Python >

Python Cheat Sheet

Contents

SYNTAX

Escape chars

```
\\      Backslash (\)
\'      Single-quote (')
\"      Double-quote (")
\a      ASCII bell (BEL)
\b      ASCII backspace (BS)
\f      ASCII formfeed (FF)
\n      ASCII linefeed (LF)
\N{name}        Character named name in the Unicode database
(Unicode only)
\r      Carriage Return (CR)
\t      Horizontal Tab (TAB)
\uxxxx  Character with 16-bit hex value xxxx (Unicode only)
\Uxxxxxxxx      Character with 32-bit hex value xxxxxxxx
(Unicode only)
\v      ASCII vertical tab (VT)
\ooo    Character with octal value ooo
\xhh    Character with hex value hh
```

multi-line print

```
print '''
some stuff here
and here and here
and here
'''
```

STRINGS

print substring

```
myString = "Hello There"
print myString   # full string
print myString[0]  # print 1st char
print myString[2:5]  # print 3rd to 5th char
```

DATA TYPES

get type of an object

```
a = 'bird'
b = 5
c = ['blue', 'gray', 'yellow']
d = { 'name' : 'fred', 'id' : 3499, 'dept' : 'sales' }

type(a)  # string
type(b)  # int
typc(c)  # list
type(d)  # dict
```

LIST (array)

List - can be different datatypes within itself

```
myList = [ 'dog','cat', 23, 'hello', 16.3 ]
```

add, remove, sort, reverse, count

```
myList.append('bird')
myList.remove('dog')
myList.sort()
myList.reverse()
myList.count('dog')  // counts # of elements in a list
```

Empty out a list
if py 3 and above
myList.clear()

if py 2

```
del myList[:]
```

sort by alphabet

```
for item in sorted(myList):
    print item
```

get total # of elements in list

```
len(myList)
```

print string 10 times
```
print "hello" * 10
```

print multi line string
```
print """ stuff here stuff here
stuff here stuff here
stuff here
stuff here
"""
```

Split string into array
```
var = "dog:cat:rabbit"
print var.split(':')
['dog', 'cat', 'rabbit']
```

Strip prefix and postfix characters
```
'red and blue balloon'.strip('red loon')
and blue bal
```

Partition a string into variables
```
file = 'somefile.csv'
filename, __, ext = file.rpartition('.')
```

Combine a string and int (concat)
```
log_file = logdir+'%d' % number+'.log'
```

remove all whitespaces
```
myString = ' '.join(myString.split()
```

remove last word from a string by delimeter
```
path = '/etc/network/sysconfig/ifcfg'
print path.rsplit('/', 1)[0]
>> /etc/network/sysconfig
```

replace whitespace with dash
```
myString = myString.replace(" ", "-")
```

remove/replace pattern from string
```
string = "Godzilla is a sea monster"
new = "green"
print string.replace("sea", new)
>> Gozilla is a green monster
```

replace everything after certain character,
```
string = "these pretzels are...making me thirsty!"
print string.split("...", 1)[0]
>> these pretzels are
```

replace string in-file
```
# cleanup double quotes inside file
with open(data_dir + 'file1.json') as file:
    data = file.read()
    content = data.replace('"[', '[').replace(']"', ']')
with open(data_dir + '/file1.json', 'w+') as file:
    file.write(content)
```

remove all text between 2 anchors in a file
xxxx

Starts with, Ends with
```
string = "tiger is orange"
print string.startswith("tiger")  # True
print string.endswith("green")  # False
```

check if string starts with a number
```
string = '345 broad street'

if not string[0].isdigit():
   print('does not start with number')
```

Lowercase, Uppercase

```
myString = myString.lower()  or upper()
```

convert all elements in List into lower or upper
```
myList = ['Joe', 'MARY', 'fred', 'JaNE']
myList = [x.lower() for x in myList]
>> ['joe', 'mary', 'fred', 'jane']
```

capitalize 1st letter
```
print 'charles'.title() # Charles
```

remove special chars
```
summary = re.sub('[!~`".,;:}?/\\\|={@#$%^&%*()_+<>\[\]]', '-',
summary)
```

print multiple variables within string
```
print("hello there {0}, and {1} and also {2}".format("Joe",
"Fred", "Mary")
```

get index of an item in the list
```
print myList.index('dog')   # index=0
```

get max, min value and index in list
```
myList[0]              # first element in list
myList.index(lst[0])   # index of 1st element in list
myList[-1]             # last element in list
myList.index(lst[-1])  # index of last element in list
```

