PYTHON 3

tutorialspoint

SIMPLYEASYLEARNING

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). Python is named after a TV Show called 'Monty Python's Flying Circus' and not after Python-the snake.

Python 3.0 was released in 2008. Although this version is supposed to be backward incompatibles, later on many of its important features have been backported to be compatible with the version 2.7. This tutorial gives enough understanding on Python 3 version programming language. Please refer to this link for our Python 2 tutorial.

Audience

This tutorial is designed for software programmers who want to upgrade their Python skills to Python 3. This tutorial can also be used 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.

Execute Python Programs

For most of the examples given in this tutorial you will find **Try it** option, so just make use of it and enjoy your learning.

Try the following example using **Try it** option available at the top right corner of the below sample code box –

```
#!/usr/bin/python3

print ("Hello, Python!")
```

Copyright & Disclaimer

Table of Contents

About the Tutorial i
Audience i
Prerequisites i
Execute Python Programs i
Copyright & Disclaimer i
Table of Contents ii

PYTHON 3 – BASIC TUTORIAL 1

1. Python 3 – What is New? 2
The __future__ module 2
The print Function 2
Reading Input from Keyboard 2
Integer Division 3
Unicode Representation 3
xrange() Function Removed 4
raise exceprion 4
Arguments in Exceptions 4
next() Function and .next() Method 4
2to3 Utility 4

2. Python 3 – Overview 6
History of Python 6
Python Features 7

3. Python 3 – Environment Setup 8
Local Environment Setup 8
Getting Python 8
Setting up PATH 9
Setting Path at Unix/Linux 10
Setting Path at Windows 10
Python Environment Variables 10
Running Python 11

4. Python 3 – Basic Syntax 13
First Python Program 13
Python Identifiers 14
Reserved Words 15
Lines and Indentation 15
Multi-Line Statements 17
Quotation in Python 17
Comments in Python 17
Using Blank Lines 18
Waiting for the User 18
Multiple Statements on a Single Line 18
Multiple Statement Groups as Suites 19
Command Line Arguments 19

Parsing Command-Line Arguments 20

5. Python 3 – Variable Types 23
Assigning Values to Variables 23
Multiple Assignment 23
Standard Data Types 24
Python Numbers 24
Python Strings 25
Python Lists 26
Python Tuples 27
Python Dictionary 27
Data Type Conversion 28

6. Python 3 – Basic Operators 30
Types of Operator 30
Python Arithmetic Operators 30
Python Comparison Operators 32
Python Assignment Operators 33
Python Bitwise Operators 35
Python Logical Operators 37
Python Membership Operators 38
Python Identity Operators 39
Python Operators Precedence 40

7. Python 3 – Decision Making 43
IF Statement 44
IF...ELIF...ELSE Statements 45
Nested IF Statements 48
Single Statement Suites 49

8. Python 3 – Loops 51
while Loop Statements 52
for Loop Statements 56
Nested loops 59
Loop Control Statements 60
break statement 61
continue Statement 63
pass Statement 65
Iterator and Generator 66

9. Python 3 – Numbers 68
Mathematical Functions 70
Number abs() Method 71
Number ceil() Method 71
Number exp() Method 72
Number fabs() Method 73
Number floor() Method 74
Number log() Method 75
Number log10() Method 76
Number max() Method 77
Number min() Method 78
Number modf() Method 79

Number pow() Method 80
Number round() Method 80
Number sqrt() Method 81
Random Number Functions 82
Number choice() Method 82
Number randrange() Method 83
Number random() Method 84
Number seed() Method 85
Number shuffle() Method 86
Number uniform() Method 87
Trigonometric Functions 88
Number acos() Method 88
Number asin() Method 89
Number atan() Method 90
Number atan2() Method 91
Number cos() Method 92
Number hypot() Method 93
Number sin() Method 94
Number tan() Method 95
Number degrees() Method 96
Number radians() Method 97
Mathematical Constants 98

