9. 파이썬 자료형 - 딕셔너리

Made By. 김규일

- 딕셔너리는 사전이라는 뜻이다.
- 영어사전처럼 ‘people’ = ‘사람’, ‘baseball’ = ‘야구’처럼 나타난다. 이때 프로그래밍에선 이를 Key-Value로 이루어져 있다고 표현한다. people이라는 Key는 사람이라는 Value와 쌍을 이루는 것이다.
- 이런 Key-Value 쌍을 보통 연관 배열(Associative Array) 또는 해시(Hash)라고 한다.
- 보통 Key와 Value의 특징은 같은 값을 가지는 Key는 존재할 수 없다. 하지만 같은 값을 가지는 Value는 존재할 수 있다.
- Key-Value가 쌍으로 이루어져 있기 때문에 추가하거나 삭제할 때는 항상 Key-Value가 한 쌍으로 추가, 삭제가 된다.
- 특정 Value를 찾기 위해서는 그 Value와 연결된 Key를 통해서만 찾을 수 있다.
- 기본적으로 파이썬의 딕셔너리 자료형에 형태는 다음과 같다.

```
>>> dic = {'name':'pey', 'phone':'010-9999-1234', 'birth': '1118'}
```

- 이를 Key-Value로 표현하면 다음과 같다.

Key	Value
name	pey
phone	010-9999-1234
birth	1118

```
>>> a = {1: 'hi'}
>>> a = { 'a': [1,2,3]}

## Key와 Value는 위처럼 데이터 타입의 제약을 크게 받지 않는다.
```

딕셔너리 쌍 추가, 삭제

- 딕셔너리 쌍 추가하기

```
>>> a = {1: 'a'}
>>> a[2] = 'b'
>>> a
{1: 'a', 2: 'b'}

## a[2] = 'b'는 a라는 딕셔너리에 Key는 2, Value는 b를 삽입하라는 의미이다.

>>> a['name'] = 'pey'
>>> a
{1: 'a', 2: 'b', 'name': 'pey'}

>>> a[3] = [1,2,3]
>>> a
{1: 'a', 2: 'b', 'name': 'pey', 3: [1, 2, 3]}
```

- 딕셔너리 요소 삭제하기

```
>>> del a[1]
>>> a
{2: 'b', 'name': 'pey', 3: [1, 2, 3]}

## Key가 1인 딕셔너리의 쌍이 삭제된다.
## 삭제 또한 Key-Value 쌍이 삭제된다.
```

딕셔너리 사용법

- 딕셔너리에서 Key 사용해 Value 얻기

```
>>> grade = {'pey': 10, 'julliet': 99}
>>> grade['pey']
10
>>> grade['julliet']
99
```

- 딕셔너리의 특징 중 하나인 Key는 불변하지 않는 고유한 값을 가져야 한다는 점이다.
- 만약 같은 Key를 입력하게 되면 뒤에 Key는 무시 된다.

```
>>> a = {1:'a', 1:'b'}
>>> a
{1: 'b'}
```

- 또한 Key는 불변해야 하기 때문에 변할 수 있는 데이터 타입인 리스트는 사용이 불가능하다.

```
>>> a = {[1,2] : 'hi'}
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unhashable type: 'list'
```

- 변하지 않는 값인 튜플은 Key로 사용할 수 있다.

딕셔너리 관련 함수들

- Key 리스트 만들기(Keys)

```
>>> a = {'name': 'pey', 'phone': '010-9999-1234', 'birth': '1118'}
>>> a.keys()
dict_keys(['name', 'phone', 'birth'])

## keys 함수는 딕셔너리의 key 값만 모아 dict_keys 객체를 반환한다.
```

```
>>> for k in a.keys():
...    print(k)
...
name
phone
birth

## keys 함수를 통해 위와 같이 활용할 수 있다.

>>> list(a.keys())
['name', 'phone', 'birth']

## 또한 위처럼 리스트로 반환하게 할 수 있다.
```

- Value 리스트 만들기(values)

```
>>> a.values()
dict_values(['pey', '010-9999-1234', '1118'])

## values 함수를 이용하면 dict_values 객체를 반환한다.
```

- Key, Value 쌍 얻기(items)

```
>>> a.items()
dict_items([('name', 'pey'), ('phone', '010-9999-1234'), ('birth', '1118')])

## items 함수를 이용하면 dict_items 객체를 반환한다.
```

- Key, Value 쌍 모두 지우기(clear)

```
>>> a.clear()
>>> a
{}

## 딕셔너리 안에 모든 요소를 삭제한다.
## 빈 리스트는 [], 빈 튜플은 (), 빈 딕셔너리는 {} 이다.
```

- Key로 Value 얻기(get)

```
>>> a = {'name':'pey', 'phone':'010-9999-1234', 'birth': '1118'}
>>> a.get('name')
'pey'
>>> a.get('phone')
'010-9999-1234'

## get() 함수 안의 key 값과 대응되는 쌍의 value를 반환한다. 딕셔너리의 a[] 역할과 같다.
## 두 방식의 다른점은 a[] 방식은 key가 없을 경우 오류를 발생하지만
## get() 함수의 경우 해당하는 key 값이 없다면 None을 반환한다.

>>> a = {'name':'pey', 'phone':'010-9999-1234', 'birth': '1118'}
>>> print(a.get('nokey'))
None
>>> print(a['nokey'])
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'nokey'

## 만약 key가 없을 경우 None 대신 원하는 값을 반환할 수 있다.
## get(key, 디폴트 값)으로 가능하다.
```

```
>>> a.get('nokey', 'foo')
'foo'

## nokey는 없지만 None 대신 foo를 반환
```

- Key가 딕셔너리 안에 있는지 확인(in)

```
>>> a = {'name':'pey', 'phone':'010-9999-1234', 'birth': '1118'}
>>> 'name' in a
True
>>> 'email' in a
False

## 해당 값의 키가 딕셔너리에 있다면 True, 없다면 False를 반환한다.
```