



CHAP.4

컨테이너 객체



좀 쓸모 있는 객체 삼인방 데이터를 담는데 사용

```
>>>a=()
```

```
>>>b=[]
```

```
>>>c={}
```

`()`, `[]`, `{}`는 해당 객체를 생성

할당 연산자는 변수가 객체를 참조하도록 한다.

변수는 할당연산자의 오른쪽 객체를 참조한다.

```
>>>type(a)
```

```
<class 'tuple'>
```

```
>>>type(b)
```

```
<class 'list'>
```

```
>>>type(c)
```

```
<class 'dict'>
```


Photo by Milena Trifonova on Unsplash

튜플 - 정수 3개를 담아 보자. 그리고 꺼집어 내 보자.

```
>>>a=(1,2,3)
>>>a
(1,2,3)
>>>a[0]
1
>>>a[1]
2
>>>a[2]
3
>>>a[3]
IndexError: tuple index out of range
```

변수 a가 참조하는 객체는?

index

Index 범위는 0,1,2

변수 a의 인덱스2가 참조하는 객체는?

Index 범위를 벗어나면 에러

튜플 - 다양하게 꼬집어 낼 수 있다. slice 슬라이싱 한다고 말한다.

```
>>>a=(1,2,3,4,5)
```

```
>>>a[1:3]      변수 a에 대해,  
(2,3)         첨자 1:3 맞게 잘라내서  
              튜플 객체를 생성해라.
```

```
>>>a[0:5]  
(1,2,3,4,5)
```

```
>>>a[:]  
(1,2,3,4,5)
```

```
>>>a[:3]  
(1,2,3)
```

```
>>>a[2:]  
(3,4,5)
```

```
>>>a[0]  
1
```

```
>>>a[-1]  
5
```

```
>>>a[-2]  
4
```

```
>>>a[:-1]  
(1,2,3,4)
```

튜플 - 튜플은 값을 바꿀 수 없다.

```
>>>a=(1,2,3)
```

```
>>>a[0]=100
```

```
TypeError: 'tuple' object does not support item assignment
```

튜플 - 이것 저것 다 담을 수 있다.

```
>>>a=('1', '2', '3')
```

```
>>>a
```

```
('1', '2', '3')
```

```
>>>a=(1.0, 2.0, 3.0)
```

```
>>>a
```

```
(1.0, 2.0, 3.0)
```

```
>>>a=(1, '2', 3.0)
```

```
>>>a
```

```
(1, '2', 3.0)
```

튜플 - 튜플도 담을 수 있다.

```
>>>a=(1, 2, 3)
>>>b=(4, 5)
>>>c=(a, b, 7, 8, 9)
>>>c
((1, 2, 3), (4, 5), 7, 8, 9)
```


튜플 - 값 한 개만 넣으면 튜플 지위를 잃는다.

```
>>>a=(1, 2)
>>>a
(1, 2)
>>>a=(1)
>>>a
1
>>>a(1,)
>>>a
(1,)
```

```
>>>a=(1)
>>>type(a)
<class 'int'>
>>>a=(1,)
>>>type(a)
<class 'tupe'>
```

튜플 - 더하기는 할 수 있다

```
>>>a=(1, 2, 3)
>>>b=(4, 5)
>>>c=a+b
>>>c
(1, 2, 3, 4, 5)
```

튜플 - 튜플에 적용할 수 있는 또 다른 연산자

```
>>>a=(1, 2, 3)
>>>b=(3, 4)
>>>a>b          사전 순서로 따진다
False
>>>a==b
False
>>>a!=b
True
```

튜플 - 튜플에 적용할 수 있는 파이썬 명령 [], {}에도 사용

```
>>>a=(1, 2, 3)
```

```
>>>len(a)
```

```
3
```

```
>>>sum(a)
```

```
6
```

```
>>>max(a)
```

```
3
```

```
>>>min(a)
```

```
1
```

```
>>>a=(3, 2, 8, 9, 4, 5)
```

```
>>>min(a)
```

```
2
```

```
>>>max(a)
```

```
9
```

```
>>sorted(a)
```

```
[2,3,4,5,8,9]
```

```
>>>a
```

```
(3,2,8,9,4,5)
```

순서대로 나열해 준다.

결과는 '리스트'

a 자체가 바뀌는 것은 아니다.

튜플은 객체였다.

```
>>>a=()
>>>dir(a)

['__add__', '__class__', '__contains__', '__delattr__',
 '__dir__', '__doc__', '__eq__', '__format__', '__ge__',
 '__getattr__', '__getitem__', '__getnewargs__',
 '__gt__', '__hash__', '__init__', '__init_subclass__',
 '__iter__', '__le__', '__len__', '__lt__', '__mul__',
 '__ne__', '__new__', '__reduce__', '__reduce_ex__',
 '__repr__', '__rmul__', '__setattr__', '__sizeof__',
 '__str__', '__subclasshook__', 'count', 'index']
```

튜플 - 튜플이 가진 함수들

```
>>>a=(1,2)
>>>b=(3,4)
>>>c=a.__add__(b)
>>>c
(1,2,3,4)
>>>a.__str__()
'(1,2)'
```

```
>>>str(a)
'(1,2)'
```

```
>>>a=(1,2,2,2,3)
>>>a.count(2)
3
>>>a.index(1)
0
>>>a.index(3)
4
>>>a.index(2)
1
```

튜플 - for 하나씩 출력하기

```
a=(1, 2, 3)
for i in range(len(a)):
    print(a[i])
```

```
a=(1, 2, 3)
for item in a:
    print(item)
```

```
1
2
3
```

```
a=(1, 2, 3)
for i, item in enumerate(a):
    print(i, item)
```

```
0 1
1 2
2 3
```

튜플 - 이런 것도 있다.

[], {}에도 사용

```
>>>a=(1,2,3)
>>>it = iter(a)
>>>type(it)
<class 'tuple_iterator'>
>>>next(it)
1
>>>next(it)
2
>>>next(it)
3
>>>next(it)
StopIteration
```

it = a.__iter__()

나중에 배운다.

Item을 하나씩 끄집어 온다.