10. Python 3 – Strings 99
Accessing Values in Strings 99
Updating Strings 99
Escape Characters 100
String Special Operators 101
String Formatting Operator 102
Triple Quotes 104
Unicode String 105
String capitalize() Method 109
String center() Method 110
String count() Method 111
String decode() Method 112
String encode() Method 112
String endswith() Method 113
String expandtabs() Method 114
String find() Method 115
String index() Method 116
String isalnum() Method 117
String isalpha() Method 118
String isdigit() Method 118
String islower() Method 119
String isnumeric() Method 120
String isspace() Method 121
String istitle() Method 122
String isupper() Method 122
String join() Method 123
String len() Method 124
String ljust() Method 125
String lower() Method 125

String lstrip() Method 126
String maketrans() Method 127
String max() Method 128
String min() Method 129
String replace() Method 129
String rfind() Method 130
String rindex() Method 131
String rjust() Method 132
String rstrip() Method 133
String split() Method 134
String splitlines() Method 135
String startswith() Method 135
String strip() Method 136
String swapcase() Method 137
String title() Method 138
String translate() Method 138
String upper() Method 140
String zfill() Method 140
String isdecimal() Method 141

11. Python 3 – Lists 143
Python Lists 143
Accessing Values in Lists 143
Updating Lists 144
Delete List Elements 144
Basic List Operations 144
Indexing, Slicing and Matrixes 145
Built-in List Functions & Methods 145
List len() Method 146
List max() Method 147
List min() Method 147
List list() Method 148
List append() Method 150
List count() Method 151
List extend() Method 151
List index() Method 152
List insert() Method 153
List pop() Method 154
List remove() Method 154
List reverse() Method 155
List sort() Method 156

12. Python 3 – Tuples 157
Accessing Values in Tuples 157
Updating Tuples 158
Delete Tuple Elements 158
Basic Tuples Operations 159
Indexing, Slicing, and Matrixes 159
No Enclosing Delimiters 160
Built-in Tuple Functions 160
Tuple len() Method 160
Tuple max() Method 161

Tuple min() Method 162
Tuple tuple() Method 162

13. Python 3 – Dictionary 164
Accessing Values in Dictionary 164
Updating Dictionary 165
Delete Dictionary Elements 165
Properties of Dictionary Keys 166
Built-in Dictionary Functions & Methods 167
Dictionary len() Method 167
Dictionary str() Method 168
Dictionary type() Method 168
Dictionary clear() Method 170
Dictionary copy() Method 171
Dictionary fromkeys() Method 172
Dictionary get() Method 172
Dictionary items() Method 173
Dictionary keys() Method 174
Dictionary setdefault() Method 174
Dictionary update() Method 175
Dictionary values() Method 176

14. Python 3 – Date & Time 178
What is Tick? 178
What is TimeTuple? 178
Getting current time 180
Getting formatted time 180
Getting calendar for a month 180
The time Module 181
Time altzone() Method 182
Time asctime() Method 183
Time clock() Method 184
Time ctime() Method 185
Time gmtime() Method 186
Time localtime() Method 187
Time mktime() Method 187
Time sleep() Method 188
Time strftime() Method 189
Time strptime() Method 191
Time time() Method 193
Time tzset() Method 194
The calendar Module 196
Other Modules & Functions 198

15. Python 3 – Functions 199
Defining a Function 199
Calling a Function 200
Pass by Reference vs Value 200
Function Arguments 202
Required Arguments 202
Keyword Arguments 202
Default Arguments 203

Variable-length Arguments 204
The Anonymous Functions 205
The return Statement 206
Global vs. Local variables 206

16. Python 3 – Modules 208
The import Statement 208
The from...import Statement 209
The from...import * Statement: 209
Executing Modules as Scripts 209
Locating Modules 210
The PYTHONPATH Variable 210
Namespaces and Scoping 211
The dir( ) Function 212
The globals() and locals() Functions 212
The reload() Function 212
Packages in Python 213

