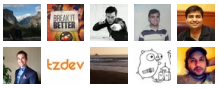
2.3K WANT ANSWERS



Latest activity: 10m ago

QUESTION TOPICS

Tips and Hacks

Ruby (programming language)

Ruby on Rails (web framework)

Python (programming language)

Computer Science

Computer Programming

Edit Topics

SHARE QUESTION

Twitter

Facebook

QUESTION STATS

Views	345,698
Want Answers	2398
Edits	

★ What are some cool Python tricks?

Write Question Details

Want Answers 2.3k Comments 3 Share 49 Downvote

127 ANSWERS ASK TO ANSWER

Balakrishnan Vasudevan, Indian | Graduate Student at UCSB | Former... (more)
Edit Biography • Make Anonymous

Write your answer, or answer later

Anubhav Yadav, A curious programmer
4.1k upvotes by Greeshma Girish, Lorenzo Hernandez, Ezugworie Ikechukwu, (more)

The Cricket World Cup fever is on and everyone in my office used to frequent websites like cricinfo and cricbuzz to see the latest scores.

I happened to use a ubuntu machine here at my office to I wrote a simple python script to fetch the latest cricket score and send me a notification every 1 minute.

Here's the script:

```
#!/usr/bin/env python2
import requests
from bs4 import BeautifulSoup
import pynotify
from time import sleep
def sendmessage(title, message):
    pynotify.init("Test")
    notice = pynotify.Notification(title, message)
    notice.show()
    return
url = "http://static.cricinfo.com/rss/livescores.xml"
while True:
    r = requests.get(url)
    while r.status_code is not 200:
            r = requests.get(url)
    soup = BeautifulSoup(r.text)
    data = soup.find_all("description")
    score = data[1].text
    sendmessage("Score", score)
    sleep(60)
```

Here is a screenshot of the script in action while the semifinal between New Zealand and South Africa is on:

RELATED QUESTIONS

What is ** in Python?

Why is python a cool programming language?

What are some cool Ruby tricks?

What are some of the best Python tricks that you have come across?

What are some tips and tricks every aspiring Python Jedi should know?

What are some cool bit manipulation tricks/hacks?

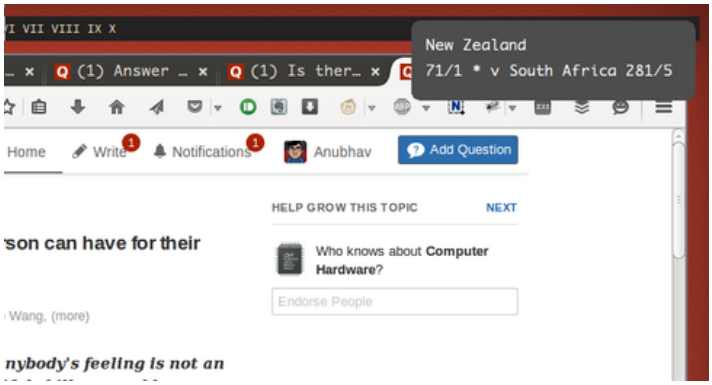
What are Python's strengths?

What are some cool easter eggs of python?

What are some cool MATLAB tricks?

What are some cool hacking tricks using cmd.exe?

More Related Questions



The code should run on any linux machine where libnotify is installed, provided the python modules imported are there in your machine.
I am now thinking to modify the script such that I should get a notification whenever runs are scored or a wicket is taken, instead of just sending me a notification every one minute.

EDIT: Who knows what this script can become. I have pushed it onto my github here: https://github.com/neo1691/score...
Feel free to contribute to it. Thanks

EDIT2: Please have a look at the github version of scorer.py . It has evolved greatly. Thank you.

Updated Thu. 78,925 views.

Upvote 4.1k Downvote Comments 45+ Share 10

Shekhar Sharma, Qurious
1.2k upvotes by Jessica Su, Namit Katariya, Colin Ulmer, (more)

A simple HTTP Server can be started in seconds.
`python -m SimpleHTTPServer`
For Python3: `python -m http.server`
The server starts on port 8000 by default which can be changed. I have found this quite handy at times.

For example: To share a complete directory with someone over the Internet, I cd to the directory and start the server. The directory is shared, and I share my IP address. The directory can be viewed and files can be downloaded easily. I can also monitor access requests on the terminal.

Updated 22 Jun, 2014. 49,774 views.

Upvote 1.2k Downvote Comments 23+ Share 9

Anders Kaseorg, MIT PhD student in Computer Science
824 upvotes by Jay Wacker, Charlie Cheever, Jessica Su, (more)

List comprehensions and generator expressions

Instead of building a list with a loop:

```
b = []
for x in a:
    b.append(10 * x)
foo(b)
```

you can often build it much more concisely with a **list comprehension**:

```
foo([10 * x for x in a])
```

or, if `foo` accepts an arbitrarily iterable (which it usually will), a **generator expression**:

```
foo(10 * x for x in a)
```

Python 2.7 supports dict and set comprehensions, too:

```
>>> {x: 10 * x for x in range(5)}
{0: 0, 1: 10, 2: 20, 3: 30, 4: 40}
>>> {10 * x for x in range(5)}
set([0, 40, 10, 20, 30])
```

Fun tricks with `zip`

Transposing a matrix:

```
>>> l = [[1, 2, 3], [4, 5, 6]]
>>> zip(*l)
[(1, 4), (2, 5), (3, 6)]
```

Dividing a list into groups of `n`:

```
>>> l = [3, 1, 4, 1, 5, 9, 2, 6, 5, 3, 5, 8]
>>> zip(*[iter(l)] * 3)
[(3, 1, 4), (1, 5, 9), (2, 6, 5), (3, 5, 8)]
```

`import 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!
```

Updated 17 Aug, 2014. 100,964 views.

Upvote 824 Downvote Comments 10+ Share 10

Joachim Krügermeister, Inquisitive
210 upvotes by Jessica Su, Nitin Madnani, Arvind Shrihari, (more)

Create infinities

Infinity, and it's brother minus infinity, comes in handy once in a while.

```
my_inf = float('Inf')
99999999 > my_inf
-> False

my_neg_inf = float('-Inf')
my_neg_inf > -99999999
-> False
```

Intuitive comparisons

A great example of the simplicity of python syntax.

```
x = 2
3 > x == 1
-> False
1 < x < 3
-> True
10 < 10*x < 30
-> True
10 < x**5 < 30
-> False
100 < x*100 >= x**6 + 34 > x <= 2*x <5
-> True
```

Enumerate it
Ever wanted to find that damn index when you're inside a loop?

```
mylist = [4,2,42]
for i, value in enumerate(mylist):
    print i, ': ', value
-> 0: 4
-> 1: 2
-> 2: 42
```

Reverse it
This has grown to become a part of my morning ritual. Reverse. Anywhere. Anytime. All the time.

```
# Reverse the list itself:
mylist = [1,2,3]
mylist.reverse()
print mylist
-> [3,2,1]
# Iterate in reverse
for element in reversed([1,2,3]): print element
-> 3
-> 2
-> 1
```

Ultra compact list generating
Using nested list comprehensions you can save a great deal typing, while having fun impressing the girls.

```
[(x**2, y**2, x**2+y**2) for x in range(1,5) for y in range(1,5) if x<=
-> [(4, 4, 8), (4, 9, 13), (4, 16, 20), (16, 16, 32)]
```

NB! Crazy nesting should be used with extreme caution.

```
Readability > Girls
-> True
```

Splat call
'*' is called the splat operator, and may make you smile. It automagically unpacks stuff in a function call.

```
def foo(a, b, c):
    print a, b, c

mydict = {'a':1, 'b':2, 'c':3}
```

```
mylist = [10, 20, 30]

foo(*mydict)
-> a, b, c
foo(**mydict)
-> 1, 2, 3
foo(*mylist)
-> 10 20 30
```

The cute empty string trick

By using two single quotes ('') and a dot (.), we have access to all the builtin string functions. This can come in handy, you see.

```
''.join('I','Want','Just','One','String')
-> 'IWantJustOneString'
```

Itertools

The itertools module provide some useful and efficient functions for us. For example

```
from itertools import chain
''.join(('Hello ', 'Kiddo'))
-> 'Hello Kiddo'
''.join((x for x in chain('XCVGOHST', 'VSBNTSKFDA') if x == 'O' or x ==
-> 'OK'
```

When Las Vegas just isn't enough

Buy in some beer, invite a few (exactly 10) friends over, and copy/paste this sexy line of python code into your favorite interpreter. The rules are:

1. Press Enter
2. The one who gets the least stars have to CHUG CHUG CHUG!
3. Press the up arrow
4. Goto 1.

```
print "\n".join(str(i)+":\t"+"*"*randint(1,10) for i in range(1,11))
```

Update:

Make python enums

I like this enumification trick:

```
class PlayerRanking:
  Bolt, Green, Johnson, Mom = range(4)

PlayerRanking.Mom
-> 4
```

Updated 21 Aug, 2012. 18,654 views.

Upvote 210 Downvote Comments 10+ Share 2

Navin Kabra, Python's my primary language for deve... (more)
144 upvotes by Nitin Madnani, Hiroshi Ono, Raj Reddy, (more)

Find the longest line in a file:

```
max(open('test.txt'), key=len)
```

Sum the digits in an unsigned integer:

```
sum(map(int, str(n)))
```

Get sparklines in text output using python: https://gist.github.com/1371985
Like so:

spark 1,5,22,13,53

A full webserver:
python -m CGIHTTPServer

Updated 29 Oct, 2013. 14,981 views.

Upvote 144 Downvote Comments 2+ Share 1

Jakub Roztočil, Pythonista
369 upvotes by Jessica Su, Jake Alon, Shoma Suzuki, (more)

```
import antigravity
```

Written 1 Dec, 2011. 14,976 views.

Upvote 369 Downvote Comments 16+ Share 8

Kailash Budhathoki, been using python for more than 3 years
50 upvotes by Anders Kaseorg, Ivo Danihelka, Saurabh Sharan, (more)

To add to Anders' suggestions, **Swapping Values**
b, a = a, b

Written 1 Dec, 2011. 7,133 views.

Upvote 50 Downvote Comment 1 Share

Avinash Kumar Sharma, Learning...
99 upvotes by Christopher VanLang, Sriram Madapusi Vasudevan, Edwin Khoo, (more)

There are many to answer but I will mention the one I use very often and which saves lot of time. It is **Reversing a string or a list in one line of code.**
Example:
For String :

```
"Hello world"[::-1]
```

This gives the output as 'dlrow olleH'
or for a list similarly it is

```
[1, 2, 3][::-1]
```

This gives the output as [3,2,1]

Updated 27 Feb, 2014. 8,893 views.

Upvote 99 Downvote Comments 1+ Share 1

Sam Thomson, Ph.D. student studying NLP at CMU
30 upvotes by Dmitri Skjorshammer, Risent Zhang, Anand Gupta, (more)

use zip to arrange words into n-grams

```
1 n = 3
2 with open('/path/to/file') as my_file:
3     words = my_file.read().split()
4 ngrams = zip(*[words[i:] for i in range(n)])
```

Written 2 Dec, 2011. 5,208 views.

Upvote 30 Downvote Comments 5 Share

Fred Cirera
73 upvotes by Alix Franquet, Sean Clarke, Ryan Balfanz, (more)

Create self contained python executables.

Every one knows the special file "__init__.py ". There is another special file "__main__.py ". If this file exists in a directory, the python interpreter will try to execute it.

Here is an example:

```
#fred:$ find .
.
./hello
./hello/__main__.py
```

In our example the content of the file is a simple loop that prints hello world.

```
#fred:$ cat ./hello/__main__.py
#
for count in range(3):
  print 'hello (cruel) world'
```

Just call python with the directory name and the python interpreter will execute the file __main__.py

```
#fred:$ python hello
hello (cruel) world
hello (cruel) world
hello (cruel) world
```

Now the interesting part...

Python is capable of importing and executing whatever python code is in a zip file. Therefore we can do the following:

```
#fred:$ (cd hello; zip -r ../hello.zip .)
  adding: __main__.py (stored 0%)

#fred:$ python hello.zip
hello (cruel) world
hello (cruel) world
hello (cruel) world
```

On any Unix like system we can use a shebang (#!) line at the beginning of a file to pass the content of that file to the interpreter specified after the shebang.

```
#fred:$ echo '#!/usr/bin/env python' > greetings
#fred:$ cat hello.zip >> greetings
#fred:$ chmod a+x greetings
```

We can now execute that file.

