Python Standard Libraries Cheatsheet

Depend on Python v3.9.8

All code snippets have been tested to ensure they work properly.

Fork me on GitHub.

- 中文
- English

Notes:

- **Every code snippet here can run independently (some need the files provided by this repo)**
- **If you want to copy the code, use command to code(Temporarily unavailable)**
- **You can use GETREADME.py to download README.md from the repository (Chinese or English, with or not with command line prefixes is up to you!)**

Contents

Internet Protocols: `webbrowser`

Multimedia Services: `wave`, `sndhdr`, `imghdr`, `colorsys`

Program Frameworks: `turtle`

Graphical Interfaces: `tkinter`

Development Tools: `typing`, `doctest`

Debugging Profiling: `timeit`, `pdb`

Software Packaging: `ensurepip`, `zipapp`

Runtime Services: `sys`, `dataclasses`, `contextlib`, `abc`, `traceback`, `__future__`, `atexit`, `builtins`, `inspect`

Importing Modules: `zipimport`, `importlib`, `runpy`

Language Services: `ast`, `keyword`, `dis`, `tabnanny`

Bonus Scene: `this`, `antigravity`

string

Attributes

```
>>> import string
>>> string.ascii_letters
'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'
>>> string.ascii_lowercase
'abcdefghijklmnopqrstuvwxyz'
>>> string.ascii_uppercase
'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
>>> string.digits
'0123456789'
>>> string.hexdigits
'0123456789abcdefABCDEF'
>>> string.octdigits
'01234567'
>>> string.punctuation
'!"#$%&\'()*+,-./:;<=>?@[\\]^_`{|}~'
>>> string.printable
'0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ!"#$%&\'()*+,-./:;<=>?@[\\]
>>> string.whitespace
' \t\n\r\x0b\x0c'
```

Formatter

```
>>> import string
>>> formatter = string.Formatter()
>>> strcmp = "my name is {name}"
>>> dct = {"name": "nick"}
>>> formatter.format(strcmp, **dct)   # use dict to format
'my name is nick'
>>> data = ("3",)   # add , to make it a tuple
>>> strcmp = "pi is about {}"
>>> formatter.format(strcmp, *data)   # use tuple to format
'pi is about 3'
```

Template

```
>>> import string
>>> strcmp = "Hello $World"   # the default delimiter is $
>>> t = string.Template(strcmp)
>>> t.substitute({"World": "nick"})   # use dict
'Hello nick'
>>> t.substitute(World = "nick")   # use args
'Hello nick'
>>> class MyTemplate(string.Template):
...     delimiter = "^"
...
>>> strcmp = "Hello ^World"
>>> mytemplate = MyTemplate(strcmp)
>>> mytemplate.substitute(World = "nick")
'Hello nick'
```

re

match, search, findall

```
>>> import re
>>> strcmp = "www.baidu.com"
>>> re.match("www", strcmp).span()   # span function to get index
(0, 3)
>>> re.match("baidu", strcmp)   # re.match only match from the beginning of the string
>>> re.search("baidu", strcmp).span()   # re.search search from all string and return the f
(4, 9)
>>> strcmp = "baidu.com/runoob.com"
>>> re.findall("com", strcmp)   # re.findall find all results and return
['com', 'com']
>>> re.findall("b(.*?).", strcmp)
['', '']
>>> re.findall("b(.*?)c", strcmp)
['aidu.', '.']
```

split, sub, escape

```
>>> import re
>>> re.split(r"\W", "hello,world")   # use regular expression
['hello', 'world']
>>> re.sub(r"Boy|Girl", "Human", "boy and girl", flags = re.I)   # re.I means ignoring apit
'Human and Human'
>>> re.escape(r"#$&*+-.^|~")
'\\#\\$\\&\\*\\+\\-\\.\\^\\/\\~'
```

difflib

Differ

```
>>> import difflib
>>> d = difflib.Differ()
>>> text1 = """difflib
... python version 3.7.4
... difflib version 3.7.4
... this is difflib document
... """
>>> text2 = """difflib
... python version 3.7.3
... this is difflib document
... feature: diff in linux
... """
>>> text1_lines = text1.splitlines()
>>> text2_lines = text2.splitlines()
>>>
>>> list(d.compare(text1_lines, text2_lines))
['  difflib', '- python version 3.7.4', '?                    ^\n', '+ python version 3.7.3
```

HtmlDiff

```
>>> import difflib
>>> d = difflib.HtmlDiff()
>>> text1 = """difflib
... python version 3.7.4
... difflib version 3.7.4
... this is difflib document
... """
>>> text2 = """difflib
... python version 3.7.3
... this is difflib document
... feature: diff in linux
... """
>>> text1_lines = text1.splitlines()
>>> text2_lines = text2.splitlines()
>>> with open("HtmlDiff.html", "w", encoding="utf-8") as f:   # make it a html file
```

```
...     HtmlDiff = d.make_file(text1_lines, text2_lines)
...     f.write(HtmlDiff)
...
3331
```

