Python 03

김은진

Data Structure

- 여러 개의 값을 한꺼번에 저장

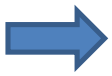
- str (string)
- list
- tuple
- dict (dictionary)
- set

list

list

- 생활 속의 리스트

팀별	멤버 이름	선곡 제목	아티스트 (원곡)
호피플라	아일, 김영소, 하현상, 최영진	《Hoppiolla》	시규어 로스
애프터문	케빈오, 디폴, 이종훈, 최영진	《Time After Time》	신디 로퍼
루시	이주혁, 신광일, 신예찬, 조원상	《Cry Bird》	테니슨
모네	자이로, 김우성, 벤지, 홍이삭, 황민재	《가져가》	자작곡
퍼플레인	양지완, 김하진, 이나우, 정광현, 채보훈	《Dream On》	에어로스미스
피플 온 더 브릿지	이찬솔, 강경윤, 김준협, 김형우, 임형빈	《Best Of You》	푸 파이터스



['호피폴라', '애프터문', "루시", "모네", '퍼플레인', "피플 온 더 브릿지"]

list 만들기

```
>>> band_list = [ ]
>>> new_band = [ '호피폴라', '애프터문', "루시", "모네", '퍼플레인', "피플 온 더 브릿지" ]
>>> print( band_list )
[]
>>> print( new_band )
['호피폴라', '애프터문', '루시', '모네', '퍼플레인', '피플 온 더 브릿지']
```

빈 list 만들기

list 만들기: [값1, 값2, 값3, 값4]

빈 list

list 원소(item)

- 원소(item)는 어떤 type이나 가능

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

정수 리스트

```
>>> num_str_list = [ 1, 2, 'Havana','No limit', 5 ]
```

정수와 문자열 리스트

```
>>> list_list = [ 1, 2, ['Havana', 3] ]
```

원소가 list, 총 원소: 3개

list item 접근: Indexing

```
>>> bands = [
  '호피폴라', '애프터문', '루시', '모네', '퍼플레인', '피플 온 더 브릿지']
```

Index: 0 1 2 3 4 5

index 는 0 부터

```
>>> bands[ 2 ]      # list이름[Index]
'루시'

>>> print( bands[2] )
루시
```

TIP: print 함수는 문자열 ' ' 없애고 출력

list item 접근: Indexing

- list의 원소가 list

```
>>> list_list = [ 1, 2, ['Havana', 3] ]
          Index:    0 1     2
>>> list_list[2]
['Havana', 3]
Index:    0        1
>>> list_list[2][1]
3
```

음수 index

```
>>> bands = [
 '호피폴라', '애프터문', '루시', '모네', '퍼플레인', '피플 온 더 브릿지']
```

0 1 2 3 4 5

index 는 0 부터

-6 -5 -4 -3 -2 -1

음수 index 는 -1 부터

```
>>> bands[-2]
>>> bands[-4]
>>> bands[-7]
```

list item 삭제, list 삭제

```
>>> num_str_list = [ 1,  2, 'Havana', 'No limit',  5 ]

>>> del num_str_list[2]
>>> num_str_list
[1, 2, 'No limit', 5]

>>> del num_str_list
>>> num_str_list
```

list의 일부 삭제

```
>>> num_str_list = [ 1, 2, 'Havana', 'No limit', 5 ]

>>> num_str_list[1:4] = [ ]
```

list의 일부에 접근 → slicing

list slicing

```
>>> bands = [
'호피폴라', '애프터문', '루시', '모네', '퍼플레인', '피플 온 더 브릿지']
```

리스트[시작 :종료: 간격]

시작: index (포함)
종료: index (미포함)
간격: 개수

```
>>> bands [ 0 : 4 : 1 ]
>>> bands [    : 4 : 1 ]
>>> bands [    : 4 ]

>>> bands [ 2:  : 2 ]
```

시작이 0이면 생략가능

간격이 1이면 생략가능

마지막 원소까지 이면 종료 생략가능

Slicing 예

```
>>> bands = [
 '호피폴라', '애프터문', '루시', '모네', '퍼플레인', '피플 온 더 브릿지']
```

```
>>> bands[0:5:2]
>>> bands[:5:2]
>>> bands[:6:2]
>>> bands[::2]
```

시작이 0이면 생략가능

종료가 index 보다 크면 마지막 원소까지 고려

시작, 종료 생략가능

Slicing 예

```
>>> bands = [
'호피폴라', '애프터문', '루시', '모네', '퍼플레인', '피플 온 더 브릿지']
```

```
>>> bands[0:6:3]
>>> bands[:6:3]
>>> bands[::3]

>>> bands[::]
>>> bands[:]
```

결과는?