All, Any
```
myList = [23,59,64]
if all([ i>20 for i in myList ])
    print("all integers larger than 20")
if any(...)
    print("some integers larger than 20")
```

create Array from string or numbers
```
list('blue')
['b','l','u','e']
```

Slice of an array
```
names = ['joe','mary','fred','anna','bob']
print(names[1:3])   # prints mary, fred

names = ['charles', 'martina', 'michael', 'florence', 'eli']
print(names[2:])  # prints michael,florence,eli
```

Get unique values from a List/Array
```
names = ['Joe', 'Mary', 'Bob', 'Mary', 'Joe', 'Ed', 'Bob']
names = list(set(names))
print names
>>> ['Ed', 'Bob', 'Joe', 'Mary']
```

remove empty values from a List
```
myList = filter(None, myList)
```

remove 1st element from List
```
mylist.pop(0)
```

TUPLES

(like arrays but data cannot be changed) - faster processing, used in JSONs instead of Li

```
myTuple = ( 'first', 234, 12.34, 'someval' )
```

append to tuple
```
animals = ('cat', 'dog')
animals = animals + ('rabbit',)
print animals
('cat', 'dog', 'rabbit')
```

sort tuple alphabetically
```
print sorted(animals)
```

DICTS (Dictionary/Hash - Key/Value pairs)

```
employees = { 'name' : 'fred', 'id' : 3499, 'dept' : 'sales' }
```

print all values in Dict
```
print employees.values()
```

print all keys in Dict
```
print employees.keys()
```

print all keys and values
```
for k, v in employees.items()
  print k, ':', v
```

print specific key
```
print employees[name]  #print specific key value
```

clear all values in Dict
```
var.clear()
```

append to a Dict
```
var['newKey'] = 'newValue'
```

multi-dim Dict
```
employee = {}

employee['joe'] = { 'id': 5, 'city':'boston' }

employee['joe'].keys()
['city','id']

employee['joe'].values()
['boston',5]

employee['joe']['city']
'boston'
```

check if Dictionary has a key
```
data = { 'name':'joe', 'id': 123 }

if data.has_key('name'):
    print "has key"
```

```
>> hello there Joe, and Fred and also Mary
```

remove leading, trailing spaces, both leading and trailing

```
myString.rstrip()  or lstrip()        or strip()
```

print string in reverse order

```
string = "Apples"
print string[::-1]
>> selppA
```

remove last character from string

```
str = 'abcdef,'
print str[:-1]
>> abcdef
```

extract word from string by delimiter

```
string = '/opt/my/dir/hello'
string = string.split('/')[-1]  // hello
```

Convert Type

List to String

```
str = ''.join(myList)
```

String to list

```
str = 'black green blue'
str = str.split()
// str = ['black', 'green', 'blue']
```

Unicode to string

```
word = u'apple'
word = word.encode('utf8')
OR
word = str(word)
```

String to Int

```
var = '3'
print type(var)
>> str
var = int(var)
print type(var)
>> int
```

INPUT

get input from cmd line

```
age = raw_input("how old are you?")
print "your age is %s" % age
```

get variables into script from user input

```
from sys import argv

script, first, second = argv
print "script is called ", script
print "first variable is ", first
print "second variable is ", second
```

read json from cmd line

```
cmd to generate json | python -c 'import sys, json; print
json.load(sys.stdin)["key"]["subkey"]'
```

FILE HANDLING

open file for reading

```
file = "somefile.txt"
content = open(file)
print content.read()
```

open file for reading & writing

```
file = "somefile.txt"
content = open(file, 'rw')
content.write("new stuff here")
content.close() # close file
```

Better Method:

```
with open("myFile.txt") as file:
    data = file.read()

file = "answer.txt"
content = open(file, 'r+')
print "i is %s" % i
var = str(i)  # convert int to str
content.write(var)
content.close()
```

check if file exists

```
if os.path.exists(file_name):
  print "exists"
```

get length of dict

```
print len(myDict)
```

Sort an unordered Dictionary (Hash) by 1st element

1. convert unordered dict/hash into an ordered tuple,

```
hash = {'rabbit': {'traits': {'color': 'gray', 'food': 'carrot'}, 'type': 'mammal'}, 'ze
{'color': 'black', 'food': 'grass'}, 'type': 'mammal'}, 'aardvark': {'traits': {'color':
'mammal'}}
```