더 이상 끄집어 올 것이 없을 때

StopIteration이라는 이름을 가진 예외(객체)

나중에 배운다.

for의 재해석

```
>>>a=(1,2,3)
>>>for i in a:
...     print(i)
1
2
3
```

내부에서는 a에 iter()을 적용하여
Iterator 객체를 가지고 온 후
이 Iterator 객체에 next()를 적용하여
얻은 객체를 i에 대입한 것
StopIteration 예외(객체)가 발생할 때 까지 반복

```
for i in range(3):
```



이 녀석도 내부에는 Iterator 객체가 있다.

튜플 - in 도 논리 연산자다!

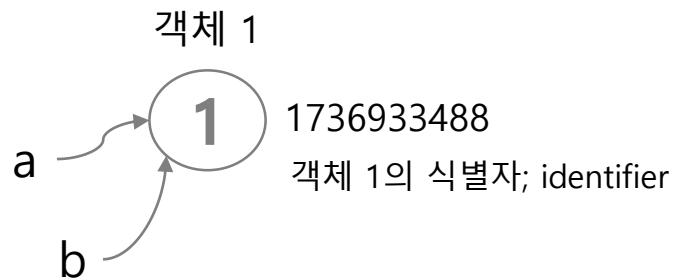
```
>>>a=(1,2,3)
>>>for i in a:
...     print(i, end=' ')
1 2 3
```

```
>>>1 in a
True
>>>4 in a
False
```

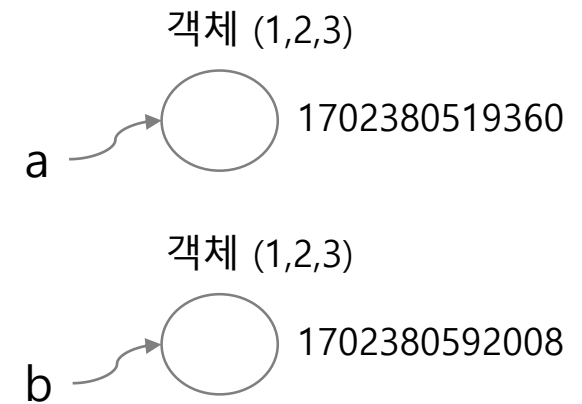
튜플 - is 다시 보기

list때 또 나온다.

```
>>>a=1
>>>b=1
>>>a == b
True
>>>a is b
True
```



```
>>>a=(1,2,3)
>>>b=(1,2,3)
>>>a == b
True
>>>a is b
False
```



문자열 - 은 튜플과 유사하다.

```
>>>a='1234'
```

```
>>>a[1]
```

```
'2'
```

```
>>>a[0]='9'
```

```
TypeError: 'str' object does not support item assignment
```

```
>>>b='56'
```

```
>>>a+b
```

```
'123456'
```

```
>>>len(a)
```

```
4
```

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

```
'123'
```

```
>>>a.index('3')
```

```
2
```

```
>>>a.count('3')
```

```
1
```

```
>>>'3' in a
```

```
True
```

```
>>>sorted(a)
```

```
['1', '2', '3', '4']
```


튜플 - 튜플을 만드는 방법

```
>>>a=(1, 2, 3)
>>>b=[4, 5, 6]      리스트
>>>c=tuple(b)
>>>c
(4, 5, 6)
```

```
>>>a=range(3)
>>>type(a)
<class 'range'>
>>>b=tuple(a)
>>>type(b)
<class 'tuple'>
```

Iterator 객체는
tuple로 변환할 수 있다.

Lab.

튜플 전화번호부

튜플 전화번호부 - 이름 입력하면 전화번호 알려준다.

```
names=('kim', 'lee', 'park')
phones=('111-1111', '222-2222', '888-8888')
name = input('Input your name:')
idx = names.index(name)
print("{0}'s phone number is {1}".format(name, phones[idx]))
```

```
names=('kim', 'lee', 'park')
phones=('111-1111', '222-2222', '888-8888')
name = input('Input your name:')
idx = names.index(name)
print("{0}'s phone number is {1}".format(name, phones[idx]))
```

Input your name: **kim**

kim's phone number is 111-1111

Input your name: **ko**

ValueError: tuple.index(x): x not in tuple


```
names=('kim', 'lee', 'park')
phones=('111-1111', '222-2222', '888-8888')
name = input('Input your name:')
if name in names:
    idx = names.index(name)
    print("{0}'s phone number is {1}".format(name, phones[idx]))
else:
    print("Not registered name")
```

이름을 대문자로 바꾸어 놓고,
이름을 소문자로 입력한다거나, 입력할 때 공백을 띄운다면 ?

```
names=('KIM', 'LEE', 'PARK')
phones=('111-1111', '222-2222', '888-8888')
name = input('input your name:')
if name in names:
    idx = names.index(name)
    print("{0}'s phone number is {1}".format(name, phones[idx]))
else:
    print("Not registered name")
```

문자열도 객체였다.

```
>>> s=""
>>> dir(s)
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__',
 '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__',
 '__getnewargs__', '__gt__', '__hash__', '__init__', '__init_subclass__',
 '__iter__', '__le__', '__len__', '__lt__', '__mod__', '__mul__', '__ne__',
 '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__',
 '__setattr__', '__sizeof__', '__str__', '__subclasshook__', 'capitalize',
 'casefold', 'center', 'count', 'encode', 'endswith', 'expandtabs', 'find',
 'format', 'format_map', 'index', 'isalnum', 'isalpha', 'isdecimal', 'isdigit',
 'isidentifier', 'islower', 'isnumeric', 'isprintable', 'isspace', 'istitle',
 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'maketrans', 'partition', 'replace',
 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines',
 'startswith', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill']
```


문자열 - 다양한 함수들

문자열 객체의 함수를 호출하면,
그 자리에 문자열 객체를 남긴다.

```
>>>a='hello, world'
>>>len(a)
12
>>>a.find('wo')
7
>>>a.split(',')
['hello', 'world']
>>>''.join(['hello', 'world'])
'helloworld'
```

```
>>>a='hello, world'
>>>a.strip()
'hello, world'
>>>a.upper()
'HELLO, WORLD'
>>>b=a.strip()
>>>b=b.upper()
'HELLO, WORLD'
>>>b=a.strip().upper()
```

다른 함수들은 인터넷을 참조. 'Python encode'

문자열 함수를 써 먹자.

```
names=('KIM', 'LEE', 'PARK')
phones=('111-1111', '222-2222', '888-8888')
name = input('input your name:')
name = name.strip().upper()
if name in names:
    idx = names.index(name)
    print("{0}'s phone number is {1}".format(name, phones[idx]))
else:
    print("Not registered name")
```

대문자로만 저장해 둔다.

