



PYTHON

programming language

tutorialspoint

SIMPLY EASY LEARNING

www.tutorialspoint.com



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



<https://twitter.com/tutorialspoint>

About the Tutorial

Python is a general-purpose interpreted, interactive, object-oriented, and high-level 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). This tutorial gives enough understanding on Python programming language.

Audience

This tutorial is designed for software programmers who need to learn Python programming language from scratch.

Prerequisites

You should have a basic understanding of Computer Programming terminologies. A basic understanding of any of the programming languages is a plus.

Disclaimer & Copyright

© Copyright 2017 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	i
Prerequisites	i
Disclaimer & Copyright.....	i
Table of Contents	ii
1. PYTHON – OVERVIEW	1
History of Python	1
Python Features	1
2. PYTHON – ENVIRONMENT.....	3
Local Environment Setup.....	3
Getting Python	3
Installing Python	4
Setting up PATH	5
Setting path at Unix/Linux	5
Setting path at Windows	6
Python Environment Variables	6
Running Python.....	6
3. PYTHON – BASIC SYNTAX.....	9
First Python Program	9
Python Identifiers.....	10
Python Keywords	11
Lines and Indentation.....	11
Multi-Line Statements.....	13

Quotation in Python.....	13
Comments in Python.....	14
Using Blank Lines	14
Waiting for the User.....	15
Multiple Statements on a Single Line	15
Multiple Statement Groups as Suites	15
Command Line Arguments	15
Accessing Command-Line Arguments.....	16
Parsing Command-Line Arguments	17
getopt.getopt method.....	17
Exception getopt.GetoptError	17
4. PYTHON – VARIABLE TYPES	20
Assigning Values to Variables.....	20
Multiple Assignment	21
Standard Data Types	21
Python Numbers	21
Python Strings.....	23
Python Lists.....	24
Python Tuples	24
Python Dictionary	26
Data Type Conversion	27
5. PYTHON – BASIC OPERATORS.....	29
Types of Operators.....	29
Python Arithmetic Operators	29
Python Comparison Operators	31

Python Assignment Operators	34
Python Bitwise Operators	36
Python Logical Operators	38
Python Membership Operators.....	38
Python Identity Operators.....	40
Python Operators Precedence.....	41
6. PYTHON – DECISION MAKING.....	44
If Statement	45
If...else Statement	46
The <i>elif</i> Statement	48
Single Statement Suites.....	49
7. PYTHON – LOOPS	51
While Loop	52
The Infinite Loop	53
Using else Statement with Loops	54
Single Statement Suites.....	55
For Loop	56
Iterating by Sequence Index	57
Using else Statement with Loops	58
Nested Loops	59
Loop Control Statements.....	60
Break Statement	61
Continue Statement	63
Pass Statement	65

8. PYTHON – NUMBERS.....	66
Number Type Conversion	67
Random Number Functions.....	69
Trigonometric Functions	69
Mathematical Constants	70
9. PYTHON – STRINGS.....	71
Accessing Values in Strings.....	71
Updating Strings.....	71
Escape Characters	72
String Special Operators.....	73
String Formatting Operator	74
Triple Quotes	76
Unicode String.....	77
Built-in String Methods	78
capitalize() Method	82
center(width, fillchar) Method	82
count(str, beg= 0,end=len(string)) Method	83
decode(encoding='UTF-8',errors='strict') Method	84
encode(encoding='UTF-8',errors='strict') Method	85
endswith(suffix, beg=0, end=len(string)) Method	86
expandtabs(tabsize=8)	87
find(str, beg=0 end=len(string)).....	88
index(str, beg=0, end=len(string))	89
isalnum() Method	90
isalpha()	90

