

Python Documentation: Tables & Lists

Functions * **boldface** not covered in this toolbox

abs()
all()
any()
ascii()
bin()
bool()
breakpoint()
bytearray()
bytes()

callable()
chr()
classmethod()
compile()
complex()
delattr()
dict()
dir()
divmod()

enumerate()
eval()
exec()
filter()
float()
format()
frozenset()
getattr()
globals()

hasattr()
hash()
help()
hex()
id()
input()
int()
isinstance()
issubclass()
iter()
len()

list()
locals()
map()
max()
memoryview()
min()
next()
object()
oct()
open()
ord()

pow()
print()
property()
range()
repr()
reversed()
round()
set()
setattr()
slice()
sorted()

staticmethod()
str()
sum()
super()
tuple()
type()
vars()
zip()
__import__()

Comparisons

Operation	Meaning
<	strictly less than
<=	less than or equal
>	strictly greater than
>=	greater than or equal
==	equal
!=	not equal
is	object identity
is not	negated object identity

Sequence Operations (4.6.1)

x in s
True if an item of s is equal to x, else False

x not in s
False if an item of s is equal to x, else True

s + t the concatenation of s and t

s * n or n * s
equivalent to adding s to itself n times

s[i] ith item of s, origin 0

s[i:j] slice of s from i to j

s[i:j:k] slice of s from i to j with step k

len(s) length of s

min(s) smallest item of s

max(s) largest item of s

s.index(x[, i[, j]]) index of the first occurrence of x in s (at or after index i and before index j)

s.count(x) number of occurrences of x in s

Mutable Sequence Operations

s[i] = x item i of s is replaced by x

s[i:j] = t slice of s from i to j is replaced by the contents of the iterable t

del s[i:j] same as **s[i:j] = []**

s[i:j:k] = t the elements of s[i:j:k] are replaced by those of t

del s[i:j:k] removes the elements of s[i:j:k] from the list

s.append(x) appends x to the end of the sequence

s.clear() removes all items from s (same as **del s[:]**)

s.copy() creates a shallow copy of s (same as **s[:]**)

s.extend(t) or **s +=** extends s with the contents of t (for the most part the same as **[len(s):len(s)] = t**)

s *= n updates s with its contents repeated n times

s.insert(i, x) inserts x into s at the index given by i (same as **s[i] = [x]**)

s.pop([i]) retrieves the item at i and also removes it from s

s.remove(x) remove the first item from s where s[i] == x

s.reverse() reverses the items of s in place

For important notes see:
<https://docs.python.org/3.6/library/stdtypes.html>

Boolean Operations

Operation Result (ascending priority)

x or y if x is false, then y, else x

x and y if x is false, then x, else y

not x if x is false, True, else False

Bitwise Operations on Integers

Operation Result

x | y bitwise *or* of x and y

x ^ y bitwise *exclusive or* of x and y

x & y bitwise *and* of x and y

x << n x shifted left by n bits

x >> n x shifted right by n bits

~x the bits of x inverted

comments and suggestions appreciated:
john@johnoakey.com

Numeric Type Operations

Operation	Result
x + y	sum of x and y
x - y	difference of x and y
x * y	product of x and y
x / y	quotient of x and y
x // y	floored quotient of x and y
x % y	remainder of x / y
-x	x negated
+x	x unchanged
abs(x)	absolute value or magnitude of x
int(x)	x converted to integer
float(x)	x converted to floating point
complex(re, im)	a complex number with real part re, imaginary part im. defaults to zero.
c.conjugate()	conjugate of the complex number c
divmod(x, y)	the pair (x // y, x % y)
pow(x, y)	x to the power y
x ** y	x to the power y

notes: <https://docs.python.org/3.6/library/stdtypes.html>

f-string Formatting : conversion types

'd' Signed integer decimal.
'i' Signed integer decimal.
'o' Signed octal value.
'u' Obsolete type – it is identical to 'd'.
'x' Signed hexadecimal (lowercase).
'X' Signed hexadecimal (uppercase).
'e' Floating point exponential format (lowercase).
'E' Floating point exponential format (uppercase).
'f' Floating point decimal format.
'F' Floating point decimal format.
'g' Floating point format. Uses lowercase exponential format if exponent is less than -4 or not less than precision, decimal format otherwise.
'G' Floating point format. Uses uppercase exponential format if exponent is less than -4 or not less than precision, decimal format otherwise.
'c' Single character (accepts integer or single character string).
'r' String (converts any Python object using repr()).
's' String (converts any Python object using str()).
'a' String (converts any Python object using ascii()).
'%' No argument is converted, results in a '%' character in the result.