2. convert unordered dict into OrderedDict

```
from collections import OrderedDict

od = OrderedDict(sorted(hash.items(),key=lambda x:x[0]))

print od

OrderedDict([('aardvark', {'traits': {'color': 'brown'}, 'type': 'mammal'}), ('rabbit',
{'color': 'gray', 'food': 'carrot'}, 'type': 'mammal'}), ('zebra', {'traits': {'color':
'grass'}, 'type': 'mammal'})])

print od['zebra']

{'traits': {'color': 'black', 'food': 'grass'}, 'type': 'mammal'}
```

Sort a Dict by multiple values (by Type and then by Time)

```
events = {
            '12/15/2017': {'name': 'SEC NEW', 'type': 'security'},
            '3/6/2013': {'name': 'INFO NEW', 'type': 'info'},
            '4/2/2004': {'name': 'SEC OLD', 'type': 'security'},
            '01/02/1960': {'name': 'SEC REALLY OLD', 'type': 'security'}
          }


od = OrderedDict(sorted(events.items(),key=lambda x: (x[1]['type'], x)))

print od

('03/06/2013', {'type': 'info', 'name': 'INFO NEW'}),
('01/02/1960', {'type': 'security', 'name': 'SEC REALLY OLD'}), |
('04/02/2004', {'type': 'security', 'name': 'SEC OLD'}),
('12/15/2017', {'type': 'security', 'name': 'SEC NEW'})])
```

sorts by #1 TYPE, #2 date order

Get value from nested dictionary, if None, can return custom value

```
pip install dictor

from dictor import dictor


  with open('sample.json') as data:
      data = json.load(data)

  print(dictor(data, 'characters', 'fallback value here'))
```

SETS

use when need to compare 2 sets of numbers or words

```
num_set {1,2,3,4,5}
word_set = set(["blue", "green", "red"])
```

add, remove to end of set

```
num_set.add(5)
num_set.remove(5)

first = {1,2,3,4,5,6}
second = {4,5,6,7}
```

combine 2 sets to form 1 set = pipe character

```
print(first|second) = 1,2,3,4,5,6,7
```

get items only in both, intersection &

```
print(first & second)
```

get items only in 1st not 2nd

```
print(first - second)
```

get item i either set but not both

```
print(first ^ second)
```

Print var with a string

```
name = Joe
```

check if file is a file or directory
```
for file in glob.glob('/etc/'):
   if os.path.isdir(file):
      print('is a directory')
```

save incoming Json into file (create file if not there)
```
import json
data = request.get_json()
with open(base_dir+'/fields.json','w+') as f:
   json.dump(jsonfile,f)
```

compare 2 files for difference
```
diff =
difflib.ndiff(open(file1).readlines(),open(file2).readlines())
log.info(''.join(diff))
```

delete file
```
os.remove(file_path)  # file_path = /tmp/myfile.txt
```

rename a file
```
os.rename('oldName.txt', 'newName.txt')
```

get file checksum
```
from hashlib import md5

def md5checksum(filePath):
    with open(filePath, 'rb') as fh:
        m = md5()
        while True:
            data = fh.read(8192)
            if not data:
                break
            m.update(data)
        return m.hexdigest()

print md5checksum('/tmp/file1')
```

read a file line by line

OS

change to a different dir
```
os.chdir( path )
```

print working dir
```
os.getcwd()
```

print Python path (where Modules are stored)
```
import sys
print sys.path
```

check if file exists
```
if not os.path.exists(base_dir+'/fields.json'):
    do something
```

delete file
```
os.remove(filename)
```

copy file
```
from shutil import copy2
copy2(source_file, target_file)
```

make a directory
```
if not os.path.exists(log_dir):
   os.makedirs(log_dir)
```

print environment variable
```
print os.environ['ENV']
```

HTTP & CURL

make a Curl request with authentication
```
import requests
url = "https://"+server+"/rest/api"
req = requests.get(url,auth=("username","pw),verify=False)
jsonfile = req.json()
```

LOGGING

```
# common.py

log_dir = 'log/'
if not os.path.exists(log_dir):
    os.makedirs(log_dir)

log_file = log_dir + 'disco.log'
global log
```

```
print "hello there", name
```

Print var inside a string
```
name = Joe
print "hello there %s, welcome!" % name
```

if variable doesnt exist, create it,
```
try:
    blah = myVar[0]['somevalue']
except KeyError:
    blah = "hello"
except IndexError:  # if [0] doesnt exist
    pass
```