SequenceMatcher

```
>>> import difflib
>>> s = difflib.SequenceMatcher(None, " abcd", "abcd abcd")
>>> s.find_longest_match(0, 5, 0, 9)
Match(a=0, b=4, size=5)
>>> s = difflib.SequenceMatcher(lambda x: x==" ", " abcd", "abcd abcd")
>>> s.find_longest_match(0, 5, 0, 9)
Match(a=1, b=0, size=4)
>>> s = difflib.SequenceMatcher(None, "abcd", "abd")
>>> s.get_matching_blocks()
[Match(a=0, b=0, size=2), Match(a=3, b=2, size=1), Match(a=4, b=3, size=0)]
```

textwrap

wrap, fill, shorten, dedent, indent

```
>>> import textwrap
>>> strcmp = "Hello,World! My name is nick, l am 14 years old"
>>> textwrap.wrap(strcmp, width = 10)   # 10 for an element
['Hello,Worl', 'd! My name', 'is nick, l', 'am 14', 'years old']
>>> textwrap.fill(strcmp, width = 10)
'Hello,Worl\nd! My name\nis nick, l\nam 14\nyears old'
>>> textwrap.shorten(strcmp, width = 45)   # content over 45 will be [...]
'Hello,World! My name is nick, l am 14 [...]'
>>> textwrap.shorten(strcmp, width = 45, placeholder = "...")   # change the placeholder
'Hello,World! My name is nick, l am 14...'
>>> strcmp = """
...     hello world!
... """
>>> textwrap.dedent(strcmp)
'\nhello world!\n'
>>> strcmp = """Hello World!
... l am nick.
... l am 14 years old.
... """
>>> textwrap.indent(strcmp, " + ", lambda line: True)
' + Hello World!\n + l am nick.\n + l am 14 years old.\n'
```

unicodedata

lookup, name, unidata_version

```
>>> import unicodedata
>>> unicodedata.lookup('LEFT CURLY BRACKET')   # get the symbol of the description
'{'
>>> unicodedata.name("(")   # reverse to lookup
'LEFT PARENTHESIS'
>>> unicodedata.unidata_version
'13.0.0'
```

readline

parse_and_bind **Notice**: If you are using Windows, you need to first install pyreadline module:

```
pip install pyreadline
```

```
>>> import readline
>>> readline.parse_and_bind('tab: complete')   # use tab to autocomplete
>>> histfile = '.pythonhistory'
```

codecs

encode, decode, getencoder, getdecoder

```
>>> import codecs
>>> codecs.encode("你好")
b'\xe4\xbd\xa0\xe5\xa5\xbd'
>>> codecs.decode(b"\xe4\xbd\xa0\xe5\xa5\xbd")
'你好'
>>> codecs.getencoder("utf-8")
<built-in function utf_8_encode>
>>> codecs.getdecoder("gbk")
<built-in method decode of MultibyteCodec object at 0x0000019E080AA078>
```

struct

pack, unpack **Notice**: The format is on struct docs

```
>>> import struct
>>> struct.pack(">l", 1024)   # return a bytes containing the values packed according to th
b'\x00\x00\x04\x00'
>>> struct.unpack(">lH", b'\x00\x00\x04\x00\xf0\xf0')
(1024, 61680)
```

datetime

MINYEAR, MAXYEAR, date More Information about strftime is on strftime docs

```
>>> import datetime
>>> datetime.MINYEAR
1
>>> datetime.MAXYEAR
9999
>>> date = datetime.date
>>> date.today()
datetime.date(2019, 7, 21)
>>> date = datetime.date(2019, 7, 21)
>>> date.today()
datetime.date(2019, 7, 21)
>>> date.weekday()
6
>>> date.isocalendar()
(2019, 29, 7)
>>> date.ctime()
'Sun Jul 21 00:00:00 2019'
>>> date.strftime("%Y %d %y, %H:%M:%S")
'2019 21 19, 00:00:00'
```

calendar

isleap, firstweekday, month

```
>>> import calendar
>>> calendar.isleap(2000)    # check if it is leap year
True
>>> calendar.firstweekday()
0
>>> print(calendar.month(2019, 7))    # get the pretty calendar
     July 2019
Mo Tu We Th Fr Sa Su
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31
```

collections

namedtuple, deque, defaultdict, OrderedDict, Counter

```
>>> import collections
>>> point = collections.namedtuple("point", ["x", "y"])    # create tuple subclasses with na
>>> p = point(2, 1)
>>> p.x, p.y
(2, 1)
```

```
>>> deque = collections.deque(["b", "c", "d"])   # list-like container with fast appends anc
>>> deque.appendleft("a")    # much faster than original list
>>> deque.append("e")
>>> deque
deque(['a', 'b', 'c', 'd', 'e'])
>>> dd = collections.defaultdict(lambda: "None")   # dict which has an default value
>>> dd ["key-1"] = "value-1"
>>> dd["key-1"]
'value-1'
>>> dd["key-2"]   # if the key is not exist, it will return the default value instead of ra
'None'
>>> od = collections.OrderedDict([("a", 1), ("b", 2)])   # dict which always keeps the orde
>>> od
OrderedDict([('a', 1), ('b', 2)])
>>> c = collections.Counter()   # dict subclass for counting hashable objects
>>> for i in "Hello, World":
...     c[i] = c[i] + 1
...
>>> c
Counter({'l': 3, 'o': 2, 'H': 1, 'e': 1, ',': 1, ' ': 1, 'W': 1, 'r': 1, 'd': 1})
```

copy

copy, deepcopy

```
>>> import copy
>>> origin = [1, 2, [3, 4]]
>>> copy1 = copy.copy(origin)
>>> copy2 = copy.deepcopy(origin)
>>> copy1 is copy2
False
>>> origin[2][0] = "Hello, copy"
>>> copy1   # change when origin changes
[1, 2, ['Hello, copy', 4]]
>>> copy2   # don't change when origin changes
[1, 2, [3, 4]]
```

pprint

pprint

```
>>> import pprint   # pretty print
>>> strcmp = ("hello world", {"nick": 13, "ben": 12}, (1, 2, 3, 4), [5, 6, 7, 8], "Hello ppr
>>> pprint.pprint(strcmp)
('hello world',
 {'ben': 12, 'nick': 13},
 (1, 2, 3, 4),
 [5, 6, 7, 8],
 'Hello pprint')
```