<code>isdigit()</code>	91
<code>islower()</code>	92
<code>isnumeric()</code>	93
<code>isspace()</code> Method.....	94
<code>istitle()</code>	95
<code>isupper()</code>	96
<code>join(seq)</code>	96
<code>len(string)</code>	97
<code>ljust(width[, fillchar])</code>	98
<code>lower()</code>	99
<code>lstrip()</code>	100
<code>maketrans()</code>	100
<code>max(str)</code>	102
<code>min(str)</code>	102
<code>replace(old, new [, max])</code>	103
<code>rfind(str, beg=0,end=len(string))</code>	104
<code>rindex(str, beg=0, end=len(string))</code>	105
<code>rjust(width,[, fillchar])</code>	106
<code>rstrip()</code>	107
<code>split(str="", num=string.count(str))</code>	108
<code>splitlines(num=string.count('\n'))</code>	109
<code>startswith(str, beg=0,end=len(string))</code>	110
<code>strip([chars])</code>	111
<code>swapcase()</code>	111
<code>title()</code>	112
<code>translate(table, deletechars="")</code>	113

upper()	114
zfill (width)	115
isdecimal()	116
10. PYTHON – LISTS	118
Python Lists	118
Accessing Values in Lists	118
Updating Lists	119
Deleting List Elements	119
Basic List Operations	120
Indexing, Slicing, and Matrixes	121
Built-in List Functions and Methods	121
Cmp(list1, list2)	122
len(List)	123
max(list)	124
min(list)	124
List.append(obj)	126
list.count(obj)	127
list.extend(seq)	128
list.index(obj)	128
list.insert(index,obj)	129
list.pop(obj=list[-1])	130
List.remove(obj)	131
List.reverse()	131
list.sort([func])	132

11. PYTHON – TUPLES	134
Accessing Values in Tuples	134
Updating Tuples	135
Deleting Tuple Elements	135
Basic Tuples Operations	136
Indexing, Slicing, and Matrixes	136
No Enclosing Delimiters.....	137
Built-in Tuple Functions.....	137
Cmp(tuple1, tuple2)	138
Len(tuple).....	139
Max(tuple)	140
Min(tuple).....	141
Tuple(seg)	141
12. PYTHON – DICTIONARY.....	143
Accessing Values in Dictionary	143
Updating Dictionary	144
Delete Dictionary Elements	144
Properties of Dictionary Keys	145
Built-in Dictionary Functions and Methods	146
Cmp(dict1, dict2).....	146
len(dict).....	147
str(dict)	148
type()	149
dict.clear().....	151
Dict.copy().....	151

Dict.fromkeys()	152
Dict.get(key,default=None)	153
Dict.has_key(key)	154
Dict.items()	155
Dict.keys()	156
dict.setdefault(key, default=None)	156
dict.update(dict2)	157
dict.values()	158
13. PYTHON – DATE AND TIME	160
What is Tick?	160
What is TimeTuple?	160
Getting Current Time	162
Getting Formatted Time	162
Getting Calendar for a Month	163
The <i>time</i> Module	163
time.altzone	165
time.actime([tupletime])	166
time.clock()	166
time.ctime([secs])	168
time.gmtime([secs])	168
time.localtime([secs])	169
time.mktime(tupletime)	170
time.sleep(secs)	171
time.strftime(fmt,[tupletime])	172
time.strptime(str,fmt='%a %b %d %H:%M:%S %Y')	174

time.time()	176
time.tzset().....	177
The <i>calendar</i> Module	179
Other Modules and Functions	181
14. PYTHON – FUNCTIONS	182
Defining a Function	182
Calling a Function	183
Passing by Reference Versus Passing by Value	184
Function Arguments.....	185
Required Arguments	185
Keyword Arguments.....	186
Default Arguments	187
Variable Length Arguments	188
The Anonymous Functions.....	189
The return Statement	190
Scope of Variables.....	190
Global vs. Local variables	191
15. PYTHON – MODULES.....	192
The <i>import</i> Statement	192
The <i>from...import</i> Statement.....	193
The <i>from...import *</i> Statement:.....	193
Locating Modules:.....	193
The PYTHONPATH Variable	194
Namespaces and Scoping	194
The dir() Function	195