17. Python 3 – Files I/O 215
Printing to the Screen 215
Reading Keyboard Input 215
The input Function 215
Opening and Closing Files 216
The open Function 216
The file Object Attributes 217
The close() Method 218
Reading and Writing Files 219
The write() Method 219
The read() Method 220
File Positions 220
Renaming and Deleting Files 221
The rename() Method 221
The remove() Method 222
Directories in Python 222
The mkdir() Method 222
The chdir() Method 223
The getcwd() Method 223
The rmdir() Method 224
File & Directory Related Methods 224
File Methods 224
File close() Method 226
File flush() Method 227
File fileno() Method 228
File isatty() Method 228
File next() Method 229
File read() Method 231
File readline() Method 232
File readlines() Method 233
File seek() Method 234
File tell() Method 236
File truncate() Method 237

File write() Method 238
File writelines() Method 240
OS File/Directory Methods 241
os.access() Method 248
os.chdir() Method 250
os.chflags() Method 251
os.chmod() Method 252
os.chown() Method 254
os.chroot() Method 255
Python os.close() Method 255
os.closerange() Method 256
os.dup() Method 258
os.dup2() Method 259
os.fchdir() Method 260
os.fchmod() Method 261
os.fchown() Method 263
os.fdatasync() Method 264
os.fdopen() Method 266
os.fpathconf() Method 267
os.fstat() Method 269
os.fstatvfs() Method 270
os.fsync() Method 272
os.ftruncate() Method 273
os.getcwd() Method 274
os.getcwdu() Method 275
os.isatty() Method 276
os.lchflags() Method 278
os.lchown() Method 279
os.link() Method 280
os.listdir() Method 281
os.lseek() Method 282
os.lstat() Method 284
os.major() Method 286
os.makedev() Method 286
os.makedirs() Method 288
os.minor() Method 288
os.mkdir() Method 289
os.mkfifo() Method 290
os.mknod() Method 291
os.open() Method 292
os.openpty() Method 293
os.pathconf() Method 294
os.pipe() Method 296
os.popen() Method 297
os.read() Method 298
os.readlink() Method 299
os.remove() Method 300
os.removedirs() Method 301
os.rename() Method 302
os.renames() Method 303
os.renames() Method 304
os.rmdir() Method 305

os.stat() Method 307
os.stat_float_times() Method 308
os.statvfs() Method 309
os.symlink() Method 310
os.tcgetpgrp() Method 311
os.tcsetpgrp() Method 312
os.tempnam() Method 313
os.tmpfile() Method 314
os.tmpnam() Method 315
os.ttyname() Method 315
os.unlink() Method 317
os.utime() Method 318
os.walk() Method 319
os.write() Method 321

18. Python 3 – Exceptions Handling 323
Standard Exceptions 323
Assertions in Python 325
What is Exception? 326
Handling an Exception 326
The except Clause with No Exceptions 328
The except Clause with Multiple Exceptions 328
The try-finally Clause 329
Argument of an Exception 330
Raising an Exception 331
User-Defined Exceptions 332

PYTHON 3 – ADVANCED TUTORIAL 333

19. Python 3 – Object Oriented 334
Overview of OOP Terminology 334
Creating Classes 335
Creating Instance Objects 336
Accessing Attributes 336
Built-In Class Attributes 337
Destroying Objects (Garbage Collection) 339
Class Inheritance 340
Overriding Methods 342
Base Overloading Methods 342
Overloading Operators 343
Data Hiding 344

20. Python 3 – Regular Expressions 346
The match Function 347
The search Function 348
Matching Versus Searching 349
Search and Replace 350
Regular Expression Modifiers: Option Flags 350
Regular Expression Patterns 351
Regular Expression Examples 353
Character classes 354

Special Character Classes 354
Repetition Cases 355
Nongreedy Repetition 355
Grouping with Parentheses 355
Backreferences 356
Alternatives 356
Anchors 356
Special Syntax with Parentheses 357

