitem() Method	369
Using clear() Method	370
93. Python – Dictionary View Objects	371
items() Method.....	371
keys() Method.....	372
values() Method.....	372
94. Python – Loop Dictionaries	374
95. Python – Copy Dictionaries	377
96. Python – Nested Dictionaries	379
97. Python – Dictionary Methods	381
Dictionary clear() Method	381

Dictionary copy() Method.....	382
Dictionary fromkeys() Method	383
Dictionary get() Method	384
Dictionary has_key() Method	384
Dictionary items() Method	385
Dictionary keys() Method	386
Dictionary setdefault() Method	386
Dictionary update() Method	387
Dictionary values() Method	388
98. Python – Dictionary Exercises.....	389
PYTHON ARRAYS	391
99. Python – Arrays.....	392
100. Python – Access Array Items	394
Changing Array Items	394
101. Python – Add Array Items	395
The append() Method.....	395
The insert() Method.....	395
The extend() Method.....	396
102. Python – Remove Array Items.....	397
array.remove() Method	397
array.pop() Method	397
103. Python – Loop Arrays	399
104. Python – Copy Arrays	401
105. Python – Reverse Arrays	403
106. Python – Sort Arrays	405
Using a Sorting Algorithm	405
Using the sort() Method from List	405
Using the Built-in sorted() Function.....	406

107. Python – Join Arrays	407
108. Python – Array Methods	409
array.reverse() Method	409
array.count() Method	409
array.index() method	410
array.fromlist() Method.....	410
array.tofile() Method	411
array.fromfile() Method.....	411
109. Python – Array Exercises	413
PYTHON FILE HANDLING	415
110. Python – File Handling	416
The open() Function	416
File Opening Modes.....	417
111. Python – Write to File	419
Writing in Binary Mode.....	419
Appending to a File	420
Using the "w+" Mode	420
112. Python – Read Files	422
Reading in Binary Mode.....	423
Read Integer Data from File.....	423
Read Float Data from File	423
Using the "r+" Mode.....	424
Python – Simultaneous Read/Write	425
The seek() Method.....	425
113. Python – Renaming and Deleting Files	427
rename() Method	427
remove() Method	427
114. Python – Directories	428

The mkdir() Method	428
The chdir() Method.....	428
The getcwd() Method	429
The rmdir() Method	429
115. Python – File Methods	430
The close() Method.....	431
The flush() Method	431
The fileno() Method.....	432
isatty() Method	433
The next() function	434
The read() Method.....	435
The readline() Method.....	436
The readlines() Method	437
The seek() Method.....	438
The tell() Method.....	439
The truncate() Method	440
The write() Method	441
The writelines() Method	443
116. Python – OS File/Directory Methods.....	445
os.closerange() Method.....	446
os.dup() Method	447
os.fdatasync() Method.....	448
os.fdopen() Method.....	450
os.fsync() Method	451
os.ftruncate() Method	452
os.lseek() Method	454
os.open() Method	455
os.read() Method.....	456

os.write() Method.....	457
PYTHON OBJECT ORIENTED PROGRAMMING.....	459
117. Python – OOP Concepts	460
Procedural Oriented Approach.....	460
Python – OOP Concepts.....	460
118. Python – Object and Classes.....	464
119. Python – Class Attributes	466
Class Variables	467
120. Python – Class Methods	468
121. Python – Static Methods	470
122. Python – Constructors.....	471
Python – Instance Methods.....	473
123. Python – Access Modifiers	475
Name Mangling	477
Python – Property Object	477
Getters and Setter Methods.....	477
124. Python – Inheritance.....	480
Python – Multiple Inheritance.....	481
Method Resolution Order (MRO)	483
125. Python – Polymorphism	484
126. Python – Method Overriding.....	486
Base Overridable Methods	488
127. Python – Method Overloading	489
128. Python – Dynamic Binding	492
Duck Typing	493
129. Python – Dynamic Typing.....	495
130. Python – Abstraction.....	497
131. Python – Encapsulation.....	499

132. Python – Interfaces	502
133. Python – Packages.....	504
Define Package List	505
Package Installation	506
134. Python – Inner Classes	508
135. Python – Anonymous Class and Objects.....	510
136. Python – Singleton Class	512
137. Python – Wrapper Classes.....	513
138. Python – Enums	514
139. Python – Reflection	517
The type() Function.....	517
The isinstance() Function.....	518
The isinstance() Function	519
The callable() Function	519
The getattr() Function.....	520
The setattr() Function	521
The hasattr() Function	521
The dir() Function	522
PYTHON ERRORS AND EXCEPTIONS	524
140. Python – Syntax Errors	525
What is Exception?	526
141. Python – Exceptions Handling	528
142. Python – The try-except Block.....	530
143. Python – The try-finally Block.....	531
Exception with Arguments	532
144. Python – Raising Exceptions	533
145. Python – Exception Chaining	535
146. Python – Nested try Block.....	538