Slicing 후 list 상태

```
>>> bands = [
 '호피폴라', '애프터문', '루시', '모네', '퍼플레인', '피플 온 더 브릿지']


>>> bands[::3]
['호피폴라', '모네']
>>> bands
['호피폴라', '애프터문', '루시', '모네', '퍼플레인', '피플 온 더 브릿지']
>>> x = bands[::3]
>>> x


>>> bands
```

Slicing 해도 원 list는 그대로

Slicing 한 결과는 새로운 list

slicing으로 동일한 list 하나 더 만들기

```
>>> bands = [
  '호피폴라', '애프터문', '루시', '모네', '퍼플레인', '피플 온 더 브릿지']
```

```
>>> bands[0:6:1]
>>> bands[:6:1]   시작 생략
>>> bands[::1]    종료 생략
>>> bands[::]     간격 생략
>>> bands[:]    두번째 :  생략 (간격 생략할 때)
```

list 할당, list slicing

```
>>> bands = [
  '호피폴라', '애프터문', '루시', '모네', '퍼플레인', '피플 온 더 브릿지']

>>> b_list = bands
>>> c_list = bands[:]
>>> del bands[1]

>>> print(b_list); print(c_list); print(bands)
['호피폴라', '애프터문', '루시', '모네', '퍼플레인', '피플 온 더 브릿지']
['호피폴라', '애프터문', '루시', '모네', '퍼플레인', '피플 온 더 브릿지']
['호피폴라', '애프터문', '루시', '모네', '퍼플레인', '피플 온 더 브릿지']
```

list 할당, list slicing

```
>>> bands = [
  '호피폴라', '애프터문', '루시', '모네', '퍼플레인', '피플 온 더 브릿지']
```

```
>>> b_list = bands
```

할당 : bands 주소 복사

bands

['호피폴라', '애프터문', '루시', '모네', '퍼플레인', '피플 온 더 브릿지']

b_list

```
>>> c_list = bands[:]
```

slicing: 새로운 리스트 생성
(할당 = 의 오른쪽)!!!

c_list

['호피폴라', '애프터문', '루시', '모네', '퍼플레인', '피플 온 더 브릿지']

list 할당, list slicing

```
>>> bands is b_list
True
```

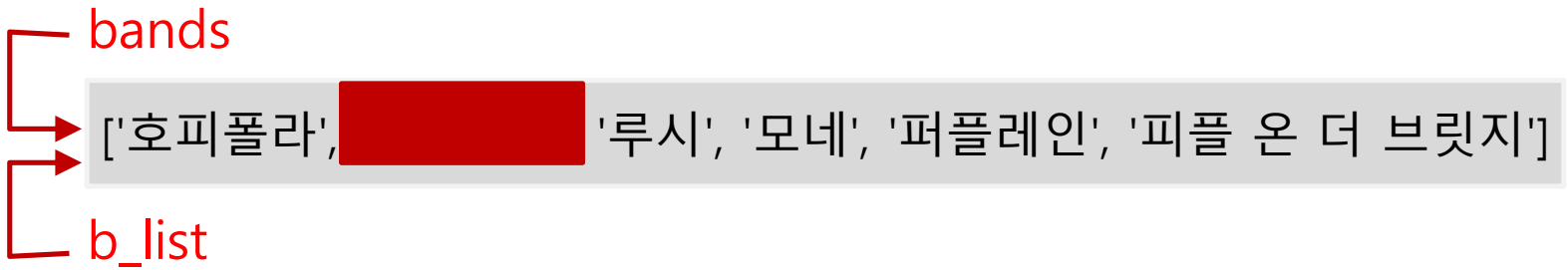
band와 b_list의 주소가 같음: 동일한 list

```
>>> bands is c_list
False
```

band와 c_list의 주소가 다름

```
>>> del bands[1]
```

band에서 삭제 → b_list에서도



list item 값 수정

```
>>> s_bands = ['호피폴라', '애프터문', '루시']
>>> id(s_bands)

>>> s_bands[0] = '루시'
>>> s_bands[1] = 100
>>> s_bands[2] = '호피폴라'
>>> s_bands;  id(s_bands)

>>> s_bands = ['호피폴라', '애프터문', '루시']
>>> id(s_bands)

>>> s_bands [::2] = ['루시', '호피폴라']
>>> s_bands; id(s_bands)
```

Item 별 수정

비교해 보기!