enum

Enum, unique, auto

```
>>> import enum
>>> class Seasons(enum.Enum):
...     Spring = 1
...     Summer = 2
...     Autumn = 3
...     Winter = 4
...
>>> Seasons.Spring
<Seasons.Spring: 1>
>>> @enum.unique   # don't allow same value
... class Unique(enum.Enum):
...     Nick = 13
...     Ben = 12
...     Jack = 13
...
Traceback (most recent call last):
  File "<stdin>", line 2, in <module>
  File "C:\Python38\lib\enum.py", line 860, in unique
    raise ValueError('duplicate values found in %r: %s' %
ValueError: duplicate values found in <enum 'Unique'>: Jack -> Nick
>>> class Auto(enum.Enum):
...     VS = enum.auto()
...     VSCode = enum.auto()
...     Pycharm = enum.auto()
...
>>> list(Auto)   # auto is from one
[<Auto.VS: 1>, <Auto.VSCode: 2>, <Auto.Pycharm: 3>]
```

bisect

bisect, bisect_left, bisect_right, insort, insort_left, insort_right

```
>>> import bisect
>>> a = [1, 2, 4, 5]
>>> bisect.bisect_left(a, 1)   # if it has the same, choose left
0
>>> bisect.bisect_right(a, 1)   # if it has the same, choose right
1
>>> bisect.bisect(a, 1)
1
```

```
>>> bisect.insort(a, 1)   # bisect and insert
>>> a
[1, 1, 2, 4, 5]
>>> bisect.insort_left(a, 2)
>>> a
[1, 1, 2, 2, 4, 5]
>>> bisect.insort_right(a, 4)
>>> a
[1, 1, 2, 2, 4, 4, 5]
```

heapq

heappush, heappop

```
>>> import heapq
>>> def heapsort(iterable):
...     h = []
...     for i in iterable:
...         heapq.heappush(h, i)   # first push all numbers
...     return [heapq.heappop(h) for i in range(len(h))]   # it was orderly extracted
...
>>> heapsort([4, 3, 6, 9, 1, 7])
[1, 3, 4, 6, 7, 9]
```

weakref

ref

```
>>> import weakref
>>> class A:
...     def method():
...         print("A")
...
>>> def b(reference):
...     print(reference)
...
>>> c = A()
>>> d = weakref.ref(c, b)   # when c is deleted, b is called
>>> del c
<weakref at 0x000001C94E12F778; dead>
```

math

ceil, factorial, floor, modf, log, pow, sqrt, pi, e

```
>>> import math
>>> math.ceil(1.4)   # return the smallest integer greater than or equal to x
```

```
2
>>> math.factorial(5)
120
>>> math.floor(1.6)   # return the largest integer less than or equal to x
1
>>> math.modf(1.6)   # return the fractional and integer parts
(0.6000000000000001, 1.0)
>>> math.log(8)
2.0794415416798357
>>> math.pow(2,5)   # pow in math produces float type number, pow in builtin function produ
32.0
>>> math.sqrt(9)
3.0
>>> math.pi   # equal to
3.141592653589793
>>> math.e   # constant e's value
2.718281828459045
```

cmath

sin, tan, cos

```
>>> import cmath   # complex maths
>>> cmath.sin(7)
(0.6569865987187891+0j)
>>> cmath.tan(7)
(0.8714479827243188+0j)
>>> cmath.cos(7)
(0.7539022543433046-0j)
```

random

random, uniform, randint, randrange

```
>>> import random
>>> random.random()   # from 0 to 1
0.6381052887323486
>>> random.uniform(5,6)   # from x to y
5.325285695528384
>>> random.randint(6, 9)   # include nine
9
>>> random.randrange(5, 10)   # don't include ten
9
```

fractions

Fraction, limit_denominator

```
>>> import fractions
>>> fractions.Fraction(16, -10)
Fraction(-8, 5)
>>> fractions.Fraction("-16/10")
Fraction(-8, 5)
>>> fractions.Fraction(8, 5) - fractions.Fraction(7, 5)   # fractions can do substraction
Fraction(1, 5)
>>> fractions.Fraction(1.1)   # sometimes it will be strange
Fraction(2476979795053773, 2251799813685248)
>>> fractions.Fraction(1.1).limit_denominator()   # use limit denominator to get right frac
Fraction(11, 10)
>>> import math
>>> math.floor(fractions.Fraction(5, 3))   # it can also be combined with math module
1
```

decimal

Decimal, getcontext

```
>>> import decimal
>>> decimal.Decimal(2)/decimal.Decimal(3)
Decimal('0.6666666666666666666666666667')
>>> context = decimal.getcontext()
>>> context
Context(prec=28, rounding=ROUND_HALF_EVEN, Emin=-999999, Emax=999999, capitals=1, clamp=0, f
>>> context.prec = 5
>>> x = decimal.Decimal(2)/decimal.Decimal(3)
>>> x
Decimal('0.66667')
>>> x.sqrt()
Decimal('0.81650')
>>> x.log10()
Decimal('-0.17609')
```

statistics

mean, harmonic_mean, median, median_low, median_high

```
>>> import statistics
>>> statistics.mean([1, 2, 3])
2
>>> statistics.harmonic_mean([2, 5, 10])
3.75
>>> statistics.median([2, 3, 5, 6])   # get the median of these numbers, median can not be
4.0
>>> statistics.median_low([2, 3, 5, 6])   # median must be in it and get the lower median
3
```

```
>>> statistics.median_high([2, 3, 5, 6])   # get the higher median
5
```