147. Python – User-Defined Exceptions	541
148. Python – Logging	542
Logging Levels	542
Logging Configuration	543
Variable Data in Logging Message	544
149. Python – Assertions	545
The assert Statement	545
150. Python – Built-in Exceptions.....	547
PYTHON MULTITHREADING	552
151. Python – Multithreading	553
152. Python – Thread Life cycle.....	554
Python – The _thread Module.....	554
Python – The threading Module.....	554
153. Python – Creating a Thread	556
154. Python – Starting a Thread	558
155. Python – Joining the Threads	559
156. Python – Naming the Threads	560
The daemon Property	560
157. Python – Thread Scheduling.....	562
158. Python – Thread Pools	565
What is a Thread Pool?	565
The Future Class	565
The ThreadPoolExecutor Class	566
159. Python – Main Thread	568
160. Python – Thread Priority	570
161. Python – Daemon Threads	573
162. Python – Synchronizing Threads.....	576

PYTHON SYNCHRONIZATION.....	578
163. Python – Inter-Thread Communication	579
The Event Object	579
The Condition Object	581
164. Python – Thread Deadlock	584
The Lock Object	584
The Semaphore Object	586
165. Python – Interrupting a Thread	589
PYTHON NETWORKING	591
166. Python – Network Programming	592
167. Python – Socket Programming	594
What are Sockets?	594
Python – The socket Module	595
Server Socket Methods.....	595
Client Socket Methods.....	596
Python – Socket Server	596
Python – Socket Client	597
Python – File Transfer with Socket Module.....	599
Python – The socketserver Module	601
168. Python – URL Processing	604
The urllib.parse Module	604
The urllib.request Module	606
The Request Object	607
The urllib.error Module	609
169. Python – Generics	610
PYTHON MISCELLANEOUS.....	612
170. Python – Date and Time	613

What are Tick Intervals?	613
What is TimeTuple?	613
Getting the Current Time.....	615
Getting the Formatted Time	615
Getting the Calendar for a Month	616
The time Module	616
Time altzone() Method	618
Time asctime() Method	618
Time perf_counter() Method.....	619
Time ctime() Method	620
Time gmtime() Method	621
Time localtime() Method	621
Time mktime() Method	622
Time sleep() Method	623
Time strftime() Method	624
Time strptime() Method	625
Time time() Method	627
Time tzset() Method	628
The calendar Module	630
datetime module	631
date.....	632
time.....	636
datetime	639
timedelta	643
171. Python – Maths.....	647
Functions in Math Module	647
Theoretic and Representation Functions	650
Power and Logarithmic Functions	676

Trigonometric Functions.....	685
Angular Conversion Functions	695
Hyperbolic Functions	698
Special Functions	705
Mathematical Constants.....	709
172. Python – Iterators	713
Asynchronous Iterator	716
173. Python – Generators	718
Asynchronous Generator.....	720
174. Python – Closures.....	723
Nested Functions	723
What is a Closure?	724
nonlocal Keyword	724
175. Python – Decorators.....	726
@classmethod Decorator	728
@staticmethod Decorator	729
@property Decorator	730
176. Python – Recursion	732
177. Python – Regular Expressions.....	735
Raw Strings	735
Metacharacters.....	736
re.match() Function	737
re.search() Function.....	738
Matching Vs Searching	739
re.findall() Function	740
re.sub() Function.....	741
re.compile() Function	742
re.finditer() Function	743

Use Cases of Python Regex	744
178. Python – PIP	745
Install a Package	745
Use requirements.txt	747
pip uninstall	747
pip list	748
pip show	749
pip freeze	749
pip download	749
pip search	750
pip config	750
179. Python – Database Access	751
The sqlite3 Module	752
The Connection Object	752
The Cursor Object	753
Creating a Database Table	753
INSERT Operation	754
READ Operation	755
Update Operation	756
DELETE Operation	757
Performing Transactions.....	757
COMMIT Operation	758
ROLLBACK Operation	758
The PyMySQL Module	759
Handling Errors	760
180. Python – Weak References.....	761
The callback Function	762
Finalizing Objects.....	763