공백을 벗겨내고, 대문자로 바꾼다.

첫 글자만 대문자로 출력되도록 했다.

Input name: kim

Kim's phone number is 111-1111

```
names=('KIM', 'LEE', 'PARK')
phones=('111-1111', '222-2222', '888-8888')
name = input('input name:')
name = name.strip().upper()
if name in names:
    idx = names.index(name)
    print("{0}{1}'s phone number is {2}".format(name[0], name[1:].lower(), phones[idx]))
else:
    print("Not registered name")
```

함수로 바꾸어 본다.

```
names=('KIM', 'LEE', 'PARK')
phones=('111-1111', '222-2222', '888-8888')
def search(name):
    if name in names:
        idx = names.index(name)
        return idx
    else:
        return None
def print_one(idx):
    name = names[idx]
    print("{0}{1}'s phone number is {2}".format(name[0], \
        name[1:].lower(), phones[idx]))
```

```
def print_all():  
    n = len(names)  
    for i in range(n):  
        print_one(i)  
name = input('Input you name: ')  
name = name.strip().upper()  
idx = search(name)  
if idx != None:  
    print_one(idx)  
else:  
    print('Not registered name')  
print_all()
```

클래스로 바꾸어 본다.

```
class PhoneBook:
    def __init__(self):
        self.names = ('KIM', 'LEE', 'PARK')
        self.phones = ('111-1111', '222-2222', '333-3333')

    def search(self, name):
        if name in self.names:
            idx = self.names.index(name)
            return idx
        else:
            return None
```

```
def print_one(self, idx):  
    name = self.names[idx]  
    print("{0}{1}'s phone number is {2}".format(name[0], \  
        name[1:].lower(), self.phones[idx]))
```

```
def print_all(self):  
    n = len(self.names)  
    for i in range(n):  
        self.print_one(i)
```

```
# PhoneBook 클래스 끝
```

```
name = input('Input you name: ')
name = name.strip().upper()

phoneBook = PhoneBook()

idx = phoneBook.search(name)
if idx != None:
    phoneBook.print_one(idx)
else:
    print('Not registered name')

phoneBook.print_all()
```


리스트



리스트 - 정수 3개를 담아 보자. 그리고 꺼집어 내 보자.

```
>>>a=[1, 2, 3]
```

```
>>>a
```

```
[1,2,3]
```

```
>>>a[0]
```

index

```
1
```

```
>>>a[1]
```

Index 범위는 0,1,2

```
2
```

```
>>>a[2]
```

```
3
```

```
>>>a[3]
```

```
IndexError: tuple index out of range
```

Index 범위를 벗어나면 에러

리스트 - 2차원, 축이 2개인 경우의 인덱싱

```
>>>a=[[1,2,3],[4,5,6]]
>>>len(a)
2
>>>len(a[0])
3
>>>a[0]
[1,2,3]
>>>a[1]
[4,5,6]
>>>a[1][2]
6
```

0번 axis에 값이 2개

1번 axis에 값이 3개

여기까진 ok~

```
>>>a[0]
[1,2,3]
>>>a[0][1] # [1,2,3][1]
2
>>>a[:]
[[1,2,3],[4,5,6]]
>>>a[:,0] a[:] ↔ a ↔ [[1,2,3],[4,5,6]]
[1,2,3]
>>>a[0][1:] a[0] ↔ [1,2,3]
[2,3]
```

리스트 - 값을 바꿀 수 있다.

```
>>>a=[1, 2, 3]
>>>a[0]=100
[100, 2, 3]
>>>a[0]=[10,20,30]
[[10,20,30], 3]
>>>del a[0] 삭제도 바꾸는 것의 일종
[2,3]
>>>del a
```

추가하기

```
>>>a=[2,3]
>>>a.append(1)
[2,3,1]
>>>a=[2,3]
>>>a.insert(0, 1)
[1,2,3]
```

리스트 - 리스트 생성하기

```
>>>a=list()      #a=[]  
>>>type(a)        객체생성 - 객체이름 뒤에()  
<class 'list'>  
>>>a=(1,2,3)  
>>>b=list(a)       튜플 (Iterator객체)을 리스트로  
[1,2,3]  
>>>b=list(range(3))  Iterator객체를 리스트로; 한 단계 더 나아가 Generator  
[0,1,2]              나중에 나온다.
```

리스트 - 또 다른 추가 방법

```
>>>a=[]
>>>a[0]=1
IndexError: list assignment index out of range

>>>a='1'+ '1'+ '1'
'111'
>>>a='1'*3   곱하기는 더하기 반복
'111'
>>>a=[-1]*3   미리 칸을 만들어 둔다; append 보다 효율적
[-1, -1, -1]   넣을 수 있는 값의 종류는 한가지로 지정
>>>a[0]=1
```

리스트도 객체였다.

```
>>> b=[]  
>>> dir(b)
```

```
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__', '__dir__',  
 '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__',  
 '__gt__', '__hash__', '__iadd__', '__imul__', '__init__', '__init_subclass__',  
 '__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__', '__new__',  
 '__reduce__', '__reduce_ex__', '__repr__', '__reversed__', '__rmul__',  
 '__setattr__', '__setitem__', '__sizeof__', '__str__', '__subclasshook__',  
 'append', 'clear', 'copy', 'count', 'extend', 'index', 'insert', 'pop',  
 'remove', 'reverse', 'sort']
```

리스트 - 꺼내고 지우기; 기본적으로 맨 뒤에 것

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

```
>>>a.pop()
```

```
5
```

맨 뒤에 것 꺼내고 삭제

```
>>>a
```

```
[1, 2, 3, 4]
```

```
>>>a.pop(3)
```

```
4
```

Index 3에 해당하는 것 꺼내고 삭제

```
>>>a.pop(-1)
```

```
3
```

마지막 index에 해당하는 것 꺼내고 삭제

```
>>>a
```

```
[1, 2]
```


리스트 - 찾아서 지우기

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

```
>>>a.remove(3)          값 3을 찾아서 지운다.
```

```
[1, 2, 4, 5]
```

```
>>>a.remove(9)          없는 값 지우라고 하면 에러
```

```
ValueError: list.remove(x): x not in list
```

```
>>>a.index(9)           없는 값 index 요구하면 에러
```

```
ValueError: list.remove(x): x not in list
```

```
>>>9 in a               들어 있는지 먼저 검사한 후 적용해야 한다.
```

```
False
```