itertools

count, repeat, groupby

```
>>> import itertools   # itertools always return a iterator
>>> for i in zip(itertools.count(1), ["A", "B", "C"]):
...     print(i)
...
(1, 'A')
(2, 'B')
(3, 'C')
>>> for i in itertools.repeat("Hello Repeat!", 5):
...     print(i)
...
Hello Repeat!
Hello Repeat!
Hello Repeat!
Hello Repeat!
Hello Repeat!
>>> [list(g) for k, g in itertools.groupby('AAAABBBCCD')]
[['A', 'A', 'A', 'A'], ['B', 'B', 'B'], ['C', 'C'], ['D']]
```

functools

lru_cache, reduce

```
>>> import functools
>>> @functools.lru_cache(None)   # None means the cache's upper limit is not limited
... def fibonacci(n):
...     if n<2:
...         return n
...     return fibonacci(n-1) + fibonacci(n-2)
...
>>> fibonacci(10)
55
>>> def add(a, b):
...     return a+b
...
>>> functools.reduce(add, range(1,100))   # add from 1 to 100
4950
```

operator

lt, eq, le, ne, gt, ge, abs, pow, concat, contains, indexOf, add

```
>>> import operator   # substitute for builtin operator
>>> operator.lt(3, 4)   # 3<4
True
>>> operator.eq(3, 4)   # 3=4
False
>>> operator.le(3, 4)   # 3<=4
True
>>> operator.ne(3, 4)   # 3!=4
True
>>> operator.gt(3, 4)   # 3>4
False
>>> operator.ge(3, 4)   # 3>=4
False
>>> operator.abs(-10)
10
>>> operator.pow(10, 2)
100
>>> operator.concat("a", "b")
'ab'
>>> operator.contains([1, 2, 3], 2)
True
>>> operator.indexOf([1, 2, 3, 2, 1], 2)
1
>>> operator.add(1, 2)
3
```

pathlib

Path

```
>>> import pathlib
>>> p = pathlib.Path(".")
>>> list(p.glob('**/*.py'))
[WindowsPath('GETREADME.py'), WindowsPath('test.py')]
>>> p/"dir"   # use / to get further path
WindowsPath('dir')
>>> (p/"GETREADME.py").name
'GETREADME.py'
>>> p.is_absolute()
False
```

os.path

exists, getsize, isfile, isdir, join

```
>>> import os.path
>>> os.path.exists(".")
```

```
True
>>> os.path.getsize("./LICENSE")
466
>>> os.path.isfile("./README.md")
True
>>> os.path.isdir("./doc")
False
>>> os.path.join("./doc", "tutorial", "basic")   # join the directory and file
'./doc\\tutorial\\basic'
```

glob

glob

```
>>> import glob
>>> glob.glob("*.pdf", recursive = True)
['README-zh-cn.pdf', 'README.pdf']
```

tempfile

TemporaryFile, mkstemp, mkdtemp

```
>>> import tempfile
>>> with tempfile.TemporaryFile() as f:
...     f.write(b"a")
...     f.seek(0)
...     f.read()
...
1
0
b'a'
>>> name = tempfile.mkstemp()   # for temporary file
>>> name
(3, 'C:\\Users\\ADMINI~1\\AppData\\Local\\Temp\\tmp___ejm5a')
>>> with open(name[1], "w", encoding="utf-8") as f:
...     f.write("Hello tempfile!")
...
15
>>> name = tempfile.mkdtemp()   # for temporary dir
>>> name
'C:\\Users\\ADMINI~1\\AppData\\Local\\Temp\\tmp5mqb0bxz'
>>> with open(name + "\\temp.txt", "w", encoding="utf-8") as f:
...     f.write("Hello tempfile!")
...
15
```

filecmp

cmp

```
>>> import filecmp
>>> filecmp.cmp("examples/cmp1.txt", "examples/cmp2.txt")
True
```

fileinput

input

```
>>> import os
>>> cmd = os.popen("python examples/fileinput_example.py examples/cmp1.txt")   # subprocess
>>> print(cmd.read())
examples/cmp1.txt | Line Number: 1 |:  1

examples/cmp1.txt | Line Number: 2 |:  2

examples/cmp1.txt | Line Number: 3 |:  3

examples/cmp1.txt | Line Number: 4 |:  4

examples/cmp1.txt | Line Number: 5 |:  5
```

shutil

copyfile, rmtree, move

```
>>> import shutil
>>> shutil.copyfile("examples/song.wav", "examples/copysong.wav")
'examples/copysong.wav'
>>> shutil.rmtree("shutil_tree")   # can delete tree has contents, os.remove can't
>>> shutil.move("examples/copysong.wav", "myapp/copysong.wav")
'myapp/copysong.wav'
```

linecache

getline

```
>>> import linecache
>>> linecache.getline("examples/GETREADME.py", 1)   # start from one, not zero
'from sys import exit\n'
```

pickle

loads, dumps

```
>>> import pickle   # pickle is a python-specific compression method
>>> data = [[1, "first"],
...         [2, "second"]]
>>> dumps = pickle.dumps(data)   # similar to json module
>>> dumps
b'\x80\x04\x95"\x00\x00\x00\x00\x00\x00\x00]\x94(]\x94(K\x01\x8c\x05first\x94e]\x94(K\x02\x8
>>> pickle.loads(dumps)
[[1, 'first'], [2, 'second']]
```

copyreg

pickle

```
>>> import copyreg
>>> import copy
>>> import pickle
>>> class A:
...     def __init__(self, a):
...         self.a = a
...
>>> def pickle_a(a):
...     print("pickle A")
...     return A, (a.a,)
...
>>> copyreg.pickle(A, pickle_a)
>>> a = A(1)
>>> b = copy.copy(a)
pickle A
>>>
>>> c = pickle.dumps(a)
pickle A
```

