5. 묶인 데이터 타입 (Collection Types)

1. 리스트 (List): `korea = ["Korea", 'KOR', 5, 8, 4]` #Olympic medals from the 2018 Pyeongchang Winter Games

- 언제든지 데이터를 추가, 삽입, 삭제하는 것이 가능(**mutable**)
- 요소가 순서가 있음(**sequence type**) → index를 이용하여 접근 가능(**indexing**)
- 데이터를 연속적인 메모리 공간에 저장하고, 각 요소(element)는 인덱스를 이용하여 임의 접근(random access)할 수 있는 가장 많이 쓰이는 기본적인 자료구조 (배열)

```
for key in majors:
    print(key)
```

```
CS
EE
MAS
ME
```

2. 튜플 (Tuple): `position = (3.14, -5, 7.5)`

- 생성된 데이터는 추가, 삽입, 삭제하는 것이 불가능(**immutable**)
- 요소가 순서가 있음(**sequence type**) → index를 이용하여 접근 가능(**indexing**)

```
for ele in odds:
    print(ele)
```

```
1
3
5
7
9
```

3. 문자열 (String): `Course = "Python"`

- 생성된 데이터는 추가, 삽입, 삭제하는 것이 불가능(**immutable**)
- 요소가 순서가 있음(**sequence type**) → index를 이용하여 접근 가능(**indexing**)

4. 사전 (Dictionary):

```
majors = {"CS": "Computer Science",
          "EE": "Electrical Engineering",
          "MAS": "Mathematical Sciences",
          "ME": "Mechanical Engineering"}
```

- 언제든지 데이터를 추가, 삽입, 삭제하는 것이 가능(**mutable**)
- 요소가 순서가 없음
- (1) 쌍으로 이루어진 데이터, (2) 다양한 속성을 가지는 하나의 객체를 관리할 때 유용
- 데이터가 key와 value로 구성 – 따라서 **key로 value의 삽입, 삭제, 탐색**이 매우 빠른 자료구조 (해시테이블)

5. 집합 (Set): `odds = {1, 3, 5, 7, 9}`

- 언제든지 데이터를 추가, 삽입, 삭제하는 것이 가능(**mutable**)
- 요소가 순서가 없음

```
>>> odds = {1, 3, 3, 5, 7, 7, 9}
>>> odds
{1, 3, 5, 7, 9}
>>> odds = { 3, 7, 1, 9}
>>> odds
{1, 3, 9, 7}
```

묶인 데이터 타입의 메소드

	리스트	튜플	문자열	사전	집합
만들기 (빈 객체)	l=[] 또는 l=list()	t=() 또는 t=tuple()	s="" 또는 s=str()	d={} 또는 d=dict()	s={}는 사용 못함 s=set()
만들기	l1=list((1,2,3)) l2=list(range(1,4)) l3=list('ABCD') l4=[[1,2],[3,4]] l5=[[[1,2],[3,4]],[[5,6],[7,8]]]	t1=(1,2,3,4) t2=5,6,7,8 t3=tuple(l2)	s='Hello world!'	d1={'one':1, 'two':2, 'three':3} d2={'four':4, 'five':5}	s1={1,2,3,4} s2={3,4,5} s3={5,6,7}
접근하기	indexing l1[0] l4[0][1] l5[1][1][1]	indexing t[-1]	indexing s[2]	key를 사용 d1['one'] d1.get('one') d1.keys() d1.values() d1.items()	-
추가하기	l3.append('E') l2.insert(1,4) l1.extend([4,5])	-	-	d1.update(d2)	s1.add(5)
삭제하기	del l1[3] 또는 del(l1[3]) l3.remove('B') l3.pop(1) l3.clear()	-	-	del d1['one'] 또는 del(d1['one']) d1.pop('two') d1.clear()	s1.discard(4) s1.remove(1) s1.pop(2) s1.clear()
추출하기	slicing 리스트명[(시작값):(끝값+1)(:증가값)] l1[1:]	slicing 튜플명[(시작값):(끝값+1)(:증가값)] t1[1:]	slicing 문자열명[(시작값):(끝값+1)(:증가값)] s[1:-1:2]	-	-
기타	l2.sort() 오름차순 l2.sort(reverse=True) 내림차순 len(l1), min(l1), sum(l1)	len(t1)	len(s) s.split() 문자열→리스트 ''.join(l2) 리스트→문자열	len(d1)	s2.union(s3) s2.difference(s3) s2.intersection(s3) len(s2)