```
#fred:$ ./greetings
hello (cruel) world
hello (cruel) world
hello (cruel) world
```

Mix that with virtualenv and you will have a powerful way to simply distribute and execute your code.

Written 26 Jul, 2014. 7,785 views.

Upvote 73 Downvote Comments 2 Share 1

Avinash SM, Py Guy!
371 upvotes by Mani Cavalieri, Madhur Bhaiya, Ken Witknow, (more)

The following are just a collection of some useful shortcuts and tools I've

found in Python over the years. Hopefully you find them helpful.

Swapping Variables

```
x = 6
y = 5
x, y = y, x

print x
>>> 5
print y
>>> 6
```

Inline if Statement

```
print "Hello" if True else "World"
>>> Hello
```

Concatenations

The last one is a pretty cool way to combine objects of two different types.

```
nfc = ["Packers", "49ers"]
afc = ["Ravens", "Patriots"]
print nfc + afc
>>> ['Packers', '49ers', 'Ravens', 'Patriots']
```

```
print str(1) + " world"
>>> 1 world
```

```
print `1` + " world"
>>> 1 world
```

```
print 1, "world"
>>> 1 world
```

```
print nfc, 1
>>> ['Packers', '49ers'] 1
```

Number Tricks

```
#Floor Division (rounds down)
print 5.0//2
>>> 2
```

```
#2 raised to the 5th power
print 2**5
>> 32
```

Be careful with division and floating point numbers.... (more)

Upvote 371 Downvote Share 9

Dibyendu Nath, Pythonista

31 upvotes by Viggnesh Venugopal, Hasan Tayyar Beşik, Stephen McInerney, (more)

For pretty printing minimized JSON string on the fly, do this:

```
$ echo '{"key":"value"}' | python -m json.tool
```

Output:

```
{
    "key": "value"
}
```

Written 16 Sep, 2014. 2,697 views.

Luciano Ramalho, Python user and instructor since 1998

32 upvotes by Hieu Tran, Seyram Komla Sapaty, Mani Cavalieri, (more)

Generate an index of of words in a text file, listing the line number of each occurrence:

```
import sys
import re

NONWORD_RE = re.compile('\W+')

idx = {}
with open(sys.argv[1], encoding='utf-8') as fp:
    for n, line in enumerate(fp, 1):
        for word in NONWORD_RE.split(line):
            if word.strip():
                idx.setdefault(word, []).append(n)

for word in sorted(idx, key=str.upper):
    print (word, idx[word])
```

Sample run (indexing the words of the code above):

```
$ python3 index.py index.py
1 [7, 8]
8 [7]
append [11]
argv [7]
as [7]
compile [4]
encoding [7]
enumerate [8]
for [8, 9, 13]
fp [7, 8]
idx [6, 11, 13, 14]
if [10]
import [1, 2]
in [8, 9, 13]
key [13]
line [8, 9]
n [8, 11]
NONWORD_RE [4, 9]
open [7]
print [14]
re [2, 4]
setdefault [11]
sorted [13]
split [9]
```

```
26 str [13]
27 strip [10]
28 sys [1, 7]
29 upper [13]
30 utf [7]
31 W [4]
32 with [7]
33 word [9, 10, 11, 13, 14, 14]
```

Written 4 Mar, 2014. 3,929 views.

Upvote 32 Downvote Comment Share 1

Riobard Zhan, Minimalist Pythonista
27 upvotes by Nitin Madnani, Sergey Kolosov, Saurabh Sharan, (more)

Accessing from the right-hand side of lists (or strings) symmetrically without the offset-by-one problem. A typical use case: check if a string is a palindrome:

```
s = '1234567890987654321'
all(s[i] == s[-i-1] for i in range(len(s))) # offset-by-one
all(s[i] == s[~i] for i in range(len(s)))  # symmetric accessing
```

This is due to the fact that

```
~0 == -1
~1 == -2
```

Written 1 Dec, 2011. 4,579 views.

Upvote 27 Downvote Comments 3+ Share

Pratik Tandel, @twitter eng
47 upvotes by Nikhil Garg, Yam Mesicka, Abhishek Shivkumar, (more)

Pretty printing JSON using python

```
$ echo '{"hello": "world"}' | python -mjson.tool
{
    "hello": "world"
}
```

or

```
$ python -mjson.tool file.json
```

Written 5 Sep, 2013. 5,930 views.

Upvote 47 Downvote Comment Share 3

Sherif Morris, Biologist, Python fanatic, Bioinforma... (more)
35 upvotes by Seyram Komla Sapaty, Mohitdeep Singh, Santiago Garza, (more)

This coolest thing, to me anyway - is Pythons slice notation. It is very straightforward and after programming in Java for a while I was amazed to see how easily Python handled this:

```
a[start:end] # items start through end-1, if you're counting from 1 a
a[start:]    # items start through the rest of the list
a[:end]      # items from the beginning through end-1
a[:]         # a copy of the whole list
```

Finally, to invert the entire string:

```
a[::-1]
```

Once i saw this, every doubt I had about Pythons usability was removed and I haven't regretted since ;)

Written 14 Apr, 2014. 3,579 views.

Upvote 35 Downvote Comments 4 Share

Ashwin Sathya, Polyglot
247 upvotes by William Chen, Mahmoud Emad, Sahil Juneja, (more)

This little devilish piece of code to swap two numbers swept me off my feet

```
a,b=b,a
```

Updated 6 Sep, 2013. 10,989 views.

Upvote 247 Downvote Comments 3+ Share 1

Rahul Khanvilkar, Opposite of a party animal
30 upvotes by David Ben Zakai, Batuhan Büyükgüzel, Bert Heymans, (more)

Best way to remove duplicates from a list in python.

Convert it to a set (a set cannot have duplicates), and then back to a list!

```
1 a=[1,2,3,2,4,5,1,4]
2 a=list(set(a))
3 print a
```

Output:
[1,2,3,4,5]

Note: Use this only if you don't care about the order of the items in the list.

Written 11 Oct. 2,078 views.

Upvote 30 Downvote Comments 3+ Share

Lingliang Zhang, Student at NYUAD
79 upvotes by Jaimal Ichharam, Miles Lauridsen, Edwin Khoo, (more)

You can use a dictionary to store a switch, so that you can use it repeatedly and keep your code clean (it also prevents you having to have the overhead of defining the switch each time).

```
1 calculator = {
2 'plus': lambda x, y: x + y,
3 'minus': lambda x, y: x - y
4 }
5 calculator['minus'](9,3)
6 >> 6
```

Written 7 Aug, 2013. 4,667 views.

Upvote 79 Downvote Comments 4+ Share 1

Yam Mesicka, Programmer, native Hebrew speaker, se... (more)
41 upvotes by Abhiroop Dabral, Tina Wong, Mahmoud Emad, (more)

Well, I can rotate matrix with one line:

```
rotated = zip(*matrix[::-1])
```

and counterclockwise:

```
rotated = zip(*matrix)[::-1]
```

Updated 17 Dec, 2013. 2,586 views.

Upvote 41 Downvote Comment Share

Martin Thoma, CS and math student, living in Karlsr... (more)
20 upvotes by Paweł Kacprzak, Miguel Oliveira, Kamyar Ghasemlou, (more)

Unpacking and short notation for tuple building:

```
1 x, y = y, x
```

Unpacking of list arguments:

```
1 my_function(*alist)
```

Unpacking of dictionaries:

```
1 my_function(**adict)
```

More unpacking (supported in Python 3.X):

```
1 >>> a, *b, c = [1,2,3,4,5]
2 >>> print(a)
3 1
4 >>> print(b)
5 [2,3,4]
6 >>> print(c)
7 5
```

A fun one:

```
1 >>> from __future__ import braces
2   File "<stdin>", line 1
3 SyntaxError: not a chance
```

Written 9 Dec. 1,213 views.

Upvote 20 Downvote Comment 1 Share 1

Vladislav Zorov, Uploading HTML since 14.4-kbit-modem-... (more)
33 upvotes by Stephen McInerney, Jesús Gómez, Viggnesh Venugopal, (more)

Python 3 supports Unicode identifiers:

```
1 Python 3.2.3 (default, Apr 11 2012, 07:12:16) [MSC v.1500 64 bit (AMD6
2 Type "copyright", "credits" or "license()" for more information.
3 >>> import math
4 >>> π = math.pi
5 >>> π
6 3.141592653589793
7 >>> Σ = sum
8 >>> Σ(n for n in range(1, 101))
9 5050
10 >>>
```

Also IPython Notebook is a pretty neat "trick" - generally because it's what literate programming deserves to be, but especially because the editor is web-based and allows you to work from your tablet and still use the processing power and storage of a (super)computer :)

Written 16 Oct. 2,203 views.

Upvote 33 Downvote Comments 1+ Share

Sriram Madapusi Vasudevan, On a Circular path.
64 upvotes by Abhishesh Sharma, Victor Jolissaint, Lex Nederbragt, (more)

Context Managers!

```
1 with open("my_file.txt") as f:
2     print(f.readlines())
```

It will automatically close the file on exit, this can be used for creating resources which are automatically cleaned up after exiting scope.

I regard this one of the best features of python.

Updated 1 Sep, 2013. 5,652 views.

Sidharth Nadhan, Physical Design Engineer, Loves Crick... (more)
37 upvotes by Edwin Khoo, Jason Zhang, Makoto Watanabe, (more)

The **defaultdict** function is one that I commonly use :
Suppose you are given a list of words and you are asked to compute frequencies. You could do something like this:

```
frequencies = {}
for word in wordlist:     frequencies[word] += 1
```

Unfortunately, Python throws a KeyError the first time through, because you cannot increment the the values for the words, as they have never been initialized. A workaround would be to catch theKeyError exception:

```
for word in wordlist:
   try:        frequencies[word] += 1
   except KeyError:        frequencies[word] = 1
```

In Python 2.5 and later, though, the collections.defaultdict class comes to the rescue! A defaultdict is just like a regular Python dict, except that it supports an additional argument at initialization: a function. If someone attempts to access a key to which no value has been assigned, that function will be called (without arguments) and its return value is used as the default value for the key. Clever, right?

```
from collections import defaultdict
frequencies = defaultdict(int)
for word in wordlist:     frequencies[word] += 1
```

Actually, there is a more simple way to do this particular task, as mentioned in the comments.

```
from collections import Counter
frequencies = Counter(wordlist)
```

Using list as the initializing function, it is easy to group a sequence of key-value pairs into a dictionary of lists:

```
>>> s = [('yellow', 1), ('blue', 2), ('yellow', 3), ('blue', 4), ('red', 1)]
>>> d = defaultdict(list)
>>> for k, v in s: d[k].append(v)
>>> d.items()
[('blue', [2, 4]), ('red', [1]), ('yellow', [1, 3])]
```

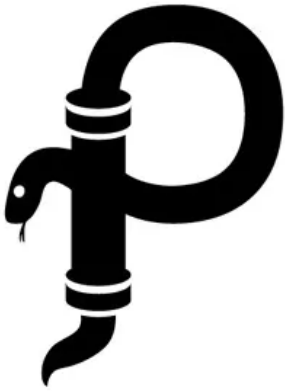
Some more cool funcionalities of the Collection module
Another cool package called xlwt for working with excel files

---

Python is not intended for using as one-liners from the command line like Perl, Awk or Sed. However we can use if we want. For eg. to print every line from an input file but remove the first two fields :

```
python -c "import sys;[sys.stdout.write(' '.join(line.split(' ')[2:])) for line in sys.stdin]" < input.txt
```

The Pyed Piper (Python Power at the Prompt) or pyp



Pyp is a linux command line text manipulation tool similar to awk or sed, but which uses standard python string and list methods as well as custom functions evolved to generate fast results in an intense production environment. Pyed Pyper was developed at Sony Pictures Imageworks to facilitate the construction of complex image manipulation "one-liner" commands during visual effects work on *Alice in Wonderland, Green Lantern,* and the *The Amazing Spiderman*.

Because pyp employs it's own internal piping syntax ("|") similar to unix pipes, complex operations can be proceduralized by feeding the output of one python command to the input of the next. This greatly simplifies the generation and troubleshooting of multistep operations without the use of temporary variables or nested parentheses. The variable "p" represents each line as a string, while "pp" is entire input as python list:
\> ls | pyp "p[0] | pp.sort() | p + ' first letter, sorted!'" #gives sorted list of first letters of every line
Pyp is not part of a default python installation though

Written 24 Oct, 2013. 3,719 views.

Upvote 37 Downvote Comments 3 Share

Simon Coulton
94 upvotes by Tracy Chou, Neel Hajare, Satoru Logic, (more)

```
@contextmanager
def ignored(*exceptions):
    """Provides the ability to not have to write try/catch blocks when
    passing on the except.

    Thanks to Raymond Hettinger from "Transforming Code into Beautiful
    This will be included in the standard library in 3.4.

