Unit 25.

Status	완료
담당자	
마감일	
완료일	

Unit 25. 딕셔너리 응용하기

딕셔너리 키-값 쌍 추가하기

키-값 쌍을 추가하는 메서드는 2가지가 있다.

- setdefault: 키-값 쌍 추가
- update: 키의 값 수정, 키가 없으면 키-값 쌍 추가

25.1.2 딕셔너리에 키와 기본값 저장하기

- setdefault에 키만 지정하면 값에 None을 저장 다음은 키 'e'를 추가하고 값에 None을 저장

```
>> x = {'a': 10, 'b': 20, 'c': 30, 'd': 40}
>>> x.setdefault('e')
>>> x
{'a': 10, 'b': 20, 'c': 30, 'd': 40, 'e': None}
```

- **setdefault(키, 기본값)**처럼 키와 기본값을 지정하면 값에 기본값을 저장한 뒤 해당 값을 반환

```
>>> x.setdefault('f', 100)
100
>>> x
{'a': 10, 'b': 20, 'c': 30, 'd': 40, 'e': None, 'f': 100}
```

!!! setdefault은 추가만 할 수 있고 수정은 안된다

25.1.3 딕셔너리에서 키의 값 수정하기

- **update(키=값)**은 딕셔너리에서 키의 값을 수정

```
>> x = {'a': 10, 'b': 20, 'c': 30, 'd': 40}
>>> x.update(a=90)
>>> x
{'a': 90, 'b': 20, 'c': 30, 'd': 40}
```

- 딕셔너리에 키가 없으면 키-값 쌍을 추가

```
>>> x.update(e=50)
>>> x
{'a': 90, 'b': 20, 'c': 30, 'd': 40, 'e': 50}
```

- update는 키-값 쌍 여러 개를 콤마로 구분해서 넣어주면 값을 한꺼번에 수정할 수 있다.

```
x.update(a=900, f=60)
>>> x
{'a': 900, 'b': 20, 'c': 30, 'd': 40, 'e': 50, 'f': 60}
```

- 키가 숫자일 경우에는 **update(딕셔너리)**처럼 딕셔너리를 넣어서 값을 수정할 수 있습니다.

```
>>> y = {1: 'one', 2: 'two'}
>>> y.update({1: 'ONE', 3: 'THREE'})
```

```
>>> y
{1: 'ONE', 2: 'two', 3: 'THREE'}
```

- 리스트는 **[[키1, 값1], [키2, 값2]]** 형식으로 키와 값을 리스트로 만들고 이 리스트를 다시 리스트 안에 넣어서 키-값 쌍을 나열해줍니다(튜플도 같은 형식).

```
>>> y.update([[2, 'TWO'], [4, 'FOUR']])
>>> y
{1: 'ONE', 2: 'TWO', 3: 'THREE', 4: 'FOUR'}
```

25.1.4 딕셔너리에서 키-값 쌍 삭제하기

- **pop(키)**는 딕셔너리에서 특정 키-값 쌍을 삭제한 뒤 삭제한 값을 반환

```
>>> x = {'a': 10, 'b': 20, 'c': 30, 'd': 40}
>>> x.pop('a')
10
>>> x
{'b': 20, 'c': 30, 'd': 40}
```

- pop 대신 del로 특정 키-값 쌍을 삭제할 수도 있다. 이때는 [ ]에 키를 지정하여 del을 사용

```
>>> x = {'a': 10, 'b': 20, 'c': 30, 'd': 40}
>>> del x['a']
>>> x
{'b': 20, 'c': 30, 'd': 40}
```

25.1.5 딕셔너리에서 모든 키-값 쌍 삭제하기

- **clear()** 는 딕셔너리의 모든 키-값 쌍을 삭제

```
>>> x = {'a': 10, 'b': 20, 'c': 30, 'd': 40}
>>> x.clear()
>>> x
{}
```

25.1.7 딕셔너리에서 키의 값을 가져오기

get(키)는 딕셔너리에서 특정 키의 값을 가져온다.

```
>>> x = {'a': 10, 'b': 20, 'c': 30, 'd': 40}
>>> x.get('a')
10
```

키가 없을 때는 기본값을 반환한다.