리스트 - 정렬

```
>>>a=[3, 1, 5, 4, 3]
>>>a.sort()          a 자체가 바뀜
>>>a
[1, 3, 3, 4, 5]
>>>a.reverse()
[5, 4, 3, 2, 1]
>>>sorted(a)         a 자체는 안 바뀜; 동작 결과로 정렬된 리스트를 남김
[1, 3, 3, 4, 5]
>>>a.clear()
[]
```

리스트 - 확장

```
>>>a=[1, 2, 3]
>>>b=[4, 5]
>>>a.extend(b)
[1, 2, 3, 4, 5]
```

```
>>>a=[1, 2, 3]
>>>a = a + b
[1, 2, 3, 4, 5]
```

리스트 - 그냥 참조

```
>>>a=[1, 2, 3]
```

```
>>>b=a
```

```
>>>print(b)
```

```
[1, 2, 3]
```

```
>>>a[0]=10
```

```
print(b)
```

```
[10, 2, 3]
```

a도 바뀐다.

```
>>>id(a)
```

```
1702380736392
```

```
>>>id(b)
```

```
1702380736392
```

위 숫자 의미는 아래 그림



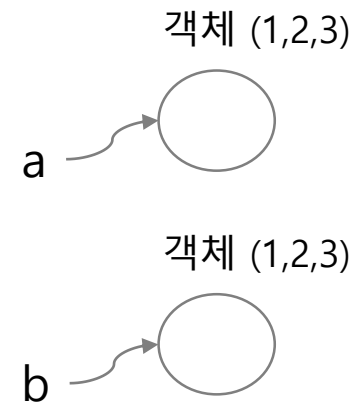
같은 객체를 가리키니까. 바뀐 것이다.

리스트 - (깊은) 복사

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

```
>>>a[0]=10
print(b)
[1, 2, 3]
```

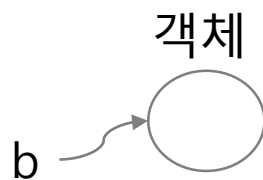
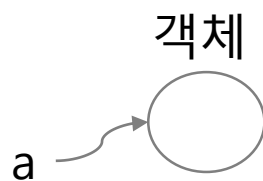
```
>>>id(a)
1702380736392
>>>id(b)
1702380717768
```





깊은 복사와 함수

기본 원칙은 간단하다.
원칙만 파악하고 있으면 쉽다.
그리고, 우리에게 멋진 도구가 있다.



함수 - 복습

```
def show():  
    print(a, "in show")  
a=1  
show()  
print(a)
```

```
1 in show  
1
```

```
def show():  
    a=2  
    print(a, "in show")  
a=1  
show()  
print(a)
```

새로운 a

```
2 in show  
1
```

함수 - 복습

```
def show():  
    a=a+1  
    print(a, "in show")  
  
a=1  
show()  
print(a)
```

UnboundLocalError: local
variable 'a' referenced before
assignment

```
def show():  
    global a  
    a=a+1  
    print(a, "in show")  
  
a=1  
show()  
print(a)
```

2 in show
2

함수 - 안 바뀐다.

```
def show(a):  
    a=2  
    print(a, 'in show')  
  
a=1  
show(a)  
print(a)
```

```
2 in show  
1
```

```
def show(a):  
    a=a+1  
    print(a, 'in show')  
  
a=1  
show(a)  
print(a)
```

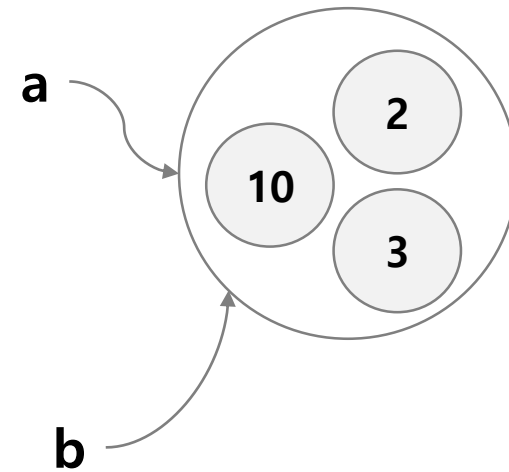
바깥쪽 a랑 상관없다.

```
2 in show  
1
```

리스트 - 함수 안에서 리스트 값은 바뀐다.

```
def show(b):  
    b[0]=10  
    print(b, 'in show')  
a=[1,2,3]  
show(a)  
print(a)
```

```
[10, 2, 3] in show  
[10, 2, 3]
```



리스트 - 함수 안에서 안 바뀌게 하고 싶다.

```
def show(a):  
    b = a.copy()  
    b[0]=10  
    print(b, 'in show')
```

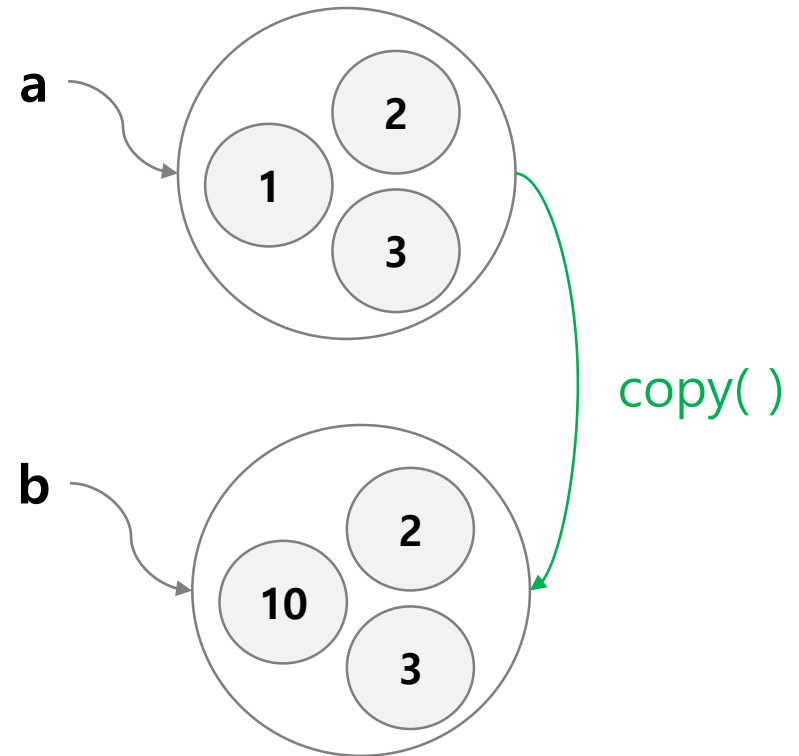
```
a=[1,2,3]
```

```
show(a)
```

```
print(a)
```

```
[10, 2, 3] in show
```

```
[1, 2, 3]
```



Lab.