zlib

compress, decompress

```
>>> import zlib
>>> zlib.compress(b"Hello World!", 5)
b'x^\xf3H\xcd\xc9\xc9W\x08\xcf/\xcaIQ\x04\x00\x1cI\x04>'
>>> zlib.decompress(b'x^\xf3H\xcd\xc9\xc9W\x08\xcf/\xcaIQ\x04\x00\x1cI\x04>')
b'Hello World!'
```

lzma

compress, decompress

```
>>> import lzma
>>> lzma.compress(b"Hello, python3!")
```

```
b"\xfd7zXZ\x00\x00\x04\xe6\xd6\xb4F\x02\x00!\x01\x16\x00\x00\x00t/\xe5\xa3\x01\x00\x0eHello,
>>> lzma.decompress(b"\xfd7zXZ\x00\x00\x04\xe6\xd6\xb4F\x02\x00!\x01\x16\x00\x00\x00t/\xe5\x
b'Hello, python3!'
```

zipfile

ZipFile

```
>>> import zipfile
>>> with zipfile.ZipFile("examples/g.zip") as f:   # extract zip file
...     f.extractall()
...
>>> with zipfile.ZipFile("a.zip", "a") as zip:   # create zip file
...     zip.write("README.md")   # use write method to write files into zip file
...
```

configparser

ConfigParser

```
>>> import configparser
>>> config = configparser.ConfigParser()   # create an instance
>>> config.read("examples/config.ini")
['examples/config.ini']
>>> config.sections()
['python', 'java']
>>> config["python"]["type"]   # all sections has values in DEFAULT
'programming language'
>>> config["java"]["popular"]
'1'
```

hashlib

md5

```
>>> import hashlib
>>> md5 = hashlib.md5()
>>> md5.update(b"Hello World")   # use update method to join in batches
>>> md5.block_size
64
>>> md5.digest_size
16
>>> md5.hexdigest()
'b10a8db164e0754105b7a99be72e3fe5'
>>> md5.digest()
b'\xb1\n\x8d\xb1d\xe0uA\x05\xb7\xa9\x9b\xe7.?\xe5'
```

hmac

new, compare_digest

```
>>> import hmac
>>> msg = b"Hello World"
>>> secret = b"key"   # use key to encrypt
>>> h = hmac.new(secret, msg, digestmod='md5')
>>> h.hexdigest()
'432c3ea3b9a503183f3d1258d9016a0c'
>>> h.digest()
b'C,>\xa3\xb9\xa5\x03\x18?=\x12X\xd9\x01j\x0c'
>>> h2 = hmac.new(secret, b"Hello world", digestmod="md5")
>>> hmac.compare_digest(h.digest(), h2.digest())
False
```

secrets

choice, token_bytes, token_hex

```
>>> import secrets
>>> secrets.choice("Hello World!")   # choose one character from the string
'd'
>>> secrets.token_bytes(32)
b'\xd7\x98\xba\xc5\x18[/\xeaLx\xdb\x962\x84\xff`(7&\xe6\xae\xd4\x17n,\xc3\x9e\xb0V\x1c\x1d\
>>> secrets.token_hex(16)
'335f8df0cb6dd60a3c41fdba7ccd1a0b'
```

os

name, getcwd

```
>>> import os
>>> os.name
'nt'
>>> os.getcwd()   # get the working directory now
'C:\\Users\\Nick'
```

time

localtime, ctime, perf_counter, sleep, strftime

```
>>> import time
>>> time.localtime()
time.struct_time(tm_year=2019, tm_mon=7, tm_mday=29, tm_hour=12, tm_min=18, tm_sec=57, tm_wo
>>> time.ctime()
'Mon Jul 29 12:19:40 2019'
>>> time.perf_counter()
```

```
174.1987535
>>> time.sleep(1)
>>> time.strftime("%d %b %Y")
'29 Jul 2019'
```

logging

log, info, debug, warning, error, critical More details about format for logging is on logging docs

```
>>> import logging
>>> logging.basicConfig(level = logging.INFO,format = '%(asctime)s - %(name)s - %(levelname)
>>> logger = logging.getLogger(__name__)
>>> logger.info("info")
2019-07-29 12:29:59,363 - __main__ - INFO - info
>>> logger.debug("debug")
>>> logger.error("error")
2019-07-29 12:30:26,729 - __main__ - ERROR - error
>>> logger.critical("critical")
2019-07-29 12:30:36,446 - __main__ - CRITICAL - critical
>>> logger.warning("warning")
2019-07-29 12:30:48,815 - __main__ - WARNING - warning
>>> logger.log(35, "log")   # customize the level
2019-07-29 12:31:59,758 - __main__ - Level 35 - log
```

getpass

getpass, getuser

```
>>> import getpass
>>> password = getpass.getpass()   # what you enter will not be displayed on the screen
Password:
>>> password
'xxx'
>>> getpass.getuser()
'Nick'
```

platform

machine, platform, python_compiler, python_version, system

```
>>> import platform
>>> platform.machine()
'AMD64'
>>> platform.platform()
'Windows-10-10.0.18362-SP0'
>>> platform.python_compiler()
```

```
'MSC v.1929 64 bit (AMD64)'
>>> platform.python_version()
'3.9.8'
>>> platform.system()
'Windows'
```

argparse

ArgumentParser

```
>>> import os
>>> def cmd(command):
...     res = os.popen(command)
...     print(res.read())
...
>>> cmd("python examples/argparse_example.py -a 1 -b 2 --sum 1 2 3 4 -r 10 -t")
Namespace(a=1, b=2, sum=[1, 2, 3, 4], required='10', true=True)
3
10

>>> cmd("python examples/argparse_example.py --help")
usage: argparse_example.py [-h] [-a A] [-b B] [-s SUM [SUM ...]] -r REQUIRED
                           [-t]

the example parser for argparse

optional arguments:
  -h, --help            show this help message and exit
  -a A                  the a number for adding
  -b B                  the b number for adding
  -s SUM [SUM ...], --sum SUM [SUM ...]
  -r REQUIRED, --required REQUIRED
  -t, --true
```

