set: 집합. 중복 입력된 값은 자동으로 없애주며,

초기화는

my\_set = set( )

my\_set = set( [ ] )

my\_set = {1, 2, 3, 4}

와 같이 이루어진다.

추가: .add( ) 사용

my\_set.add(6)

add 사용시 오직 하나의 원소만, 하나의 number, string, tuple 만 가능

여러개 추가: .update( ) 사용

여러 개의 원소를 추가하므로 list 또는 set 안에 각 원소들을 넣어서 update( ) 로 전달한다.

my\_set.update([1, 2, 3, 4, 5, 6])

my\_set.update({6, 7, 8, 8})

my\_list = [0 ,1, 2, 3]

my\_set.update(my\_list)

set 안의 값 제거

remove, discard: set 안의 값 제거

remove: 없는 값 제거시 오류

discard : 없는 값 제거해도 오류 x

집합 연산 가능

합집합: | or .union( ), 교집합: & or .intersection( ), 차집합: - or .difference( ), 대칭차집합: ^ or .symmetric\_difference

dictonary: 바뀔 수 없는 (immutable)한 key와 바뀔 수 있는 (mutable) 한 값 value로 이루어진,

순서가 없는 집합.

초기화 : 중괄호 사용

ex) my\_dict = { } // 빈 딕셔너리 생성

my\_dict\_1 = {“a” : 1, “b”, 2} //key를 a, value가 1 , key가 b, value가 2인 두 딕셔너리 생성

key 값으로 들어갈 수 있는 Immutable 한 값들:

int, tuple, float, bool, string

>>> a = {1: 5, 2: 3} # int 사용

>>> a

{1: 5, 2: 3}

>>> a = {(1,5): 5, (3,3): 3} # tuple사용

>>> a

{(1, 5): 5, (3, 3): 3}

>>> a = { 3.6: 5, "abc": 3} # float 사용

>>> a

{3.6: 5, 'abc': 3}

>>> a = { True: 5, "abc": 3} # bool 사용

>>> a

{True: 5, 'abc': 3}

key 값으로 들어갈 수 없는 것들:

# mutable 예

>>> a = { {1, 3}: 5, {3,5}: 3} #set 사용 에러

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

TypeError: unhashable type: 'set'

>>> a = {[1,3]: 5, [3,5]: 3} #list 사용 에러

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

TypeError: unhashable type: 'list'

>>> a = { {"a":1}: 5, "abc": 3} #dict 사용 에러

Traceback (most recent call last):

File "<stdin>", line 1, in <module>

TypeError: unhashable type: 'dict'

value 값으로 들어갈 수 있는 mutable 값들:

키 값이 중복되면, 마지막 값으로 덮어씌워진다.

순서가 없기에, 인덱스로 접근할 수 없으며 키로 접근할 수 있다.

객체에 키로 접근하여 값 변경 가능하다.

my\_dict [“key”] = new\_value

딕셔너리에 존재하지 않는 키로 접근하면 새 값을 추가한다

my\_dict [“new\_key”] = some\_value

얕은 복사와 깊은 복사

얕은 복사

a = { ‘alice’: 3, ‘bob’: 20, ‘tony’:15, ‘suzy’: 30 }

b = a.copy( ) or b = dict(a)

if b is modified, a is also modicfied

깊은 복사

import copy

b = copy.deepcopy(a)

if b is modified, a is not modified

딕셔너리 수정:

단일 값은 직접 접근하여 할당,

여러값 수정은 .update(dict addeddict) , set과 비슷하게 사용한다.

for문 사용

for key in my\_dict:

print(key)

순서가 없으므로 랜덤, 기본적으로 key 값이 할당되어 반복문을 돌게된다.

마찬가지로 순서가 없으므로 랜덤, my\_dict.values( ) 를 사용하면 각 value 값이 할당된다.

for val in my\_dict.values():

print(val)

두 값을 모두 반복하려면 dict.items( ) 사용

for key, val in my\_dict.items():

print(key, value)

dictionary 요소 삭제: del 사용

del my\_dict[‘key’]

tuple: 변하지 않는, immutable 한 순서가 있는 객체의 집합이다.

list형과 비슷하지만 한번 생성되면 갑 변겅이 불가능하다.

https://wikidocs.net/16042