25.1.8 딕셔너리에서 키와 쌍의 값을 모두 가져오기

- items: 키-값 쌍을 모두 가져옴

```
>>> x = {'a': 10, 'b': 20, 'c': 30, 'd': 40}
>>> x.items()
dict_items([('a', 10), ('b', 20), ('c', 30), ('d', 40)])
```

- keys: 키를 모두 가져옴

```
>>> x.keys()
dict_keys(['a', 'b', 'c', 'd'])
```

- values: 값을 모두 가져옴

```
>>> x.values()
dict_values([10, 20, 30, 40])
```

25.2 반복문으로 딕셔너리 키-값 쌍을 모두 출력하기

반복문으로 키와 값을 모두 출력하려면

딕셔너리를 지정하고 items( )를 사용 해야한다

```
for 키, 값 in 딕셔너리.items():
     반복할 코드
```

```
>>> x = {'a': 10, 'b': 20, 'c': 30, 'd': 40}
>>> for key, value in x.items():
...     print(key, value)
...
a 10
b 20
c 30
d 40
```

- 딕셔너리를 직접 지정하고 items를 사용해도 관계없다.

```
for key, value in {'a': 10, 'b': 20, 'c': 30, 'd': 40}.items():
    print(key, value)
```

25.2.1 딕셔너리의 키만 출력하기

키만 가져오거나 값만 가져올 수도 있다

keys: 키를 모두 가져옴

```
>>> x = {'a': 10, 'b': 20, 'c': 30, 'd': 40}
>>> for key in x.keys():
...     print(key, end=' ')
...
a b c d
```

25.2.2 딕셔너리의 값만 출력하기

values: 값을 모두 가져옴

```
>>> x = {'a': 10, 'b': 20, 'c': 30, 'd': 40}
>>> for value in x.values():
...     print(value, end=' ')
...
10 20 30 40
```

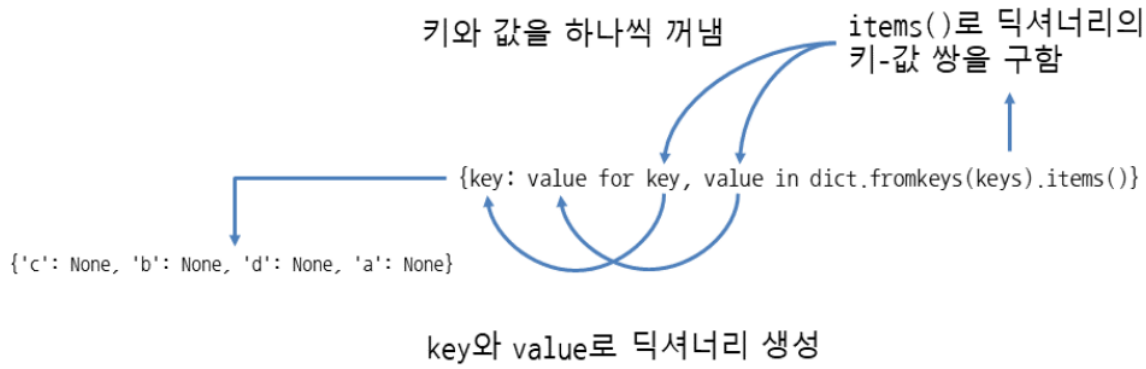
25.3 딕셔너리 표현식 사용하기

- **{키: 값 for 키, 값 in 딕셔너리}**
- **dict({키: 값 for 키, 값 in 딕셔너리})**

```
>>> keys = ['a', 'b', 'c', 'd']
>>> x = {key: value for key, value in dict.fromkeys(keys).items()}
>>> x
{'a': None, 'b': None, 'c': None, 'd': None}
```

- 딕셔너리 표현식을 사용할 때는

`x = {key: value for key, value in dict.fromkeys(keys).items()}`



- keys로 키만 가져온 뒤 특정 값을 넣거나, values로 값을 가져온 뒤 값을 키로 사용할 수도 있다.

