Tweetable Python Documentation

Release 3.7

Agiliq

Jul 29, 2018

Table of Contents:

Judge me by my size, do you? Hmm, hmm. - Yoda.

A book of 128 short (280 chars or less) python commands which get the job done.

Inspired by Peteris Krumins' Perl One Liners.

The book is a collection of 128 small python programs. Each of these programs does something useful and/or fun.

Take this example

```
import random, string
"".join([random.choice(string.ascii_letters + string.digits) for i in range(8)])
```

In less than a 100 chars, you can generate truly random passwords.

We will look at things like file manipulation, file conversion, password generation and even a game.

Lets get started.

CHAPTER 1

String manipulation

1.1 Invert letter case of string

1.2 Rot13 a String

1.3 left pad

1.4 Speaking in ubbi dubbi

https://www.youtube.com/watch?v=rfR03gibh6M https://en.wikipedia.org/wiki/Ubbi_dubbi

1.5 Pig latin

https://en.wikipedia.org/wiki/Pig_Latin

1.6 convert repeated spaces to one space

```
import re; re.sub(r"[ ]+", ' ', 'this    sentence          has              non-
↪uniform      spaces')
```

The above snippet clears out the repeated spaces in a text and replaces it with single space. re is a regular expression module to find more than one occurrences of space with '[ ]+'.

1.7 Check if a string is a valid IP v4 address

1.8 Check if a string is a valid IP v6 address

1.9 Check if string is palindrome

1.10 Find all valid anagrams of a word

CHAPTER 2

File manipulation

2.1 oxford comma

2.2 count words in file

```
len(open('data/test.txt', 'r').read().split())
```

Returns the number of words in a text file, test.txt. `open('data/test.txt', 'r').read()` gets us the text of the file, we get the word using `.split()`, and `len` gives us the count of the words.

To run it for arbitrary files

```
$ python -c "import sys; print(len(open(sys.argv[1], 'r').read().split()))" data/test.
↪txt
47
```

2.3 count lines in file

```
len(open('data/test.txt', 'r').read().split('\n'))
```

Returns the number of lines in a text file, test.txt `open('data/test.txt', 'r').read()` gets us the text of the file, we get the word using `.split('\n')`, and `len` gives us the count of lines.

To run it for arbitrary files

```
$ python -c "import sys;len(open(sys.argv[1], 'r').read().split('\n'))" data/test.txt
54
```

2.4 add spaces after punctuation

```
def repl(*args): obj=args[0]; return obj.string[obj.start()]+' '+obj.string[obj.
↪start()+1]
import re, string; re.sub('['+string.punctuation+'][a-zA-Z0-9]+', repl, "this'will;be.
↪formatted,with! spaces")
```

Parse text and add a space after punctuations if its not present. If the space after the punctuation is present it will remain intact.

2.5 add line numbers to text file

```
out=open('data/test-out.txt', 'w')
for i, j in enumerate(open('data/test.txt', 'r')): out.write(str(i+1) + j)
out.close()
```

Creates a new file with line number before each line.

2.6 add line numbers to text file, don't number empty lines

```
out=open('data/test-out.txt', 'w'); i=0
for line in open('data/test.txt', 'r'):
    if line.strip(): out.write(str(i)+line);i+=1
    else: out.write(line)
out.close()
```

Creates a new file with line number before each line. If the line is empty it will be skipped.

2.7 delete trailing spaces

2.8 delete multiple newlines between paragraphs to keep only one line

```
out=open('data/out-single-line-gap.txt', 'w')
out.write((re.sub('(\n\n)[\n]*', '\n\n', open('data/test.txt','r').read())))
```

Delete multiple new lines from a file between paragraphs and save it in a new file.

To run it for arbitrary files

```
$ python -c "import sys;out=open('data/out-single-line-gap.txt', 'w');out.write((re.
↪sub('(\n\n)[\n]*', '\n\n', open(sys.argv[1],'r').read())))" data/test.txt
```

2.9 first ten lines of file

```
open('data/100west.txt', 'r').read().split('\n')[:10]
```

Returns first 10 lines of a file.

To run it for arbitrary files

```
$ python -c "import sys; open(sys.argv[1], 'r').read().split('\n')[:10]" data/test.txt
```

2.10 last ten lines of file

```
open('data/100west.txt', 'r').read().split('\n')[-10:]
```

Returns last 10 lines of a file

To run it for arbitrary files

```
$ python -c "import sys; open(sys.argv, 'r').read().split('\n')[-10:]" data/test.txt
```

2.11 Reverse a file line by line

2.12 Get alternate lines from files starting from the top

2.13 Find the most common words in a file

2.14 Find the lines which match a specified text

CHAPTER 3

File conversions

3.1 csv to json

```
import csv,json;reader = csv.DictReader(open('data/example.csv', 'r'), fieldnames=(
↪"User","Country","Age"))
out=open('data/out.json','w'); out.write(json.dumps([row for row in reader]))
```

Converts a given file of comma separated values to json and store it in another file. The csv used here doesn't contain headers for the columns.