CONDITIONALS

If, Else If
```
if var = 1 :
    print 'var is one'
elif var = 2 :
    print 'var is two'
else :
    print 'var is zero'
```

If NOT
```
if not var = 5:
    do stuff

or var != 5
```

One-line If
var = 100
if (var == 100) : print "value of variable is 100"

While Loop
```
while True:
    try:
        print next(it)

i = 0
while i < 10:
 print i
 i = i + 1
```

For Loop
```
for x in var:
    print x
```

complex if then,
```
 if all( [cond1 == 'val1', cond2 == 'val2', cond3 == 'val3', cond4 == 'val4'] ):

 if any( [cond1 == 'val1', cond2 == 'val2', cond3 == 'val3', cond4 == 'val4'] ):
```

complex If, parse all values in a list,
```
if all(val == 'NA' for val in [timestamp, hostname, env, sudoFrom, sudoTo, cmd]):
    log.warning("empty row, skipping..")
```

check value with Assert
```
x = 5
assert (x > 10), "x is less than 10!!!"
AssertionError: x is less than 10!!!
```

FUNCTIONS

convert to integer, string or float
```
int(), str(), float()
```

Iteration (using **Next** function)
```
list = [1,2,3,4]

iterator = iter(list)
while True
    try:
        print (next(iterator))
```

Print args multiple args
```
def print_two(*args):
   arg1, arg2 = args
   print "arg1: %r, arg2: %r" % (arg1, arg2)
```

Generator Functions
```
def fibonacci(n): #generator function
    a, b, counter = 0, 1, 0
```

```
log_format = "%(asctime)s [%(levelname)s] %(message)s"
log = logging.getLogger('myLogger')

# 5MB max, 3 files max before rollover
handler = RotatingFileHandler(log_file, maxBytes=5242880,
backupCount=3)
formatter = logging.Formatter(log_format)
handler.setFormatter(formatter)
log.addHandler(handler)
log.setLevel(logging.DEBUG)


# another.py

from common import log
log.info('test log entry')
```

To log to both File and Syslog (with Tag)

```
def init_log(pluginpath):
    ''' configures Logging for each plugin '''

    log_dir = base_dir + '/logs'

    if not os.path.exists(log_dir):
        os.makedirs(log_dir)

    if pluginpath == 'maestro':
        plugin = 'maestro'
    else:
        plugin = pluginpath.split('/')[-1]

    # set level to DEBUG for trace logging
    #logging.basicConfig(stream=sys.stdout,
level=logging.DEBUG)

    # logging globals
    formatter = logging.Formatter("TagName: myApp - %
(asctime)s [%(levelname)s] "+plugin+": %(message)s")
    log = logging.getLogger(plugin)
    log.setLevel(logging.ERROR)

    # set Syslog handler
    from logging.handlers import SysLogHandler
    syslog_handler = SysLogHandler(address='/dev/log')
    syslog_handler.setLevel(logging.ERROR)
    syslog_handler.setFormatter(formatter)
    log.addHandler(syslog_handler)


    # set file log handler
    from logging.handlers import RotatingFileHandler
    file_handler =
RotatingFileHandler(log_dir+'/'+plugin+'.log',
maxBytes=2000000, backupCount=2)
    file_handler.setFormatter(formatter)
    file_handler.setLevel(logging.DEBUG)
    log.addHandler(file_handler)

    #log.propagate = 1
    return log
```

to log just to syslog

```
log = logging.getLogger(__name__)
log.setLevel(logging.DEBUG)
handler = logging.handlers.SysLogHandler(address =
'/dev/log')
formatter = logging.Formatter('%(module)s.%(funcName)s: [%
(levelname)s] %(message)s')
handler.setFormatter(formatter)
log.addHandler(handler)
```

PIP & PIPENV

configure pip to work behind proxy and not use SSL

mkdir ~/.config/pip

vim ~/.config/pip/pip.conf

```
[global]

proxy="http://proxyserver.yourcompany.com:8080"
index-url="http://pypi.python.org/simple"
trusted-host=pypi.python.org
```

Install via proxy & no SSL check

```
sudo pip install --proxy $PROXY --trusted-host pypi.python.org
pylzma
```

PIPENV (new and improved pkg handler)

install pipenv

```
pip install --user pipenv
```

start new project

```
cd /home/user && pipenv install
```

```
    while True:
        if (counter > n):
            return
        yield a

        a, b = b, a + b
        counter = counter + 1


## Call the Function
f = fibonacci(27) #f is iterator object
```

```
def add(a, b):
   print "ADDING %d + %d" % (a, b)
   return a + b
add(30,55)

Yield - returns multiple values from function
```