The <i>globals()</i> and <i>locals()</i> Functions.....	196
The <i>reload()</i> Function	196
Packages in Python	196
16. PYTHON – FILES I/O	198
Printing to the Screen.....	198
Reading Keyboard Input.....	198
The <i>raw_input</i> Function	198
The <i>input</i> Function	199
Opening and Closing Files.....	199
The <i>open</i> Function	199
The file Object Attributes	201
The <i>close()</i> Method	202
Reading and Writing Files.....	203
The <i>write()</i> Method.....	203
The <i>read()</i> Method	204
File Positions	204
Renaming and Deleting Files	205
The <i>rename()</i> Method	206
The <i>remove()</i> Method	206
Directories in Python.....	207
The <i>mkdir()</i> Method	207
The <i>chdir()</i> Method	207
The <i>getcwd()</i> Method.....	208
The <i>rmdir()</i> Method.....	208
File and Directory Related Methods	209

file.close()	210
File.flush()	211
File.fileno()	212
File.isatty()	213
File.next()	214
File.read([size])	215
File.readline([size])	216
file.readline([sizehint])	218
file.seek(offset[,whence])	219
file.tell()	221
file.truncate([size])	222
file.write(str)	224
file.writelines(sequence)	225
OS Object Methods	227
17. PYTHON – EXCEPTIONS	233
Assertions in Python	235
The <i>assert</i> Statement	235
What is Exception?	236
Handling an Exception	236
The <i>except</i> Clause with No Exceptions	238
The <i>except</i> Clause with Multiple Exceptions	239
The try-finally Clause	239
Argument of an Exception	240
Raising an Exception	241
User-Defined Exceptions	242

18. PYTHON – CLASSES AND OBJECTS	244
Overview of OOP Terminology	244
Creating Classes	245
Creating Instance Objects.....	246
Accessing Attributes.....	246
Built-In Class Attributes.....	248
Destroying Objects (Garbage Collection)	249
Class Inheritance	251
Overriding Methods	252
Base Overloading Methods	253
Overloading Operators.....	254
Data Hiding	255
19. PYTHON – REGULAR EXPRESSIONS	257
The match Function	257
The search Function	259
Matching Versus Searching	260
Search and Replace	261
Regular-Expression Modifiers: Option Flags	261
Regular-Expression Patterns	262
Regular-Expression Examples	265
Grouping with Parentheses	267
Backreferences.....	267
20. PYTHON – CGI PROGRAMMING.....	270
What is CGI?.....	270
Web Browsing.....	270

CGI Architecture.....	271
Web Server Support and Configuration.....	271
First CGI Program	272
HTTP Header	273
CGI Environment Variables.....	274
GET and POST Methods.....	275
Passing Information using GET method:.....	276
Simple URL Example : Get Method.....	276
Simple FORM Example: GET Method.....	277
Passing Information Using POST Method	278
Passing Checkbox Data to CGI Program.....	279
Passing Radio Button Data to CGI Program	280
Passing Text Area Data to CGI Program.....	281
Passing Drop Down Box Data to CGI Program	283
Using Cookies in CGI.....	284
How It Works?.....	284
Setting up Cookies.....	285
Retrieving Cookies.....	285
File Upload Example.....	286
How To Raise a "File Download" Dialog Box?.....	288
21. PYTHON – DATABASE ACCESS.....	289
What is MySQLdb?	289
How do I Install MySQLdb?	290
Database Connection	290
Creating Database Table	292

INSERT Operation.....	293
READ Operation	295
Update Operation	296
DELETE Operation	297
Performing Transactions	298
COMMIT Operation.....	299
ROLLBACK Operation	299
Disconnecting Database	299
Handling Errors	300
22. PYTHON – NETWORK PROGRAMMING.....	302
What is Sockets?	302
The <i>socket</i> Module	303
Server Socket Methods	303
Client Socket Methods	304
General Socket Methods	304
A Simple Server	304
A Simple Client	305
Python Internet modules	306
Further Readings	307
23. PYTHON – SENDING EMAIL.....	308
Sending an HTML e-mail using Python	309
Sending Attachments as an E-mail	310
24. PYTHON – MULTITHREADING.....	313
Starting a New Thread.....	313
The <i>Threading</i> Module	314

