연습문제 10장

1

In [1]:
```
def frange(start, stop=None, step=1.0):
    if stop is None:
        stop = start
        start = 0
    print start, stop, step
```

In [2]:
```
frange(1)
```

```
0 1 1.0
```

In [3]:
```
frange(0, 10)
```

```
0 10 1.0
```

In [4]:
```
frange(0, 10, 2)
```

```
0 10 2
```

In [5]:
```
def frange(start, stop=None, step=1.0):
    if stop is None:
        stop = start
        start = 0
    L = []
    value = start
    if step >= 0:
        while value < stop:
            L.append(value)
            value += step
    else:
        while value > stop:
            L.append(value)
            value += step
    return L
```

In [6]:
```
frange(10)
```

Out[6]:
```
[0, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0]
```

In [7]:
```
frange(5, 10)
```

Out[7]:
```
[5, 6.0, 7.0, 8.0, 9.0]
```

In [8]:
```
frange(0, 5, 0.5)
```

Out[8]:
```
[0, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5]
```

사실은 이러한 함수는 이미 준비되어 있다. numpy의 arange 함수를 사용하면 된다. 파이썬 표준 라이브러리는 아니므로 콘솔창(도스창)에서 다음과 같이 설치해야 한다.

```
pip install numpy
```

In [9]:
```
import numpy

numpy.arange(10)
```

Out[9]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [10]:
```
numpy.arange(5, 10)
```

Out[10]:
```
array([5, 6, 7, 8, 9])
```

In [11]:
```
numpy.arange(0, 5, 0.5)
```

Out[11]:
```
array([ 0. ,  0.5,  1. ,  1.5,  2. ,  2.5,  3. ,  3.5,  4. ,
4.5])
```

2

In [12]:
```
def adder(a, b):
    r = a + b
    x = int(bool(r & 0x02))
    y = r & 0x01
    return x, y
```

In [13]:
```
adder(0, 0)
```

Out[13]:
```
(0, 0)
```

In [14]:
```
adder(0, 1)
```

Out[14]:
```
(0, 1)
```

In [15]:
```
adder(1, 0)
```

Out[15]: (0, 1)

In [16]:
```
adder(1, 1)
```

Out[16]: (1, 0)

3

In [17]:
```
def mysum(*args):
    return sum(args)

mysum(1,2,3)
```

Out[17]: 6

In [18]:
```
import operator

def mysum(*args):
    return reduce(operator.add, args)

mysum(1,2,3)
```

Out[18]: 6

In [19]:
```
def mysum(*args):
    acc = 0
    for v in args:
        acc += v
    return acc

mysum(1,2,3)
```

Out[19]: 6

4

In [20]:
```
flist = []
for i in range(10):
    flist.append('img_{:03d}.jpg'.format(i))

for i in range(10):
    flist.append('img_{:03d}_thumb.jpg'.format(i))
```

In [21]:
```
flist
```

Out[21]:
```
['img_000.jpg',
 'img_001.jpg',
 'img_002.jpg',
 'img_003.jpg',
 'img_004.jpg',
 'img_005.jpg',
 'img_006.jpg',
 'img_007.jpg',
 'img_008.jpg',
 'img_009.jpg',
 'img_000_thumb.jpg',
 'img_001_thumb.jpg',
 'img_002_thumb.jpg',
 'img_003_thumb.jpg',
 'img_004_thumb.jpg',
 'img_005_thumb.jpg',
 'img_006_thumb.jpg',
 'img_007_thumb.jpg',
 'img_008_thumb.jpg',
 'img_009_thumb.jpg']
```

In [22]:
```
filter(lambda s: '_thumb' in s, flist)
```

Out[22]:
```
['img_000_thumb.jpg',
 'img_001_thumb.jpg',
 'img_002_thumb.jpg',
 'img_003_thumb.jpg',
 'img_004_thumb.jpg',
 'img_005_thumb.jpg',
 'img_006_thumb.jpg',
 'img_007_thumb.jpg',
 'img_008_thumb.jpg',
 'img_009_thumb.jpg']
```

In [23]:
```
filter(lambda s: '_thumb' not in s, flist)
```

Out[23]:
```
['img_000.jpg',
 'img_001.jpg',
 'img_002.jpg',
 'img_003.jpg',
 'img_004.jpg',
 'img_005.jpg',
 'img_006.jpg',
 'img_007.jpg',
 'img_008.jpg',
 'img_009.jpg']
```

5

In [24]:
```
s = 'as soon as possible'
```

In [25]:
```
''.join(map(lambda e: e[0].upper(), s.split()))
```

Out[25]: 'ASAP'

6

In [26]:
```
%%file data.txt
1 2 3
4 5 6
7 8 9
```

```
Writing data.txt
```

In [27]:
```
rows = []
for line in open('data.txt'):
    rows.append(map(int, line.split()))
```

In [28]:
```
rows
```

Out[28]: [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

In [29]:
```
x, y, z = zip(*rows)
```

In [30]:
```
x
```

Out[30]: (1, 4, 7)

In [31]:
```
y
```

Out[31]: (2, 5, 8)

In [32]:
```
z
```

Out[32]: (3, 6, 9)

7

In [33]:
```
def fact(n):
    if n <= 1:
        return 1
    return n * fact(n-1)
```

In [34]:
```
fact(1)
```

Out[34]: 1

In [35]:
```
fact(3)
```

Out[35]: 6

In [36]:
```
fact(10)
```

Out[36]: 3628800

In [37]:
```
fact(100)
```

Out[37]:
```
933262154439441526816992388562667004907159682643816214685929638952175999932299156089414639761565182862536979208272237582511852109168640000000000000000000000000L
```

8

In [38]:
```
def replace(L, a, b):
    for i, ele in enumerate(L):
        if type(ele) in (list, tuple):
            replace(ele, a, b)
        elif L[i] == a:
            L[i] = b
```

In [39]:
```
L = [3,2,[3,[[3],4]]]

replace(L, 3, 5)
```

In [40]:
```
L
```

Out[40]: [5, 2, [5, [[5], 4]]]

In [41]:
```

```