연습문제 5장

1

In [1]:
```
L = [1, 2, 3, 4, 5]
L[1:3] = [100]
L
```

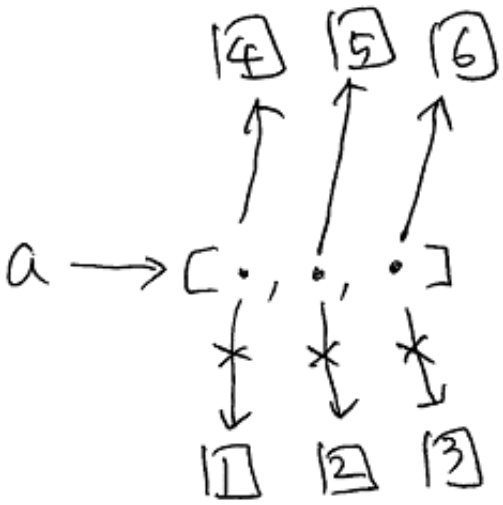
Out[1]:
```
[1, 100, 4, 5]
```

In [2]:
```
L = [1, 2, 3, 4, 5]
L[1:3] = 100
L
```

```
---------------------------------------------------------------------
----------
TypeError                                 Traceback (most recent
call last)
<ipython-input-2-e0bf36d463f0> in <module>()
      1 L = [1, 2, 3, 4, 5]
----> 2 L[1:3] = 100
      3 L

TypeError: can only assign an iterable
```

2



In [3]:
```
a = [1, 2, 3]  # 리스트를 생성한 후 심볼 a가 그 리스트를 참조하게 한다.
```

In [4]:
```
a[:] = [4, 5, 6]  # 리스트는 1, 2, 3 객체 대신에 4, 5, 6 객체를 참조하게 된다. a의 참조 값은 변경되지 않는다.
a
```

Out[4]: [4, 5, 6]

3

In [5]:
```
s = 'Sometimes I feel like a motherless child'
```

In [6]:
```
# 가)
' '.join(reversed(s.split()))
```

Out[6]: 'child motherless a like feel I Sometimes'

In [7]:
```
# 나)
''.join(reversed(list(s)))
```

Out[7]: 'dlihc sselrehtom a ekil leef I semitemoS'

In [8]:
```
# 나)
''.join(reversed(s))
```

Out[8]: 'dlihc sselrehtom a ekil leef I semitemoS'

In [9]:
```
# 나)
s[::-1]
```

Out[9]: 'dlihc sselrehtom a ekil leef I semitemoS'

In [10]:
```
# 다)
''.join(s.split())
```

Out[10]: 'SometimesIfeellikeamotherlesschild'

4

In [11]:
```
a = [1, 2, 3]
b = a * 3
c = [a] * 3
```

In [12]:
```
# 가)
b
```

Out[12]: [1, 2, 3, 1, 2, 3, 1, 2, 3]

In [13]:
```
c
```

Out[13]: [[1, 2, 3], [1, 2, 3], [1, 2, 3]]

In [14]:
```
# 나), 다)
a[0] = 0
```

In [15]:
```
b  # b는 새로운 리스트이므로 a의 변화에 영향 없음
```

Out[15]: [1, 2, 3, 1, 2, 3, 1, 2, 3]

In [16]:
```
c  # c는 a를 기초로한 리스트이므로 a의 변화에 민감
```

Out[16]: [[0, 2, 3], [0, 2, 3], [0, 2, 3]]

In [17]:
```
# 라)
a = [1, 2, 3]
c2 = [a[:]] * 3  # a의 참조를 복사한 후 반복하는 것이므로 a의 변화와 관계 없다
c2
```

Out[17]: [[1, 2, 3], [1, 2, 3], [1, 2, 3]]

In [18]:
```
a[0] = 0
c2
```

Out[18]: [[1, 2, 3], [1, 2, 3], [1, 2, 3]]

In [19]:
```
# 마)
a = [1, 2, 3]
c3 = [a[:], a[:], a[:]]
c3
```

Out[19]: [[1, 2, 3], [1, 2, 3], [1, 2, 3]]

In [20]:
```
# c3가 c, c2와 다른 점 : c3의 0, 3, 6 번째 데이터가 연동되어 있지 않다.

a = [1, 2, 3]
c = [a] * 3
c[0][0] = 0  # 값을 하나만 바꿔도 세 개의 값이 바뀐다
c
```

Out[20]: [[0, 2, 3], [0, 2, 3], [0, 2, 3]]

In [21]:
```
a = [1, 2, 3]
c2 = [a[:]] * 3
c2[0][0] = 0 # 값을 하나만 바꿔도 세 개의 값이 바뀐다
c2
```

Out[21]: [[0, 2, 3], [0, 2, 3], [0, 2, 3]]

In [22]:
```
a = [1, 2, 3]
c3 = [a[:], a[:], a[:]]
c3[0][0] = 0
c3
```

Out[22]: [[0, 2, 3], [1, 2, 3], [1, 2, 3]]

5

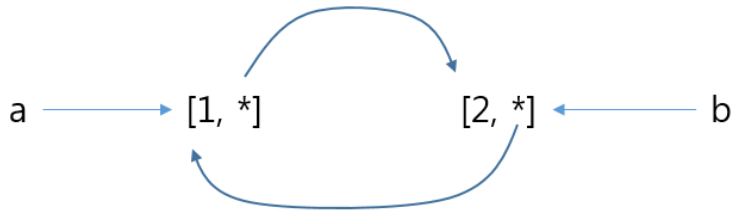
In [23]:
```
L = range(10)
L.sort(key=lambda e: e%5)
L
```

Out[23]: [0, 5, 1, 6, 2, 7, 3, 8, 4, 9]

6

In [24]:
```
a = [1]
b = [2]
```

In [25]:
```
a.append(b)
b.append(a)
```



In [26]:
```
# 나)
a
```

Out[26]: [1, [2, [...]]]