리스트 전화번호부

리스트 전화번호부- 입력, 검색, 수정, 삭제가 기본

CRUD; Create, Read, Update, Delete

<pre>Names=[] Phones=[] def isEmpty(): pass def search(): pass def insert(): pass def delete(): pass def listAll(): pass</pre>	<pre>while True: print('\n-----') print('[S] Search') print('[N] Insert') print('[U] Update') print('[D] Delete') print('[A] List All') print('[Q] Quit') print('-----') menu = input('Menu: ')</pre>	<pre>if menu == 'S': search() elif menu == 'N': insert() elif menu == 'U': update() elif menu == 'D': delete() elif menu == 'A': listAll() elif menu == 'Q': break</pre>
--	---	--

프로그램 전체에서 사용되므로, 대문자로 시작했다.

```
Names=['KIM']
```

```
Phones=['111-1111']
```

메뉴 입력을 잘 못 했을 경우 고려

```
menuList = ['S', 'N', 'U', 'D', 'A', 'Q']
bFinish = False
while not bFinish:
    print('\n-----')
    print('[S] Search')
    print('[N] Insert')
    print('[U] Update')
    print('[D] Delete')
    print('[A] List All')
    print('[Q] Quit')
    print('-----')
```

```
menu = "    이걸 안해 주면,
while True:
    menu = input('Menu: ')
    menu = menu.strip().upper()
    print()
    if menu in menuList:
        break

if menu=='S':    여기 menu는
                 처음 보는 menu가
                 된다.
    search()
elif menu=='Q':
    bFinish = True    나머지 부분은 생략...
```

비었는지 검사

```
def isEmpty():  
    n = len(Names)  
    if n==0:  
        return True  
    else:  
        return False
```


모든 내용 출력

```
def listAll():  
    print('LIST')  
    print('-----')  
    bEmpty = isEmpty()  
    if bEmpty == True:  
        print('Phone Book is empty')  
        return  
    for i, name in enumerate(Names):  
        phone = Phones[i]  
        print("{0} {1}{2} {3}".format(i+1, name[0], name[1:].lower(), phone))  
  
print('Total: {} items'.format(len(Names)))
```

함수를 끝내는 용도로 사용했다.

검색 - 수정, 삭제에서 사용되니까 찾은 인덱스를 남긴다. 못 찾으면 None을 남긴다.

```
def search():
    print('SEARCH')
    name = input('Input the name: ')
    name = name.strip().upper()
    if name in Names:
        idx = Names.index(name)
        phone = Phones[idx]
        print("{0}{1}'s phone number is {2}".format(name[0], name[1:].lower(), phone))
        return idx
    else:
        print("Not Found")
        return None
```

동일 이름을 가진 사람이 여러 명이면?

신규 입력

```
def insert():  
    print('INSERT')  
    name = input('Input new name: ')  
    name = name.strip().upper()  
    phone = input('input the phone number: ')  
    phone = phone.strip().upper()  
  
    Names.append(name)  
    Phones.append(phone)  
  
    print("Updated")
```

수정 – 검색 결과 활용

```
def update():  
    print('UPDATE')  
    idx = search()  
    if idx == None:  
        return  
    else:  
        name = input('Input a name: '); name = name.strip().upper()  
        phone = input('input the phone number: ')  
        phone = phone.strip().upper()  
        Names[idx] = name  
        Phones[idx] = phone  
        print("Updated")
```

삭제 – 검색 결과 활용

```
def delete():  
    print('DELETE')  
    idx = search()  
    if idx == None:  
        return  
    else:  
        name = Names[idx]  
        del Names[idx]  
        del Phones[idx]  
        print("{0}{1} deleted".format(name[0], name[1:].lower()))
```

객체로 바꿀 수 있어야 한다.

변수를 공유하는 함수가 여럿 있으니까.

```
Names=[]  
Phones=[]  
def isEmpty()  
def search()  
def insert()  
def update()  
def delete()  
def listAll()
```



```
class
```

객체로 바꿀 수 있어야 한다.

변수를 공유하는 함수가 여럿 있으니까.

```
Names=[]  
Phones=[]  
def isEmpty()  
def search()  
def insert()  
def update()  
def delete()  
def listAll()
```



```
class PhoneBook():  
    def __init__(self):  
        self.Names=[]  
        self.Phones=[]  
    def isEmpty(self):  
    def search(self):  
    def insert(self):  
    def update(self):  
    def delete(self):  
    def listAll(self):
```


Lab.

행렬전치




```
a=[[1,1,1],[2,2,2]]
```

```
b=[[3,3,3],[4,4,4]]
```

```
def print_mat(m, name):
```

```
    rows = len(m)
```

```
    cols = len(m[0])
```

```
    print(name, '=')
```

```
    for j in range(rows):
```

```
        for i in range(cols):
```

```
            print('{:3d}'.format(m[j][i]), end="")
```

```
        print()
```

```
print_mat(a, 'a')
```

```
print_mat(b, 'b')
```

```
def add_mat(a, b):
```

```
    rows = len(a)
```

```
    cols = len(a[0])
```

```
    print('add')
```

```
    c=[]
```

```
    #c=[[0]*cols]*rows
```

```
    #c=a.copy()
```

```
    for j in range(rows):
```

```
        c.append([])
```

```
        for i in range(cols):
```

```
            c[j].append(a[j][i] + b[j][i])
```

```
    return c
```

```
c = add_mat(a, b)
```

```
print_mat(c, 'c')
```