비교해 보기!

Slicing 으로 수정

: slicing을 할당 = 의 왼쪽에 사용

→ list의 해당 item만 골라내서 수정

list item 값 수정

```
>>> s_bands = ['호피폴라', '애프터문', '루시'] ; id( s_bands )

>>> s_bands [:] = ['루시', 100, '호피폴라']
>>> s_bands

>>> id( s_bands )
```

비교해 보기!

```
>>> s_bands = ['호피폴라', '애프터문', '루시'] ; id( s_bands )

>>> s_bands  = ['루시', 100, '호피폴라']
>>> s_bands

>>> id( s_bands )
```

새로운 list 만들어 변수 이름 할당 비교해 보기!

Item 추가

- + 연산자 list에 사용하기

```
>>> new_list = [ '루시', '호피폴라', '애프터문', '루시' ]
>>> new_list = new_list + [ '모네' ]
>>> new_list
```

llist는 list 하고만 + 할 수 있음

- append( ) 함수로 item 추가

```
>>> new_list.append(  '퍼플레인'  )
>>> new_list
```

Item 값 : 끝에 추가함

함수 호출(function call)
: 작업대상 함수호출 방법

```
>>> new_list . append (  '퍼플레인'  )
```

작업대상.함수명(인자값)

작업대상에 인자값을 (함수명)하세요
new_list에 '퍼플레인' 을 추가하세요

함수 호출(function call)
: built-in 함수호출 방법

- Built-in 함수

Built-in Functions				
abs()	dict()	help()	min()	setattr()
all()	dir()	hex()	next()	slice()
any()	divmod()	id()	object()	sorted()
ascii()	enumerate()	input()	oct()	staticmethod()
bin()	eval()	int()	open()	str()
bool()	exec()	isinstance()	ord()	sum()
bytearray()	filter()	issubclass()	pow()	super()
bytes()	float()	iter()	print()	tuple()
callable()	format()	len()	property()	type()
chr()	frozenset()	list()	range()	vars()
classmethod()	getattr()	locals()	repr()	zip()
compile()	globals()	map()	reversed()	__import__()
complex()	hasattr()	max()	round()	
delattr()	hash()	memoryview()	set()	

함수 호출(function call)
: built-in 함수호출 방법

```
>>> print ( 'Hello world' )
```

함수명(인자값)

'Hello world'을 출력하세요

함수 호출(function call) 방법

작업대상 함수 호출 법

작업대상.함수명(인자값)

특정 '**작업대상**'만 실행 경우

Built-in 함수 호출 법

함수명(인자값)

범용적인 작업인 경우

list 늘리기: * 연산자 / list함수-extend()

```
>>> nums = [ 1, 2, 3 ]
>>> new_nums = nums * 2
>>> new_nums
[1, 2, 3, 1, 2, 3]

>>> numbers = [ 4, 5, 6 ]
>>> new_numbers = numbers * 3
>>> new_numbers
[4, 5, 6, 4, 5, 6, 4, 5, 6]

>>> n = [ 3, 5 ]  ;  n2 = [ 100, 200 ]
>>> n.extend( n2 )
>>> n
[3, 5, 100, 200]
>>> n2
[100, 200]
```

list * 자연수

n2 : 인자값은 list 이어야 함

list 로 2차원 데이터(table) 만들기

- list 로 성적 table 만들기

이름	국어	수학
홍길동	70	80
김돌쇠	50	50
강철수	90	100

list 로 성적 table 만들기

- 행 list (row list) 만들기

```
>>> l1 = [ '이름', '국어', '수학' ]
>>> l2 = [ '홍길동', 70, 80 ]
>>> l3 = [ '김돌쇠', 50, 50 ]
>>> l4 = [ '강철수', 90, 100 ]
```