WeakKeyDictionary	764
WeakValueDictionary	765
181. Python – Serialization	768
Pickle Protocols.....	768
dump() and load()	768
dumps() and loads().....	769
Pickler Class	770
Unpickler Class.....	770
182. Python – Templating	771
substitute()	771
safe_substitute().....	772
is_valid().....	773
get_identifiers()	773
183. Python – Output Formatting	775
String Formatting Operator	775
The format() Method.....	775
f-strings.....	776
Template Strings.....	776
The textwrap Module	777
The shorten() Function	779
The pprint Module.....	780
184. Python – Performance Measurement	784
185. Python – Data Compression	787
The zlib Module	787
The gzip Module	788
The bz2 Module	790
The lzma Module	791
The zipfile Module	793

The tarfile Module	796
Command Line Interface	798
186. Python – CGI Programming	799
What is CGI?	799
Web Browsing.....	799
CGI Architecture Diagram	800
Web Server Support and Configuration	800
First CGI Program.....	801
HTTP Header	802
CGI Environment Variables	802
GET and POST Methods.....	804
Passing Information using GET method.....	804
Simple URL Example – Get Method.....	804
Simple FORM Example – GET Method.....	805
Passing Radio Button Data to CGI Program	808
Passing Text Area Data to CGI Program.....	809
Passing Drop Down Box Data to CGI Program	810
Using Cookies in CGI	811
How It Works?	811
Setting up Cookies	811
Retrieving Cookies	812
File Upload Example	813
How To Raise a "File Download" Dialog Box ?	814
187. Python – XML Processing	815
What is XML?	815
Parsing XML with SAX APIs	817
Parsing XML with DOM APIs	820
ElementTree XML API	822

Create an XML File	822
Parse an XML File	824
Modify an XML file	825
188. Python – GUI Programming	826
Tkinter Programming	826
Tkinter Widgets	828
Tkinter Button	829
Tkinter Canvas	832
Tkinter Checkbutton	835
Tkinter Entry	838
Tkinter Frame	842
Tkinter Label	844
Tkinter Listbox	847
Tkinter Menubutton	851
Tkinter Menu	854
Tkinter Message	858
Tkinter Radiobutton	860
Tkinter Scale	864
Tkinter Scrollbar	868
Tkinter Text	871
Tkinter Toplevel	876
Tkinter Spinbox	879
Tkinter PanedWindow	882
Tkinter LabelFrame	885
Tkinter tkMessageBox	887
Standard Attributes	888
Tkinter Dimensions	888
Tkinter Colors	889

Tkinter Fonts.....	890
Tkinter Anchors.....	891
Tkinter Relief styles.....	892
Tkinter Bitmaps.....	893
Tkinter Cursors.....	894
Geometry Management	895
Tkinter pack() Method	896
Tkinter grid() Method	897
Tkinter place() Method	898
SimpleDialog	900
The FileDialog Module	902
ttk module	904
Combobox Widget	905
Progressbar.....	906
Notebook	908
Treeview	909
Sizegrip	911
Separator	912
189. Python – Command-Line Arguments	914
The getopt Module	916
The argparse Module.....	918
190. Python – Docstrings	922
191. Python – JSON.....	926
JSONEncoder Class.....	927
JSONDecoder class.....	929
JSON with Files/Streams.....	929
192. Python – Sending Emails	931

smtplib.SMTP() Function	931
The smtpd Module	932
Using gmail SMTP	934
193. Python – Further Extensions	936
Pre-Requisites for Writing Extensions	936
First look at a Python Extension	936
The Header File Python.h	936
The C Functions	936
The Method Mapping Table	937
The Initialization Function	938
Building and Installing Extensions	940
Importing Extensions	940
Passing Function Parameters.....	941
The PyArg_ParseTuple Function	942
Returning Values.....	943
The Py_BuildValue Function	944
194. Python – Tools / Utilities	946
The <i>dis</i> Module	946
The <i>pdb</i> Module.....	947
The <i>profile</i> Module	948
The <i>tabnanny</i> Module	949
195. Python – GUIs	950
IDLE.....	950
Jupyter Notebook	951
VS Code.....	953
PyCharm	955

Python Basics

1. Python – Overview

Python is a high-level, multi-paradigm programming language. As Python is an interpreter-based language, it is easier to learn compared to some of the other mainstream languages. Python is a dynamically typed language with very intuitive data types.

Python is an open-source and cross-platform programming language. It is available for use under **Python Software Foundation License** (compatible to GNU General Public License) on all the major operating system platforms Linux, Windows and Mac OS.