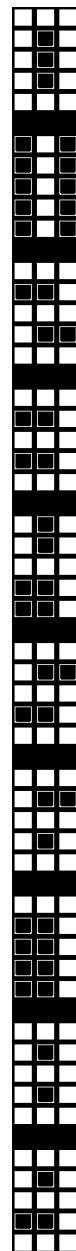
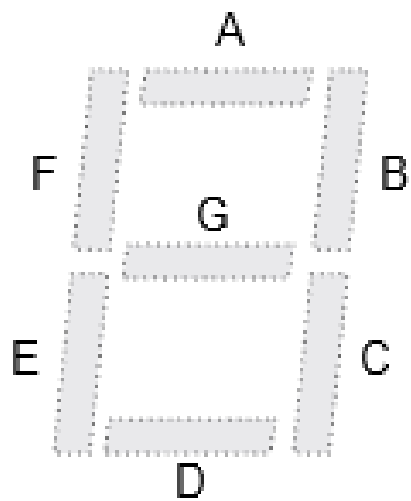
```
def trans_mat(a):  
    rows = len(a)  
    cols = len(a[0])  
    c=[]  
    for i in range(cols):  
        c.append([])  
        for j in range(rows):  
            c[i].append(a[j][i])  
    return c
```

```
print_mat(a, 'a')  
c = trans_mat(a)  
print_mat(c, 'c')
```

Lab.

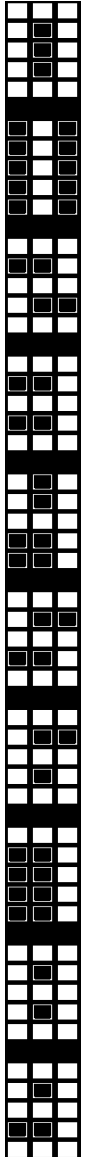
숫자전광판





```
d1=[
['□', '■', '□'],
['□', '■', '□'],
['□', '■', '□'],
['□', '■', '□'],
['□', '■', '□']]
```

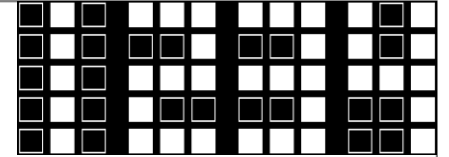
<pre> d1=[['□','■','□'], ['□','■','□'], ['□','■','□'], ['□','■','□'], ['□','■','□']] for j in range(5): for i in range(3): print(d1[j][i], end="") print() </pre>	<pre> d0=[[1,1,1], [1,0,1], [1,0,1], [1,0,1], [1,1,1]] d1=[[0,1,0], [0,1,0], [0,1,0], [0,1,0], [0,1,0]] </pre>	<pre> d2=[[1,1,1], [0,0,1], [1,1,1], [1,0,0], [1,1,1]] d3=[[1,1,1], [0,0,1], [1,1,1], [0,0,1], [1,1,1]] </pre>	<pre> d4=[[1,0,1], [1,0,1], [1,1,1], [0,0,1], [0,0,1]] d5=[[1,1,1], [1,0,0], [1,1,1], [0,0,1], [1,1,1]] </pre>	<pre> d6=[[1,1,1], [1,0,0], [1,1,1], [1,0,1], [1,1,1]] d7=[[1,1,1], [0,0,1], [0,0,1], [0,0,1], [0,0,1]] </pre>	<pre> d8=[[1,1,1], [1,0,1], [1,1,1], [1,0,1], [1,1,1]] d9=[[1,1,1], [1,0,1], [1,1,1], [0,0,1], [1,1,1]] </pre>	<pre> digit_pattern = [d0, d1, d2, d3, d4, d5, d6, d7, d8, d9] </pre>
--	--	--	--	--	--	---



```
def show(d):
    for j in range(5):
        for i in range(3):
            if d[j][i]==1:
                c='■'
            else:
                c='□'
            print(c, end="")
        print()
    print()

for i in range(10):
    show(digit_pattern[i])
```

```
def display(digits):
    for j in range(5):
        n = len(digits)
        for k in range(n):
            d = int(digits[k])
            for i in range(3):
                if digit_pattern[d][j][i]==1:
                    c = '■'
                else:
                    c = '□'
                print(c, end="")
            print(' ', end="")
        print()
display('1234')
```

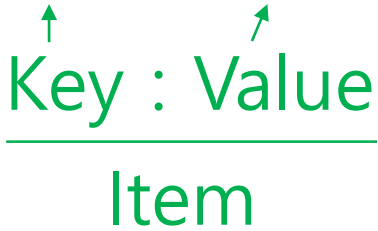


딕셔너리

Photo by Naassom Azevedo on Unsplash

딕셔너리- 데이터 쌍; Key → Value를 다룬다.

```
>>>a=dict()
>>>a={}
>>>a={'kim':'111-1111', 'lee':'222-2222', 'park':'333-3333'}
>>>a['kim']    Index로 Key를 사용
'111-1111'     대응되는 Value 남긴다.
>>>a[1]        없는 Key를 넣으면 에러
KeyError: 1
>>>a.keys()    어떤 키가 있는지 알려준다.
dict_key(['kim', 'lee', 'park'])
```



딕셔너리- 어떤 Key가 있나

```
>>>a.keys()
dict_key(['kim', 'lee', 'park']) 어떤 키가 있는지 알려준다.
>>>for key in a.keys():           Iterator란 얘기
...     print(key, end=' ')
kim lee park
>>>a.keys().__iter__().__next__() # next(iter(a.keys()))
'kim'
>>>keys = list(a.keys())          list로 바로 변환해 볼 수 있다.
['kim', 'lee', 'park']           데이터가 큰 경우, 오래 걸릴 수 있다.
```

```
dir(a.keys())
```

```
[ '__iter__', ...]
```

딕셔너리- 어떤 Value가 있나

```
>>>a.values()
dict_values(['111-1111', '222-2222', '333-3333'])
>>>for val in a.values():
...     print(val, end=' ')
111-1111 222-2222 333-3333

>>>values = list(a.values())
['111-1111', '222-2222', '333-3333']
```

list로 바로 변환해 볼 수 있다.