errno

errorcode

```
>>> import errno
>>> errno.errorcode
{19: 'ENODEV', 10065: 'WSAEHOSTUNREACH', 122: 'ENOMSG', 120: 'ENODATA', 40: 'ENOSYS', 32: 'E
```

io

StringIO, BytesIO

```
>>> import io
>>> stringio = io.StringIO()   # similar as file operation
>>> stringio.write("Hello World!")
12
>>> stringio.seek(6)
6
>>> stringio.read()
'World!'
>>> stringio.close()
>>> bytesio = io.BytesIO()
>>> bytesio.write(b"Hello World")
11
>>> bytesio.seek(0)
0
>>> bytesio.read()
b'Hello World'
>>> bytesio.close()
```

socket

socket Run in bash:

```
python socker_server.py
Connected by ('127.0.0.1', 64346)
```

```
python socket_client.py
Received b'Hello, world'
```

json

dumps, loads

```
>>> import json
>>> x = json.dumps({"Nick": 13, "Ben": 10})
>>> x
'{"Nick": 13, "Ben": 10}'
>>> json.loads(x)
{'Nick': 13, 'Ben': 10}
```

html

escape, unescape

```
>>> import html
>>> html.escape("As we all know, 2>1")
'As we all know, 2&gt;1'
>>> html.unescape('As we all know, 2&gt;1')
'As we all know, 2>1'
```

webbrowser

open, open_new, open_new_tab

```
>>> import webbrowser
>>> webbrowser.open("www.baidu.com")
True
>>> webbrowser.open_new("www.baidu.com")
True
>>> webbrowser.open_new_tab("www.baidu.com")
True
```

wave

open

```
>>> import wave
>>> f = wave.open("examples/song.wav", "rb")
>>> f.getparams()
_wave_params(nchannels=2, sampwidth=2, framerate=44100, nframes=442368, comptype='NONE', co
```

sndhdr

what

```
>>> import sndhdr
>>> sndhdr.what("examples/song.wav")
SndHeaders(filetype='wav', framerate=44100, nchannels=2, nframes=442368, sampwidth=16)
```

imghdr

what

```
>>> import imghdr
>>> imghdr.what("examples/china.jpg")
'jpeg'
```

colorsys

rgb_to_yiq, rgb_to_hls, rgb_to_hsv

```
>>> import colorsys
>>> colorsys.rgb_to_yiq(128, 128, 0)
(113.91999999999999, 41.177600000000005, -39.948799999999984)
>>> colorsys.rgb_to_hls(128, 128, 0)
(0.16666666666666666, 64.0, -1.0158730158730158)
>>> colorsys.rgb_to_hsv(128, 128, 0)
(0.16666666666666666, 1.0, 128)
```

turtle

pensize, pencolor, begin_fill, forward, right, end_fill

```
>>> import turtle   # it can draw a five-pointed star
>>> turtle.pensize(5)
>>> turtle.pencolor("yellow")
>>> turtle.begin_fill()
>>> for _ in range(5):
...     turtle.forward(200)
...     turtle.right(144)
...
>>> turtle.end_fill()
```

tkinter

Tk, Label details in tkinter docs

run in bash:

```
python tkinter_example.py
```

typing

List, NewType

```
>>> import typing
>>> lst = typing.List[int]   # list full of int
>>> float_lst = typing.List[float]
>>> def foo(x: float, lst: lst)->float_lst:
...     return [x*num for num in lst]
...
>>> foo(1.5, [3, 4, 5])
[4.5, 6.0, 7.5]
>>> ID = typing.NewType("ID", int)
>>> ID(70)
70
```

doctest

testfile

```
>>> import doctest
>>> doctest.testfile("../examples/doctest_example.txt", verbose=True)
Trying:
    from doctest_example import factorial
Expecting nothing
ok
Trying:
    [factorial(n) for n in range(6)]
Expecting:
    [1, 1, 2, 6, 24, 120]
ok
Trying:
    factorial(30)
Expecting:
    265252859812191058636308480000000
ok
1 items passed all tests:
   3 tests in doctest_example.txt
3 tests in 1 items.
3 passed and 0 failed.
Test passed.
TestResults(failed=0, attempted=3)
```

timeit

timeit, Timer

```
>>> import timeit
>>> timeit.timeit("[i for i in range(10000)]", number = 1000)
1.0810747999999961
>>> timeit.timeit("lst = []\nfor i in range(10000):\n    lst.append(i)", number = 1000)
1.770644500000003
>>> a = timeit.Timer("[i for i in range(10000)]")
>>> a.timeit(number = 1000)
0.9840328999999883
```

pdb

set_trace

```
>>> import pdb
>>> def foo():
...     lst = []
...     for i in range(2):
...         pdb.set_trace()
...         lst.append(i)
...     return lst
...
>>> foo()
> <stdin>(5)foo()
(Pdb) p i
0
(Pdb) p lst
[]
```

```
(Pdb) n
> <stdin>(3)foo()
(Pdb) list
[EOF]
(Pdb) n
> <stdin>(4)foo()
(Pdb) r
> <stdin>(5)foo()
(Pdb) p i
1
(Pdb) p lst
[0]
(Pdb) q
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "<stdin>", line 5, in foo
  File "<stdin>", line 5, in foo
  File "C:\Python37\lib\bdb.py", line 88, in trace_dispatch
    return self.dispatch_line(frame)
  File "C:\Python37\lib\bdb.py", line 113, in dispatch_line
    if self.quitting: raise BdbQuit
bdb.BdbQuit
```

ensurepip

version, bootstrap

```
>>> import ensurepip
>>> ensurepip.version()
'19.0.3'
>>> ensurepip.bootstrap(upgrade=True)
Looking in links: C:\Users\Nick\AppData\Local\Temp\tmpus54fm12
Requirement already up-to-date: setuptools in c:\python37\lib\site-packages (41.2.0)
Requirement already up-to-date: pip in c:\python37\lib\site-packages (19.2.3)
```