- 모든 행 list를 잇는 list 만들기

```
>>> s = [ l1, l2, l3, l4 ]
>>> print( s )
```

list 로 성적 table 만들기

이름	국어	수학
홍길동	70	80
김돌쇠	50	50
강철수	90	100

```
>>> print( s )
[
  ['이름', '국어', '수학'],
  ['홍길동', 70, 80],
  ['김돌쇠', 50, 50],
  ['강철수', 90, 100]
]
```

list들을 연결하는 2차원 list

table 접근

김돌쇠의 행 list 출력

```
>>> print( s[2] )
```

강철수의 수학점수 item 출력

```
>>> print( s[3][2] )
```

이름	국어	수학
홍길동	70	80
김돌쇠	50	50
강철수	90	100

list 함수– pop( ) : item 제거

pop (index) 도 가능!!!

```
>>> bands = [
  '호피폴라', '애프터문', '루시', '모네', '퍼플레인', '피플 온 더 브릿지']
>>> bands2 = bands[ : ]
```

②

①

```
>>> print( bands2.pop( ) )
피플 온 더 브릿지
```

list에서 마지막 item을 삭제하고 반환

```
>>> print( bands2 )
```

bands2

제거

['호피폴라', '애프터문', '루시', '모네', '퍼플레인', '피플 온 더 브릿지']

['호피폴라', '애프터문', '루시', '모네', '퍼플레인']

list 함수– insert( ) : item 추가

```
>>> bands3 = bands2[ : ]

>>> bands3.insert( 3, "today" )
                  insert ( 삽입index, item값 )
>>> print( bands3 )
```

bands3

```
[ '호피폴라', '애프터문', '루시', '모네', '퍼플레인' ]
```

```
['호피폴라', '애프터문', '루시', 'today', '모네', '퍼플레인' ]
```

기존 item들은 뒤로 밀려남

list 함수– sort( ) : item 정렬

```
>>> bands4 = bands [:]
>>> print(bands4)

>>> bands4.sort()
>>> print(bands4)

>>> bands4.sort( reverse = True )
>>> print(bands4)
```

오름차순

내림차순

bands4

['호피폴라', '애프터문', '루시', '모네', '퍼플레인', '피플 온 더 브릿지']

['루시', '모네', '애프터문', '퍼플레인', '피플 온 더 브릿지', '호피폴라']

['호피폴라', '피플 온 더 브릿지', '퍼플레인', '애프터문', '모네', '루시']

list 함수– index( ) : index 찾기

```
>>> bands5 = bands[ : ]
>>> bands5.index( '퍼플레인' )
4
>>> bands5.index( '모네' )
3
```

bands5

['호피폴라', '애프터문', '루시', '모네', '퍼플레인', '피플 온 더 브릿지']

list 함수– remove( ) : 값으로 item 제거

```
>>> bands6 = [
'호피폴라', '애프터문', '모네', '퍼플레인', '피플 온 더 브릿지', '모네']
>>> bands6.remove('모네')
>>> bands6
```

bands6

['호피폴라', '애프터문', '모네', '퍼플레인', '피플 온 더 브릿지', '모네']

처음 만나는 '모네' 만 제거

['호피폴라', '애프터문', '퍼플레인', '피플 온 더 브릿지', '모네']

Built-in 함수 : len() – item 개수

```
>>> len(bands6)
5
```

bands6

```
['호피폴라', '애프터문',  '퍼플레인', '피플 온 더 브릿지', '모네']
```

list

- list 만들기
- indexing tuple, str에도 적용
- slicing tuple, str에도 적용
- item / list 일부 / list 전부 제거
- list 할당
- slicing 할당
- item 값 변경
- item 추가
- +, * operations tuple, str에도 적용
- item 정렬
- index 찾기
- item 개수
- 2차원 list 활용

** Function 호출 방법

tuple

tuple

```
>>> t_days = ( 'Sun', 'Mon', 'Tues', 'Wed', 'Thr', 'Fri', 'Sat' )
```

tuple 만들기: (값1, 값2, 값3, 값4)
: 값1, 값2, 값3, 값4

```
>>> t_days[1]
```

indexing 방법 list와 동일

```
'Mon'
```

```
>>> del t_days[1]
```

한번 만든 tuple은 절대 수정불가!

```
Traceback (most recent call last):
  File "<pyshell#190>", line 1, in <module>
    del t_days[1]
TypeError: 'tuple' object doesn't support item deletion
```