    Usage:
        # instead of...
        try:
            do_something()
        except:
            pass

        # use this:
        with ignored(Exception):
            do_something()

```

```
    """
    try:
        yield
    except exceptions:
        pass
```

Written 3 Sep, 2013. 10,662 views.

Upvote 94 Downvote Comments 2 Share 1

Abhinav Saini, Frequent User
34 upvotes by Hiroshi Ono, Lucas Wiman, Jon Noronha, (more)

Launch an interactive shell to examine program state

```
import code; code.interact(local = locals())
```

Written 2 Dec, 2011. 3,449 views.

Upvote 34 Downvote Comments 2 Share

Lucas Wiman
21 upvotes by Nitin Madnani, Morimoto Tetsuya, Jennifer Lee, (more)

Notes from my python startup file

Copy and paste from clipboard (OS X only)

```
import subprocess
def pbpaste():
    p = subprocess.Popen(['pbpaste'], stdout=subprocess.PIPE)
    retcode = p.wait()
    data = p.stdout.read()
    return data

def pbcopy(data):
    p = subprocess.Popen(['pbcopy'], stdin=subprocess.PIPE)
    try:
        p.stdin.write(data)
    except TypeError:
        p.stdin.write(repr(data))
    p.stdin.close()
```

Unzip function:

```
def unzip(tuples):
        return zip(*tuples)
```

Decorator to print calls to decorated functions:

```
from decorator import decorator
@decorator
def trace(f, *args, **kw):
    print "calling %s with args %s, %s" % (f.func_name, args, kw)
    return f(*args, **kw)
```

Tab completion in the interpreter:

```
import readline
import rlcompleter
histfile = os.path.join(os.environ["HOME"], ".pyhist")
try:
    readline.read_history_file(histfile)
except IOError:
    pass
readline.parse_and_bind('tab: complete')
```

```
 9
10 atexit.register(readline.write_history_file, histfile)
11 del histfile
```

Also useful from the Python standard library: random.shuffle, collections.defaultdict, collections.namedtuple, itertools.count, itertools.product, itertools.everything_else_in_the_module

Written 7 Dec, 2011. 3,964 views.

Upvote 21 Downvote Comments 2 Share

Yogesh Pandey, my best answers are anonymous
62 upvotes by Ayelet Bitton, Seyram Komla Sapaty, Danyal Rehman, (more)

I often use this to shorten my tweets to 140

```
print raw_input("To SMS lingo").translate(None,'aeiou')
```

Updated 26 May, 2014. 3,157 views.

Upvote 62 Downvote Comments 3+ Share 1

Victor Jolissaint, tech & innovation enthusiast
17 upvotes by Christ Azika-Eros, Ankur Mittal, Abhishesh Sharma, (more)

Debugging a SMTP server running on localhost:

```
1 python -m smtpd -n -c DebuggingServer -d localhost:25
```

Implementing a switch-case statement:

```
1 def f(x):
2     return {
3         'foo': 1,
4         'bar': 2,
5     }.get(x, 3)    # 3 will be the default value
```

Updated 16 Jul, 2013. 1,964 views.

Upvote 17 Downvote Comments 1+ Share

Bhuvan Arumugam, openstack contributor
15 upvotes by Burak Bekdemir, Hiroshi Ono, Greg Price, (more)

Interactive shell in python with ipython.

It's a powerful interactive terminal, useful to understand methods, arguments and apidoc. It's very useful for all programmers to quickly get used to specific library. It has readline support wherein the method names are auto completed.

Consider this scenario. You are dealing with path. You like to know the full path for the given directory. You know the method reside in os.path library, but unsure about the exact method name. You may use ipython to figure out the right method name.

Import the library and use tab completion to list all available methods in the library, similar to how we do command completion in bash. Simple!

```
1 whitepearl:~ bhuvan$ ipython
2
3 In [1]: from os import path
4
5 In [2]: path.
6 path.abspath                    path.altsep                    path
```

```
path.devnull                    path.dirname                    path
path.genericpath                path.getatime                   path
path.isdir                      path.isfile                     path
path.normcase                   path.normpath                   path
path.relpath                    path.samefile                   path
path.splitdrive                 path.splitext                   path
path.warnings

In [2]: path.realpath?
Type:       function
Base Class: <type 'function'>
String Form:<function realpath at 0x10a331578>
Namespace:  Interactive
File:       /System/Library/Frameworks/Python.framework/Versions/2.7/l
Definition: path.realpath(filename)
Docstring:
Return the canonical path of the specified filename, eliminating any
symbolic links encountered in the path.

In [3]: path.realpath('.')
Out[3]: '/Users/bhuvan'
```

Written 24 May, 2012. 2,277 views.

Upvote 15 Downvote Comment Share 1

Nitin Madnani, Rabid Pythonista
76 upvotes by Ishan Mandhan, Rohit Dholakia, Baptiste Fontaine, (more)

```
Python 2.7.2 (default, Sep  4 2011, 00:16:50)
[GCC 4.2.1 (Based on Apple Inc. build 5658) (LLVM build 2335.15.00)] on
Type "help", "copyright", "credits" or "license" for more information.
>>> from __future__ import braces
  File "<stdin>", line 1
SyntaxError: not a chance
>>>
```

Written 1 Dec, 2011. 4,644 views.

Upvote 76 Downvote Comment Share

Baptiste Fontaine, CS Student
9 upvotes by Joachim Krügermeister, Lucas Wiman, Jakub Zarzycki, (more)

```
>>> my_list = list("python") # ['p', 'y', 't', 'h', 'o', 'n']
>>> a, *b = my_list #a: head, b: tail (only with Python3)
>>> a
'p'
>>> b
['y', 't', 'h', 'o', 'n']
>>> my_list[::-1] # reverse
['n', 'o', 'h', 't', 'y', 'p']
```

Updated 21 Sep, 2013. 2,567 views.

Upvote 9 Downvote Comments 2+ Share

Vivek Nagarajan, Programmer for 25 years
29 upvotes by Vladislav Zorov, Saurav Verma, Matt Richardson, (more)

Argument unpacking :

```
def test(arg1, arg2):
  print arg1, arg2

def fn(dct):
```

```
5   test(**dct)
6
7 fn({'arg1': 'Hell', 'arg2': 'World'})
```

This lets you pass a dictionary as named arguments to a function.

It may not look terribly useful, but when you get a dictionary of values - for example from HTTP POST/GET requests - you can use this idiom very effectively.

Moreover the runtime will check that the names of the arguments match the keys of the dictionary and throw a TypeError automatically if they don't - saves a lot of manual checking.

Written 19 Nov. 1,861 views.

Upvote 29 Downvote Comment Share

Deepanshu Mehndiratta, Code, Beer, BITS, Goa, Apple, Floyd, ... (more)
29 upvotes by Christophe Grosjean, Victor Jolissaint, Viswa Murali, (more)

Get a method and its documentation for a particular use case without remembering its name or even the module it belongs to.

```
1 # Get a list of all available modules
2 help('modules')
3 # Import a module from the list of modules
4 import <name of module>
5 # Get the available class/functions of the module
6 dir(<name of module>)
7 # Get the documentation of the class/method of the module
8 help(<name of module>.<class/method in module>)
```

Updated 8 Mar, 2014. 3,647 views.

Upvote 29 Downvote Comment 1 Share 1

Sean Wang, S.W. Test Dev, especially mobile at p... (more)
30 upvotes by Tolga Yilmaz, Sherif Morris, Christophe Grosjean, (more)

Insert two or more elements to an existing list:

```
1 >>> a= [1,2,3,4,7]
2 >>> a[4:4] = [5,6]
3 >>> print a
4 [1,2,3,4,5,6,7]
```

Written 17 Jun, 2014. 1,623 views.

Upvote 30 Downvote Comments 1+ Share

Vineet Naik, I like using Python for writing CLIs
15 upvotes by Gabriel H Pugliese, Seyram Komla Sapaty, Julie Prentice, (more)

Don't know if it qualifies as a Python trick, but I use json.tool all the time to pretty-print json in the shell

```
1 $ echo '{"greeting": "hello", "name": "world"}' | python -m json.tool
```

Written 25 Nov, 2012. 1,530 views.

Upvote 15 Downvote Comment 1 Share

Gabriel H Pugliese, Pythonista
33 upvotes by Hiroshi Ono, Dan Loewenherz, Donald Huang, (more)

Circular lists :)

```
1 >>> def make_circular_list(a_list):
```

```
...   while True:
...     for an_item in a_list:
...       yield an_item
...
>>> a_list = [1,2,3]
>>> a_circular_list =  make_circular_list(a_list)
>>> a_circular_list.next()
1
>>> a_circular_list.next()
2
>>> a_circular_list.next()
3
>>> a_circular_list.next()
1
>>>
```

Written 22 Jul, 2012. 2,623 views.

Upvote 33 Downvote Comments 3+ Share 1

Ron Reiter, Programmer, Co-founder of a young sta... (more)
42 upvotes by Tigran Sloyan, Jared Morrow, Alistair Bong, (more)

I've created a website just for cool Python tricks.
http://facts.learnpython.org

Written 6 Dec, 2011. 3,799 views. Suggestions Pending

Upvote 42 Downvote Comments 4+ Share 1

Denzil Correa, Computer Scientist
24 upvotes by Franck Dernoncourt, Bulat Bochkariov, Seyram Komla Sapaty, (more)

I love how easy `itertools` makes it for algebra.

```
>>> import itertools
>>> print list(itertools.product('ABCD', repeat=2))
[('A', 'A'), ('A', 'B'), ('A', 'C'), ('A', 'D'), ('B', 'A'), ('B', 'B'
>>> print list(itertools.permutations('ABCD', 2))
[('A', 'B'), ('A', 'C'), ('A', 'D'), ('B', 'A'), ('B', 'C'), ('B', 'D'
>>> print list(itertools.combinations('ABCD', 2))
[('A', 'B'), ('A', 'C'), ('A', 'D'), ('B', 'C'), ('B', 'D'), ('C', 'D'
>>> print list(itertools.combinations_with_replacement('ABCD', 2))
[('A', 'A'), ('A', 'B'), ('A', 'C'), ('A', 'D'), ('B', 'B'), ('B', 'C'
>>> print list(itertools.izip('ABCD', 'xy'))
[('A', 'x'), ('B', 'y')]
```

Find the odd numbers in a list

```
>>> print list(itertools.ifilter(lambda x: x%2, range(10)))
[1, 3, 5, 7, 9]
```

Written 10 Nov, 2012. 2,038 views.

Upvote 24 Downvote Comments 1+ Share

Jun-Geun Park
56 upvotes by Yair Livne, John Clover, Nikhil Garg, (more)

Finding the most frequent element in a list, x:

```
max(set(x), key=x.count)
```

Written 29 Oct, 2013. 3,452 views.

Upvote 56 Downvote Comments 3+ Share

Ryan Rembert, Polypassionate geoscientist, software... (more)
18 upvotes by Viggnesh Venugopal, Stefan Rieger, Tuan-Anh Hoang-Vu, (more)

This is a really useful idiom to set arbitrary breakpoints in your code. Set a shortcut for it and go crazy - a step up (or just different) from plastering your code with print statements.

```
1 import ipdb; ipdb.set_trace()
```

When this blurb gets hit, it will exit to the interpreter and allow you to check the state of your code.

Written 16 Sep, 2014. 931 views.

Upvote 18 Downvote Comments 2 Share

Jim Pivarski
33 upvotes by Richard West, Tim Burgess, Brendan Shillingford, (more)

Problem: you're calling a function that returns a list of results when you know that there's only one result. It will be a singleton list, but you don't want to bother with that--- you just want the value of the one item.

Pedestrian solution: put a [0] at the end of the expression:

```
1 result = getResults()[0]
```

Slick solution: put a comma (,) at the end of the variable:

```
1 result, = getResults()
```

Why does this work? The trailing comma turns the variable into a single-element tuple. When you assign to a tuple, Python unpacks whatever is on the right-hand side into each element of the tuple. (You've probably seen

```
1 x, y = first(), second()
```

or something like that.) In this case, it unpacks the only item. If the function actually returns 0 or more than 1 result, you'll get a ValueError ("need more than 0 values to unpack" or "too many values to unpack"). If you had used the pedestrian method with the square brackets, you'd get an IndexError.

I find that I need to unpack things quite often, and it's good to have little idioms like this to remind yourself of your assumptions. (In this case, the comma indicates that you expect exactly one result; a [0] would suggest that you expect at least one result.)

Written 22 May, 2014. 2,243 views.

Upvote 33 Downvote Comments 2+ Share

Seyram Komla Sapaty, from __future__ import braces
30 upvotes by Swetha Sampath, Vladimir Krestyannikov, Harit Himanshu, (more)

```
1 # python 2.x only
2 print  'ernqnovyvgl pbhagf'.decode('rot13')
```

Use of a single underscore in IDLE.
An underscore represents the result of a previous expression