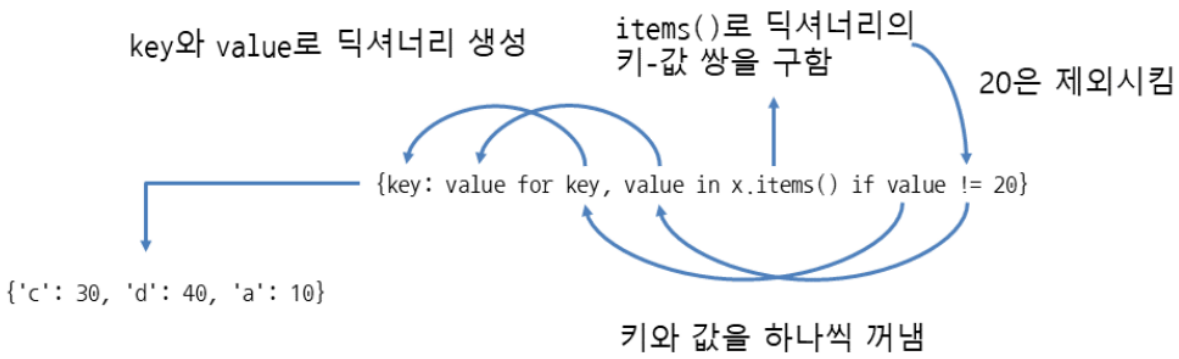
```
>>> {key: 0 for key in dict.fromkeys(['a', 'b', 'c', 'd']).keys()}              # 키만 가져옴
{'a': 0, 'b': 0, 'c': 0, 'd': 0}
>>> {value: 0 for value in {'a': 10, 'b': 20, 'c': 30, 'd': 40}.values()}       # 값을 키로 사용
{10: 0, 20: 0, 30: 0, 40: 0}
```

25.3.1 딕셔너리 표현식에서 if조건문 사용하기

- 딕셔너리 표현식은 특정 값을 찾아서 삭제하는데 유용하다
- **{키: 값 for 키, 값 in 딕셔너리 if 조건식}**
- **dict({키: 값 for 키, 값 in 딕셔너리 if 조건식})**

```
>>> x = {'a': 10, 'b': 20, 'c': 30, 'd': 40}
>>> x = {key: value for key, value in x.items() if value != 20}
>>> x
{'a': 10, 'c': 30, 'd': 40}
```

for반복문으로 반복하면서 키-값 쌍을 삭제하면 안된다.



25.4 딕셔너리 안에 딕셔너리 사용하기

- **• 딕셔너리 = {키1: {키A: 값A}, 키2: {키B: 값B}}**
- 딕셔너리 안에 들어있는 딕셔너리에 접근하려면 딕셔너리 뒤에 _를 단계만큼 붙이고 키를 지정

25.5 딕셔너리의 할당과 복사

- 리스트와 마찬가지로 딕셔너리 또한 할당과 복사는 큰 차이가 있다

```
>>> x = {'a': 0, 'b': 0, 'c': 0, 'd': 0}
>>> y = x
```

위와 같이 다른 변수에 할당하면 딕셔너리는 2개가 될꺼 같지만 실제로는 한 개다.

딕셔너리를 x와y를 완전히 2 개로 만드려면 copy메서드로 모드 키-값 쌍을 복사해야 한다.

```
>>> x = {'a': 0, 'b': 0, 'c': 0, 'd': 0}
>>> y = x.copy()
```

- 중첩 딕셔너리는 완전히 복사하려면 copy가 아닌 copy모듈의 deepcopy함수르 사용해야 한다.

```
>>> x = {'a': {'python': '2.7'}, 'b': {'python': '3.6'}}
>>> import copy             # copy 모듈을 가져옴
>>> y = copy.deepcopy(x)    # copy.deepcopy 함수를 사용하여 깊은 복사
>>> y['a']['python'] = '2.7.15'
>>> x
{'a': {'python': '2.7'}, 'b': {'python': '3.6'}}
>>> y
{'a': {'python': '2.7.15'}, 'b': {'python': '3.6'}}
```