In [27]:
```
b
```

Out[27]: [2, [1, [...]]]

다) sys.getrefcount(a)는 레퍼런스 카운트 값 +1을 넘겨준다. 왜냐하면 getrefcount 함수에서 인수 참조를 한 번 더 하기 때문이다. 따라서 아래의 경우에 참조수가 2가 나온다.

In [28]:
```
# 다)
import sys

a = [1]
b = [2]
sys.getrefcount(a)
```

Out[28]: 2

그런데 b에서 a를 추가했기 때문에 다음 코드의 실행 결과는 3이 된다.

In [29]:
```
a.append(b)
b.append(a)
sys.getrefcount(a)
```

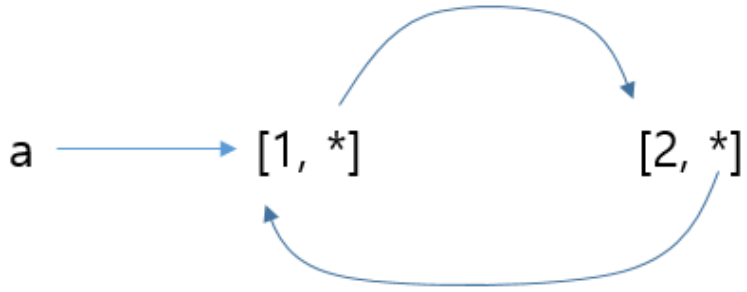
Out[29]: 3

라) b를 삭제해도 참조수는 줄어들지 않는다. 왜냐하면 b가 참조하던 리스트 자체는 남아있고 리스트 내에서 a 리스트를 참조하기 때문이다.

In [30]:
```
del b
sys.getrefcount(a)
```

Out[30]: 3

마)



바) a를 삭제해도 두 개의 리스트는 상호 참조하고 있기 때문에 메모리에서 제거되지 않는다. 이러한 현상을 메모리 누수 현상이라고 한다.

7

In [31]:
```
s1 = ['Spam', 'egg', 'Ham']
s1.sort(key=lambda s: s.lower())
s1
```

Out[31]: ['egg', 'Ham', 'Spam']

8

In [32]:
```
s = ' first item : second item : third item '
L = []
for w in s.split(':'):
    L.append(w.strip())
L
```

Out[32]: ['first item', 'second item', 'third item']

9

In [33]:
```
[w.strip() for w in s.split(':')]
```

Out[33]: ['first item', 'second item', 'third item']

10

In [34]:
```
import glob
import os

for fpath in glob.glob('*.py'):
    fsize = os.path.getsize(fpath)
    if fsize > 500:
        print fpath, fsize
```

11

In [35]:
```
import glob
import os
import time

curtime = time.time()
for fpath in glob.glob('*.*'):
    mtime = os.path.getmtime(fpath)
    if mtime > curtime - 60*60*20:
        print fpath, time.ctime(mtime)
```

12

옵션 처리방법은 getopt모듈 보다는 argparse 모듈을 사용하도록 변경되었기 때문에 argparse 모듈을 이용한 방법의 해제를 보인다.

In [36]:
```
import argparse


parser = argparse.ArgumentParser()
parser.add_argument("names", nargs='+', help="names")
parser.add_argument("--condition", default='bar', help="condition
option")
parser.add_argument("--testing", default=False, action="store_fals
e", help="testing option")
parser.add_argument("-x", default=False, action="store_false", hel
p="short option")
parser.add_argument("--output-file", default='out.txt', help="outp
ut file name")

sample_options = '--condition=foo --testing --output-file abc.def
-x a1 a2'
args = parser.parse_args(sample_options.split())

print args
print args.condition, args.names
```

```
Namespace(condition='foo', names=['a1', 'a2'], output_file='abc.d
ef', testing=False, x=False)
foo ['a1', 'a2']
```

인수 처리 예제

다음 코드를 getpage.py 이름으로 저장한다.

import urllib import sys import argparse def getpage(url): f = urllib.urlopen(url) return f.read() if __name__ == '__main__': parser = argparse.ArgumentParser() parser.add_argument("urls", nargs='+', help="URL of web page") args = parser.parse_args() print(args) for url in args.urls: try: text = getpage(url) # print text except IOError: pass

콘솔에서 실행한다. 인수를 주지 않았을 경우의 예:

> python getpage.py getpage.py usage: getpage.py [-h] urls [urls ...] getpage.py: error: too few arguments

인수를 두 개 이상 입력했을 때:

> python getpage.py http://python.org http://sf.net getpage.py http://python.org http://sf.net Namespace(urls=['http://python.org', 'http://sf.net']) ...

In [37]:
```

```