Run in bash:

```
python -m ensurepip   # download pip
python -m ensurepip --upgrade   # upgrade pip
```

zipapp

create_archive

```
>>> import zipapp
>>> zipapp.create_archive("myapp", "examples/myapp.pyz")
>>> __import__("os").popen("python examples/myapp.pyz").read()   # pyz file can also be exe
'Hello, everyone!\n'
```

sys

exc_info, implementation, maxsize, platform, version

```
>>> import sys
>>> try:
...     1/0
... except Exception:
...     print(sys.exc_info())   # traceback.print_exc is a beautful version of sys.exc_info
...
(<class 'ZeroDivisionError'>, ZeroDivisionError('division by zero'), <traceback object at 0x
>>> sys.implementation
namespace(cache_tag='cpython-37', hexversion=50791664, name='cpython', version=sys.version_i
>>> sys.maxsize
9223372036854775807
>>> sys.platform
'win32'
>>> sys.version
'3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 20:34:20) [MSC v.1916 64 bit (AMD64)]'
>>> sys.exit()
```

dataclasses

dataclass

```
>>> import dataclasses
>>> @dataclasses.dataclass
... class User:
...     name: str
...     age: int
...     def get_info(self):
...         return self.name + " is " + str(self.age) + " years old."
...
>>> pynickle = User("pynickle", 14)
>>> pynickle.get_info()
'pynickle is 14 years old.'
```

contextlib

contextmanager

```
>>> import contextlib
>>> @contextlib.contextmanager
... def cm(name):
...     print("__enter__ cm")
...     yield "Hello," + name
...     print("__exit__ cm")
...
```

```
>>> with cm("pynickle") as value:
...     print(value)
...
__enter__ cm
Hello,pynickle
__exit__ cm
>>> with cm("pynickle") as a, cm("bob") as b:
...     print(a, b)
...
__enter__ cm
__enter__ cm
Hello,pynickle Hello,bob
__exit__ cm
__exit__ cm
```

abc

ABCMeta, abstractmethod

```
>>> import abc
>>> class User(metaclass=abc.ABCMeta):
...     def hello(self, name):
...         print("Hello," + name)
...     @abc.abstractmethod
...     def unique_hello(self):
...         self.hello()
...     @property
...     @abc.abstractmethod
...     def age():
...         pass
...
>>> class UserOne(User):
...     def unique_hello(self):
...         self.hello()
...         print("l am coming!")
...     def age():
...         return "13"
...
>>> user1 = UserOne()
>>> dir(user1)
['__abstractmethods__', '__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq
>>> isinstance(user1, User)
True
>>> class UserTwo():
...     pass
...
```

```
>>> User.register(UserTwo)   # register only made UserTwo subclass of Uuser, but none of the
<class '__main__.UserTwo'>
>>> user2 = UserTwo()
>>> dir(user2)
['__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '__format__', '__ge
>>> issubclass(UserTwo, User)
True
```

traceback

print_exc

```
>>> import traceback
>>> try:
...     1/0
... except Exception:
...     traceback.print_exc()
...
Traceback (most recent call last):
  File "<stdin>", line 2, in <module>
ZeroDivisionError: division by zero
```

future

division, absolute_import, print_function, unicode_literals

```
>>> from __future__ import division, absolute_import, print_function, unicode_literals
```

atexit

register

```
>>> import atexit
>>> def bye():
...     print("bye!")
...
>>> atexit.register(bye)
<function bye at 0x004B0858>
>>> exit()
bye!
```

builtins

range

```
>>> import builtins
>>> for i in builtins.range(10):
...     print(i)
```

```
...
0
1
2
3
4
5
6
7
8
9
```

inspect

getmembers, ismethod, isfunction, isclass, isbuiltin

```
>>> import inspect
>>> inspect.getmembers([1, 2, 3])
[('__add__', <method-wrapper '__add__' of list object at 0x0000021147AB4880>), ('__class__',
of list object at 0x0000021147AB4880>), ('__doc__', 'Built-in mutable sequence.\n\nIf no arg
at 0x0000021147AB4880>), ('__rmul__', <method-wrapper '__rmul__' of list object at 0x000002
at 0x00007FFA415E9AF0>), ('append', <built-in method append of list object at 0x0000021147A
clear of list object at 0x0000021147AB4880>), ('copy', <built-in method copy of list object
<built-in method count of list object at 0x0000021147AB4880>), ('extend', <built-in method e
object at 0x0000021147AB4880>), ('pop', <built-in method pop of list object at 0x0000021147
>>> print(inspect.getdoc([1, 2, 3]))
Built-in mutable sequence.

If no argument is given, the constructor creates a new empty list.
The argument must be an iterable if specified.
>>> inspect.ismethod(inspect.getmembers)
False
>>> inspect.isbuiltin(repr)
True
>>> inspect.isfunction(lambda x:x+1)
True
>>> inspect.isclass(inspect.Signature)
True
```

zipimport

importer

```
>>> import zipimport
>>> zip = zipimport.zipimporter("examples/g.zip")
>>> zip.archive
'examples\\g.zip'
```

```
>>>
>>> a = zip.load_module("a")
>>> a
<module 'a' from 'examples\\g.zip\\a.py'>
>>> a.main()
Hello everyone
```