```
1 >>> 1 + 2
2 3
3 >>> _ + 1
4 4
```

Updated 20 Jul, 2012. 1,717 views.

Upvote 30 Downvote Comments 2 Share 1

R. Drew Davis
7 upvotes by Michael Herman, Ksaver Frikov, Lee Hanxue, (more)

Python Generators are pretty neat and not generally found in other programming languages. You can chain multiple generators together to program in a style reminiscent of shell command line pipelines. But where my mind was really blown was when it was shown how useful it can be to generate a sequence of dictionary objects.

Updated 22 Aug, 2014. 740 views.

Upvote 7 Downvote Comment Share

Pradipta Bora, Loving Python since Day One. Created ... (more)
38 upvotes by Seyram Komla Sapaty, Vladislav Zorov, Mark Wong, (more)

1. Decorators

Decorators allow to wrap a function or method in another function that can add functionality, modify arguments or results, etc. You write decorators one line above the function definition, beginning with an "at" sign (@). Decorators are very powerful and allows us to modify functions as and when needed.

For example, you can use a Function Metadata decorator:

```
def print_metadata(f):
    def r(*args,**kwargs):
        print (f.__name__ , 'function ')
        print ('Arguments: ', args, kwargs)
        if not f.__doc__ is None:
            print ('Documentation: ', f.__doc__)
        return f(*args, **kwargs)
    return r

@print_metadata
def print_two(A,b):
    """
    Print  A and b
    """
    print (A, b)

print_two("Hello", "Python")
```

2. Sending Values into Generators

This is a very cool generator method that allows us to send a value to the generator using the send method.

For example,

```
def mygen():
    """Yield 5 until something else is passed back via send()"""
    a = 5
    while True:
        f = (yield a) #yield a and possibly get f in return
        if f is not None:
            a = f
```

And then you can use the Generator:

```
>>>s =  mygen()
>>>s.next()
```

```
3 5
4 >>>s.send(19)
5 19
6 >>>s.next()
7 19 #19 forever until you have sent anything else
```

3. __future__ module

Future module allows you to effectively use Python 3 language features in Python 2.6 and 2.7.

Things like Unicode Literals, Floor Division with //, Print Function etc can be easily used in Python 2.7 and 2.6.

Just add the following to the top of your script:

```
1 from __future__ import division, unicode_literals, print_function
```

Full Docs here: 28.11. __future__ - Future statement definitions - Python 2.7.9 documentation

And __future__ can be used with Python 2.4 and 2.5 too by installing from PyPI: Python Package Index

4. Readable Regular Expressions

Regular Expressions are very difficult to comprehend due to their syntax and because of the fact that they are mostly compressed together. Complex Regular Expressions can be difficult to read for the programmer himself. Python supports comments in Regular Expressions that can be safely ignored.

Example:

```
1 pattern = """
2 ^                   # beginning of string
3 M{0,4}              # thousands - 0 to 4 M's
4 (CM|CD|D?C{0,3})    # hundreds - 900 (CM), 400 (CD), 0-300 (0 to 3 C's
5                                  #            or 500-800 (D, fo
6 (XC|XL|L?X{0,3})    # tens - 90 (XC), 40 (XL), 0-30 (0 to 3 X's),
7                     #        or 50-80 (L, followed by 0 to 3 X's)
8 (IX|IV|V?I{0,3})    # ones - 9 (IX), 4 (IV), 0-3 (0 to 3 I's),
9                     #        or 5-8 (V, followed by 0 to 3 I's)
10 $                  # end of string
11 """
```

5. Variable Swapping

This is a very powerful feature that reduces lots of complexity. They can be used for swapping List elements as well. This can be very useful in Bubble Sorting a list.

```
1 for index in range(len(array)-1):
2     if array[index] > array[index+1]:
3         array[index], array[index+1] =  array[index+1], array[index]
```

6. Properties

Remember getters and setters? Functions that are used for setting and getting private variables in other languages. Although, Private Variables are unavailable in Python, getters and setters can still be used. For example, we can have a temperature attribute of a Kelvin to Celsius Class.

Now temperature below 0 does not exist in Kelvin so the only way to prevent that is by using a setter (which calls a check for the passed argument).

But Python has a more beautiful way to do that. Properties

```
class kelvin_to_celsius(object):
    def __init__(self, temp):
        self.temperature = temp
    @property
    def temperature(self):
        return self._temperature
    @temperature.setter
    def temperature(self, value):
        if value>=0:
            self._temperature = value
        else:
            raise ValueError("Temperature in Kelvin cannot be below 0"

obj = kelvin_to_celsius(-1) #This will raise an Error
```

Written Wed. 2,106 views.

Upvote 38 Downvote Comment Share 1

Rahul Choudhury
15 upvotes by Tuan-Anh Hoang-Vu, Vladislav Zorov, Saurav Verma, (more)

Using default dictionaries to represent simple trees:

```
import json
tree = lambda: collections.defaultdict(tree)
root = tree()
root['menu']['id'] = 'file'
root['menu']['value'] = 'File'
root['menu']['menuitems']['new']['value'] = 'New'
root['menu']['menuitems']['new']['onclick'] = 'new();'
root['menu']['menuitems']['open']['value'] = 'Open'
root['menu']['menuitems']['open']['onclick'] = 'open();'
root['menu']['menuitems']['close']['value'] = 'Close'
root['menu']['menuitems']['close']['onclick'] = 'close();'
print json.dumps(root, sort_keys=True, indent=4, separators=(',', ': '))
{
   "menu": {
      "id": "file",
      "menuitems": {
         "close": {
            "onclick": "close();",
            "value": "Close"
         },
         "new": {
            "onclick": "new();",
            "value": "New"
         },
         "open": {
            "onclick": "open();",
            "value": "Open"
         }
      },
      "value": "File"
   }
}
```

Written 24 Nov. 587 views.

Upvote 15 Downvote Comment Share

Koushik Ramachandra, Curious.
13 upvotes by Jon Barker, Colin Lin, Edwin Khoo, (more)

Creating a new type dynamically; my personal favorite. Awesome for code injections. I discovered this when trying to generate dynamic forms using Django.

```
class Custom(object):
        var = "custom"

c = Custom()
```

This is equivalent to

```
c = type("Custom", (object,), {'var':'custom'})()
```

Written 18 Jan, 2014. 1,448 views.

Upvote 13 Downvote Comment Share 1

Agu Gabriel
12 upvotes by Mohitdeep Singh, Vlad-Doru Ion, Makoto Watanabe, (more)

Positional and Keyword arguments(* and ** in function signatures):

```
def daily_sales_total(*all_sales):
    total = 0.0
    for each_sale in all_sales:
        total += float(each_sale)
    return total
```

```
daily_sales_total()
daily_sales_total(10.00)
daily_sales_total(5.00, 1.50, '128.75') # Any type allowed, not just fl
```

Written 6 Nov, 2013. 1,320 views.

Upvote 12 Downvote Comment 1 Share

Jeff Cumpston
18 upvotes by Pranav Angara, Ryan Fox Squire, Thomas Aquinas, (more)

Alternately apply two functions in an iterative algorithm:

```
funcs = [func1, func2]
resultlist=[]

for i in range(idxmax):
     nowfunc = funcs[i % 2]
     resultlist.append(nowfunc(args))
```

EDIT (something new)
Store data in different variables with each iteration:

```
For idx in range(idxmax):
    exec "var%i = func(idx)" %idx
```

Updated 30 Dec. 610 views.

Upvote 18 Downvote Comments 2 Share

Gopal Adhikari
4 upvotes by Bala Krishnan, Chuan Lu, Agu Gabriel, (more)

To host a pydoc server locally:

```
pydoc -p 8080
```

The docs will be accessible at http://localhost:8080

Written 22 Jul, 2013. 659 views.

Upvote 4 Downvote Comment Share

Paweł Kacprzak, chasethered.com - competitive program... (more)
23 upvotes by Kevin Lin, Sandip Chatterjee, Viswa Murali, (more)

The **get()** method of a dict. The signature is:

```
mydict.get(key, default=None)
```

and it returns mydict[key] if there is a value associated with this key, otherwise it returns default, so we can many things simpler, for example:

```
1 mydict = {}
2 for e in mylist:
3     mydict[e] = mydict.get(e, 0) + 1
```

instead of:

```
1 mydict = {}
2 for e in mylist:
3     if not e in mydict:
4         mydict[e] = 0
5     mydict[e] = mydict[e] + 1
```

Written 20 Oct. 874 views.

Upvote 23 Downvote Comments 1+ Share

Mahdi Yusuf, curator @pycoder and screencaster @ne... (more)
24 upvotes by Bramie Jim, Edwin Khoo, Pete Hunt, (more)

If you don't like using whitespace to denote scopes, you can use the C-style {} by issuing:

```
from __future__ import braces
```

Neckbeard Republic - bite-sized python screencasts

Written 27 Mar, 2013. 1,543 views.

Upvote 24 Downvote Comments 7+ Share

Grzegorz Gwardys, Interested in machine learning, compu... (more)
12 upvotes by Seyram Komla Sapaty, Abhay Bothra, Julie Prentice, (more)

I'm not sure, whether these things are tricks - you would down-vote me in the worst case ;-)

adding to *.py script **#!/usr/bin/env python** line, make it executable and run by: ./script.py

generating HTML documentation: pydoc -w ./script.py

else statement after for/while loops (go into, if there was no break)

What I have ?
pip freeze
Is there something ?
pip search something
Do I need something ?

pip install something

We can not:

```
def yetanotherfunc():
    yaf_int = 1
    def childfunc():
        global yaf_int
        yaf_int = 3
```

but we can:

```
def yetanotherfunc():
    yaf_int = [1]
    def childfunc():
        yaf_int[0] = 3
```

deep copy:

import copy
sth_deepcopy = copy.deepcopy(sth)

Why do we have "private" attributes in Python:
http://stackoverflow.com/questio...

for in protocole:
Understanding Python's "for" statement

Generators can be use as alternative to callbacks - for example yields if there some big data has arrived.

Tools:
virualenv
pdb debugger (pdb.set_trace() feature)
pylint for code analysis

Written 11 Nov, 2012. 840 views.

Upvote 12 Downvote Comments 1+ Share 2

Sayed Atif Ali
71 upvotes by Mohitdeep Singh, Adriano Sá Nascimento, Adil Mujeeb, (more)

I'm a bit amazed that this hasn't come up here yet. Stack Overflow is a repository of such tips and tricks, you just need to dig a bit deeper. Here are some of the most comprehensive list of lesser known but useful features of python:

Hidden features of Python

Similar lists also exist for:

Hidden features of C
Hidden Features of C++?
tips and tricks - Hidden Features of C#? - Stack Overflow
Hidden Features of Java
Hidden Features of JavaScript?
Hidden features of HTML
Hidden features of CSS
Hidden Features of PHP?
Hidden Features of VB.NET?
.net - Hidden Features of ASP.NET - Stack Overflow
Hidden features of Ruby
Hidden Features of Ruby on Rails

Hidden features of Perl?

You're most welcome! :)

Written 1 May, 2013. 4,858 views.

Upvote 71 Downvote Comment Share 5

Mario Hari, A Natural Language Processor
9 upvotes by Michel Poisson, Miguel Oliveira, Ryan Fox Squire, (more)

Here is the collection of must read subtleties compiled by Peter Norvig,

Infrequently Answered Questions

Written 1 Feb. 517 views.

Upvote 9 Downvote Comment 1 Share

Mustafa Zengin, I write computer programs
83 upvotes by Seyram Komla Sapaty, Gary Prahandono, Mahmoud Emad, (more)

Here is the hack for the ternary operator of C-like languages:

To simulate the behavior of

```
1 int greater = a > b ? a : b;
```

you can write in Python:

```
1 greater = [b, a][a>b]
```

So how does this work?
In the Python code, the comparison "(a > b)" is converted to index. If "a" is greater than "b" then "(a > b)" becomes "True" and because it is used as an index, it is converted to 1. So, it will retrieve "a" from the list and assign it to "greater" since a has the index 1. Otherwise, "b" will be retrieved from the list and assigned to "greater" by making the comparison result "False", and converting it to "0".

Written 16 May, 2013. 2,696 views.

Upvote 83 Downvote Comments 4+ Share

Pranav Angara, Stack-to-Stack Engineer | Computer Sc... (more)
18 upvotes by Tuan-Anh Hoang-Vu, Prashanth Ramanathan, Mani Cavalieri, (more)

A fast way to compute large fibonacci numbers using generators:

```
1 def fibonacci_generator():
2     a,b = 0,1
3     while True:
4           yield a
5            a,b = b, a+b
6 f = fibonacci_generator()
7 fib_nums = [f.next() for i in xrange(300)]
```

Updated 25 Nov. 830 views.

Upvote 18 Downvote Comments 1+ Share

AJ Sethi, by chance
4 upvotes by Nate Ferrero, Peter Strömberg, Sadjad Fouladi, (more)

If its about language and the system then "virtualenv" is my favorite.

Written 18 May, 2012. 1,049 views.

Upvote 4 Downvote Comment Share

Dipanjan Nag
14 upvotes by Gerardo Andres Riaño Briceño, Vladislav Zorov, Reed Oei, (more)

This is the coolest thing I found some days ago.
Say I've a list: `['a','b','b','c','a','a']`
Now I want to count how may a,b,c is there?
It's simple:

```
1 >>> from collections import Counter
2 >>> Counter(['a','b','b','c','a','a'])
```

The output will be:

```
Counter({'a': 3, 'b': 2, 'c': 1})
```

If you feel this is cool here is the details http://pymotw.com/2/collections/...

Now, one more evergreen thing:

```
1 >>> a = 'abcd'
2 >>> a[::-1]
```

The output is: `'dcba'`

Written Thu. 677 views.

Upvote 14 Downvote Comments 1+ Share 2

Suresh Alse, Well..I'm not just a geek!
22 upvotes by Fabian Okeke, Adwait Sharma, Balaji Kulkarni, (more)

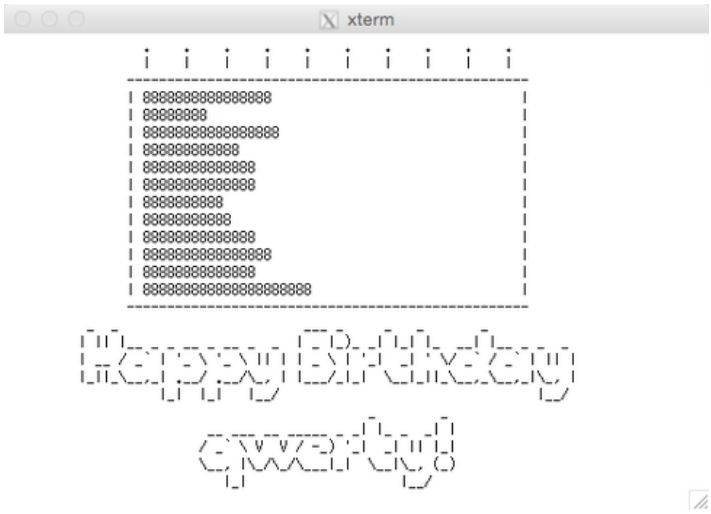
A geek way to wish Happy Birthday!

There is nothing more awesome than doing regular things in a geeky way. Few days back I was wondering what would be the geekiest way to wish someone happy birthday. It should be geeky alright, but should also have regular things like **cake + candles + birthday song + wishing happy birthday**.

Now mixing all this with my very little knowledge in signal processing, I wrote a python code which does this:

A virtual cake on **terminal** who's shade has an **equalizer effect** corresponding to the **"happy birthday"** song that is played in the background. Here, the cake has candles with flames fluttering randomly. Also, we have a fancy display of happy birthday message.

Snap of the virtual cake:



How it works?

Read my blog post: A geek way to wish Happy Birthday! .
Source on Github: alseambusher/HappyBirthday-term

Here is the python code:

```
import scipy.io.wavfile as wavfile
import numpy as np
import pylab as pl
import time
import os
import sys
import subprocess
from scipy import mean
from random import randint

# music wav file
FILE = "karaoke.wav"
rate, data = wavfile.read(FILE)
t_total = len(data[:,0])/rate
display_rate = 1500 #number of frames processed in one iteration
sample_size = 120
max_display = 90
data_length  = len(data) #total number of frames
_min = min([abs(x) for x in data[:,0]]) #max amplitude in the wav
_max = max([abs(x) for x in data[:,0]]) #min amplitude in the wav

# IMPORTANT: correction factor. Change this value to match the song wi
correction = 0.645

# cake settings
cols = int(subprocess.Popen("tput cols",shell=True,stdout=subprocess.P
display_char = "8"
cake_size = 50

# flame control
flame_flutter_rate = 50
FLAMES = [ " . ", ".  ", "  ." ]
current_flame = ""

os.system("tput civis") #hide cursor

# TODO open file with some player. If you are on mac, uncomment follow
#os.system("open "+FILE)

```

```
for _f in range(data_length/display_rate):

        # fluttering effect to candle flames
        if _f%flame_flutter_rate == 0:
                current_flame = (" "*(cols/2 - cake_size/2))+((" "+FLA
        print current_flame

        # candles
        print (" "*(cols/2 - cake_size/2))+("  |  "*(cake_size/5))
        # cake top layer
        print (" "*(cols/2 - cake_size/2))+("-"*cake_size)

        bucket = []
        mug = []
        # mug contains the current frame samples (absolute values) of
        # average of mugs are put into bucket
        for value in data[:,0][_f*display_rate+1:(_f+1)*display_rate]:
                mug.append(abs(value))
                if len(mug) == sample_size:
                        bucket.append(mean(mug))
                        mug = []
        bucket = [ (float)((x - _min) * max_display)/(_max - _min) for

        # print the equalizer from the bucket
        for value in bucket:
                print (" "*(cols/2 - cake_size/2))+"| "+("8"*(value%(c

        # bottom crust of the cake
        print (" "*(cols/2 - cake_size/2))+("-"*cake_size)

        # print happy birthday message
        os.system("figlet -c -f small Happy Birthday qwerty!")

        # sleep to match with the audio
        """
                NOTE: correction has to be multiplied to sleep time
                This is because of several factors like time taken to
                CHANGE THE VALUE OF correction TO FIT YOUR NEED
        """
        time.sleep(((float)(display_rate * t_total) / data_length)*cor

        # clear screen
        if _f != data_length/display_rate-1:
                os.system("clear")

raw_input()
```

Written Thu. 380 views.

Upvote 22 Downvote Comments 1+ Share

Vishnu Ashok, Py Dev. Qurious Netizen.
24 upvotes by Leandro Toshiyuki Kinoshita, Cyril Anderson, Pranav Angara, (more)

Easy way to reverse a string :
print str[::-1]

eg :
>> str = 'hello world'
>>print str[::-1]
dlrow olleh

Written 20 Feb. 649 views.

Upvote 24 Downvote Comment 1 Share 8

Adwait Sharma
16 upvotes by Eliot Ball, Makoto Watanabe, Sam Bodanis, (more)

Running a simple HTTP server in current directory:

```
python -m SimpleHTTPServer 8000
```

Written 6 Jun, 2014. 777 views.

Upvote 16 Downvote Comment Share

Thomas J Garcia
5 upvotes by Sean Clarke, Cyril Anderson, Adarsh Jois, (more)

This is numpy specific but python wouldn't be half as useful to me without numpy.

Stride tricks in numpy!

Generating a rolling window over a 1D array (can extend this to n-dimensional array) to compute sliding statistics:

```
1 def rolling_window(a, window):
2   """Directly taken from Erik Rigtorp's post to numpy-discussion.     <
3   shape = a.shape[:-1] + (a.shape[-1] - window + 1, window)
4   strides = a.strides + (a.strides[-1],)
5   return np.lib.stride_tricks.as_strided(a, shape=shape, strides=stride
```

Here's a portion of code for decoding the forward error correction encoding in a GPS NAV message that has been stored as an array of 32 bit integers. I bring it up because I think it demonstrates the utility of strides/fancy indexing to simplify more complex array manipulations.

The NAV message is made up of several subframes containing 300 bits for each subframe made up of 10 words of length 30 each. This is the code for decoding a single subframe:

```
1 from itertools import product # product as in cross product
2 import numpy as np
3 from numpy.lib.stride_tricks import as_strided
4
5 for i, subframe in enumerate(nav_frames) ...
6 # Decode each word of the subframe based on section 20.3.5.2 of IS-GPS-
7
8 framebits= [0x1 & (w >> (29-k)) for w,k in product(subframe,xrange(30))
9 framebits = np.array(framebits, dtype=np.uint8)
```

This first part converts 10 integer words of length 30 into a flat array of binary digits. It is essentially a double for loop where w is the integer word and k is the bit position index.

```
1 datawords = as_strided(framebits, shape=(10,24), strides=(30,1))
```

Grab the data/non-parity bits (the first 24 bits) for each word as a memory view. (a memory view is just a reference to the underlying data, no copies are performed).

```
1 datawords[1:] ^= framebits[29::30].reshape(10,1)[:-1]
```

XOR the last 9 words of the subframe with the 30th bit of the previous word for the first 9 words of the subframe. Notice the double index in 29::30 - this grabs the 30th bit of each word. The :-1 index excludes the 30th bit of the last word from the operation.

Written 7 Dec. 492 views.

Upvote 5 Downvote Comment Share

Angus Hollands
3 upvotes by Christophe Grosjean, Yuan Jian, and Gilles Castel.

Descriptors, how have we failed to mention those every other comment!? And, metaclasses. Metamagic for everyone!

Also, in the rare occasions when you're trying to debug an application which cannot be easily tested (I.E A game framework) and you want to use the output of the terminal to find an object, use "_ctypes.PyObj_FromPtr(id_)" to find an object by its id().

This is only really useful if you're using a print() statement to print out lots of ids and you manually inspect the console readout to determine which one you need. Or you could just store everything in an indexed global variable.

Written 13 Aug, 2014. 818 views.

Upvote 3 Downvote Comment Share

Rajat Khandelwal, CSE@IITD, 2009-2013
6 upvotes by Jacob Oscarson, Mathew Kellogg, Bayo Opadeyi, (more)

One word for you: 'itertools'

Written 18 May, 2012. 1,062 views.

Upvote 6 Downvote Comment Share

Rohan Agrawal
4 upvotes by Ashish Shavarna, Tushar Suresh, Hardik Akbari, (more)

The gift of time travel

```
import __future__
```

Written 23 Jul, 2012. 582 views.

Upvote 4 Downvote Comment Share

Ashish Kashyap
10 upvotes by Mohitdeep Singh, Yongliang Wang, Abhiroop Dabral, (more)

```
>>> import itertools
>>> print list(itertools.product('ABCD', repeat=2))
[('A', 'A'), ('A', 'B'), ('A', 'C'), ('A', 'D'), ('B', 'A'), ('B', 'B'), ('B', 'C'), ('B', 'D'), ('C', 'A'), ('C', 'B'), ('C', 'C'), ('C', 'D'), ('D', 'A'), ('D', 'B'), ('D', 'C'), ('D', 'D')]
>>> print list(itertools.permutations('ABCD', 2))
[('A', 'B'), ('A', 'C'), ('A', 'D'), ('B', 'A'), ('B', 'C'), ('B', 'D'), ('C', 'A'), ('C', 'B'), ('C', 'D'), ('D', 'A'), ('D', 'B'), ('D', 'C')]
>>> print list(itertools.combinations('ABCD', 2))
[('A', 'B'), ('A', 'C'), ('A', 'D'), ('B', 'C'), ('B', 'D'), ('C', 'D')]
>>> print list(itertools.combinations_with_replacement('ABCD', 2))
[('A', 'A'), ('A', 'B'), ('A', 'C'), ('A', 'D'), ('B', 'B'), ('B', 'C'), ('B', 'D'), ('C', 'C'), ('C', 'D'), ('D', 'D')]
>>> print list(itertools.izip('ABCD', 'xy'))
[('A', 'x'), ('B', 'y')]
```

Written 14 Apr, 2013. 1,079 views.

Upvote 10 Downvote Comment Share

Paul Sawaya, New age fun with a vintage feel!
4 upvotes by Ralph Ning, Krzysztof Kozielczyk, Sumit Bisht, (more)

Simple HTTP Server (just serve files on a given port, no CGI).

```
python -m SimpleHTTPServer 5000
```

Map a tuple or list to function arguments.