`csv.DictReader(open('data/example.csv', 'r'), fieldnames=( "User","Country", "Age"))` gets us the DictReader object from the csv file with three columns specified as fieldnames. `json.dumps([row for row in reader])`, converts the json object to string for writing it in output file and `out.write()` method writes the data into the file.

To run it for arbitrary files

```
$ python -c "import csv,json,sys;reader = csv.DictReader(open(sys.argv[1], 'r'),␣
↪fieldnames=( "User","Country","Age"));out=open('data/out.json','w'); out.write(json.
↪dumps([row for row in reader]))" data/example.csv
```

3.2 json to csv

```
import json, csv; _json=json.loads(open('data/example.json', 'r').read())
out=open('data/converted.csv', 'w')
[out.write(','.join([x[x.keys()[i]] for i in range(len(x))]) + '\n')  for x in _json]
```

Creates a csv file from a json file.

`json.loads(open('data/example.json', 'r').read())` Creates a json object from json file.

3.3 csv to sqlite

3.4 base64 encoding

```
import base64;base64.encode(open('data/100west.txt', 'r'), open('data/encoded.txt', 'w
↪'))
```

Converts an input file to base64 encoded file.

To run it for arbitrary files

```
$ python -c "import base64,sys,sys;base64.encode(open(sys.argv[1], 'r'), open('data/
↪encoded.txt', 'w'))" data/test.txt
```

3.5 base64 decoding

```
import base64;base64.decode(open('data/encoded.txt', 'r'), open('data/decoded.txt', 'w
↪'))
```

Decodes the base64 encoded file back to its original encoding.

To run it for arbitrary files

```
$ python -c "import base64,sys;base64.decode(open(sys.argv[1], 'r'), open('data/
↪decoded.txt', 'w'))" data/test.txt
```

3.6 zip all .txt files in directory

```
import zipfile, os; myzip = zipfile.ZipFile('test.zip', 'w'); [myzip.write(each) for␣
↪each in os.listdir() if each.endswith('.txt')]
```

Creates a zip file called test.zip of all the .txt files present in your current directory. zipfile.ZipFile creates a new zip file. `os.listdir()` lists all the files in the current directory.

To run it for arbitrary directory. You must provide the absolute path to the directory

```
$ python -c "import zipfile,os,sys; myzip=zipfile.ZipFile('test.zip', 'w'); [myzip.
↪write(each) for each in os.listdir(sys.argv[1]) if each.endswith('.txt')]" /User/
↪xyz/files/
```

3.7 batch rename files in directory

Change all files with `.txt` extension to `.rst`.

3.8 Rename all files in directory to add .bak extension

Copy all files with name `filename.ext` to *filename.bak.ext*

3.9 Managing your downloads folder

When you download files, it generally goes `~/Downloads`. This folder grows and becomes unwieldy as time goes.

So to manage this, we will create a folder of format `YYYY-MM` in the `~/Downloads` folder and move all files tothe correct folder.

3.10 prettify json

```
import json; json.dumps([{"one":123,"two":455,"three":789}], indent=4)
```

Returns a prettified json string for the given json object. The above json object will be converted as below:

```
[
    {
        "one": 123,
        "two": 455,
        "three": 789
    }
]
```

CHAPTER 4

Password generation and validation

4.1 Random Alpha-numeric passwords

```
import random, string; "".join([random.choice(string.ascii_letters + string.digits)␣
↪for i in range(8)])
```

Creates a random alpha-numberic string of 8 characters that can be used as a password.

4.2 Pronounceable passwords

```
import random;words=open('/usr/share/dict/words').read().split(); "-".join([random.
↪choice(words) for _ in range(4)])
```

Creates a random '-' separated 4 word password from the given input text file to be used as a password

4.3 Passwords with alternate vowels

```
import random, string, itertools;
"".join(itertools.chain(*zip([random.choice(string.ascii_lowercase) for _ in␣
↪range(6)],  [random.choice('aeiou') for _ in range(6)])))
```

4.4 Caesar Cipher

CHAPTER 5

Ascii art

5.1 asterisk triangle

```
print('\n'.join(['*' for _ in range(x+1)]) for x in range(5)]))
```

