Beyond Basic Programming - Intermediate Python
recluze.net/learn

Scenario

You're trying to download tweets and process them for sentiments:

```
 - "I have a @Microsoft Surface Pro 4 and LOVE it!"
- "#surfacepro @surface Why is this so expensive?"
```

How do we solve this problem?

- Strategy 1: Download all tweets and then process them
  - `download, loop [ process ]`

- Strategy 2: Download each tweet in small batches
  - `loop [ download tweet, process ]`

Need a function that can implement strategy 1 but does not need to keep everything in memory!

Same design pattern is used in many cases. For instance, machine learning: load images and train machine based on them.

- There are thousands (or even millions) of images so you can't load them all in memory.
- The processes are complicated so you don't want to couple loading logic with learing.

Generators

Let's first see what we're trying to do and then define generators.

In [19]:
```python
for i in [0, 1, 2, 3, 4]:    # keeps the list in memory
    print(i)                 # process each item
```

```
0
1
2
3
4
```

In [21]:
```python
for i in range(5):  # does not generate all five elements in memory!
    print(i)
```

```
0
1
2
3
4
```

In [20]:
```python
range(5)     # NOT a list but a generator object
```

Out[20]:
```
range(0, 5)
```

In [22]:
```python
def myrange(n):
    x = 0
    while x < n:
        yield x   # 'yield' turns a function into a generator
        x += 1
```

In [23]:
```
type(myrange(5))
```
Out[23]:
```
generator
```

In [24]:
```
for i in myrange(5):
    print(i)
```
```
0
1
2
3
4
```

In [25]:
```
def countdown(n):
    while n > 0:
        print("Computing next number ... ")
        yield n
        n -= 1
```

In [26]:
```
for i in countdown(5):
    print(i)
```
```
Computing next number ...
5
Computing next number ...
4
Computing next number ...
3
Computing next number ...
2
Computing next number ...
1
```

In [27]:
```
v = countdown(5)
```

In [33]:
```
next(v)
```
```
---------------------------------------------------------------------------
StopIteration                             Traceback (most recent call last)
<ipython-input-33-10c82e4dde46> in <module>()
----> 1 next(v)

StopIteration: 
```

In [34]:
```
import random

def random_gen(low, high, num):
    i = 0
    while i < num:
        yield random.randrange(low, high)
        i += 1
```

In [35]:
```
r = random_gen(0, 100, 5)
```

In [37]:
```
type(r)
```
Out[37]:
```
generator
```

In [38]:
```
list(r)
```
Out[38]:
```
[59, 6, 44, 78, 75]
```

In [39]:
```
def random_gen_inf(low, high):
    while True:
        yield random.randrange(low, high)
```

In [40]:
```
r = random_gen_inf(0, 100)
```

In [73]:
```
next(r)
```

Out[73]:
```
67
```

Generator Syntax

In [74]:
```
%time v = [ i**2 for i in range(10000000) ]
```

```
CPU times: user 5.41 s, sys: 394 ms, total: 5.81 s
Wall time: 6.16 s
```

In [75]:
```
print(v[:10])
```

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

In [76]:
```
%time g = ( i**2 for i in range(10000000) )
```

```
CPU times: user 11 µs, sys: 1e+03 ns, total: 12 µs
Wall time: 15.7 µs
```

In [92]:
```
next(g)
```

Out[92]:
```
196
```

Real World Example

In [93]:
```
wwwlog = open("access-log")
for line in wwwlog:
    print(line)
    break
```

```
140.180.132.213 - - [24/Feb/2008:00:08:59 -0600] "GET /ply/ply.html HTTP/1.1" 200 97238
```

In [94]:
```
wwwlog = open("access-log")
total = 0
```

In [95]:
```
for line in wwwlog:
    bytestr = line.rsplit(None,1)[1]
    if bytestr != '-':
        total += int(bytestr)

print("Total", total)
```

```
Total 230741830
```

The generator way:

In [96]:
```
wwwlog      = open("access-log")
bytecolumn  = (line.rsplit(None,1)[1] for line in wwwlog)
bytes       = (int(x) for x in bytecolumn if x != '-')

print("Total", sum(bytes))
```

```
Total 230741830
```

Tailing a File

```
In [97]: import time
         def follow(thefile):
             thefile.seek(0, 2)         # Go to the end of the file
             while True:
                 line = thefile.readline()
                 if not line:
                     time.sleep(0.1)    # Sleep briefly
                     continue
                 yield line
```

```
In [98]: logfile  = open("test-log")
```

```
In [99]: loglines = follow(logfile)
```

```
In [100]: type(loglines)
Out[100]: generator
```

```
In [101]: for line in loglines:
              print(line, )

              if line[:1] == '.':
                  break
```

```
Something

Something-else

.
```

```
In [ ]:
```