```
1 def foo(a,b):
2     print "a=%s b=%s" % (a,b)
3 q = [1,2]
4 foo(*q)
```

Written 1 Dec, 2011. 1,757 views.

Upvote 4 Downvote Comment Share

Joseph Freemind, Software developer
12 upvotes by Denzil Correa, Brent Fitzgerald, Yuhao Huang, (more)

Reversing a string

```
1 hello = "Hello world"
2 print hello[::-1]
```

Updated 29 Sep, 2012. 700 views.

Upvote 12 Downvote Comment Share 1

Yves Dorfsman
7 upvotes by Greg Price, Sai Teja Pratap, Behzad Nouri, (more)

python -i

If you are going to need more than 2 or 3 lines, do **not** use the interpreter. Instead, write it into a file, then run

> python -i *filename*

This way, python enters the interactive mode after executing the script. If you want to try it, write a small file where you define a long list, run python -i against it, then, when inside the interpreter, type the name of the list.

Written 7 Dec, 2011. 1,435 views.

Upvote 7 Downvote Comment 1 Share

Akash Raj, Traveller
9 upvotes by Makoto Watanabe, Ksaver Frikov, Jesús Gómez, (more)

what if i tell you that
1. To get the last character of a string 'A', simply type

```
A[-1]
```

2. To clear a python list 'A' type-

```
A[:] = []
```

3. You can create functions without a name

```
(lambda x: x*2)
```

4.Get all possible permutations of string in 2 line code(#import itertools)

```
for p in itertools.permutations([1, 2]):
    print ''.join(str(x) for x in p)
```

Written 17 Jun, 2014. 616 views.

Upvote 9 Downvote Comments 1+ Share

Ashwini Chaudhary, http://stackoverflow.com/users/846892
17 upvotes by Nico Ekkart, Mani Cavalieri, Chen Liu, (more)

In Python3 **print()** is a function, and you can take the advantage of argument unpacking.

So, this code:

```
>>> lis = range(10)
>>> print (' '.join([str(x) for x in lis]))
0 1 2 3 4 5 6 7 8 9
```

can be written as:

```
>>> print (*lis, sep=' ')
0 1 2 3 4 5 6 7 8 9
```

To use **print()** as a function in Python2 you need to import it first:

```
from __future__ import print_function
```

True and **False** are subclass of **int** class in Python, and they can be used interchangeably except when it comes to their string representation. **True == 1** and **False == 0**.

So, expressions like:

```
>>> lis = [1, 4, 6, 1, 8, 9, 1]
>>> sum(1 for x in lis if x%2)
4
```

can be rewritten directly as generator expressions:

```
>>> sum(x%2 for x in lis)
4
```

But IMO the first one is a bit more explicit so meets the **explicit is better than implicit** point of **Zen of Python**.

And this one is not a trick per se but a general info about **str.join** that many people don't know of. **str.join** always works better when passed an iterable like **list**, **tuple** or **collections.deque** etc. And beats a generator expression and an iterator in terms of both memory efficiency and speed as two iterations are required when an iterator/generator expression is used. (Raymond Hettinger's answer on SO: Page on Stackoverflow)

```
>>> %timeit ' '.join(str(x) for x in xrange(10**5))
10 loops, best of 3: 84.2 ms per loop
>>> %timeit ' '.join([str(x) for x in xrange(10**5)])
10 loops, best of 3: 74.4 ms per loop
```

Updated 23 Feb, 2014. 1,356 views.

Upvote 17 Downvote Comment 1 Share

Roshan Sam
4 upvotes by David Dérus, Viggnesh Venugopal, Oszkár Józsa, (more)

You are looking at the wrong place. Just browse stackoverflow and you'll be staggered at the versatility of Python.

How do you check if every element in a long list has the same value?
>>> if len(set(longList)) == 1: #VOILA!!

If you have a list of strings of various lengths, how would you pad them all (with '#') so that all have the same length, i.e. the length of the largest string?

In one line?
>>> [x + '#'*(max([len(y) for y in stringList]) - len(x)) for x in stringList]
#MAGIC OF LIST COMPREHENSIONS

If you want to make multiple comparisons in one if statement?
Say you want to check if x is within a specific bunch of bounds, [(1,3), (4,7), (9,12)].
>>> bList = [(1,3),(4,7),(9,12)]
>>> x = 11
>>> y = 8
>>> any([a<x<b for a,b in bList]) #NOT x>a and x<b JUST a<x<b
True
>>> any([a<y<b for a,b in bList])
False

And I haven't even gotten to the functions map, filter and reduce yet!

Python may not be an outright performer like C++, but it sure makes programming deliriously enjoyable. :)

Written 26 Jul, 2014. 485 views.

Upvote 4 Downvote Comments 2+ Share

Aditya Sarawgi
8 upvotes by John Clover, Baptiste Fontaine, Ben Mordecai, (more)

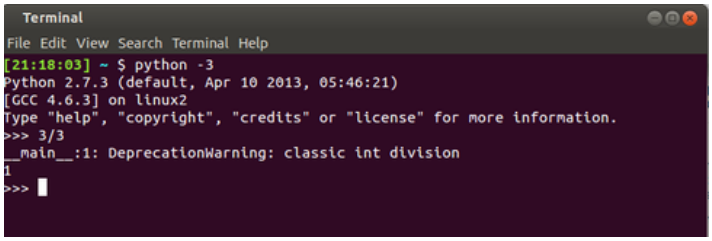
[0] * k gives you an zeroed list of length k

Written 8 Oct, 2012. 756 views.

Upvote 8 Downvote Comments 3+ Share 1

Dhiraj Thakur, a day without python is a day wasted!
5 upvotes by Nelson Jacobsen, Charles Wu, Viola Yee, (more)

Just stumbled upon a cool piece of info :
Running your script with -3 option turned on can warn you about incompatibilities with Python 3 that cannot be fixed with automated Python 2 to 3 translation library.



Written 2 Jul, 2013. 587 views.

Upvote 5 Downvote Comment Share 1

Korey Hinton, http://koreyhinton.com
2 upvotes by Diliup Gabadamudalige and Viola Yee.

Multiple assignment:

a, b, c, d = 1, 2, 3, 4

Written 22 Aug, 2013. 397 views.

Upvote 2 Downvote Comment Share

Luna Ruan
12 upvotes by Fredrik Dahlgren, Yunyun Chen, Mathew Kellogg, (more)

I'm not sure if this is cool or plain disturbing, but you can set False equal to True and True equal to False. Good luck!

```
>>> True = False
>>> False
False
>>> True
False
>>> True == False
True
```

Updated 22 Jul, 2012. 673 views.

Upvote 12 Downvote Comments 4+ Share

Himanshu Mishra, Writer | IIT Kharagpur
14 upvotes by Adib Behjat, Mark Wong, Vladislav Zorov, (more)

import os

Now you need not to worry about the type of operating system you're program will be running on. os module contains all that information.
For example
Linux uses the forward slash**(/)** for denoting the path (e.g. /etc/environment) while Windows uses the backward slash (\ ) (e.g. C:\Windows\System32)

This can be the basic need of os module. While there are many more uses you can take advantage of.

You might want to see this 15.1. os - Miscellaneous operating system interfaces - Python 2.7.9 documentation

Written 1 Feb. 1,195 views.

Upvote 14 Downvote Comment Share

Jakub Zarzycki, Django developer
2 upvotes by Ziya Wings and Charles Wu.

Simple counter

```
>>> from collections import defaultdict
>>> counter = defaultdict(int)
>>> counter['a'] += 1
>>> counter['b'] += 1
>>> counter['a'] += 1
>>> counter
defaultdict(<type 'int'>, {'a': 2, 'b': 1})
```

Default value

```
>>> text = ''
>>> option = text or 'empty'
>>> option
'empty'
```

Center text

```
>>> 'foo'.center(10)
'   foo    '
```

Written 7 Dec, 2011. 1,107 views.

Upvote 2 Downvote Comments 2 Share

Morimoto Tetsuya, Programmer
2 upvotes by Nana A. Thompson and Nate Ferrero.

I like this snippet reading file with each block.

```
>>> with open(file_name) as f:
```

```
2 ...   for block in iter(lambda: f.read(10), ''):
3 ...     do_something
```

Written 7 Dec, 2011. 1,176 views.

Upvote 2 Downvote Comments 1+ Share

Anonymous
55 upvotes by Abhishek Shivkumar, Niyikiza Aimable, Ali Ahmadvand, (more)

Sending automated mails to Gmail/ Email bomb

```
def send_email():
    import smtplib

    gmail_user = "gmail_user@gmail.com   "
    gmail_pwd = "password"
    FROM = 'gmail_uswr@gmail.com   '
    TO = ['gmail_target@gmail.com   '] #must be a list
    SUBJECT = "Testing sending using gmail"
    TEXT = "Testing sending mail using gmail servers"

    # Prepare actual message
    message = """\From: %s\nTo: %s\nSubject: %s\n\n%s""" % (FROM, ",
".join(TO), SUBJECT, TEXT)
    try:
        #server = smtplib.SMTP(SERVER)
        server = smtplib.SMTP("smtp.gmail.com   ", 587) #or port 465 doesn't
seem to work!
        print "print 1"
        server.ehlo()
        print "print 2"
        server.starttls()
        print "print 3"
        server.ehlo()
        server.login(gmail_user, gmail_pwd)
        print "print 4"
        server.sendmail(FROM, TO, message)
        print "print 5"
        #server.quit()
        server.close()
        print 'successfully sent the mail'
    except:
        print "failed to send mail"

send_email()
```

In order to send 100's of mail to an user just put send_mail() inside a loop your email bomb is ready. :P

SOURCE: Travel Coding Linux ... My Views (my blog :))

Written 16 May, 2013. 2,411 views.

Upvote 55 Downvote Comment Share 3

Silvery Fu, in the zone
18 upvotes by Eliot Ball, Pavan Vemana, Taro Sato, (more)

```
import antigravity
```

Written 9 Apr, 2014. 771 views.

Upvote 18 Downvote Comment Share

Frank P Mora, Studied philosophy and logic in schoo... (more)


7 upvotes by Seyram Komla Sapaty, Adolfo De Unanue, Stephen McInerney, (more)

list comprehensions

[# list comprehensions
 # In-line anonymous list comprehensions functions

```
# sum
nl = [2, 3, 6, 18]
[j for j in [0] for i in nl for j in [j + i]][-1]

# product
[j for j in [1] for i in nl for j in [j * i]][-1]

# factorial
fac = 3
[j for j in [1] for i in range(2, fac+1) for j in [j*i]][-1]


# with prefix functions

T= [set(s) for s in [(1,2,3,4,5),(2,3,4,5,6),(3,4,5,6,7),(4,5,6,7,8)]]

f= set.intersection
[ s for s in [f(*T[:2])] for i in range(2,len(T)) for s in [f(s,T[i])] ][-1]
set([4, 5])     ## issues the null set if no intersection between all given sets

f= set.union
[ s for s in [f(*T[:2])] for i in range(2,len(T)) for s in [f(s,T[i])] ][-1]
set([1, 2, 3, 4, 5, 6, 7, 8])


# join
T= [(1,2,3,4,5,6,7),(3,4,5,6,7),(5,6,7),(7,8,9,2,1,5)]
[ s for s in [T[0]] for i in range(1,len(T)) for s in [s + T[i]] ][-1]
(1, 2, 3, 4, 5, 6, 7, 3, 4, 5, 6, 7, 5, 6, 7, 7, 8, 9, 2, 1, 5)

T= ["abcd","efgh","ijkl","mnop","qrst"]
[ s for s in [T[0]] for i in range(1,len(T)) for s in [s + T[i]] ][-1]
'abcdefghijklmnopqrst'


# with predicates

l= [20,30,40,50,60,70,80,90,1,2,3,4,5]

# "all" function
[ p for p in [l[0]>42] for i in range(1,len(l)) for p in [p and l[i]>42] ][-1]
False

# "any" function
[ p for p in [l[0]>42] for i in range(1,len(l)) for p in [p  or l[i]>42] ][-1]
True
```

 # Also I work a lot with strings but don't like quoting everything

 # dictionary comprehensions

```
ls= "one ten two twenty three thirty four forty five fifty".split()

{ ls[i]: ls[i+1] for i in range(0,len(ls),2) }
{'four': 'forty', 'three': 'thirty', 'five': 'fifty', 'two': 'twenty', 'one': 'ten'}

ls1='one two three four five six seven eight nine ten'
ls2='aa1 ab2 ac3   ad4  ae5  af6 ag7   ag8   ah9  ai0'
```

{ l1:l2 for l1,l2 in zip(ls1.split(), ls2.split()) }
{'seven': 'ag7', 'ten': 'aio', 'nine': 'ah9', 'six': 'af6', 'three': 'ac3',
'two': 'ab2', 'four': 'ad4', 'five': 'ae5', 'eight': 'ag8', 'one': 'aa1'}

Updated 24 Feb. 497 views.

Upvote 7 Downvote Comments 3+ Share

Stephen McAleese
6 upvotes by Ryan Fox Squire, Haoyue Wang, Tigran Sloyan, (more)

"""Prints out even numbers in list/array"""
my_array = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
print filter(lambda x: x % 2 == 0, my_array)

"""Prints odd numbers in list/array"""
my_list = range(1, 11)
print my_list[::2]

"""Floor: Returns 3 (instead of remainder) """
x = 10 // 3
print x

Written 1 Feb. 547 views.

Upvote 6 Downvote Comment 1 Share

Ilian Iliev, Python & Django Developer
1 upvote by Mohitdeep Singh.

Multiplicate elements in a list is simpler than you think:

```
1 from operator import mul
2 reduce(mul, range(1,5)) # 1 * 2 * 3 * 4
```

Written 25 Apr, 2013. 463 views.

Upvote 1 Downvote Comment Share 1

Nitin Bansal
7 upvotes by Julie Prentice, Joseph Freemind, Dan Goldin, (more)

I used backdoor recently. It is an interpreter to a running process. You can debug a process, see how states changes in a process. It is also possible to change variables from backdoor in the running process.

edit: Added the link
http://eventlet.net/doc/modules/...

Updated 3 Oct, 2012. 652 views.

Upvote 7 Downvote Comments 1+ Share

Amit Dev, Life long learner, Eclectic, Computer... (more)
5 upvotes by Victor Jolissaint, Yifei Zhang, Sean Zhu, (more)

The *-m* option of the interpreter to try quick commands, like Http server or **timeit**:

```
1 $ python -mtimeit -s 'd = {1:2}' 'd.has_key(1)'
2 10000000 loops, best of 3: 0.0666 usec per loop
3 $ python -mtimeit -s 'd = {1:2}' '1 in d'
4 10000000 loops, best of 3: 0.0365 usec per loop
```

which, btw, shows why you should never use has_key on dicts.

Python 3:

PEP 3132 -- Extended Iterable Unpacking :

```
a, *b, c = range(5) # a=0, b = [1,2,3], c=4
```

PEP 3104 -- Access to Names in Outer Scopes : Finally closures are easier in python:

```
def counter():
  n = 0
  def count():
     nonlocal n
     n += 1
     return n
return count
```

Updated 28 Oct, 2013. 806 views.

Upvote 5 Downvote Comments 1+ Share

Sang Han
2 upvotes by Benjamin Fox and Sushant Srivastava.

Build A Recursive Tree

```
from collections import defaultdict

Tree = lambda: defaultdict(Tree)
```

Lay Down Nodes

```
taxonomy = tree()
taxonomy['Animalia']['Chordata']['Mammalia']['Carnivora']['Felidae']['F
taxonomy['Animalia']['Chordata']['Mammalia']['Carnivora']['Felidae']['P
taxonomy['Animalia']['Chordata']['Mammalia']['Carnivora']['Canidae']['C
taxonomy['Animalia']['Chordata']['Mammalia']['Carnivora']['Canidae']['C
taxonomy['Plantae']['Solanales']['Solanaceae']['Solanum']['tomato']
taxonomy['Plantae']['Solanales']['Solanaceae']['Solanum']['potato']
taxonomy['Plantae']['Solanales']['Convolvulaceae']['Ipomoea']['sweet po
```

Represent as JSON

```
import json
print(json.dumps(taxonomy, indent=4))

{'Animalia': {'Chordata': {'Mammalia': {'Carnivora': {'Canidae': {'Cani
```

Written 15 Oct. 2,455 views.

Upvote 2 Downvote Comment Share

Rainfiel Lei, game programmer
7 upvotes by Kailash Budhathoki, Rafael Oliveira, Harit Himanshu, (more)

This is cool, I like it.

```
>>> data = {'arg1': 'tom', 'arg2': 'jerry'}
>>> '%(arg1)s is a cat, %(arg2)s is a mouse' % data
>>> 'tom is a cat, jerry is a mouse'
```

Written 6 Dec, 2011. 1,353 views.

Upvote 7 Downvote Comment 1 Share

Michael Herman
2 upvotes by Zhongliang Li and Diliup Gabadamudalige.

Ternary Expressions

```
>>> y = 1
>>> x = 10 if (y == 1) else 20
>>> x
10
>>> y = 2
>>> x = 10 if (y == 1) else 20
>>> x
20
```

Written 17 Jun, 2013. 396 views.

Upvote 2 Downvote Comment Share

Rav Onaus
1 upvote by Charles Wu.

Execute arbitrary, multi-line python code on the command line by calling exec():

```
python -c "exec('for x in range(5): print x\nprint \'Finished\'')"
```

Written 27 Apr, 2013. 323 views.

Upvote 1 Downvote Comments 1+ Share

Prashant Gaur, Python and JavaScipt Developer
4 upvotes by Amit Saha, Asad Moosvi, Hardik Akbari, (more)

Clear Python shell:

```
import os
os.system('clear')
```

Updated 7 May, 2013. 515 views.

Upvote 4 Downvote Comments 2+ Share

Juan Sánchez
1 upvote by Henry Williams.

def sample():
    z = 1
    y = 2
    x = 3
    print locals()

Written 12 Aug, 2014. 181 views.

Upvote 1 Downvote Comments 1+ Share

Benny Tsui, Since 2005

Sort using operator.itemgetter:

```
>>> import operator
>>> l = [('foo', 2, 'b'), ('bar', 1, 'a'), ('baz', 2, 'a')]
>>> sorted(l, key=operator.itemgetter(1), reverse=True)
[('foo', 2, 'b'), ('baz', 2, 'a'), ('bar', 1, 'a')]
>>> sorted(l, key=operator.itemgetter(1, 2))
[('bar', 1, 'a'), ('baz', 2, 'a'), ('foo', 2, 'b')]
```

Written 20 Mar, 2013. 334 views.

Upvote Downvote Comment Share

Nithin Saji, physics and computer science under gr... (more)
1 upvote by Derek Ho.

```
>>> from __future__ import braces
```

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

Written 10 Jul, 2013. 403 views.

Upvote 1 Downvote Comment Share

Mat Mathews, Started using Python in 2004 to build... (more)
9 upvotes by Julie Prentice, Amir Tarighat, Denzil Correa, (more)

I apologize for being so simplistic, but the best Python trick is to replace about 50K lines of a complex, dependency injected java code with about 10K lines of Python.. Badda Bing.. its a wonderful maintainable thing.

Written 26 Nov, 2012. 458 views.

Upvote 9 Downvote Comments 2 Share

Abel Paul, A Procrastinator by Choice and A Prog... (more)
11 upvotes by Christopher VanLang, Seyram Komla Sapaty, Rajath Ramakrishna, (more)

For loop with an If-Else for printing , all in one line ...

```
1 print ['even' if i%2 == 0 else 'odd' for i in range(0,10)]
2 ['even', 'odd', 'even', 'odd', 'even', 'odd', 'even', 'odd', 'even', 'o
```

Now that was pretty impressive to me.. \m/

Written 22 Apr, 2014. 792 views.

Upvote 11 Downvote Comments 3+ Share

Ben Roberts, CTO, chief architect, Nutrislice

python's console is super useful for just testing things out and learning how the language and various objects work. I even use it as my calculator. A few really useful introspection tools when consoling are the dir() and help(). Use these on any object to figure out what's going on with them.

```
1 >>> d= {}
2 >>> dir(d)
3 ['__class__', '__cmp__', '__contains__', '__delattr__', '__delitem__',
4 >>>help(d)
5
6 Help on dict object:
7
8 class dict(object)
9  |  dict() -> new empty dictionary
10 |  dict(mapping) -> new dictionary initialized from a mapping object'
11 |      (key, value) pairs
12 |  dict(iterable) -> new dictionary initialized as if via:
13 |      d = {}
14 |      for k, v in iterable:
15 |          d[k] = v
16 |  dict(**kwargs) -> new dictionary initialized with the name=value p
17 |      in the keyword argument list.  For example:  dict(one=1, two=2
18 |
19 |  Methods defined here:
20 |
21 |  __cmp__(...)
22 |      x.__cmp__(y) <==> cmp(x,y)
23 . . .
```

Updated 2 Mar, 2013. 352 views.

Upvote Downvote Share 1

Ryan Barnes, Legend

Enter interactive debugger where you can examine variables and stack frames among other things:

```
import pdb;pdb.set_trace()
```

Written 20 May, 2013. 323 views.

Upvote Downvote Comment Share

Diliup Gabadamudalige, Composer/Audio Engineer/Python Coder/... (more)

if you need to convert the key item of a dict to a list:

adict={'a': [], 'count': 1, 'EA': False}
alist = adict.items()
this gives you
[('a', []), ('count', 1), ('EA', False)]

Written 20 Apr, 2014. 265 views.

Upvote Downvote Comment Share

Anonymous
1 upvote by Bala Krishnan.

Autovivification (https://en.wikipedia.org/wiki/Au...)

```
1 >>> from collections import defaultdict
2 >>> def tree(): return defaultdict(tree)
3
4 >>> t = tree()
5 >>> t['A']['child_of_A']['child_of_child_of_A'] = 'some data'
6 >>> t['B']['child_of_B'] = 'some other data'
```

This seems to be the source hrldcpr/tree.md

Written 21 May, 2013. 355 views.

Upvote 1 Downvote Comment Share 1

Anonymous
5 upvotes by Martin Mattis, Swateek Jena, Abhijit Banerjee, (more)

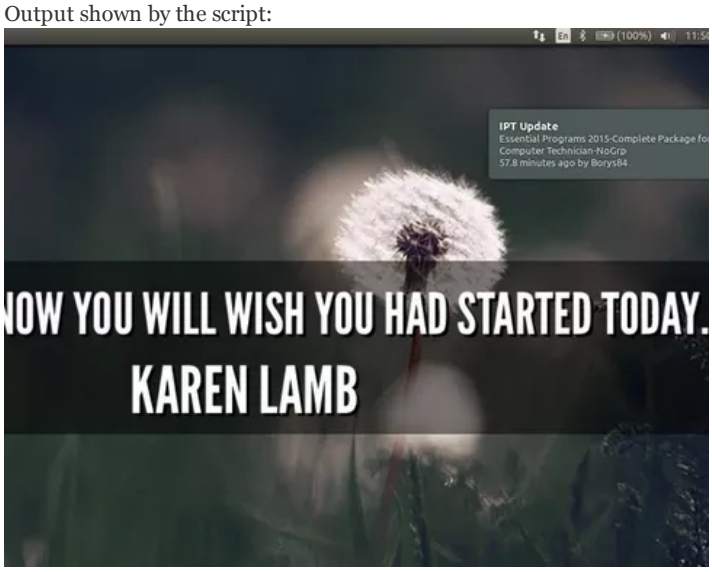
From a long time I have been thinking about writing a python script which can show the freeleech torrent that come upon IPT . For those who are not aware of what IPT is, it's a private torrent website. There is a problem of upload credit in this site in which you have to maintain upload/download ratio. So, in order to upload you have to download a freeleech torrent (which will then seed to a leecher) and that too just after it came(max 5-10 minutes after it came on site).

So I made a python script in order to avoid visiting the site to know if a new freeleech torrent came to the site. Here is the code:

```
1 from requests import session
2 from bs4 import BeautifulSoup
3 import pynotify
4 from time import sleep
5
```

```
def sendmessage(title, message):
    pynotify.init("Test")
    notice = pynotify.Notification(title, message)
    notice.show()
    return

payload = {
    'action'   : 'Login',
    'username' : 'USERNAME',
    'password' : 'PASSWORD'
}
login_url='https://www.iptorrents.com/t'

prev_title = ""

while True:
    with session() as c:
        c.post(login_url, data = payload)
        request = c.get('https://www.iptorrents.com/t')
        soup = BeautifulSoup(request.text)
        data = soup.find_all('span',class_="t_tag_free_leech")
        i = -1
        for freeleech in data:
            i = i + 1
            if i > 0:
                parent = freeleech.parent
                print parent
                title =  parent.find('a').text
                time =  parent.find('div').text
                if prev_title != title :
                    sendmessage("IPT Update", title + '\n' + time)
                    prev_title = title
                break
```

Output shown by the script:



Written Tue. 97 views.

Upvote 5 Downvote Comments 1+ Share

Somesh Singh, Growing up ..
3 upvotes by Usman Qazi, Bharath G.M, and Ankit Sharma.

```
max(map(int, myList))
```

get max value from a list without any jitters

Updated 25 May, 2014. 395 views.

Upvote 3 Downvote Comment Share

Marcin Szamotulski

There is a very cool module to build command line with Python interfaces, read this post: Command line user interface with Python .

Written 9 Jun, 2013. 312 views.

Upvote Downvote Comment Share 1

Jabba Laci

If you want bash-like functionalities in your command-line script like browsing previous commands with the Up arrow, or move the cursor with the Left/Right arrows, simply add one import:

```
import readline
```

Written 18 Jun, 2013. 328 views.

Upvote Downvote Comment Share

Qiangjun Ran, Programmer

python -m pyftpdlib -w -p 21
python -m SimpleHTTPServer

Written 22 Oct. 83 views.

Upvote Downvote Comment Share

Dmitry Loginov

```
>>> a = [1,2,3]
>>> a.append(a)
>>> print a
[1, 2, 3, [...]]
>>> print a[3][3][1]
2
```

Written 15 Dec. 55 views.

Upvote Downvote Comment Share

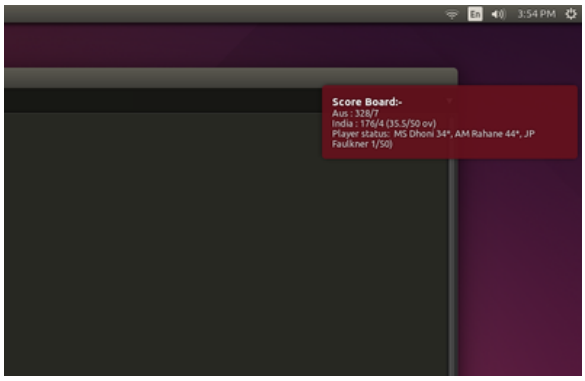
Karan Dev, Algoholic, Python Python everywhere, ... (more)
24 upvotes by Love Sharma, Dragos Serban, Om Patri, (more)

I just wrote the code sitting in my office to see International Cricket Match score without actually browsing the website. Answer of Anubhav Yadav inspired me to write something similar. It will show the latest score every 2 minutes. It also shows the player's and overs's status.