The design philosophy of Python emphasizes on simplicity, readability and unambiguity. Python is known for its batteries included approach as Python software is distributed with a comprehensive standard library of functions and modules.

Python's design philosophy is documented in the **Zen of Python**. It consists of nineteen aphorisms such as:

- Beautiful is better than ugly
- Explicit is better than implicit
- Simple is better than complex
- Complex is better than complicated

To obtain the complete Zen of Python document, type **import this** in the Python shell:

```
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
```


Now is better than never.

Although never is often better than **right** now.

If the implementation is hard to explain, it's a bad idea.

If the implementation is easy to explain, it may be a good idea.

Namespaces are one honking great idea -- let's do more of those!

Python supports imperative, structured as well as object-oriented programming methodology. It provides features of functional programming as well.

2. Python – History

Guido Van Rossum, a Dutch programmer, created Python programming language. In the late 80's, he had been working on the development of ABC language in a computer science research institute named **Centrum Wiskunde & Informatica** (CWI) in the Netherlands. In 1991, Van Rossum conceived and published Python as a successor of **ABC** language.

For many uninitiated people, the word Python is related to a species of snake. Rossum though attributes the choice of the name Python to a popular comedy series "**Monty Python's Flying Circus**" on BBC.

Being the principal architect of Python, the developer community conferred upon him the title of "**Benevolent Dictator for Life** (BDFL). However, in 2018, Rossum relinquished the title. Thereafter, the development and distribution of the reference implementation of Python is handled by a nonprofit organization **Python Software Foundation**.

Important stages in the history of Python:

Python 0.9.0

Python's first published version is 0.9. It was released in February 1991. It consisted of support for core object-oriented programming principles.

Python 1.0

In January 1994, version 1.0 was released, armed with functional programming tools, features like support for complex numbers etc.

Python 2.0

Next major version – Python 2.0 was launched in October 2000. Many new features such as list comprehension, garbage collection and Unicode support were included with it.

Python 3.0

Python 3.0, a completely revamped version of Python was released in December 2008. The primary objective of this revamp was to remove a lot of discrepancies that had crept in Python 2.x versions. Python 3 was backported to Python 2.6. It also included a utility named as **python2to3** to facilitate automatic translation of Python 2 code to Python 3.

EOL for Python 2.x

Even after the release of Python 3, Python Software Foundation continued to support the Python 2 branch with incremental micro versions till 2019. However, it decided to discontinue the support by the end of year 2020, at which time Python 2.7.17 was the last version in the branch.

Current Version

Meanwhile, more and more features have been incorporated into Python's 3.x branch. As of date, Python **3.11.2** is the current stable version, released in February 2023.

What's New in Python 3.11?

One of the most important features of Python's version 3.11 is the significant improvement in speed. According to Python's official documentation, this version is faster than the previous version (3.10) by up to 60%. It also states that the standard benchmark suite shows a 25% faster execution rate.

- Python 3.11 has a better exception messaging. Instead of generating a long traceback on the occurrence of an exception, we now get the exact expression causing the error.
- As per the recommendations of PEP 678, the **add_note()** method is added to the **BaseException** class. You can call this method inside the except clause and pass a custom error message.
- It also adds the **cbroot()** function in the **maths** module. It returns the cube root of a given number.
- A new module **tomllib** is added in the standard library. TOML (Tom's Obvious Minimal Language) can be parsed with tomllib module function.

3. Python – Features

In this chapter, let's highlight some of the important features of Python that make it widely popular.

Python is Easy to Learn

This is one of the most important reasons for the popularity of Python. Python has a limited set of keywords. Its features such as simple syntax, usage of indentation to avoid clutter of curly brackets and dynamic typing that doesn't necessitate prior declaration of variable help a beginner to learn Python quickly and easily.

Python is Interpreter Based

Instructions in any programming languages must be translated into machine code for the processor to execute them. Programming languages are either compiler based or interpreter based.

In case of a compiler, a machine language version of the entire source program is generated. The conversion fails even if there is a single erroneous statement. Hence, the development process is tedious for the beginners. The C family languages (including C, C++, Java, C Sharp etc) are compiler based.

Python is an interpreter based language. The interpreter takes one instruction from the source code at a time, translates it into machine code and executes it. Instructions before the first occurrence of error are executed. With this feature, it is easier to debug the program and thus proves useful for the beginner level programmer to gain confidence gradually. Python therefore is a beginner-friendly language.