importlib

import, reload

```
>>> import importlib
>>> sys = importlib.__import__("sys")   # equal to built in function __import__
>>> importlib.reload(sys)
<module 'sys' (built-in)>
```

runpy

run_module, run_path

```
>>> import runpy
>>> runpy.run_module("myapp")
Hello, runpy!
{'__name__': 'myapp.__main__', '__file__': 'C:\\Users\\pc\\Desktop\\python-cheatsheet-redefi
round>, 'setattr': <built-in function setattr>, 'sorted': <built-in function sorted>, 'sum':
'False': False, 'True': True, 'bool': <class 'bool'>, 'memoryview': <class 'memoryview'>, 'b
'BaseException': <class 'BaseException'>, 'Exception': <class 'Exception'>, 'TypeError': <cl
or Ctrl-Z plus Return to exit, 'exit': Use exit() or Ctrl-Z plus Return to exit, 'copyright'
All Rights Reserved.

Copyright (c) 2000 BeOpen.com.
All Rights Reserved.

Copyright (c) 1995-2001 Corporation for National Research Initiatives.
All Rights Reserved.

Copyright (c) 1991-1995 Stichting Mathematisch Centrum, Amsterdam.
All Rights Reserved., 'credits':     Thanks to CWI, CNRI, BeOpen.com, Zope Corporation and a
    for supporting Python development.  See www.python.org for more information., 'license':
see the full license text, 'help': Type help() for interactive help, or help(object) for hel
>>> runpy.run_path("myapp")
Hello, runpy!
{'__name__': '<run_path>', '__doc__': None, '__package__': '', '__loader__': <_frozen_import
'_frozen_importlib.BuiltinImporter'>, '__spec__': ModuleSpec(name='builtins', loader=<class
function abs>, 'all': <built-in function all>, 'any': <built-in function any>, 'ascii': <bui
<built-in function iter>, 'len': <built-in function len>, 'locals': <built-in function local
sum>, 'vars': <built-in function vars>, 'None': None, 'Ellipsis': Ellipsis, 'NotImplemented'
```

```
'bytes'>, 'classmethod': <class 'classmethod'>, 'complex': <class 'complex'>, 'dict': <class
'RuntimeError'>, 'RecursionError': <class 'RecursionError'>, 'NotImplementedError': <class '
All Rights Reserved.

Copyright (c) 2000 BeOpen.com.
All Rights Reserved.

Copyright (c) 1995-2001 Corporation for National Research Initiatives.
All Rights Reserved.

Copyright (c) 1991-1995 Stichting Mathematisch Centrum, Amsterdam.
All Rights Reserved., 'credits':     Thanks to CWI, CNRI, BeOpen.com, Zope Corporation and a
    for supporting Python development.  See www.python.org for more information., 'license':
```

ast

literal_eval, parse, dump

```
>>> import ast
>>> ast.literal_eval("__import__('os')")   # safer than built in function eval
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "C:\Users\Lenovo\AppData\Local\Programs\Python\Python37-32\lib\ast.py", l
ine 91, in literal_eval
    return _convert(node_or_string)
  File "C:\Users\Lenovo\AppData\Local\Programs\Python\Python37-32\lib\ast.py", l
ine 90, in _convert
    return _convert_signed_num(node)
  File "C:\Users\Lenovo\AppData\Local\Programs\Python\Python37-32\lib\ast.py", l
ine 63, in _convert_signed_num
    return _convert_num(node)
  File "C:\Users\Lenovo\AppData\Local\Programs\Python\Python37-32\lib\ast.py", l
ine 55, in _convert_num
    raise ValueError('malformed node or string: ' + repr(node))
ValueError: malformed node or string: <_ast.Call object at 0x00B11C50>
>>> ast.literal_eval("[1, 2, 3]")
[1, 2, 3]
>>> hello_world = ast.parse("print('Hello World!')", "<string>", "exec")   # abstract synta
>>> ast.dump(hello_world)
"Module(body=[Expr(value=Call(func=Name(id='print', ctx=Load()), args=[Constant(value='Hell
```

keyword

kwlist, iskeyword

```
>>> import keyword
>>> keyword.kwlist
```

```
['False', 'None', 'True', '__peg_parser__', 'and', 'as', 'assert', 'async', 'await', 'break
>>> keyword.iskeyword("True")
True
```

dis

dis, show_code, code_info

```
>>> import dis
>>> def func():
...     print("Hello World")
...
>>> dis.dis(func)
  2           0 LOAD_GLOBAL              0 (print)
              2 LOAD_CONST               1 ('Hello World')
              4 CALL_FUNCTION            1
              6 POP_TOP
              8 LOAD_CONST               0 (None)
             10 RETURN_VALUE
>>> dis.show_code(func)
Name:              func
Filename:          <stdin>
Argument count:    0
Kw-only arguments: 0
Number of locals:  0
Stack size:        2
Flags:             OPTIMIZED, NEWLOCALS, NOFREE
Constants:
   0: None
   1: 'Hello World'
Names:
   0: print
>>> dis.code_info(func)
"Name:              func\nFilename:          <stdin>\nArgument count:    0\nKw-only argumen
```

tabnanny

verbose, check

```
>>> import tabnanny
>>> tabnanny.verbose = True
>>> tabnanny.check("examples/tabnanny_example.py")
'examples/tabnanny_example.py': *** Line 3: trouble in tab city! ***
offending line: '\tprint(i)'
indent not greater e.g. at tab sizes 1, 2, 3, 4
```

this

this

```
>>> 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!
```

antigravity

antigravity

```
>>> import antigravity
```