x

K-Digital Training 스마트 팩토리 2기

x

K-Digital Training 스마트 팩토리 3기

튜플, 셋, 딕셔너리

튜플(tuple)

- () 사용
- 리스트처럼 요소를 일렬로 저장
- 하지만, 저장된 요소를 변경, 추가, 삭제할 수 없음
- 한 마디로, 읽기 전용 리스트

```
# 튜플 생성
t1 = (10, 20, 30)

# 튜플 요소 접근
print(t1[0]) # 10

# 튜플 삭제
del(t1)

# 요소가 한 개 있는 튜플 생성하기
t2 = (10) # 10
t3 = (10, ) # (10,)
t4 = 10, # (10,)
```

셋(set)

- 집합을 표현하는 자료형
- 중복된 문자는 한번만 저장된다.
- 요소의 순서가 정해져 있지 않아 셋을 출력해보면 매번 요소의 순서가 바뀐다.

셋(set)

```python
# 셋 만들기
numbers1 = {1, 2, 3, 4, 5} # {1, 2, 3, 4, 5}
numbers2 = set([1, 2, 3, 4, 5, 5]) # {1, 2, 3, 4, 5}
apple = set('apple') # {'e', 'l', 'p', 'a'}

# 값 in 셋
print(1 in numbers1) # True

# 값 not in 셋
print('c' in apple) # False

# 요소 추가
numbers1.add(6)
print(numbers1) # {1, 2, 3, 4, 5, 6}

# 요소 삭제
numbers1.remove(1)
print(numbers1) # {2, 3, 4, 5, 6}

# 임의의 요소 삭제
print(numbers1.pop()) # 2
print(numbers1) # {3, 4, 5, 6}

# 모든 요소 삭제
numbers1.clear()
print(numbers1) # set()
```

셋(set)

```
[1]  s1 = {1,2,3,4,5}
     s2 = {4,5,6,7,8}
```

교집합

```
[2]  s1 & s2
```

```
{4, 5}
```

```
[3]  s1.intersection(s2)
```

```
{4, 5}
```

합집합

```
[4]  s1 | s2
```

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

```
[5]  s1.union(s2)
```

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

차집합

```
[6]  s1 - s2
```

```
{1, 2, 3}
```

```
[7]  s2 - s1
```

```
{6, 7, 8}
```

```
[10] s1.difference(s2)
```

```
{1, 2, 3}
```

딕셔너리(dictionary)

- 두 개의 쌍이 하나로 묶이는 자료구조
- 중괄호({})로 묶여 있으며 키와 값의 쌍으로 이루어져 있다.

```
딕셔너리변수 = {키1: 값1, 키2: 값2, 키3: 값3}
```

```
# 딕셔너리 생성
dict1 = {1: 'a', 2: 'b', 3: 'c'}
print(dict1) # {1: 'a', 2: 'b', 3: 'c'}

# 빈 딕셔너리 생성
dict3 = {}
dict4 = dict()

# 딕셔너리에 요소 추가
dict1[4] = 'd'
print(dict1) # {1: 'a', 2: 'b', 3: 'c', 4: 'd'}

# 요소 삭제
del(dict1[4])
print(dict1) # {1: 'a', 2: 'b', 3: 'c'}
```

딕셔너리(dictionary)

- 딕셔너리 값에 접근하기
  - 딕셔너리변수[키] vs 딕셔너리변수.get(키)

```
fruits = {'apple': '사과', 'banana': '바나나', 'peach': '복숭아'}
print(fruits['pineapple']) # KeyError: 'pineapple'
print(fruits.get('pineapple')) # None
```

- 딕셔너리의 모든 키 반환

```
print(fruits.keys()) # dict_keys(['apple', 'banana', 'peach'])
print(list(fruits.keys())) # ['apple', 'banana', 'peach']
```

- 딕셔너리의 모든 값 반환

```
print(fruits.values()) # dict_values(['사과', '바나나', '복숭아'])
```

딕셔너리(dictionary)

- For문을 활용하여 딕셔너리 모든 값 출력하기

```
for fruit in fruits.keys():
    print(fruit, fruits[fruit])

'''
apple 사과
banana 바나나
peach 복숭아
'''
```

딕셔너리(dictionary)

- 딕셔너리 모두 지우기
  - fruits.clear()
- Key가 딕셔너리 안에 있는지 조사

```
"apple" in fruits
```

```
False
```

추가) map()

- map(함수, 반복 가능한 객체) : 반복 가능한 객체를 함수에 넣어서 실행한다

```
# 반복문 사용
result2 = []
for i in ["a", "b", "c"]:
    result2.append(i.upper())

print(result2) # ['A', 'B', 'C']
```

```
# map() 사용
def capitalize(str):
    return str.upper()

result1 = map(capitalize, ["a", "b", "c"]) # <map object at 0x0000013AD40E3D30>
print(list(result1)) # ['A', 'B', 'C']
```