딕셔너리- Key-Value 조합인 Item

```
>>>a.items()
dict_items([('kim', '111-1111'), ('lee', '222-2222'), ('park', '333-3333')])
>>>for item in a.items():
...     print(item)
('kim', '111-1111')
('lee', '222-2222')
('park', '333-3333')
>>>k, v = ('kim', '111-1111') # (k, v) = ('kim', '111-1111')
>>>k
'kim'
>>>v
'111-1111'
```

```
>>>for key, value in a.items():
```

```
...     print(key, value)
```

```
kim 111-1111
```

```
lee 222-2222
```

```
park 333-3333
```

```
>>>for key, value in a:  참고로 이것은 에러
```

```
...     print(key, value)
```

```
ValueError: too many values to unpack (expected 2)
```

딕셔너리- 동일 키가 들어 있으면

```
>>>a={1:100, 3:300, 5:500, 3:400}  
{1:100, 3:400, 5:500}
```

나중 것만 남긴다.
키는 유일 해야 한다.

딕셔너리- 수정, 추가, 삭제

Key가 숫자인 경우임에 유의

```
>>>a={1:100, 3:300, 5:500}
>>>a[1]=150                있는 Key 넣으면 수정
{1:150, 3:300, 5:500}
>>>a[4]=400                없는 Key 넣으면 추가
>>>a[2]=200
{1: 100, 3: 300, 5: 500, 4: 400, 2: 200}
>>>del a[4]                Key 4를 찾아서 있으면 삭제한다.
{1: 100, 3: 300, 5: 500, 2: 200}
>>>del a[4]
KeyError: 4                Key 4가 없으면 에러 발생; 대체 방법은 뒤에서 설명
```

딕셔너리도 객체였다.

```
>>>c={}
>>>dir(c)
```

```
['__class__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc__',
 '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__gt__',
 '__hash__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__',
 '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__setattr__',
 '__setitem__', '__sizeof__', '__str__', '__subclasshook__', 'clear', 'copy',
 'fromkeys', 'get', 'items', 'keys', 'pop', 'popitem', 'setdefault', 'update',
 'values']
```

딕셔너리- 꺼내고 지우기; pop, popitem

```
>>>a={'1':100, '3':300, '5':500}
```

```
>>>a.pop()
```

TypeError: pop expected at least 1 arguments, got 0 Key를 넣어야 한다.

```
>>>a.pop('2') 없는 Key를 넣으면 에러 발생
```

```
KeyError '2'
```

```
>>>a.pop('3')
```

```
300
```

```
>>>a
```

```
{'1':100, '5':500} 꺼내고 지운다.
```

list에서는 맨 마지막 것 꺼내 왔었다.

Pop이라는 것은 원래 맨 마지막 것 꺼내는 것
popitem이 이 역할을 하고 있다.

```
>>>a.popitem()  
( '5':500)
```

맨 끝에 아이템 꺼내고 지운다.

딕셔너리- Key가 있는지 검사

```
>>>a={'1':100, '3':300, '5':500}
```

```
>>>'1' in a.keys()
```

```
True
```

```
>>>'1' in a    그냥 딕셔너리를 사용할 수 있다.
```

```
True
```

```
>>>'2' in a
```

```
False
```

딕셔너리- Key 있는지 검사하지 않고 그냥 꺼내 오기; get

```
>>>a={'1':100, '3':300, '5':500}
```

```
>>>a.get('5')
```

```
500
```

```
>>>a
```

```
{'1':100, '5':500}
```

```
>>>a.get('2') == None    없으면 None을 남긴다. 에러가 나지 않아 좋다.
```

```
True
```

```
>>>a.get('2', '1234')    없으면 뒤에 설정한 값을 남긴다. 유용할 듯.
```

```
'1234'
```

Lab.

딕셔너리 전화번호부

딕셔너리 전화번호부

```
PhoneBook={'KIM':'111-1111'}
```

```
def isEmpty():  
    n = len(PhoneBook)  
    if n==0:  
        return True  
    else:  
        return False
```

```
def search():
    print('SEARCH')
    name = input('Input the name: ')
    name = name.strip().upper()

    phone = PhoneBook.get(name)    검색해서 없으면 None을 남겨준다.
    if phone != None:
        print("{0}{1}'s phone number is {2}".format(name[0], name[1:].lower(), phone))
        return name
    else:
        print("Not found")
        return None
```



```
def insert():  
    print('INSERT')  
    name = input('Input new name: ')  
    name = name.strip().upper()  
    phone = input('input the phone number: ')  
    phone = phone.strip().upper()  
  
    PhoneBook[name] = phone  
    print("updated")
```

```
def update():  
    print('UPDATE')  
    name = search()  
    if name == None:  
        return      #return None  
    else:  
        #name = input('Input a name: ')  
        #name = name.strip().upper()  
        phone = input('input the phone number: ')  
        phone = phone.strip().upper()  
        PhoneBook[name] = phone  
        print("updated")
```

```
def delete():  
    print('DELETE')  
    name = search()  
  
    if name == None:  
        return  
    else:  
        PhoneBook.pop(name)  
        print("{0}{1} deleted".format(name[0], name[1:].lower()))
```



```
def listAll():
    print('LIST')
    print('-----')
    bEmpty = isEmpty()
    if bEmpty == True:
        print('Phone Book is empty')
        return
    i=0
    for name, phone in PhoneBook.items():
        i += 1
        print("{0} {1}{2} {3}".format(i, name[0], name[1:].lower(), phone))
    print('Total: {} items'.format(len(PhoneBook)))
```

```
bFinish = False
while not bFinish:
    print()
    print('-----')
    print('[S] Search')
    print('[N] Insert')
    print('[U] Update')
    print('[D] Delete')
    print('[A] List All')
    print('[Q] Quit')
    print('-----')
    menuList=['S','N','U','D','A','Q']
    menu = "
```

```
while True:
    menu = input('Menu: ')
    menu = menu.strip()
    menu = menu.upper()
    print()

    if menu in menuList:
        break

    if menu == 'Q':
        print('Bye')
        bFinish = True
```

```
elif menu == 'S':
    search()
elif menu == 'N':
    insert()
elif menu == 'U':
    update()
elif menu == 'D':
    delete()
elif menu == 'A':
    listAll()
```