Python is Interactive

Standard Python distribution comes with an interactive shell that works on the principle of REPL (Read – Evaluate – Print – Loop). The shell presents a Python prompt `>>>`. You can type any valid Python expression and press Enter. Python interpreter immediately returns the response and the prompt comes back to read the next expression.

```
>>> 2*3+1
7

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

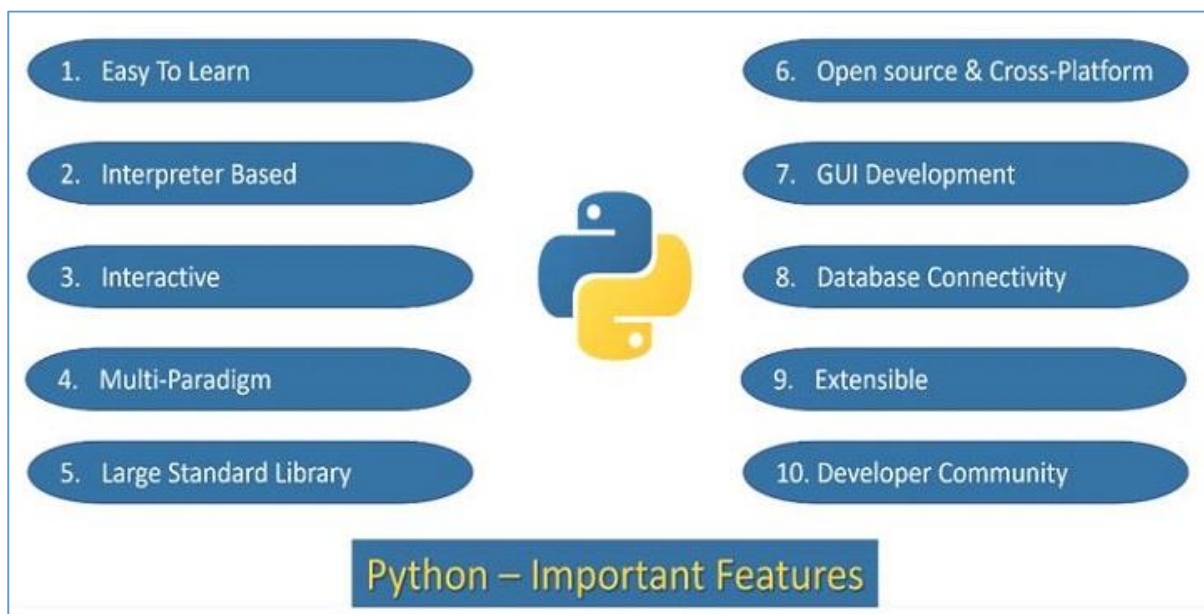
The interactive mode is especially useful to get familiar with a library and test out its functionality. You can try out small code snippets in interactive mode before writing a program.

Python is Multi-Paradigm

Python is a completely object-oriented language. Everything in a Python program is an object. However, Python conveniently encapsulates its object orientation to be used as an imperative or procedural language – such as C. Python also provides certain functionality that resembles functional programming. Moreover, certain third-party tools have been developed to support other programming paradigms such as aspect-oriented and logic programming.

Python's Standard Library

Even though it has a very few keywords (only Thirty Five), Python software is distributed with a standard library made of large number of modules and packages. Thus Python has out of box support for programming needs such as serialization, data compression, internet data handling, and many more. Python is known for its batteries included approach.



Python is Open-Source and Cross-Platform

Python's standard distribution can be downloaded from <https://www.python.org/downloads/> without any restrictions. You can download pre-compiled binaries for various operating system platforms. In addition, the source code is also freely available, which is why it comes under open source category.

Python software (along with the documentation) is distributed under Python Software Foundation License. It is a BSD style permissive software license and compatible to GNU GPL (General Public License).

Python is a cross-platform language. Pre-compiled binaries are available for use on various operating system platforms such as Windows, Linux, Mac OS, Android OS. The reference implementation of Python is called CPython and is written in C. You can download the source code and compile it for your OS platform.

A Python program is first compiled to an intermediate platform independent byte code. The virtual machine inside the interpreter then executes the byte code. This behaviour

makes Python a cross-platform language, and thus a Python program can be easily ported from one OS platform to other.

Python for GUI Applications

Python's standard distribution has an excellent graphics library called TKinter. It is a Python port for the vastly popular GUI toolkit called TCL/Tk. You can build attractive user-friendly GUI applications in Python. GUI toolkits are generally written in C/C++. Many of them have been ported to Python. Examples are PyQt, WxWidgets, PySimpleGUI etc.

Python's Database Connectivity

Almost any type of database can be used as a backend with the Python application. DB-API is a set of specifications for database driver software to let Python communicate with a relational database. With many third party libraries, Python can also work with NoSQL databases such as MongoDB.

Python is Extensible

The term extensibility implies the ability to add new features or modify existing features. As stated earlier, CPython (which is Python's reference implementation) is written in C. Hence one can easily write modules/libraries in C and incorporate them in the standard library. There are other implementations of Python such as Jython (written in Java) and IPython (written in C#). Hence, it is possible to write and merge new functionality in these implementations with Java and C# respectively.

Python's Active Developer Community

As a result of Python's popularity and open-source nature, a large number of Python developers often interact with online forums and conferences. Python Software Foundation also has a significant member base, involved in the organization's mission to "promote, protect, and advance the Python programming language"

Python also enjoys a significant institutional support. Major IT companies Google, Microsoft, and Meta contribute immensely by preparing documentation and other resources.

4. Python vs C++

Both Python and C++ are among the most popular programming languages. Both of them have their advantages and disadvantages. In this chapter, we shall take a look at their characteristic features.

Compiled vs Interpreted