Creating Thread Using <i>Threading</i> Module:	315
Synchronizing Threads	317
Multithreaded Priority Queue	319
25. PYTHON – XML PROCESSING	323
What is XML?	323
XML Parser Architectures and APIs:	323
Parsing XML with SAX APIs.....	325
The <i>make_parser</i> Method	325
The <i>parse</i> Method	325
The <i>parseString</i> Method.....	326
Parsing XML with DOM APIs.....	329
26. PYTHON – GUI PROGRAMMING	332
Tkinter Programming	332
Tkinter Widgets.....	333
Button.....	335
Canvas.....	338
Checkbutton.....	340
Entry	344
Frame.....	349
Label	351
Listbox.....	354
MenuButton.....	358
Menu	362
Message	366
Radiobutton	369

Scale.....	373
Scrollbar	378
Text.....	381
TopLevel.....	387
SpinBox	390
PanelWindow.....	393
LabelFrame	396
tkMessageBox.....	398
Standard Attributes.....	400
Dimensions	400
Colors.....	401
Fonts.....	402
Anchors.....	403
Relief styles.....	404
Bitmaps.....	406
Cursors.....	407
Geometry Management.....	408
Python Tkinter pack() Method	409
Python Tkinter grid() Method.....	410
Python Tkinter place() Method	411
27. PYTHON – FURTHER EXTENSIONS.....	413
Pre-Requisites for Writing Extensions	413
First Look at a Python Extension	413
The Header File <i>Python.h</i>	413
The C Functions.....	414

The Method Mapping Table	415
The Initialization Function	416
Building and Installing Extensions	418
Importing Extensions	418
Passing Function Parameters	418
The PyArg_ParseTuple Function	420
Returning Values	421
The <i>Py_BuildValue</i> Function	423

1. Python – Overview

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **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.

History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, Unix shell, and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

Python Features

Python's features include:

- **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.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below:

- 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.

2. Python – Environment

Python is available on a wide variety of platforms including Linux and Mac OS X. Let's understand how to set up our Python environment.

Local Environment Setup

Open a terminal window and type "python" to find out if it is already installed and which version is installed.

- Unix (Solaris, Linux, FreeBSD, AIX, HP/UX, SunOS, IRIX, etc.)
- Win 9x/NT/2000
- Macintosh (Intel, PPC, 68K)
- OS/2
- DOS (multiple versions)
- PalmOS
- Nokia mobile phones
- Windows CE
- Acorn/RISC OS
- BeOS
- Amiga
- VMS/OpenVMS
- QNX
- VxWorks
- Psion
- Python has also been ported to the Java and .NET virtual machines

Getting Python

The most up-to-date and current source code, binaries, documentation, news, etc., is available on the official website of Python: <http://www.python.org/>.

You can download Python documentation from www.python.org/doc/. The documentation is available in HTML, PDF, and PostScript formats.

Installing Python

Python distribution is available for a wide variety of platforms. You need to download only the binary code applicable for your platform and install Python.

If the binary code for your platform is not available, you need a C compiler to compile the source code manually. Compiling the source code offers more flexibility in terms of choice of features that you require in your installation.

Here is a quick overview of installing Python on various platforms:

Unix and Linux Installation

Here are the simple steps to install Python on Unix/Linux machine.

- Open a Web browser and go to <http://www.python.org/download/>.
- Follow the link to download zipped source code available for Unix/Linux.
- Download and extract files.
- Editing the *Modules/Setup* file if you want to customize some options.
- **run** `./configure` script
- **make**
- **make install**

This installs Python at standard location `/usr/local/bin` and its libraries at `/usr/local/lib/pythonXX` where XX is the version of Python.

Windows Installation

Here are the steps to install Python on Windows machine.

- Open a Web browser and go to <http://www.python.org/download/>
- Follow the link for the Windows installer *python-XYZ.msi* file where XYZ is the version you need to install.
- To use this installer *python-XYZ.msi*, the Windows system must support Microsoft Installer 2.0. Save the installer file to your local machine and then run it to find out if your machine supports MSI.
- Run the downloaded file. This brings up the Python install wizard, which is really easy to use. Just accept the default settings, wait until the install is finished, and you are done.

Macintosh Installation

Recent Macs come with Python installed, but it may be several years out of date. See <http://www.python.org/download/mac/> for instructions on getting the current version along with extra tools to support development on the Mac. For older Mac OS's before Mac OS X 10.3 (released in 2003), MacPython is available.