Range

range(stop)

range(start, stop, [step size])

```
var = range(10)
print var
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

for x in range(0, 30, 5)
[0, 5, 10, 15, 20, 25]
```

Get variables in scope

- `dir()` will give you the list of in scope variables:
- `globals()` will give you a dictionary of global variables
- `locals()` will give you a dictionary of local variables

Kwargs

provide unlimited key=value pairs into a function

```
def print_values(**kwargs):
    for key, value in kwargs.items():
        print("The value of {} is {}".format(key, value))

print_values(my_name="Sammy", your_name="Casey")
```

MATH

compare

compare 2 variables x,y (1,2 = -1 1,1 = 0 2,1 = 1)

```
print cmp(1,2)
print cmp(2,2)
print cmp(2,1)

output:
-1
0
1
```

Return

```
def multiply(a,b):
 return a * b

# convert input string to int
a = int(raw_input("enter first number: "))
b = int(raw_input("enter second number: "))

print multiply(a,b)
```

get min, max, sum

```
digits = [2,5,343,11,0,-1]
print(min(digits))
print(max(digits))
print(sum(digits))
```

generate random number

```
from random import randint
print randint(0,15)
```

install dependencies from requirements file
```
pipenv install -r requirements.txt
```

start Pipenv virtual env
```
pipenv shell  (type 'exit' to leave virtual env)
```

run a script w/o switching to shell
```
pipenv run python myscript.py
```

generate Lock file (willl force same packages for all users)
```
pipenv lock
```

generate a setup.py file from all current variables
```
pipenv install -e .
```

remove all modules installed outside of Lock file
```
pipenv clean
```

remove virtual environment
```
pipenv --rm
```

show all installed packages in virtual env
```
pipenv graph
```

Virtual ENV

start venv
```
virtualenv venv
source venv/bin/activate
```

stop venv
```
deactivate
```

JSON

convert multidimensional Dictionary to Json
```
import json


employee = {}
employee['joe'] = { 'id': 5, 'city':'boston' }

json_object = json.dumps(employee)

print(json_object)
```
```
{"joe": {"city": "boston", "id": 5}}
```

traverse JSON data object
```
json_object = {"joe": {"city": "boston", "id": 5}}

data = json.loads(json_object)

for name in data:
    print name
    print data[name]['id']
    print data[name]['city']
```

open json file for reading
```
with open('file.json') as json_data:
    data = json.load(json_data)
    print data['somevalue']
```

pretty print large JSON file from cmd line,
```
cat file.json | python -m json.tool
```

YAML

open and read YAML file
```
import yaml

with open("myfile.yaml", "r") as file:
    try:
        print(yaml.load(stream))
    except yaml.YAMLError as e:
        print(str(e))
```

REGEX

import re

find string between 2 patterns
```
string = "{font color=red}Mary had a little lamb{/font}"
text = re.search("}(.*){", string).group(1)
>> Mary had a little lamb
```

Fabionacci generator
```
fibs = [0, 1]
for i in range(8):
    fibs.append(fibs[-2] + fibs[-1])

## 0,1,1,2,3,5,8,13,21,34
```

PyPI

upload a package to PyPI
create setup.py (see this as example: https://github.com/perfecto25/dictor/blob/master/setup.py)

install setuptools and twine
```
python3 -m pip install --user --upgrade setuptools wheel
python3 -m pip install --user --upgrade twine
```

generate distributables
```
python3 setup.py sdist bdist_wheel
```

upload to PyPI
```
python3 -m twine upload dist/*
```

string wildcard

```
import fnmatch

for file in os.listdir('.'):
    if fnmatch.fnmatch(file, '*.txt'):
        print file
```

EXCEPTIONS & DEBUG

Read in json, if fails, ignore

```
try: json_object = json.loads(myjson)
except ValueError, e: pass
```

run script with full debug

```
python -m trace -t script.py
```

print literal parameter name, not value
data = { "name": "Joe" }
value = data['name]

Port & Network Connectivity

sping up simple web server over a custom port to check basic connectivity

```
serverA> python -m SimpleHTTPServer 8331
```

now check for connectivity using netcat

```
serverB> nc serverA 8331 -vvv
```

ENCRYPT & HASH

generate hash of a password:

```
python -c 'import crypt,getpass;
print(crypt.crypt(getpass.getpass(),
crypt.mksalt(crypt.METHOD_SHA512)))'
```