Like C, C++ is also a compiler-based language. A compiler translates the entire code in a machine language code specific to the operating system in use and processor architecture.

Python is interpreter-based language. The interpreter executes the source code line by line.

Cross-platform

When a C++ source code such as `hello.cpp` is compiled on Linux, it can be only run on any other computer with Linux operating system. If required to run on other OS, it needs to be compiled.

Python interpreter doesn't produce compiled code. Source code is converted to byte code every time it is run on any operating system without any changes or additional steps.

Portability

Python code is easily portable from one OS to other. C++ code is not portable as it must be recompiled if the OS changes.

Speed of Development

C++ program is compiled to the machine code. Hence, its execution is faster than interpreter based language.

Python interpreter doesn't generate the machine code. Conversion of intermediate byte code to machine language is done on each execution of program.

If a program is to be used frequently, C++ is more efficient than Python.

Easy to Learn

Compared to C++, Python has a simpler syntax. Its code is more readable. Writing C++ code seems daunting in the beginning because of complicated syntax rule such as use of curly braces and semicolon for sentence termination.

Python doesn't use curly brackets for marking a block of statements. Instead, it uses indents. Statements of similar indent level mark a block. This makes a Python program more readable.

Static vs Dynamic Typing

C++ is a statically typed language. The type of variables for storing data need to be declared in the beginning. Undeclared variables can't be used. Once a variable is declared to be of a certain type, value of only that type can be stored in it.

Python is a dynamically typed language. It doesn't require a variable to be declared before assigning it a value. Since, a variable may store any type of data, it is called dynamically typed.

OOP Concepts

Both C++ and Python implement object oriented programming concepts. C++ is closer to the theory of OOP than Python. C++ supports the concept of data encapsulation as the visibility of the variables can be defined as public, private and protected.

Python doesn't have the provision of defining the visibility. Unlike C++, Python doesn't support method overloading. Because it is dynamically typed, all the methods are polymorphic in nature by default.

C++ is in fact an extension of C. One can say that additional keywords are added in C so that it supports OOP. Hence, we can write a C type procedure oriented program in C++.

Python is completely object oriented language. Python's data model is such that, even if you can adapt a procedure oriented approach, Python internally uses object-oriented methodology.

Garbage Collection

C++ uses the concept of pointers. Unused memory in a C++ program is not cleared automatically. In C++, the process of garbage collection is manual. Hence, a C++ program is likely to face memory related exceptional behavior.

Python has a mechanism of automatic garbage collection. Hence, Python program is more robust and less prone to memory related issues.

Application Areas

Because C++ program compiles directly to machine code, it is more suitable for systems programming, writing device drivers, embedded systems and operating system utilities.

Python program is suitable for application programming. Its main area of application today is data science, machine learning, API development etc.

The following table summarizes the comparison between C++ and Python:

Criteria	C++	Python
Execution	Compiler based	Interpreter based
Typing	Static typing	Dynamic typing
Portability	Not portable	Highly portable
Garbage collection	Manual	Automatic

Syntax	Tedious	Simple
Performance	Faster execution	Slower execution
Application areas	Embedded systems, device drivers	Machine learning, web applications

5. Python – Hello World

Hello World program is a basic computer code written in a general purpose programming language, used as a test program. It doesn't ask for any input and displays a Hello World message on the output console. It is used to test if the software needed to compile and run the program has been installed correctly.

It is very easy to display the Hello World message using the Python interpreter. Launch the interpreter from a command terminal of your operating system and issue the print statement from the Python prompt as follows:

```
PS C:\Users\mlath> python
Python 3.11.2 (tags/v3.11.2:878ead1, Feb  7 2023, 16:38:35) [MSC v.1934
64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print ("Hello World")
Hello World
```

Similarly, Hello World message is printed in Linux.