딕셔너리 전화번호부-클래스로 구현

```
class PhoneBook():  
    def __init__(self):  
        self.phoneBook={'KIM':'111-1111'}  
  
    def isEmpty(self):  
        n = len(self.phoneBook)  
        if n==0:  
            return True  
        else:  
            return False
```

```
def search(self):  
    print('SEARCH')  
    name = input('Input the name: ')  
    name = name.strip().upper()  
  
    phone = self.phoneBook.get(name)  
    if phone != None:  
        print("{0}{1}'s phone number is {2}".format(name[0], \br/>                                                    name[1:].lower(), phone))  
  
        return name  
    else:  
        print("Not found")  
        return None
```

```
def insert(self):  
    print('INSERT')  
    name = input('Input new name: ')  
    name = name.strip().upper()  
    phone = input('input the phone number: ')  
    phone = phone.strip().upper()  
  
    self.phoneBook[name] = phone  
    print("updated")
```

```
def update(self):  
    print('UPDATE')  
    name = self.search()  
    if name == None:  
        return  
    else:  
        #name = input('Input a name: ')  
        #name = name.strip().upper()  이름은 바꾸지 못하게 하자.  
        phone = input('input the phone number: ')  
        phone = phone.strip().upper()  
        self.phoneBook[name] = phone  
        print("updated")
```

```
def delete(self):  
    print('DELETE')  
    name = self.search()  
  
    if name == None:  
        return  
    else:  
        self.phoneBook.pop(name)  
        print("{0}{1} is deleted".format(name[0], name[1:].lower()))
```

```
def listAll(self):
    print('LIST')
    print('-----')
    bEmpty = self.isEmpty()
    if bEmpty == True:
        print('Phone Book is empty')
        return
    i=0
    for name, phone in self.phoneBook.items():
        i += 1
        print("{0} {1}{2} {3}".format(i, name[0], name[1:].lower(), phone))
    print('Total: {} items'.format(len(self.phoneBook)))
```



```

def Main():
    myPhoneBook= PhoneBook()
    bFinish = False
    while not bFinish:
        print()
        print('-----')
        print('[S] Search')
        print('[N] Insert')
        print('[U] Update')
        print('[D] Delete')
        print('[A] List All')
        print('[Q] Quit')
        print('-----')
        menuList=['S','N','U','D','A','Q']

        menu = "

```

```

while True:
    menu = input('Menu: ')
    menu = menu.strip().
    menu = menu.upper()
    print()

    if menu in menuList:
        break

    if menu == 'Q':
        print('Bye')
        bFinish = True

```

```

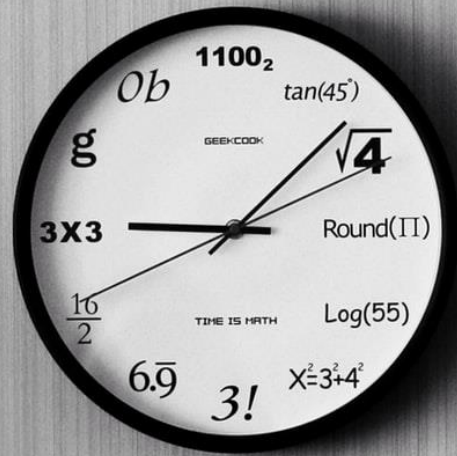
elif menu == 'S':
    myPhoneBook.search()
elif menu == 'N':
    myPhoneBook.insert()
elif menu == 'U':
    myPhoneBook.update()
elif menu == 'D':
    myPhoneBook.delete()
elif menu == 'A':
    myPhoneBook.listAll()

del myPhoneBook

Main()

```

집합



집합- 동일한 원소는 있을 수 없다, 순서도 없다.

```
>>>a=set()
>>>a=set((1,2,2,3))
>>>a=set([1,2,2,3])
{1, 2, 3}
>>>a='Hello, World'
>>>a=list(a)
['H', 'e', 'l', 'l', 'o', ',', ' ', 'W', 'o', 'r', 'l', 'd'] 12개
>>>a=set(a)
{' ', 'H', 'o', 'd', 'r', 'W', 'e', ',', 'l'} 9개, l 2, O 1개 제거
>>>a=set('Hello, World')
```

집합- 원소 꼬집어 오기; 순서가 없어 index가 없다.

```
>>>a=set([1,2,3])
```

```
>>>a[0]
```

```
TypeError: 'set' object does not support indexing
```

```
>>>b=list(a)
```

```
[1,2,3]
```

집합- 교집합, 합집합, 차집합

```
>>>a=set([1,2,3])
```

```
>>>b=set([3,4,5])
```

```
>>>a & b
```

```
{3}
```

```
>>>a | b
```

```
{1,2,3,4,5}
```

```
>>>a - b
```

```
{1, 2}
```

집합도 객체

```
>>>c=set()
```

```
>>>dir(c)
```

```
['__and__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__',  
 '__eq__', '__format__', '__ge__', '__getattribute__', '__gt__', '__hash__',  
 '__iand__', '__init__', '__init_subclass__', '__ior__', '__isub__', '__iter__',  
 '__ixor__', '__le__', '__len__', '__lt__', '__ne__', '__new__', '__or__', '__rand__',  
 '__reduce__', '__reduce_ex__', '__repr__', '__ror__', '__rsub__', '__rxor__',  
 '__setattr__', '__sizeof__', '__str__', '__sub__', '__subclasshook__', '__xor__',  
 'add', 'clear', 'copy', 'difference', 'difference_update', 'discard',  
 'intersection', 'intersection_update', 'isdisjoint', 'issubset', 'issuperset',  
 'pop', 'remove', 'symmetric_difference', 'symmetric_difference_update',  
 'union', 'update']
```

집합- add, update, pop

```
>>>a=set([1])
>>>a.add(2)
{1,2}
>>>a.add(2)    1개 추가
{1,2}
>>>a.update([2,3])  여러 개 추가
{1,2,3}
>>>a.pop()      Index 줄 수 없다.
1              맨 앞에 것 꺼내 온다.
```

```
>>>a.pop()
2
>>>a.pop()
3
>>>a.pop()
KeyError: 'pop from an empty set'
>>>len(a)      길이를 체크하면 비어
0              있는지 알 수 있다.
```

집합- remove, discard

```
>>>a=set([1,2,3])
>>>a.remove(2)
{1,3}
>>>a.remove(5) 없는 것 제거하려면 에러
KeyError: 5
>>>5 in a 있는지 검사
False
```

```
>>>a=set([1,2,3])
>>>a.discard(2)
{1, 3}
>>>a.discard(5) == None
True
없는 것 제거하려면 None을 남긴다.
```