Generates an incremental triangle of asterisk '*'. Example:

```
*
**
***
****
*****
```

```
print('\n'.join(['*' for _ in range(5-x)]) for x in range(5)]))
```

Prints out asterisk triangle in decremental order. Example:

```
*****
****
***
**
*
```

```
print('\n'.join([ (11-x)*' '+ ''.join(['* ' for _ in range(x)]) for x in range(1,11,␣
↪2)]))
```

Prints out a asterisk triangle as below:

```
|             *
|           * * *
|         * * * * *
|       * * * * * * *
|     * * * * * * * * *
```

5.2 Alternating Pattern

```
print('\n'.join([(x+1)*'#'+(7-x)*'*' for x in range(7)]))
```

Prints alternating patterns in # and * based on a given integer n. n=7 here.

```
#*******
##******
###*****
####****
#####***
######**
#######*
```

5.3 banners (cowsay)

CHAPTER 6

Mathematic

6.1 Decimal to Binary

6.2 Binary to decimal

6.3 Factorial sequence

6.4 Fibonacci Sequence

6.5 GCD of two number

6.6 LCM of two numbers

6.7 pascal's triangle

```
print('\n'.join([ (6-x)*' '+ ''.join(['{} '.format(p) for p in str(11**x)]) for x in␣
↪range(6) if x!=1]))
```

Prints out Pascal's triangle as below

```
|              1
|            1 2 1
|          1 3 3 1
|        1 4 6 4 1
|      1 6 1 0 5 1
```

6.8 OEIS sequence A127421

```
[(x-1)*(10**len(str(x))) + x for x in range(1,19)]
```

Prints out an OEIS sequence A127421, numbers whose decimal expansion is a concatenation of 2 consecutive increasing non-negative numbers. 1, 12, 23, 34, 45, 56, 67, 78, 89, 910, 1011, 1112, ...

6.9 Check if a number is a prime

6.10 Find largest number among numbers passed on command line

6.11 Find largest number among numbers passed on command line

6.12 unit converter

6.13 transpose a matrix

6.14 Sieve of Eratosthenes

CHAPTER 7

Networking

7.1 get local hostname

```
import os; os.uname().nodename

# or

import socket; print(socket.gethostname())
```

Run this as

```
$ python -c "import os; print(os.uname().nodename)"
shabdas-MacBook-Pro.local
```

7.2 Get IP Address

```
import socket; s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM);s.connect(("8.8.8.
↪8", 80));print(s.getsockname()[0])
```

Run it as

```
$ python -c "import socket; s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM);s.
↪connect(('8.8.8.8', 80));print(s.getsockname()[0])"
192.168.31.253
```

7.3 Get IP Address using a remote service

```
import urllib.request, json
json.loads(urllib.request.urlopen('http://jsonip.com').read())['ip']
```

Run it as

```
$ python -c "import urllib.request, json; print(json.loads(urllib.request.urlopen(
↪'http://jsonip.com').read())['ip'])"
183.83.214.40
```

7.4 Generate a Random IPv6 Address

7.5 Generate a gravatar url from email

CHAPTER 8

Date and Calendar

8.1 Display current year's calendar

```
python -c "import calendar, datetime; today = datetime.datetime.today();
↪print(calendar.TextCalendar().formatyear(today.year))"
```

8.2 Display current month's calendar

```
$ python -c "import calendar, datetime; today = datetime.datetime.today();
↪print(calendar.TextCalendar().formatmonth(today.year, today.month))"
    July 2018
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
```

8.3 Number of days remaining to Christmas

8.4 Is the current year a leap year?

8.5 Unix epoch time

CHAPTER 9

python -m

9.1 pretty print a json

`python -m json.tool` will format a string

```
$ cat data/example.json
[{"fieldname0": "User", "fieldname1": "Country", "fieldname2": "Age"}, {"fieldname0":
↪"Alex", "fieldname1": "US", "fieldname2": "25"}, {"fieldname0": "Ben", "fieldname1
↪": "US", "fieldname2": "24"}, {"fieldname0": "Dennis", "fieldname1": "UK",
↪"fieldname2": "25"}, {"fieldname0": "Yuvi", "fieldname1": "IN", "fieldname2": "24"}
↪](django-admin-cookbook)

$ cat data/example.json | python -m json.tool
[
    {
        "fieldname0": "User",
        "fieldname1": "Country",
        "fieldname2": "Age"
    },
    {
        "fieldname0": "Alex",
        "fieldname1": "US",
        "fieldname2": "25"
    },
    {
        "fieldname0": "Ben",
        "fieldname1": "US",
        "fieldname2": "24"
    },
    {
        "fieldname0": "Dennis",
        "fieldname1": "UK",
        "fieldname2": "25"
    },
```