21. Python 3 – CGI Programming 358
What is CGI? 358
Web Browsing 358
CGI Architecture Diagram 359
Web Server Support and Configuration 359
First CGI Program 360
HTTP Header 361
CGI Environment Variables 361
GET and POST Methods 363
Passing Information using GET method 363
Simple URL Example – Get Method 363
Simple FORM Example – GET Method 364
Passing Radio Button Data to CGI Program 367
Passing Text Area Data to CGI Program 368
Passing Drop Down Box Data to CGI Program 369
Using Cookies in CGI 370
How It Works? 370
Setting up Cookies 371
Retrieving Cookies 371
File Upload Example 372
How To Raise a "File Download" Dialog Box ? 374

22. Python 3 – MySQL Database Access 375
What is PyMySQL ? 375
How do I Install PyMySQL? 376
Database Connection 376
Creating Database Table 377
INSERT Operation 378
READ Operation 380
Update Operation 382
DELETE Operation 383
Performing Transactions 383
COMMIT Operation 384
ROLLBACK Operation 384
Disconnecting Database 384
Handling Errors 385

23. Python 3 – Network Programming 387
What is Sockets? 387
The socket Module 388
Server Socket Methods 388
Client Socket Methods 388
General Socket Methods 389

A Simple Server 389
A Simple Client 390
Python Internet Modules 391
Further Readings 392

24. Python 3 – Sending Email using SMTP 393
Sending an HTML e-mail using Python 394
Sending Attachments as an E-mail 395

25. Python 3 – Multithreaded Programming 398
Starting a New Thread 398
The Threading Module 400
Creating Thread Using Threading Module 400
Synchronizing Threads 402
Multithreaded Priority Queue 404

26. Python 3 – XML Processing 407
What is XML? 407
XML Parser Architectures and APIs 407
Parsing XML with SAX APIs 408
The make_parser Method 409
The parse Method 409
The parseString Method 409
Parsing XML with DOM APIs 412

27. Python 3 – GUI Programming (Tkinter) 415
Tkinter Programming 415
Tkinter Widgets 416
Tkinter Button 418
Tkinter Canvas 420
Tkinter Checkbutton 423
Tkinter Entry 427
Tkinter Frame 431
Tkinter Label 433
Tkinter Listbox 435
Tkinter Menubutton 439
Tkinter Menu 442
Tkinter Message 446
Tkinter Radiobutton 449
Tkinter Scale 453
Tkinter Scrollbar 457
Tkinter Text 460
Tkinter Toplevel 464
Tkinter Spinbox 467
Tkinter PanedWindow 471
Tkinter LabelFrame 473
Tkinter tkMessageBox 475
Standard Attributes 477
Tkinter Dimensions 477
Tkinter Colors 478
Tkinter Fonts 479
Tkinter Anchors 480

Tkinter Relief styles 481
Tkinter Bitmaps 482
Tkinter Cursors 484
Geometry Management 485
Tkinter pack() Method 486
Tkinter grid() Method 487
Tkinter place() Method 488

28. Python 3 – Extension Programming with C 490
Pre-Requisites for Writing Extensions 490
First look at a Python Extension 490
The Header File Python.h 490
The C Functions 491
The Method Mapping Table 491
The Initialization Function 492
Building and Installing Extensions 494
Importing Extensions 494
Passing Function Parameters 495
The PyArg_ParseTuple Function 496
Returning Values 497
The Py_BuildValue Function 498

Python 3 – Basic Tutorial

1. Python 3 – What is New?

The __future__ module

Python 3.x introduced some Python 2-incompatible keywords and features that can be imported via the in-built __future__ module in Python 2. It is recommended to use __future__ imports, if you are planning Python 3.x support for your code.