```
import requests
import pynotify
import re
from time import sleep
import json


def popup(title, message):
    pynotify.init("Test")
    pop = pynotify.Notification(title, message)
    pop.show()
```

```
    return


def getscore():
        url = "http://www.espncricinfo.com/icc-cricket-world-cup-2015/
        r = requests.get(url)
        while r.status_code is not 200:
                r = requests.get(url)
        data = json.loads(r.text)
        player_status = data['match']['current_summary'].strip()
        team1_name = data['other_scores']['international'][0]['team1_n
        team1_score = data['other_scores']['international'][0]['team1_
        team2_name = data['other_scores']['international'][0]['team2_n
        team2_score = data['other_scores']['international'][0]['team2_
        if not team1_score:
                team1_score = 'Yet to bat'
        if not team2_score:
                team2_score = 'Yet to bat'
        score = str(team1_name) + ' : ' + str(team1_score) + '\n\n' +
        player_status = re.sub(r'.*ov,','', str(player_status))
        score = score + '\nPlayer status: ' + player_status
        popup("Score Board:-", score)
        sleep(120)


if __name__ == "__main__":
        while True:
                getscore()
```

Here is the screen shot.



You just need to download package Pynotify and place it in the same directory where your code file is.

Thanks to Anubhav Yadav for such a nice idea.

Written Thu. 544 views.

Upvote 24 Downvote Comment 1 Share 1

Ayush Goel, Learner, Worker

Just adding to the points:

any object to a string format (readable format)
repr(object)
or str(object)

```
>>> repr(this)
"<module 'this' from 'C:\\Python27\\lib\\this.pyc'>"
```

ascii related functions (quite useful sometimes)

```
>>> ord('a')
97
>>> chr(97)
'a'
```

Where are you?

```
>>> __name__
'__main__'
```

If making a standalone script, the statements to be executed should be put like:

```
## scr1.py
## classes and functions
if __name__ == "__main__":
    ## statements to be exceuted
```

This makes your classes and functions reusable by other scripts as this module can be imported easily

```
>>> import scr1
```

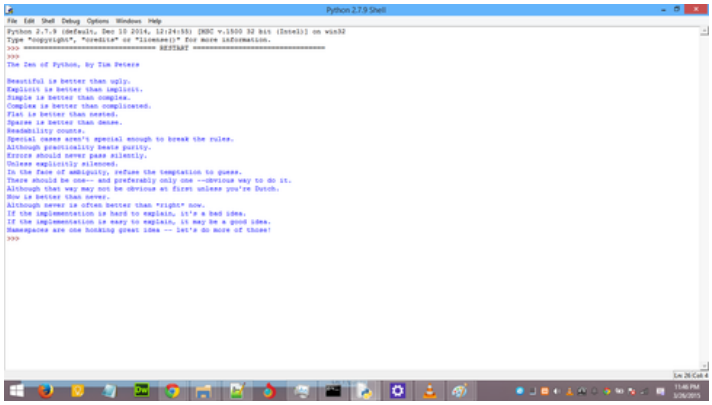
And the last, to check for the effectiveness of your code, the following modules are available which can help you to analyze your code:
timeit
cProfile
pstats

Written 7 Dec, 2011. 1,110 views.

Upvote Downvote Comments 1+ Share

Rakesh Yadav, Tech junkie@IIT DELHI
7 upvotes by Cyril Anderson, Andreas Iversen, Mimansa Jaiswal, (more)

"import this"
output of this is Perhaps the best-known and most-quoted passage of Python philosophy by Tim Peters, a longtime Python guru who wrote down many of the principles that guide Python's own development process. The 19 lines he came up with, called the Zen of Python, have been so influential to Python programmers over time that they are immortalized as Python Enhancement Proposal (PEP) 201 and in the Python distribution itself, as an "Easter egg" module called this



Written Thu. 339 views.

Upvote 7 Downvote Comment Share

Paul Davies C, Rails Developer

Invoking functions in a class using strings

```
class myClass:
    def test(self):
        print "Hello"
    def test1(self):
        print "Hello world!"
    def main(self):
        ip=raw_input("Enter 'test' or 'test1' : ")
        try:
            func=getattr(self,ip)
        except AttributeError:
            print "No such function defined"
        try:
            func()
        except Exception as e:
            pass
ob=myClass()
ob.main()
```

Written 25 Apr, 2013. 333 views.

Upvote Downvote Comment Share

Anand Gupta, Engineer@WalmartLabs
5 upvotes by Frédéric Grosshans-André, Prakash Chandrasekaran, Sudarshan Shetty, (more)

if you don't like the complicated and clumsy way of initializing dictionaries in python

replace:

```
dict1 = {"a":1, "b":2, "c":3)
```

with :

```
def getdict(**kwrgs):
      return kwrgs

newdict = getdict(a=1, b=2,c=3)
```

==========

Converting all the integers in a list to string :

```
map(str, a)
```

Before:

```
a = [1,2,3,4,5]
```

After:

```
a = ['1', '2', '3', '4', '5']
```

Updated 18 May, 2013. 688 views.

Upvote 5 Downvote Comments 1+ Share

Nicflocka Nickmweezy, Nairobian,geek,informaniac
3 upvotes by Lorenzo Hernandez, Bharath Mylarappa, and Sundar Lakshmanan.

type "python -m SimpleHTTPServer " on the terminal or command line for windows to share files instantly on your LAN What are some cool Python tricks?

Written 21 Mar. 213 views.

Upvote 3 Downvote Comment 1 Share

Originally answered on What are some cool Python tricks/hacks/short cuts?

Dhruv Kapur, seeking Interactions.
1 upvote by Surya Prakash.

One of the popular ones of all times, I guess :P

```
a = 1
b = 2
a,b = b,a
```

And the two are swapped!

Written 29 Jun, 2014. 165 views.

Upvote 1 Downvote Comment Share

Arpan Mehrotra, minimalist

```
#!/usr/bin/python  import os, glob, Image  pic_list = []  pic_list = glob.glob('*.JPG') pic_list.sort()  print len(pic_list), 'images found...' for temp in pic_list:     print temp   cmd = 'fulla -g 0.0216776:-0.0799067:0.0566601:1 ' for temp in pic_list:     repair = cmd + temp     os.popen(repair).read()
```

Written 14 Oct. 118 views.

Upvote Downvote Comment Share

Kranthi Rajoli

I use pyparsing module regularly and found it to be very effective: pyparsing - home

```
1 from pyparsing import Word, alphas
2
3 greet = Word( alphas ) + "," + Word( alphas ) + "!"
4 greeting = greet.parseString( "Hello, World!" )
5 print greeting
6
7 ['Hello', ',', 'World', '!']
```

Written 16 May, 2013. 324 views.

Upvote Downvote Comment Share

Sumit Raj, Python Developer, Growth Hacker, Free... (more)
1 upvote by Sandhan Sarma.

Check palindrome:-

```
a="madam"
if a=a[::-1]:
   print "palindrome"
else:
   print "Not palindrome"
```

Written 20 Aug, 2014. 161 views.

Upvote 1 Downvote Comment Share 1

Saibal Banerjee, All scripts no play.

The cool lambda

```
swap = lambda x,y:(y,x)
a=3
b=4
(a,b)=swap(a,b)
print a,b
```

Written 20 May, 2013. 308 views.

Upvote Downvote Comments 1+ Share

Anonymous
1 upvote by Ezugworie Ikechukwu.

import antigravity

antigravity looks something like this:

> [code]
> import webbrowser
>
> webbrowser.open("http://xkcd.com/353/ ")
> [/code]

It leads us to an already existing url. So you can also create your own modules with your own cool links in the place of "http://xkcd.com/353/ '
;)
For example:
Create a module called quora with the following content:

```
1 import webbrowser
2
3 webbrowser.open('http://www.quora.com')
```

Save this in python library. Open Python interpretter. Type import quora. And Voila!

Written 6 Feb. 159 views.

Upvote 1 Downvote Comment Share

Avinasha Br, Team Lead
2 upvotes by Miroslav Georgiev and Rajat Khandelwal.

converting string to uppercase without going python prompt

python -c 'print "abc".upper()'

Written 19 Jan, 2012. 317 views.

Upvote 2 Downvote Comment 1 Share

Chuan Hu

You can not use
`var++`

Written 14 Jul, 2013. 349 views.

Upvote Downvote Comment Share

Satyam Pujari, #tech #dev #hack
5 upvotes by Tolga Yilmaz, Jonathan Ellenberger, Bhanukiran Perabathini, (more)

call lambda

```
(lambda x : 2*x)(2)
```

Written 18 Dec, 2013. 394 views.

Upvote 5 Downvote Comments 2 Share

Ali Jina

Using set() as an execution engine. I use this when I have a generator (even

better when chained) that I want to run but do not necessarily care about the return value (or they all return None, like a regular function call). Just run set(generator) and done.

Written 24 Aug, 2013. 409 views.

Upvote Downvote Comments 2 Share

Vivek Shah, BIM Revit CAD Technician at Hi-Tech E... (more)
1 upvote by Dmitry Loginov.

Hidden features of Python...Here is the link..

Hidden features of Python

Written 6 Nov. 188 views.

Upvote 1 Downvote Comments 1+ Share

Rishi Giri, Code is my Poetry
5 upvotes by Saurav Verma, Octave Julien, Devesh Sawant, (more)

```
1 import urllib2
2 from BeautifulSoup import BeautifulSoup
3 x = 1
4 while ( x <= 100 ):
5         print BeautifulSoup(urllib2.urlopen("https://twitter.com/"+str(
6         x+=1
```

The above code will fetch the user name whose twitter handle start from @1 to @100.

Written 21 Mar. 352 views.

Upvote 5 Downvote Comment 1 Share 2

Anonymous
4 upvotes by Jay Rambhia, Sugavanesh Balasubramanian, Ashwin Jagadeesha, (more)

The most frequent and useful two things that I use in python are -

Using **pickle**

> ***import pickle***
> ***d=[1,2,3,4,4,5,3,1,21,2,431,5,315,13,531,5,135]***
> ***pickle.dump(d,open("filename","w")***
> ***s=pickle.load(open("filename"))***
> ***print s***
> ***>>[1,2,3,4,4,5,3,1,21,2,431,5,315,13,531,5,135]***

The pickle creates a sort of dump where python objects like lists, dict, etc can be loaded from and dumped to. This becomes efficient if you want to use the output of one program to another without involving a db (even across systems)

Sometimes after going halfway through the code I want to experiment with things. in the later stages to check variable values, etc.

To do so in IDE one may copy paste the whole code as an option,
another way to do so is using the execfile command

```
>>> execfile("filename.py   ")
locals()   # to check the current variables
```

Now you can normally experiment from the end of the filename.py execution.

Other things which I use very often are ->

- regular expression for any url if you're looking to parse a page -"
  ((?:https?://|www\.)[-\w]+(?:\.[-\w]+)*(?::\d+)?(?:/(?:(?:[~\w\+%-]|(?:[,.;@:][^\s$]))+)?)*(?:\?[\w\+%&=.;:-]+)?(?:\#[\w\-\.]*)?)(?:\p{P}|\s|<|$)"

- If you're running a url fetch command on python and your server is not hosted a server which has other ports (like ftp ssh) set to timeout then you can use the
  socket.defaulttimeout() function
  otherwise the inefficient urllib2 timeout would work...

Updated 27 Sep, 2013. 359 views.

Upvote 4 Downvote Comment Share

Haotian Xu, DEV
1 upvote by Binu Jasim.

import this

Written 25 Sep, 2013. 266 views.

Upvote 1 Downvote Comment Share

12 Answers Collapsed

Top Stories from Your Feed

Greeshma Girish upvoted this • Wed

What are some pictures (without text) with deep meaning(s)?

Sushant Somkuwar, member of League of Assassins
1.4k upvotes by Greeshma Girish, Sophia Noor, Abdurraheem Raqib, (more)

Read In Feed

Virali Modi upvoted this • Mon

What are some of the best questions/answers asked/given by kids?

Divya Murugesan
2.3k upvotes by Virali Modi, Mohan Ram Karthik, Padmaja Anand, (more)

I met this amazing 6 yr old kid who is crazy about cars. I heard this conversation between him and his mother. Mom (typical Indian mother): You need to study well to get a good job so that you can ...

Read In Feed

Poorvisha Ravi upvoted this • 20 Mar

What are some cool photos depicting India/Indian-ness?

Ghanshyam Lele, CS undergrad
2.1k upvotes by Poorvisha Ravi, Rajender Reddy, Abha Pandey, (more)

Read In Feed