(continues on next page)

(continued from previous page)

```
    {
        "fieldname0": "Yuvi",
        "fieldname1": "IN",
        "fieldname2": "24"
    }
]
```

9.2 Expose a folder to a static file server

`python -m http.server` will start a server on port 8000 which will server the files from current dir.

```
$ python -m http.server 8844
Serving HTTP on 0.0.0.0 port 8844 (http://0.0.0.0:8844/) ...
```

9.3 A simple editor for Python

Every python install comes with the idle editor, you can start it like this `python -m idlelib.idle`

..image:: _static/idle.png

9.4 Debugging emails

python -m smtpd -n -c DebuggingServer localhost:1025

CHAPTER 10

List, sets and other groups

CHAPTER 11

Misc

11.1 fizzbuzz

11.2 UPside down text

```
def replacement(x): return dict(zip(string.ascii_letters+string.digits,
↪'qpluodbsnxzqpHIWNOQSΛMXZ0986')).get(x, ' ')
print("".join([replacement(ch) for ch in "Guido is a smart guy"]))
```

CHAPTER 12

Ester eggs

12.1 The world's smallest hello world program?

```
import __hello__
```

Or `python -c 'import __hello__'`.

12.2 What would Guido do?

```
import this
```

The `import this` is well known, but let's look at the source code.

```
s = """Gur Mra bs Clguba, ol Gvz Crgref

Ornhgvshy vf orggre guna htyl.
Rkcyvpvg vf orggre guna vzcyvpvg.
Fvzcyr vf orggre guna pbzcyrk.
Pbzcyrk vf orggre guna pbzcyvpngrq.
Syng vf orggre guna arfgrq.
Fcnefr vf orggre guna qrafr.
Ernqnovyvgl pbhagf.
Fcrpvny pnfrf nera'g fcrpvny rabhtu gb oernx gur ehyrf.
Nygubhtu cenpgvpnyvgl orngf chevgl.
Reebef fubhyq arire cnff fvyragyl.
Hayrff rkcyvpvgyl fvyraprq.
Va gur snpr bs nzovthvgl, ershfr gur grzcgngvba gb thrff.
Gurer fubhyq or bar-- naq cersrenoyl bayl bar --boivbhf jnl gb qb vg.
Nygubhtu gung jnl znl abg or boivbhf ng svefg hayrff lbh'er Qhgpu.
Abj vf orggre guna arire.
Nygubhtu arire vf bsgra orggre guna *evtug* abj.
```

(continues on next page)

(continued from previous page)

```
Vs gur vzcyrzragngvba vf uneq gb rkcynva, vg'f n onq vqrn.
Vs gur vzcyrzragngvba vf rnfl gb rkcynva, vg znl or n tbbq vqrn.
Anzrfcnprf ner bar ubaxvat terng vqrn -- yrg'f qb zber bs gubfr!"""

d = {}
for c in (65, 97):
    for i in range(26):
        d[chr(i+c)] = chr((i+13) % 26 + c)

print("".join([d.get(c, c) for c in s]))
```

It doesn't contain the Zen of Python, but `d[chr(i+c)] = chr((i+13) % 26 + c)` gives it away. It is the Zen of Python with a Rot13 transform.

12.3 Are we Java yet?

```
>>> from __future__ import braces

File "<stdin>", line 1
SyntaxError: not a chance
```

How does this work? Look at `__futures__.py`. You won't find any `braces`. This error gets propagated to `parso/python/errors.py`, where we find:

```
@ErrorFinder.register_rule(type='import_from')
class _FutureImportRule(SyntaxRule):
    message = "from __future__ imports must occur at the beginning of the file"

    def is_issue(self, node):
        if _is_future_import(node):
            if not _is_future_import_first(node):
                return True

            for from_name, future_name in node.get_paths():
                # ...
                if name == 'braces':
                    self.add_issue(node, message = "not a chance")
                # ...
```

12.4 I believe I can fly

```
import antigravity
```

This takes you to https://xkcd.com/353/. Let's look at `antigravity.py`.

```
import webbrowser
import hashlib

webbrowser.open("https://xkcd.com/353/")

def geohash(latitude, longitude, datedow):
```

(continues on next page)

(continued from previous page)

```
    '''Compute geohash() using the Munroe algorithm.
    # ...
```

`webbrowser.open("https://xkcd.com/353/")` is what opens the web page. This is another example of how "batteries included" Python is. It even comes with and *Interfaces for launching and remotely controlling Web browsers*.

CHAPTER 13

Indices and tables

- genindex
- modindex
- search