For example, if we want Python 3.x's integer division behavior in Python 2, add the following import statement.

```
from __future__ import division
```

The print Function

Most notable and most widely known change in Python 3 is how the **print** function is used. Use of parenthesis () with print function is now mandatory. It was optional in Python 2.

```
print "Hello World" #is acceptable in Python 2
print ("Hello World") # in Python 3, print must be followed by ()
```

The print() function inserts a new line at the end, by default. In Python 2, it can be suppressed by putting ',' at the end. In Python 3, "end=' '" appends space instead of newline.

```
print x,           # Trailing comma suppresses newline in Python 2
print(x, end=" ")  # Appends a space instead of a newline in Python 3
```

Reading Input from Keyboard

Python 2 has two versions of input functions, **input()** and **raw_input()**. The input() function treats the received data as string if it is included in quotes '' or "", otherwise the data is treated as number.

In Python 3, raw_input() function is deprecated. Further, the received data is always treated as string.

```
In Python 2
>>> x=input('something:')
something:10 #entered data is treated as number
>>> x
10
>>> x=input('something:')
```

```
something:'10' #eentered data is treated as string
>>> x
'10'
>>> x=raw_input("something:")
something:10 #entered data is treated as string even without ''
>>> x
'10'
>>> x=raw_input("something:")
something:'10' #entered data treated as string including ''
>>> x
"'10'"
In Python 3
>>> x=input("something:")
something:10
>>> x
'10'
>>> x=input("something:")
something:'10' #entered data treated as string with or without ''
>>> x
"'10'"
>>> x=raw_input("something:") # will result NameError
Traceback (most recent call last):
  File "", line 1, in

    x=raw_input("something:")
NameError: name 'raw_input' is not defined
```

Integer Division

In Python 2, the result of division of two integers is rounded to the nearest integer. As a result, 3/2 will show 1. In order to obtain a floating-point division, numerator or denominator must be explicitly used as float. Hence, either 3.0/2 or 3/2.0 or 3.0/2.0 will result in 1.5

Python 3 evaluates 3 / 2 as 1.5 by default, which is more intuitive for new programmers.

Unicode Representation

Python 2 requires you to mark a string with a **u** if you want to store it as Unicode.

Python 3 stores strings as Unicode, by default. We have Unicode (utf-8) strings, and 2 byte classes: byte and byte arrays.

xrange() Function Removed

In Python 2 range() returns a list, and xrange() returns an object that will only generate the items in the range when needed, saving memory.

In Python 3, the range() function is removed, and xrange() has been renamed as range(). In addition, the range() object supports slicing in Python 3.2 and later .

raise exceprion

Python 2 accepts both notations, the 'old' and the 'new' syntax; Python 3 raises a SyntaxError if we do not enclose the exception argument in parenthesis.

```
raise IOError, "file error" #This is accepted in Python 2
raise IOError("file error") #This is also accepted in Python 2
raise IOError, "file error" #syntax error is raised in Python 3
raise IOError("file error") #this is the recommended syntax in Python 3
```

Arguments in Exceptions

In Python 3, arguments to exception should be declared with 'as' keyword.

```
except Myerror, err: # In Python2
except Myerror as err: #In Python 3
```

next() Function and .next() Method

In Python 2, next() as a method of generator object, is allowed. In Python 2, the next() function, to iterate over generator object, is also accepted. In Python 3, however, next(0 as a generator method is discontinued and raises **AttributeError.**

```
gen = (letter for letter in 'Hello World') # creates generator object
next(my_generator) #allowed in Python 2 and Python 3
my_generator.next() #allowed in Python 2. raises AttributeError in Python 3
```

2to3 Utility

Along with Python 3 interpreter, 2to3.py script is usually installed in tools/scripts folder. It reads Python 2.x source code and applies a series of fixers to transform it into a valid Python 3.x code.

```
Here is a sample Python 2 code (area.py):
def area(x,y=3.14):
    a=y*x*x
    print a
    return a
```

```
a=area(10)
print "area",a
To convert into Python 3 version:
$2to3 -w area.py
Converted code :
def area(x,y=3.14): # formal parameters
    a=y*x*x
    print (a)
    return a
a=area(10)
print("area",a)
```

2. Python 3 – 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 whereas the other languages use punctuations. 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, and 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 1.0 was released in November 1994. In 2000, Python 2.0 was released. Python 2.7.11 is the latest edition of Python 2.

