5강

지능형 자료구조(딕셔너리)

5-1 딕셔너리 조작

- 딕셔너리 생성
  - ✓ 파이썬에서는 연관된 값(key, value)을 묶어서 저장하는 용도로 딕셔너리를 제공한다.
  - ✓ 딕셔너리의 키는 문자열 뿐만 아니라 정수, 실수, 불도 사용할 수 있으며 자료형을 섞어서 사용해도 된다.
  - ✓ 값에는 리스트, 튜플 등을 포함하여 모든 자료형을 사용할 수 있다.

```
data1 = {'name':'kim', 'age':20, 'address':'seoul'}
print(data1)
print(type(data1))
```

[결과]
{'name': 'kim', 'age': 20, 'address': 'seoul'}
<class 'dict'>

```
data2 = {'names':['kim','lee','choi'],
         'info':{'ages':[10,20,30],
         'height':[180,172,158]}}
print(data2)
```

[결과]
{'names': ['kim', 'lee', 'choi'], 'info': {'ages': [10, 20, 30], 'height': [180, 172, 158]}}

```
data3 = {10:[40,50], True:[22.2,444.56], 3.3:['kim','choi']}
print(data3)
print(data3[10])
print(data3[True])
print(data3[3.3])
```

[결과]
{10: [40, 50], True: [22.2, 444.56], 3.3: ['kim', 'choi']}
[40, 50]
[22.2, 444.56]
['kim', 'choi']

5-1 딕셔너리 조작

- 딕셔너리 생성
  - ✓ dict는 다음과 같이 키와 값을 연결하거나, 리스트, 튜플, 딕셔너리로 딕셔너리를 만들 때 사용한다.

```
data4 = dict(name='kim', age=30)
print(data4)
print(data4['name'])
```

```
[결과]
{'name': 'kim', 'age': 30}
kim
```

```
data5 = dict(zip(['health','name'],[100,'kim']))
print(data5)
```

```
[결과]
{'health': 100, 'name': 'kim'}
```

```
data6 =
dict([('name','lee'),('age',50),('weight',80)])
print(data6)
```

```
[결과]
{'name': 'lee', 'age': 50, 'weight': 80}
```

- 딕셔너리 할당
  - ✓ 딕셔너리에서 키의 값을 출력할 때와 마찬가지로 [ ]에 키를 지정한 뒤 값을 할당하면 된다.

```
data7 = {'age':30}
data7['age'] = 50
print(data7)
data7['weight'] = 80
print(data7)
```

```
[결과]
{'age': 50}
{'age': 50, 'weight': 80}
```

5-1 딕셔너리 조작

- 딕셔너리 기본 값 할당
  - ✓ setdefault(키)는 딕셔너리에 키-값 쌍을 추가한다. 이때 setdefault()에 키만 지정하면 값에 None이 저장된다.

```
data7 = {'age':30}
data7.setdefault('height',180)
print(data7)
data7.setdefault('info')
print(data7)
```

```
[결과]
{'age': 30, 'height': 180}
{'age': 30, 'height': 180, 'info': None}
```

- 딕셔너리 수정
  - ✓ update(키=값)은 딕셔너리에서 키의 값을 수정한다.

```
data8 = {'name':'lee', 'age':20}
data8.update(name=40)
print(data8)
```

```
[결과]
{'name': 40, 'age': 20}
```

  - ✓ 딕셔너리에 키가 없으면 키-값 쌍을 추가한다.

```
data8.update(weight=78)
print(data8)
```

```
[결과]
{'name': 40, 'age': 20, 'weight': 78}
```

5-1 딕셔너리 조작

- 딕셔너리 함수 생성
  - ✓ dict.fromkeys(키리스트)는 키 리스트로 딕셔너리를 생성하며 값은 모두 None으로 저장한다.
  - ✓ dict.fromkeys(키리스트, 값)처럼 키 리스트와 값을 지정하면 해당 값이 키의 값으로 저장된다.

```
keys = ['a', 'b', 'c', 'd']
data1 = dict.fromkeys(keys)
print(data1)
data2 = dict.fromkeys(keys, 100)
print(data2)
```

```
[결과]
{'a': None, 'b': None, 'c': None, 'd': None}
{'a': 100, 'b': 100, 'c': 100, 'd': 100}
```

