Table of Contents

Python Notes **1**
- regex pattern for any number 1
- remove comments 1
- xor 1
- divide a list 1
- set-difference between two lists 1
- emulate switch with a dict 2
- timestamps 2
- dict extract 2
- Sorting Coupled Lists 2
- Default Values for Mutable Parameters 2
- Enumerate 3
- Path of This File 3
- Console Input 3
- Console Output 4
- Get IP address 4

Python Notes

regex pattern for any number

```
[-+]?[0-9]*\.?[0-9]+([eE][-+]?[0-9]+)?
```

Advice from here.

remove comments

```
for line in f:
    line = line.split('#')[0].strip() # remove comments
    if line:
        do_something()
```

xor

```
xor = lambda x, y: True if (x or y) and not (x and y) else False
```

divide a list

```
def divide_list(li, n):
    """ Yield successive n lists of even size.
    """
    start = 0
    for i in xrange(n):
        stop = start + len(li[i::n])
        yield li[start:stop]
        start = stop

def chunk_list(li, n):
    """ Yield successive n-sized chunks from l.
    """
    for i in xrange(0, len(li), n):
        yield li[i:i+n]
```

Example:

```
>>> a = range(10)
>>> list( divide_list(a, 3) )
[[0, 1, 2, 3], [4, 5, 6], [7, 8, 9]]
>>> list( chunk_list(a, 3) )
[[0, 1, 2], [3, 4, 5], [6, 7, 8], [9]]
```

Advice taken from here.

See also:

```
group = lambda t, n: zip(*[t[i::n] for i in range(n)])
```

Advice taken from here.

set-difference between two lists

```
diff_list = [item for item in list1 if not item in list2]
```

Advice taken from here.

```
def check_for_missing(haves, wants):
    return [item for item in wants if not item in haves]
```

emulate switch with a dict

```
def f(x):
    return {
        'a': 1,
        'b': 2,
        }.get(x, 9)
```

Advice taken from here.

timestamps

```
timestamp = datetime.datetime.now().strftime('%Y-%m-%d_%a_%H:%M:%S')
'2011-04-13_Wed_11:49:27'
>>> time.strftime('%Y-%m-%d %H:%M:%S')
'2011-06-08 13:29:49'
>>> time.strftime('%Y-%m-%d-%Hh%M')
'2011-06-08-13h30'
```

dict extract

```
def extract(d, keys):
    return dict((k, d[k]) for k in keys if k in d)
```

Sorting Coupled Lists

Often one has several lists where the information stored in the i-th entry of each list is related, and one wants to sort all the lists together according to the entries in one of them. The following sorts two coupled lists according to the first.

```
zipped = zip(list1, list2)
zipped.sort()
list1, lists2 = map(list, zip(*zipped))
```

It can be generalized to several lists by

```
zipped = zip(*list_of_lists)
zipped.sort()
list_of_lists = map(list, zip(*zipped))
```

Default Values for Mutable Parameters

This advice was taken from here.

This is a common mistake that beginners often make. Even more advanced programmers make this mistake if they don't understand Python names.

```
def bad_append(new_item, a_list=[]):
    a_list.append(new_item)
    return a_list
```

The problem here is that the default value of `a_list`, an empty list, is evaluated at function definition time. So every time you call the function with the default parameter, you get the same instance of the list. Try it several times:

```
>>> print bad_append('one')
['one']

>>> print bad_append('two')
['one', 'two']
```

Lists are a mutable objects; you can change their contents. The correct way to get a default list (or dictionary, or set) is to create it at run time instead, inside the function:

```
def good_append(new_item, a_list=None):
    if a_list is None:
        a_list = []
    a_list.append(new_item)
    return a_list
```

For cases where it isn't natural to represent the default case by `None`, I like to make a default object as a place holder for default parameters.

```
_default = object()
def f(mut_param=_default):
    if mut_param is _default:
        mut_param = ['some', 'default', 'mutable']
```

Enumerate

Use `enumerate` when you want to do a for loop over an iterable with an index. So when you would do

```
i = 0
for e in elements:
    print i, e
    i += 1
```

do this instead

```
for i, e in enumerate(elements):
    print i, e
```

Path of This File

```
import os
path_of_this_file = os.path.abspath( __file__ )
dir_of_this_file = os.path.dirname(os.path.abspath( __file__ ))
```

Old stupid way:

```
path_of_this_file = os.path.realpath(os.path.dirname(sys.argv[0]))
```

Console Input

```
>>> uin = raw_input('my prompt:').strip()
my prompt:   hello world, 1234
>>> uin
'hello world, 1234'
```

Console Output

```
import subprocess
p = subprocess.check_output('ls -l -a', stderr=subprocess.STDOUT, shell=True)
print p
```

Get IP address

```
s = subprocess.check_output("/sbin/ifconfig | grep -w inet | grep -v 127.0.0.1 | awk '{print $2}'
```

-- RyanReece - 17 Oct 2008

This topic: Sandbox > PythonNotes
Topic revision: r23 - 2013-09-17 - RyanReece