1. 리스트와 관련된 함수

.append(args)	args을 배열의 값으로 추가한다. 이때 args은 어떤 자료형이든 상관없다.
.count(args)	리스트 내부에서 args과 같은 값을 지니는 원소의 개수를 반환한다.
.index(args)	리스트 내부에서 args이 최초로 등장하는 인덱스를 반환한다.
.insert(index, args)	리스트의 index에 args값을 삽입한다.
.pop(), .pop(index)	리스트의 최우측값 or index에 위치한 값을 출력하고 삭제한다.
.reverse()	리스트 요소들을 거꾸로 정렬한다.
.sort()	리스트 요소들을 오름차순으로 정렬한다.

2. 튜플과 관련된 함수

.count(args)	튜플 내에 args과 일치하는 원소의 개수를 반환한다.
.index(args)	튜플 내에 특정 args값과 최초로 일치하는 원소의 인덱스값을 반환한다.

3. 딕셔너리와 관련된 함수

.has_key(key)	특정 key값이 딕셔너리 안에 존재하는지를 boolean값으로 반환한다. = key in dict
.items()	딕셔너리에 저장된 데이터들을 리스트로 반환한다.
.keys()	딕셔너리에 저장된 킷값들을 리스트로 반환한다.
.values()	딕셔너리에 저장된 밸류값들을 리스트로 반환한다.
.update(dict)	또다른 딕셔너리를 해당 딕셔너리에 합친다.
.pop(key)	딕셔너리에서 특정 key에 해당하는 value를 반환하고 삭제한다.
.get(key)	특정 키의 밸류값을 반환한다.
.clear()	모든 키와 밸류를 삭제한다.

4. 리스트 vs 튜플

(1) 공통점

- 받을 수 있는 원소의 타입이 뭐든 상관 없다.
- 순서가 존재한다.
- 슬라이싱이 가능하다.
- iterable하다 -> for문 사용이 가능하다.

(2) 차이점

- 리스트는 가변적(mutable)이며 튜플은 불가변적이다.
- 딕셔너리의 key값은 immutable해야하므로 리스트를 사용할 수 없다.
- 리스트를 복사하면 동일 값을 가진 새 객체가 생성되지만, 튜플을 복사하면 동일 identity를 가진 객체가 만들어진다.
- for문을 돌렸을 시에 튜플이 좀더 빠르다.
- 즉, 튜플을 사용하는 것이 메모리 측면에 있어서 경제적이다.

(3) 문화적 용례

- 리스트는 단일 종류의 요소를 가지는 경우, 그 요소가 몇 개인지 명확하지 않은 경우 사용한다.
- 튜플은 들어 있는 요소의 수를 사전에 알고 있는 경우에 사용한다.

```
성적정보 = {'학생1' : (30, 60, 70),
           '학생2' : (70, 80, 50),
           '학생3' : (50, 79, 20)}
```

- 각각의 인덱스를 국어, 영어, 수학점수라고 하면 각 인덱스만을 이용하여 과목별 평균을 구할 수 있다.

```
def korean_avg(dict):
        temp = 0
        for stu in dict:
                temp = temp +dict[stu][0]
        return temp/3
```

- 이때 평균점수가 50점을 넘는 학생들을 알아보고 싶다면 리스트를 반환해주는 함수를 사용하면 된다.

```
def who_over_50(dict):
        temp=[]
        for stu in dict:
                avg=0
                for grade in dict[stu]:
                        avg = avg+grade
                if(avg/3>50):
                        temp.append(stu)
        return temp
```

5. 딕셔너리의 용례

- 딕셔너리의 장점은 key와 value를 유기적으로 연결지어 많은 정보를 경제적으로 전달할 수 있다는 것이다.

- ex) 인간 = {'신체':{('머리', '머리카락') : ['눈', '코', '입'],
몸통':['팔','가슴','배'],
'하체':['허벅지','종아리','발']
},
'정신':{'자의식':{'정체성':['나','아들','백수'],'자존감':'높음'
},
'감정':['기쁨','분노','행복'],
'감각':['배고픔','통증','가려움']
}
}
- 인간['신체'][('머리','머리카락')] => [입', '코', '눈']
- 인간['신체']['몸통'] => [배', '가슴', '팔'}
- 인간['정신']['자의식']['자존감'] => '높음'

– 또 다른 사례 : http에서의 헤더 정보

– '요청 –> 응답' 메커니즘

– 이러한 방식으로 이미지, 영상, 텍스트 등 다양한 데이터를 전달할 수 있다.