Keywords

and	as	assert	async	await	break	class
continue	def	del	elif	else	except	False
for	from	global	if	import	in	is
nonlocal	None	not	or	pass	raise	return
try	while	with	yield			True

(keywords = reserved words)

Operator Precedence

Lambda (Multiplication, matrix multiplication, division, floor division, remainder) **+x, -x, ~x**

if – else (Positive, negative, bitwise NOT) ****** (exponentiation)

or/and/not x **await x** (Await expression)

in, not in, is, is not, **x[index], x[index:index], x** (arguments..., x.attribute)

<, <=, >, >=, !=, == (subscription, slicing, call, attribute reference)

|/^/&

<<, >>

+, -

***, @, /, //, %**

Open File Modes

Character	Meaning
'r'	open for reading (default)
'w'	open for writing, truncating the file first
'x'	open for exclusive creation, fails if it already exists
'a'	open for writing, appending to the end of the file if it exists
'b'	binary mode
't'	text mode (default)
'+'	open a disk file for updating (reading and writing)
'U'	universal newlines mode (deprecated)

Built-in Constants

False, True, None, NotImplemented, Ellipsis (same as literal '...'), __debug__, quit(), exit(), copyright, credits, license

f-string : conversion flags

'#' conversion will use the "alternate form"

'0' conversion zero padded for numerics

'.' value is left adjusted (overrides the '0')

' ' (space) A blank should be left before a + number (or empty string)

'+' A sign character ('+' or '-') will precede the conversion (overrides a "space" flag).

Built-in Types

numerics, sequences, mappings, classes, instances, exceptions

Escape Sequences

\ newline

\\ Backslash (\)

\' Single quote (')

\\" Double quote (")

\a ASCII Bell (BEL)

\b ASCII Backspace (BS)

\f ASCII Formfeed (FF)

\n ASCII Linefeed (LF)

\r ASCII Carriage Return (CR)

\t ASCII Horizontal Tab (TAB)

\v ASCII Vertical Tab (VT)

\ooo Character with octal value ooo (1,3)

\xhh Character with hex value hh (2,3)

www.wikipython.com

The real power of Python is its transformer-like ability to add functions and abilities to fit just about any conceived programming need. This is done through the importation of specialized **MODULES** that integrate with, and extend, Python; adding abilities that become part of the program. About 230 of these modules are downloaded automatically when Python is installed. If you can't find what you need in this "Standard Library", there are over another 1,000,000 packages contributed by users in the PyPi online storage waiting for your consideration. A few highlights of the modules in the "The Python Standard Library" and a couple of others in PyPi are noted below. Find PyPi at: <https://pypi.org/>

The Python Standard Library

Text Processing Services - 7 modules including:

string — Common string operations
re — Regular expression operations
textwrap — Text wrapping and filling

Binary Data Services - 2 modules

Data Types - 13 modules including:

datetime — Basic date and time types
calendar — General calendar-related functions
collections — Container datatypes
array — Efficient arrays of numeric values

Numeric and Mathematical Modules - 7 modules including:

numbers — Numeric abstract base classes
math — Mathematical functions
decimal — Decimal fixed point and floating-point arithmetic

random — Generate pseudo-random numbers
statistics — Mathematical statistics functions

Functional Programming Modules - 3 modules:

File and Directory Access - 11 modules including:
pathlib — Object-oriented filesystem paths
os.path — Common pathname manipulations
shutil — High-level file operations

Data Persistence - 6 modules including:

pickle — Python object serialization
marshal — Internal Python object serialization
sqlite3 — DB-API 2.0 interface for SQLite databases

Data Compression and Archiving - 6 modules including:

zipfile — Work with ZIP archives
tarfile — Read and write tar archive files

File Formats - 5 modules including:

csv — CSV File Reading and Writing

Cryptographic Services - 3 modules:

Generic Operating System Services - 16 modules including:

os — Miscellaneous operating system interfaces
time — Time access and conversions
curses — Terminal handling for character-cell displays

Concurrent Execution - 10 modules including:

threading — Thread-based parallelism
multiprocessing — Process-based parallelism

Interprocess Communication and Networking - 9 modules:

Internet Data Handling - 10 modules:

Structured Markup Processing Tools - 13 modules:

Internet Protocols and Support - 21 modules:

Multimedia Services - 9 modules including:

wave — Read and write WAV files

Internationalization - 2 modules:

Program Frameworks - 3 modules including:

turtle — Turtle graphics

Graphical User Interfaces with Tk - 6 modules including:

tkinter — Python interface to Tcl/Tk
IDLE

Development Tools - 9 modules:

Debugging and Profiling - 7 modules:

Software Packaging and Distribution - 4 modules including:

distutils — Building and installing Python modules

Python Runtime Services - 14 modules including:

sys — System-specific parameters and functions
sysconfig — Provide access to Python's configuration information

__main__ — Top-level script environment

inspect — Inspect live objects

Custom Python Interpreters - 2 modules:

Importing Modules - 5 modules including:

zipimport — Import modules from Zip archives

Python Language Services - 13 modules:

Miscellaneous Services - 1 module:

MS Windows Specific Services - 4 modules including:

winsound — Sound-playing interface for Windows

Unix Specific Services - 13 modules:

Superseded Modules - 2 modules:

Undocumented Modules - 1 module:

Cherrypicked Useful Standard Library Module Methods

calendar: many many functions; ex:
weekdays = ['M', 'Tu', 'W', 'Th', 'F', 'S', 'S']
print('birth day is a: ' + weekdays[calendar.weekday(1948, 1, 19)])
↳ birth day is a: M

copy: .copy(x), .deepcopy(x)

datetime: .date(year, month, day),
.date.today(), .datetime.now(),
.timedelta(days or seconds), ex:
start = datetime.date(2019, 1, 1)
duration = datetime.timedelta(days=180)
enddate = start + duration
print(enddate) ↳ 2019-06-30 *also in
PyPi see new python-dateutil module

decimal: accounting level precision,
from decimal import *
.Decimal(value="0", context=None) ex:
from decimal import *
import math
print(math.sqrt(2), '\n', Decimal(2).sqrt()) ↳
1.4142135623730951
1.414213562373095048801688724

math: .ceil(x), .fsum(iterable), .sqrt(x),
.log(x[,base]), .factorial(x), .floor(), .log(x[,base]), log1p(x), .sqrt(x), all trig
and hyperbolic functions constants: .pi, .e

pathlib: new in 3.5, Unless you understand the "PurePath" class, you want to use "concrete paths" and should import using "from pathlib import Path"; this is the assumption in the following where p = Path:
p.cwd() current directory; p.home();
p.exists(str); p.is_dir(); p.is_file();
p.iterdir() ↳ iterates directory paths

for file in p.iterdir(p.cwd()):
print(file) ↳ all files in working dir
p.mkdir(mode=0o777, parents=False, exist_ok=False) create new directory
FileExistsError if it already exists
p.open(mode='r', buffering=-1, encoding=None, errors=None, newline=None)
p.read_text(); p.rename(target);
p().resolve(strict=False) - make absolute path; p.glob(pattern) - creates
iterator for files filtered by pattern, "*" ↳ all dir and subdirs, "**.*" ↳ all files in path
dir, "**/*/*" ↳ all dir and their files
p.rglob(pattern) - like ** in front of .glob; p.rmdir() - remove empty directory; p.write_text(data,
encoding=None, errors=None) - open,
write, close - all in one fell swoop

os: os.environ['HOME'] home directory,
.chdir(path) change working dir, .getcwd()
current working dir, .listdir(path),
.mkdir(path), .remove(), .curdir,
note: os.path is a different module

random: .seed([x]), .choice(seq),

.randint(a, b), .random() - floating point
[0.0 to 1.0], reuse seed to reproduce value

sys: .exit([arg]), .argv, .exe_info(),
.getsizeof(object [,default]), .path,
.version, __stdin__, __stdout__

string: constants: ascii_letters,
ascii_lowercase, ascii_uppercase, digits,
hexdigits, octdigits, punctuation,
printable, whitespace

statistics: .mean(), .median(), .mode(),
.pstdev(), .pvariance(), p is for
population

time: sleep(secs), localtime(), clock(),
asctime(struct_time tuple)

wave: .open(file, mode='rb' or 'wb')
read or write, read_object.close(),
write_object.close()

**pickle tarfile shelve sqlite json
filecmp fileinput zipfile filecmp**

see **Data on Disk Toolbox**

Complex modules where single method examples are not useful:

tkinter: best gui but equivalent to learning Python twice - see 10 page
tkinter toolbox on www.wikipython.com

re: exigent find & match functions

collections: use mostly for named
tuples and ordered dictionaries

array: very fast, efficient, single type

turtle: intro graphics based on tkinter

Raspberry Pi Aficionados

Rpi.GPIO - module to control
Raspberry Pi GPIO channels - see GPIO
toolbox on www.wikipython.com,
download module from: <https://pypi.org/search/?q=rpi.gpio>

Selected Other PYPI Frequently Downloaded Packages

pip, pillow, numpy, python-dateutil,
doctils, pyasn1, setuptools (also see
pbr), jmespath 0.9.3, cryptography,
ipaddress, pytest, decorator py parsing,
psutil, flask, scipy, scikit-learn (requires
3.5, Numpy and SciPy), pandas, django,
cython, imagesize, pyserial, fuzzywuzzy,
multidict, yarl

**Can important key methods of your
favorite module be briefly
summarized? We would really like
to hear your suggestion(s)! email:**

oakey.john@yahoo.com

www.wikipython.com

"No registration, cookies, fees, contributions, ads, &
nobody looking for a job - secure site & downloads."