Jack Jansen maintains it and you can have full access to the entire documentation at his website - <http://www.cwi.nl/~jack/macpython.html>. You can find complete installation details for Mac OS installation.

Setting up PATH

Programs and other executable files can be in many directories, so operating systems provide a search path that lists the directories that the OS searches for executables.

The path is stored in an environment variable, which is a named string maintained by the operating system. This variable contains information available to the command shell and other programs.

The **path** variable is named as PATH in Unix or Path in Windows (Unix is case-sensitive; Windows is not).

In Mac OS, the installer handles the path details. To invoke the Python interpreter from any particular directory, you must add the Python directory to your path.

Setting path at Unix/Linux

To add the Python directory to the path for a particular session in Unix:

- **In the csh shell:** type `setenv PATH "$PATH:/usr/local/bin/python"` and press Enter.
- **In the bash shell (Linux):** type `export PATH="$PATH:/usr/local/bin/python"` and press Enter.
- **In the sh or ksh shell:** type `PATH="$PATH:/usr/local/bin/python"` and press Enter.

Note: /usr/local/bin/python is the path of the Python directory

Setting path at Windows

To add the Python directory to the path for a particular session in Windows:

At the command prompt: type `path %path%;C:\Python` and press Enter.

Note: C:\Python is the path of the Python directory

Python Environment Variables

Here are important environment variables, which can be recognized by Python:

Variable	Description
PYTHONPATH	It has a role similar to PATH. This variable tells the Python interpreter where to locate the module files imported into a program. It should include the Python source library directory and the directories

	containing Python source code. PYTHONPATH is sometimes preset by the Python installer.
PYTHONSTARTUP	It contains the path of an initialization file containing Python source code. It is executed every time you start the interpreter. It is named as .pythonrc.py in Unix and it contains commands that load utilities or modify PYTHONPATH.
PYTHONCASEOK	It is used in Windows to instruct Python to find the first case-insensitive match in an import statement. Set this variable to any value to activate it.
PYTHONHOME	It is an alternative module search path. It is usually embedded in the PYTHONSTARTUP or PYTHONPATH directories to make switching module libraries easy.

Running Python

There are three different ways to start Python:

(1) Interactive Interpreter

You can start Python from Unix, DOS, or any other system that provides you a command-line interpreter or shell window.

Enter **python** the command line.

Start coding right away in the interactive interpreter.

<code>\$python</code>	<code># Unix/Linux</code>
<code>or</code>	
<code>python%</code>	<code># Unix/Linux</code>
<code>or</code>	
<code>C:>python</code>	<code># Windows/DOS</code>

Here is the list of all the available command line options:

Option	Description
-d	It provides debug output.
-O	It generates optimized bytecode (resulting in .pyo files).
-S	Do not run import site to look for Python paths on startup.
-v	verbose output (detailed trace on import statements).
-X	disable class-based built-in exceptions (just use strings); obsolete starting with version 1.6.
-c cmd	run Python script sent in as cmd string
file	run Python script from given file

(2) Script from the Command-line

A Python script can be executed at command line by invoking the interpreter on your application, as in the following:

```
$python script.py          # Unix/Linuxor
python% script.py          # Unix/Linuxor C:>python script.py      #
Windows/DOS
```

Note: Be sure the file permission mode allows execution.

(3) Integrated Development Environment

You can run Python from a Graphical User Interface (GUI) environment as well, if you have a GUI application on your system that supports Python.

- **Unix:** IDLE is the very first Unix IDE for Python.
- **Windows:** PythonWin is the first Windows interface for Python and is an IDE with a GUI.
- **Macintosh:** The Macintosh version of Python along with the IDLE IDE is available from the main website, downloadable as either MacBinary or BinHex'd files.

If you are not able to set up the environment properly, then you can take help from your system admin. Make sure the Python environment is properly set up and working perfectly fine.

Note: All the examples given in subsequent chapters are executed with Python 2.4.3 version available on CentOS flavor of Linux.

We already have set up Python Programming environment online, so that you can execute all the available examples online at the same time when you are learning theory. Feel free to modify any example and execute it online.

End of ebook preview

If you liked what you saw...

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