- Meanwhile, Python 3.0 was released in 2008. Python 3 is not backward compatible with Python 2. The emphasis in Python 3 had been on the removal of duplicate programming constructs and modules so that "There should be one -- and preferably only one -- obvious way to do it." Python 3.5.1 is the latest version of Python 3.

Python Features

Python's features include-

- **Easy-to-learn:** Python has few keywords, simple structure, and a clearly defined syntax. This allows a 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. A 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.

3. Python 3 – Environment Setup

Try it Option Online

We have set up the Python Programming environment online, so that you can compile and execute all the available examples online. It will give you the confidence in what you are reading and will enable you to verify the programs with different options. Feel free to modify any example and execute it online.

Try the following example using our online compiler available at CodingGround

```
#!/usr/bin/python3

print ("Hello, Python!")
```

For most of the examples given in this tutorial, you will find a **Try it** option on our website code sections, at the top right corner that will take you to the online compiler. Just use it and enjoy your learning.

Python 3 is available for Windows, Mac OS and most of the flavors of Linux operating system. Even though Python 2 is available for many other OSs, Python 3 support either has not been made available for them or has been dropped.

Local Environment Setup

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

Getting Python

Windows platform

Binaries of latest version of Python 3 (Python 3.5.1) are available on this download page

The following different installation options are available.

- Windows x86-64 embeddable zip file
- Windows x86-64 executable installer
- Windows x86-64 web-based installer
- Windows x86 embeddable zip file
- Windows x86 executable installer
- Windows x86 web-based installer

Note: In order to install Python 3.5.1, minimum OS requirements are Windows 7 with SP1. For versions 3.0 to 3.4.x, Windows XP is acceptable.

Linux platform

Different flavors of Linux use different package managers for installation of new packages.

On Ubuntu Linux, Python 3 is installed using the following command from the terminal.

```
$sudo apt-get install python3-minimal
```

Installation from source

```
Download Gzipped source tarball from Python's download URL:
https://www.python.org/ftp/python/3.5.1/Python-3.5.1.tgz
Extract the tarball
tar xvfz Python-3.5.1.tgz
Configure and Install:
cd Python-3.5.1
./configure --prefix=/opt/python3.5.1
make
sudo make install
```

Mac OS

Download Mac OS installers from this URL:https://www.python.org/downloads/mac-osx/

- Mac OS X 64-bit/32-bit installer : python-3.5.1-macosx10.6.pkg
- Mac OS X 32-bit i386/PPC installer : python-3.5.1-macosx10.5.pkg

Double click this package file and follow the wizard instructions to install.

The most up-to-date and current source code, binaries, documentation, news, etc., is available on the official website of Python:

Python Official Website : http://www.python.org/

You can download Python documentation from the following site. The documentation is available in HTML, PDF and PostScript formats.

Python Documentation Website : www.python.org/doc/

Setting up PATH

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

The important features are-

- 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/python3" and press Enter.

- **In the bash shell (Linux):** type export PATH="$PATH:/usr/local/bin/python3" and press Enter.

- **In the sh or ksh shell:** type PATH="$PATH:/usr/local/bin/python3" and press Enter.

Note: /usr/local/bin/python3 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 are 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.

```
$python           # Unix/Linux
or
python%           # Unix/Linux
or
C:>python         # Windows/DOS
```

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

Option	Description
-d	provide debug output
-O	generate 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 the command line by invoking the interpreter on your application, as shown in the following example.

```
$python  script.py          # Unix/Linux
or
python% script.py           # Unix/Linux
or
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 the help of 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 3.4.1 version available on Windows 7 and Ubuntu Linux.

We have already set up Python Programming environment online, so that you can execute all the available examples online while 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**