5-2 딕셔너리 할당과 복사

- 딕셔너리 할당과 복사
  - ✓ 딕셔너리를 아래와 같이 생성 후 다른 변수에 할당한다.
  - ✓ data1 = data2로 할당하면 두 변수는 같은 딕셔너리를 지정하고 있다.

```
data1 = {'a':10,'b':20}
data2 = data1
print(data1 is data2)
data2['b'] = 40
print(data1)
print(data2)
```

```
[결과]
True
{'a': 10, 'b': 40}
{'a': 10, 'b': 40}
```

  - ✓ 복사하기 위해서는 copy()를 이용한다.

```
data3 = {'a':10,'b':20}
data4 = data3.copy()
print(data3 is data4)
data4['b'] = 40
print(data3)
print(data4)
```

```
[결과]
False
{'a': 10, 'b': 20}
{'a': 10, 'b': 40}
```

5-2 딕셔너리 할당과 복사

- 중첩딕셔너리 할당과 복사
  - ✓ 딕셔너리를 아래와 같이 생성 후 다른 변수에 copy() 사용하여 할당한다.

```
data5 = {'a':{'name':'kim'},'b':{'name':'choi'}}
data6 = data5.copy()
data6['a']['name'] = 'oh'
print(data5)
print(data6)
```

[결과]
{'a': {'name': 'oh'}, 'b': {'name': 'choi'}}
{'a': {'name': 'oh'}, 'b': {'name': 'choi'}}

  - ✓ 중첩 딕셔너리를 완전히 복사하려면 copy 메서드 대신 copy 모듈의 deepcopy 함수를 사용해야 한다.

```
import copy
data7 = {'a':{'name':'kim'},'b':{'name':'choi'}}
data8 = copy.deepcopy(data7)
data8['a']['name']='lee'
print(data7)
print(data8)
```

[결과]
{'a': {'name': 'kim'}, 'b': {'name': 'choi'}}
{'a': {'name': 'lee'}, 'b': {'name': 'choi'}}

5-3 딕셔너리 컴프리핸션

- 딕셔너리 컴프리핸션 사용
  - ✓ 리스트와 마찬가지로 딕셔너리도 for 반복문과 if 조건문을 사용하여 딕셔너리를 생성할 수 있다.

```
keys = ['a','b','c','d']
data1 = {k : v for k, v in dict.fromkeys(keys).items()}
print(data1)
```

```
[결과]
{'a': None, 'b': None, 'c': None, 'd': None}
```

  - ✓ keys()로 키만 가져온 뒤 특정 값을 넣거나, values로 값을 가져온 뒤 값을 키로 사용할 수도 있다.

```
data2 = {k : 100 for k in dict.fromkeys(keys).keys()}
print(data2)
```

```
[결과]
{'a': 100, 'b': 100, 'c': 100, 'd': 100}
```

```
data3 = {'name':['kim','lee','choi']}
data4 = {v : 0 for value in data3.values() for v in value}
print(data4)
```

```
[결과]
{'kim': 0, 'lee': 0, 'choi': 0}
```

5-3 딕셔너리 컴프리핸션

- 딕셔너리 컴프리핸션 사용
  - ✓ 딕셔너리는 for 반복문으로 반복하면서 키-값 쌍을 삭제하면 안 된다.

```
data5 = {'one':10, 'two':20, 'three':30, 'four':40}
for k, v in data5.items():
    if v == 20:
        del data5[k]
```

```
[결과]
Traceback (most recent call last):
  File "C:/Users/HSGlobal/python_lec/dictEx.py", line 2, in <module>
    for k, v in data5.items():
RuntimeError: dictionary changed size during iteration
```

  - ✓ 딕셔너리 컴프리핸션에 if 조건문을 사용하여 삭제할 값을 제외하면 딕셔너리의 특정 값을 가지고 있는 요소를 제거할 수 있다.

```
data5 = {k : v for k, v in data5.items() if v != 20}
print(data5)
```

```
[결과]
{'one': 10, 'three': 30, 'four': 40}
```