값변경, 추가, 제거, 정렬 모두 안됨

tuple slicing

```
>>> t_days = ('Sun', 'Mon', 'Tues', 'Wed', 'Thr', 'Fri', 'Sat')
>>> t_days[ 1: 6: 2 ]
('Mon', 'Wed', 'Fri')
```

slicing 방법 list와 동일

tuple slicing 결과 → tuple

tuple : + * operations

```
>>> a = ( 10, 20, 30 ); b = ( 50, 10 )

>>> a + b
(10, 20, 30, 50, 10)
```

tuple + 연산 결과 –> 새로운 tuple

```
>>> a + ( 90, )
(10, 20, 30, 90)
```

원소 1개 tuple 표현법: (값,)

```
>>> b * 3
(50, 10, 50, 10, 50, 10)
```

tuple +, * 연산: list, str 과 동일

tuple assignment

- 한번에 할당하기 (Chapter02)

```
>>> width , height , length = 20, 5, 5
>>> width , height , length = (20, 5, 5)
>>> (width , height , length) = 20, 5, 5
>>> (width , height , length) = (20, 5, 5)

>>> [width , height , length] = [20, 5, 5]
```

dict

dict (dictionary)

```
>>> bands_list = ['호피폴라', '애프터문', '퍼플레인']
```
list

```
>>> days_tuple = ( 'Sun', 'Mon', 'Tues', 'Wed', 'Thr', 'Fri', 'Sat')
```
tuple

```
>>> d = dict (  )
>>> d2 = { }
```
원소 없는 dict 만들기

dict 만들기

(key, value)의 item들 집합

```
singer ={
        "이름" : "김광석" ,
        "출생년도" : 1980 ,
        "데뷔곡" : "너에게"
 }
```

dict 에서 key는 반드시 유일해야(unique) 한다

item 추가 / value 변경

```
>>> player = { }

>>> player['이름'] = '김광석'
>>> player['출생년도'] = 1980
>>> player['데뷔곡'] = '너에게'

>>> player
{'이름': '김광석', '출생년도': 1980, '데뷔곡': '너에게'}

>>> player['출생년도'] = 1964

>>> player
{'이름': '김광석', '출생년도': 1964, '데뷔곡': '너에게'}
```

item 추가:
dict이름[새 key값] = 새 value값

value 변경:
dict이름[key값] = 새 value값

dict indexing

```
>>> player[ '이름' ]
'김광석'
```

dict이름[key값]

Value값 알려줌

```
>>> years={ 2016 : "원숭이", 2017 : "닭" , 2018: "개" }

>>> years[2017]
'닭'
>>> years[2019]
Traceback (most recent call last):
  File "<pyshell#222>", line 1, in <module>
    years[2019]
KeyError: 2019
```

없는 key값으로 indexing

dict – item 제거

```
>>> years={ 2016 : "원숭이", 2017 : "닭", 2018: "개" }

>>> del years[2017]
```

key값이 2017인 item 제거

```
>>> years
{2016: '원숭이', 2018: '개'}
```

built-in 함수 – len() / dict함수 – items()

```
>>> player= {'이름': '김광석', '출생년도': 1964, '데뷔곡': '너에게
'}
```

item 개수

```
>>> len(player)
3
```

dict → dict_items type으로

```
>>> player.items()
dict_items(
  [ ('이름', '김광석'),  ('출생년도', 1964), ('데뷔곡', '너에게')]  )
```

dict_items type: (key, value) tuple 들의 모임, list와 비슷

dict함수 – keys(), values()

```
>>> player
{'이름': '김광석', '출생년도': 1964, '데뷔곡': '너에게'}

>>> player.keys()
dict_keys(['이름', '출생년도', '데뷔곡'])
```

[key1, key2, key3, ...]

```
>>> player.values()
dict_values(['김광석', 1964, '너에게'])
```

[value1, value2, value3, ...]