```
mv1@GNVBGL3:~$ python3
Python 3.10.6 (main, Mar 10 2023, 10:55:28) [GCC 11.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print ("Hello World")
Hello World
```

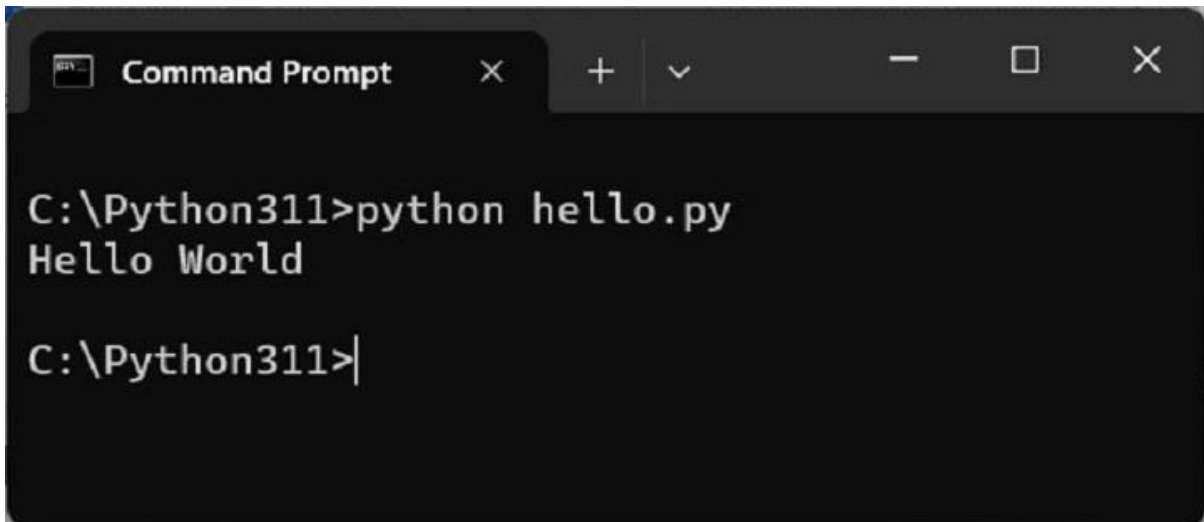
Python interpreter also works in scripted mode. Open any text editor, enter the following text and save as Hello.py

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

For Windows OS, open the command prompt terminal (CMD) and run the program as shown below:

```
C:\Python311>python hello.py
Hello World
```

The terminal shows the Hello World message.

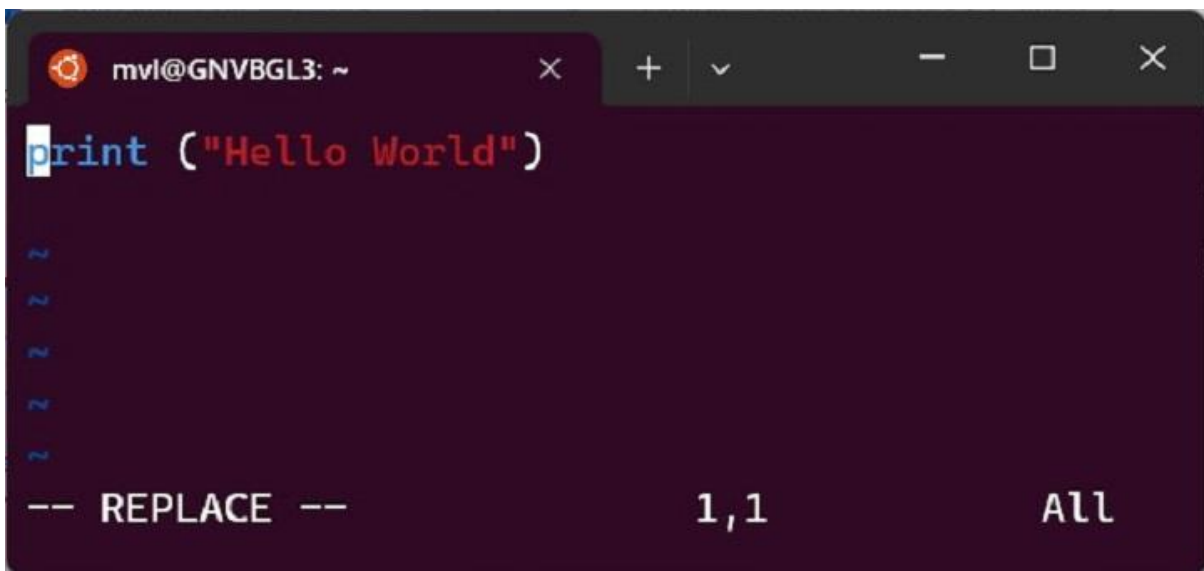


