

제7장 리스트, 튜플, 딕셔너리

■ 학습목표

- 리스트의 개념을 이해하고 활용할 수 있다.
- 튜플의 개념을 이해하고 활용할 수 있다.
- 딕셔너리의 개념을 이해하고 활용할 수 있다.

튜플(Tuple)

■ 튜플의 생성

- 튜플은 읽기 전용 자료를 저장할 때 사용
- 튜플은 ()로 생성, () 생략 가능, 항목이 하나인 튜플은 쉼표(,) 붙임

■ 예:

<pre>t1 = (10,20,30) print(t1) t2 = 10,20,30 print(t2) t3 = (10) print(t3) t4 = 10 print(t4) t5 = (10,) print(t5) t6 = 10, print(t6)</pre>	⇒	<pre>(10, 20, 30) (10, 20, 30) 10 10 (10,) (10,)</pre>
--	---	--

튜플(Tuple)

- 튜플 삭제 : `del(튜플)`, 예 : `del(t1)`

- 튜플 항목 접근 : `튜플명[위치]`

■ 예

<pre>t1 = (10,20,30, 40) t1[0] t1[0] + t1[1] + t[2]</pre>	⇒	<pre>10 60</pre>
---	---	------------------

- 튜플 범위 접근 : `튜플(시작값:끝값+1)`

■ 예

<pre>t1 = (10,20,30, 40) t1[1:3] t1[1:] t1[:3]</pre>	⇒	<pre>(20,30) (20,30,40) (10,20,30)</pre>
--	---	--

- 튜플 덧셈, 곱셈

■ 예

<pre>t2 = ('A', 'B') t1 + t2 t2 * 3</pre>	⇒	<pre>(10,20,30,'A','B') ('A','B', 'A','B', 'A','B,')</pre>
---	---	--

딕셔너리(Dictionary)

■ 딕셔너리의 개념

- 쌍 2개(키, 값)가 하나로 묶인 자료구조
 - 예 : 'apple:사과'
- 문법 : 딕셔너리변수 = {키1:값1, 키2:값2, 키3:값3, ...}
 - 키(유일함)와 값은 사용자가 지정하는 것이지 규정은 없음.
 - Tip : 다른 프로그래밍 언어에서는 해시(Hash), 연관 배열 (Associative Array)이라 함. {키(Key), 값(Value)}로 묶어 구성
 - 주의 : 딕셔너리에는 순서가 없고, 생성 순서대로 딕셔너리가 구성되어 있다는 보장 없음

딕셔너리(Dictionary)

■ 딕셔너리 생성

- 빈 딕셔너리 생성하고 **key**와 **value** 쌍을 추가

■ 예

```
season={}  
season[1]='봄'  
season[2]='여름'  
print(season)
```

 {1: '봄', 2: '여름'}

- 딕셔너리 생성하면서 초기화

■ 예

```
season={1:'봄', 2:'여름'}  
print(season)  
print(season[1])  
print(season[2])
```

 {1: '봄', 2: '여름'}
봄
여름

딕셔너리(Dictionary)

■ 딕셔너리 항목 추가

- **딕셔너리명[키] = 값**

■ 예

```
season={1:'봄', 2:'여름'}  
print(season)  
  
season[3]='가을'  
print(season)
```

 {1: '봄', 2: '여름'}
{1: '봄', 2: '여름', 3: '가을'}

■ 딕셔너리 수정

- **딕셔너리명[키] = 값**

■ 예

```
season={1:'봄', 2:'여름', 3:'가을'}  
print(season)  
  
season[3]='Fall'  
print(season)
```

 {1: '봄', 2: '여름', 3: '가을'}
{1: '봄', 2: '여름', 3: 'Fall'}

딕셔너리(Dictionary)

■ 딕셔너리 항목 삭제 : **del**, **pop()**, **clear()**

- **del 딕셔너리명[키]** : 키에 해당하는 항목 삭제
- **딕셔너리명.pop()** : 지정 항목 삭제하면서 삭제
- **딕셔너리명.clear()** : 모든 항목 삭제

■ 예

```
season={1:'봄', 2:'여름', 3:'가을'}  
print(season)  
  
del season[1]  
print(season)  
  
season.pop(2)  
print(season)  
  
season.clear()  
print(season)
```

 {1: '봄', 2: '여름', 3: '가을'}
{2: '여름', 3: '가을'}
{3: '가을'}
{}

딕셔너리(Dictionary)

■ 딕셔너리의 사용

- 딕셔너리에서 가장 중요한 것은 **키**를 가지고 연관된 **값** 찾는 것
- 값 접근 : **딕셔너리명[키]** , **딕셔너리명.get(키)**
- **get()** 함수 : 존재하지 않는 키에 접근할 경우 **None** 출력

■ 예

```
season={1:'봄', 2:'여름', 3:'가을'}  
print(season)  
print(season[1])  
print(season.get(2))  
print(season.get(4))
```

 {1: '봄', 2: '여름', 3: '가을'}
봄
여름
None

- 딕셔너리항목 갯수 : **len(딕셔너리명)**

■ 예

```
season={1:'봄', 2:'여름', 3:'가을'}  
print(len(season))
```

 3

딕셔너리(Dictionary)

■ 딕셔너리 사용

- 모든 키 반환 : `딕셔너리명.keys()`, `list(딕셔너리명.keys())`:

예

```
season={1:'봄', 2:'여름'}
print(season.keys())
print(list(season.keys()))
```

```
dict_keys([1, 2])
[1, 2]
```

- 딕셔너리의 모든 값을 알고 싶을 때 : `딕셔너리명.values()`

예

```
season={2:'여름', 1:'봄'}
print(season.values())
```

```
dict_values(['여름', '봄'])
```

- 튜플 형태로 항목 반환 : `딕셔너리명.items()`

예

```
season={1:'봄', 2:'여름'}
print(season.items())
```

```
dict_items([(1, '봄'), (2, '여름')])
```

딕셔너리(Dictionary)

■ 딕셔너리 사용

- 딕셔너리 안에 해당 키 있는지 여부 확인 : `키 in 딕셔너리`

예

```
season={1:'봄', 2:'여름', 3:'가을'}

key1 = 1 in season
key2 = 5 in season

print(key1)
print(key2)
```

```
True
False
```

- 딕셔너리의 모든 값 출력

예

```
season={1:'봄', 2:'여름', 3:'가을'}
for k in season.keys():
    print(season[k])
```

```
봄
여름
가을
```

딕셔너리(Dictionary)

■ 딕셔너리 사용

- 딕셔너리 키 정렬 방법

- 딕셔너리는 '키'로 구분하는 것이 중요하기 때문에 순서는 중요하지 않으나 `sorted()` 를 사용하여 정렬 가능함

예

```
season={2:'여름', 1:'봄'}
for key in sorted(season.keys()):
    print(key, season[key])
```

```
1 봄
2 여름
```

- 딕셔너리의 키는 유일해야 함

- 동일한 키가 있다면 마지막에 있는 키가 적용됨

예

```
season={1:'봄', 2:'여름', 3:'가을', 1:'spring'}
print(season)
```

```
{1: 'spring', 2: '여름', 3: '가을'}
```