dict 함수 – items(), keys(), values()의 결과

- 엄밀히 list type이 아니므로 list로 변환하여 사용해야 함

```
>>> print( list( player.keys() )  )
['이름', '출생년도', '데뷔곡']

>>> print( list(  player.values() )  )
['김광석', 1964, '너에게']

>>> print( list(  player.items() )  )
[('이름', '김광석'), ('출생년도', 1964), ('데뷔곡', '너에게')]
```

in 연산자

```
>>> '이름' in player
True

>>> 1964 in player
False
```

key 에서만 찾음

in 연산자 : list, tuple, dict, set, str 모두에 사용 가능

key

```
>>> years={ [2016, 2017] : "원숭이닭"}
Traceback (most recent call last):
  File "<pyshell#241>", line 1, in <module>
    years={ [2016, 2017] : "원숭이닭"}
TypeError: unhashable type: 'list'
```

list는 key로 사용 못함

```
>>> years={ (2016, 2017) : "원숭이닭"}
>>> years
{(2016, 2017): '원숭이닭'}
```

tuple은 key로 사용 가능

tuple, dict

- tuple
  - tuple 만들기
  - indexing
  - slicing
  - +, * 연산
- dict
  - dict 만들기 (key, value, item)
  - item 추가 / value 변경
  - indexing
  - item 제거
  - dict 함수 - keys(), values(), items()
  - in 연산자 list, tuple, str, set 에도 적용
  - key 제한사항

set

set 만들기

```
>>> bands_list = ['호피폴라', '애프터문', '퍼플레인']
>>> days_tuple = ( 'Sun', 'Mon', 'Tues', 'Wed', 'Thr', 'Fri', 'Sat')
>>> years={ 2016 : "원숭이", 2017 : "닭" , 2018: "개" }

>>> nums2 = set()
>>> nums2
set()

>>> nums = { 1, 2, 3 }
>>> type(nums)
<class 'set'>
>>> nums
{1, 2, 3}
```

empty set 만들기

{값1, 값2, 값3}

set 만들기

```
>>> days_set = set(days_tuple)
>>> days_set
{'Sun', 'Wed', 'Sat', 'Fri', 'Tues', 'Mon', 'Thr'}

>>> player_set = set([ 1, 2, 2, 2, 3])
>>> player_set
{1, 2, 3}

>>> player_set = set("Hello python")
>>> player_set
{'t', 'h', 'l', 'H', 'e', ' ', 'p', 'o', 'y', 'n'}
```

item값 중복 허용 안함

item들의 순서가 없음

주로 set을 중복 제거를 위한 필터로 사용

indexing

```
>>> player_set = set([1,2,3,4,5,6,7,8])
```

set 내 item값에 접근할 방법이 없다
① indexing 불가 (순서가 없음)
② dict와 같이 key가 존재하지 않음

```
>>> player_list = list(player_set)

>>> player_tuple = tuple(player_set)
```

Indexing 가능한 type으로 변환

```
>>> player_list[3]
4
>>> player_tuple[6]
7
```

item 추가, 삭제

```
>>> player_set
{1, 2, 3, 4, 5, 6, 7, 8}

>>> player_set.add(10)
>>> player_set
{1, 2, 3, 4, 5, 6, 7, 8, 10}

>>> player_set.remove(5)
>>> player_set
{1, 2, 3, 4, 6, 7, 8, 10}

>>> player_set.pop()
1
>>> player_set
{2, 3, 4, 6, 7, 8, 10}
```

10 추가

5 제거

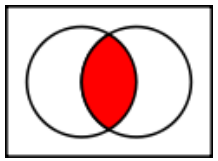
임의의 item 제거

set 연산 / set 함수

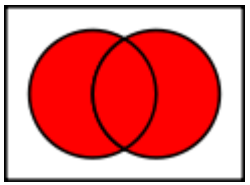
```
>>> a=set([1, 2, 3, 4])
>>> b=set([3, 4, 5, 6])

>>> a & b
{3, 4}
>>> a.intersection(b)
{3, 4}

>>> a | b
{1, 2, 3, 4, 5, 6}
>>> a.union(b)
{1, 2, 3, 4, 5, 6}
```



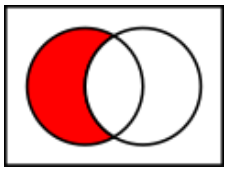
교집합



합집합

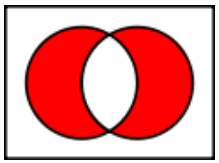
set 연산 / set 함수

```
>>> a - b
{1, 2}
>>> a.difference(b)
{1, 2}
```



차집합

```
>>> a ^ b
{1, 2, 5, 6}
>>> a.symmetric_difference(b)
{1, 2, 5, 6}
```



대칭차집합

set

- set
  - unordered, not duplicated
  - indexing 불가
- item 추가, 삭제 set 함수
- set 연산