```

Command Prompt
C:\Python311>python hello.py
Hello World
C:\Python311>

```

While working on Ubuntu Linux, you have to follow the same steps, save the code and run from Linux terminal. We use vi editor for saving the program.



```

mv1@GNVBGL3: ~
print ("Hello World")
~
~
~
~
~
-- REPLACE --          1,1          All

```

To run the program from Linux terminal

```

mv1@GNVBGL3:~$ python3 hello.py
Hello World

```

In Linux, you can convert a Python program into a self executable script. The first statement in the code should be a shebang. It must contain the path to Python executable. In Linux, Python is installed in `/usr/bin` directory, and the name of the executable is `python3`. Hence, we add this statement to `hello.py` file

```

#!/usr/bin/python3
print ("Hello World")

```

You also need to give the file executable permission by using the `chmod +x` command

```

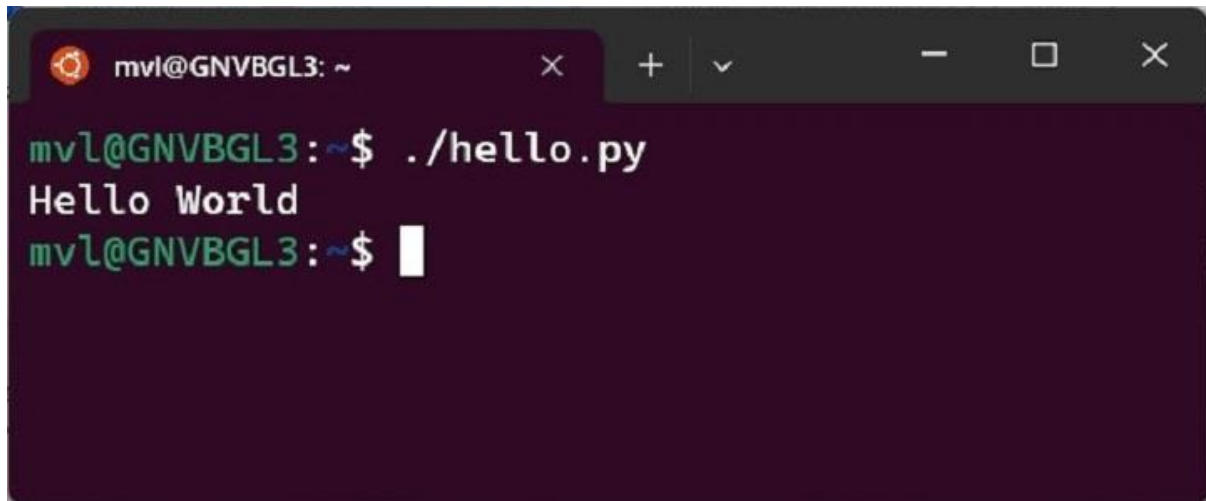
mv1@GNVBGL3:~$ chmod +x hello.py

```

Then, you can run the program with following command line:

```
mv1@GNVBGL3:~$ ./hello.py
```

The **output** is shown below:

A terminal window with a dark purple background. The window title is 'mv1@GNVBGL3: ~'. The prompt is 'mv1@GNVBGL3:~\$'. The command './hello.py' is entered and executed, resulting in the output 'Hello World'. The prompt returns to 'mv1@GNVBGL3:~\$' with a white cursor. The terminal window has standard window controls (close, maximize, minimize) in the top right corner.

```
mv1@GNVBGL3:~$ ./hello.py
Hello World
mv1@GNVBGL3:~$
```

Thus, we can write and run Hello World program in Python using the interpreter mode and script mode.

=====

End of ebook preview

If you liked what you saw...

Buy it from our store @ <https://store.tutorialspoint.com>