파이썬 기초

튜플, 집합, 딕셔너리

튜플

튜플

- 튜플(tuple)
  - 리스트와 유사하지만 값을 수정할 수 없으며, 읽기만 가능
  - 튜플은 괄호( )로 생성

```
>>> t1 = ()
>>> t2 = (1,)              # 항목이 하나인 경우 주의
>>> t3 = (1, 2, 3)
>>> t4 = 1, 2, 3           # 괄호 생략 가능
```

  - 프로그램이 동작하는 중 변경되지 않는 데이터의 저장

튜플 특성

- 튜플 인덱싱, 슬라이싱

```
>>> t = (1, 2, 3, 4, 5, 6)
>>> t[0]
1
>>> t[:3]
(1, 2, 3)
>>> t[4:6]
(5, 6)

>>> t[0] = 7 ???
```

- 튜플 더하기, 곱하기

```
>>> t1 = (1, 2)
>>> t2 = ('a', 'b')

>>> t3 = t1 + t2
>>> t3
(1, 2, 'a', 'b')

>>> t1 * 3
(1, 2, 1, 2, 1, 2)
```

튜플 패킹, 언패킹

- **튜플 패킹(Tuple Packing)**

  : 여러 데이터를 튜플로 묶는 것

```
>>> a = 1, 2, 3
>>> a
(1, 2, 3)
```

- **튜플 언패킹(Tuple Unpacking)**

  : 튜플의 각 항목을 여러 개의 변수에 할당하는 것

```
>>> one, two, three = a
>>> print (one, two, three)
1 2 3
```

- **튜플 관련 함수**
  - **index(), count() 제공**

```
# 값 1의 위치
>>> a.index(1)

# 값 1과 일치하는 요소의 개수
>>> a.count(1)
```

집합

집합

- 집합(set)

  - 순서 없이 항목을 저장, 항목이 중복 될 수 없음

※ Python 3.x

```
>>> s1 = set([1,2,3])  # 리스트기반
>>> s2 = {3,4,5}

>>> word = "Hello" * 2
>>> word
'HelloHello'
>>> word_list = list(word)
>>> word_list
['H', 'e', 'l', 'l', 'o', 'H', 'e', 'l', 'l', 'o']
>>> s3 = set(word_list)
```

```
>>> s3
{'H', 'e', 'l', 'o'}

>>> s4 = set("Hello") #문자열기반
>>> s4
{'H', 'e', 'l', 'o'}
```

※ 자료형의 중복을 제거하기 위한 필터로 종종 사용

```
>>> t = tuple(s4)
>>> t
('H', 'e', 'l', 'o')
```

집합 관련 함수

※ Python 3.x

```
>>> s = {1, 2, 3, 4, 5, 3, 4}
>>> s
{1, 2, 3, 4, 5}
>>> s.add(10)
>>> s
{1, 2, 3, 4, 5, 10}
>>> s.remove(2)
>>> s
{1, 3, 4, 5, 10}
```

```
>>> s.update([1,3,5,7,9])
>>> s
{1, 3, 4, 5, 7, 9, 10}
>>> s.discard(7)
>>> s
{1, 3, 4, 5, 9, 10}
>>> s.clear()
>>> s
set()
```

집합

- 합집합, 교집합, 차집합

※ Python 3.x

```
>>> s1 = {1, 2, 3}
>>> s2 = {3, 4, 5}

# 합집합
>>> s1.union(s2)
{1, 2, 3, 4, 5}

# 교집합
>>> s1.intersection(s2)
{3}
```

```
# 차집합( - ) s1.difference(s2)
>>> s1 - s2
{1, 2}

# 합집합( | )
>>> s1 | s2
{1, 2, 3, 4, 5}

# 교집합( & )
>>> s1 & s2
{3}
```

딕셔너리

딕셔너리

- 딕셔너리(dictionary)
  - 중괄호{ }로 묶여 있으며, key와 value의 쌍으로 이루어짐

    d = {key1:value1, key2:value2}
  - key로 value를 관리

    key는 중복 될 수 없음. 예)학번, 주민번호
  - 항목들 사이에 순서가 없음

딕셔너리

```
>>> d = {}
>>> d['kim'] = 1        # 새 항목 추가
>>> d['park'] = 2       # 새 항목 추가
>>> d                   # 전체 출력
{'park': 2, 'kim': 1}
>>> d['park']    # key를 사용하여 값에 접근
2
# 이미 있는 키의 경우 기존의 값 변경
>>> d['kim'] = 3
>>> d
{'park': 2, 'kim': 3}
>>> d['youn'] = 1     # 새 항목 추가
>>> d
{'youn': 1, 'park': 2, 'kim': 3}
```

```
>>> d.keys()          # 키만 추출
dict_keys(['kim', 'park', 'youn'])
>>> d.values()        # 값만 추출
dict_values([3, 2, 1])
# 키와 값의 항목을 튜플 형태로
>>> d.items()
dict_items([('kim', 3), ('park', 2), ('youn', 1)])

# 딕셔너리에 키가 있는 경우
>>> 'kim' in d
True
 # 딕셔너리에 키가 없는 경우
>>> 'lee' in d
False
```

※ Python 3.x

딕셔너리

```
>>> d
{'youn': 1, 'park': 2, 'kim': 3}

# 항목 1개 삭제
>>> del d['kim']
>>> d
{'youn': 1, 'park': 2}

# 전체 삭제
>>> d.clear()
>>> d
{}
```

딕셔너리 – 실습 1

- 어떤 문장을 입력 받으면 해당 문장에서 각 알파벳이 몇 개씩 나오는지 저장하는 딕셔너리를 만든 후, 아래와 같이 출력하시오.
- 실행예시(밑줄은 사용자 입력)

```
Enter a sentence: Python is fun!
{'!': 1, ' ': 2, 'f': 1, 'i': 1, 'h': 1, 'o': 1, 'n': 2, 'P': 1,
's': 1, 'u': 1, 't': 1, 'y': 1}
```