데이터 구조

1. 리스트

- 대괄호 [ ]를 이용한다.
- [ ] 안에 [ ]를 넣으면 차원이 증가한다.
- 인덱스를 이용한 읽기와 쓰기를 지원한다.
- 부분 데이터셋을 뽑아내는 슬라이싱을 지원한다.

listData = [ ]	리스트를 만들어 줌
len()	길이(항목의 수)
min() / max()	최댓값 / 최솟값
+	두 리스트를 연결함
*	리스트를 곱한 수만큼 반복
append()	단일 항목을 맨 뒤에 추가. 리스트를 append하면 리스트가 항목으로 추가됨
extend()	리스트를 항목별로 맨 뒤에 추가
insert(index, value)	list의 index위치에 value를 삽입
[start:stop]	start 위치부터 stop 위치까지 부분 데이터를 추출 (stop 위치의 항목은 포함 안 됨)
count(value)	리스트에서 value의 개수를 반환
index(value, position=0)	position위치 이후에서 value의 값이 있는 인덱스를 반환
remove(value)	리스트에서 해당 값을 삭제
del listData[index]	리스트에서 인덱스를 이용해 항목을 삭제
pop()	리스트에서 가장 마지막 항목을 반환하고 삭제
clear()	list의 모든 항목을 삭제
reverse() / [::-1]	리스트의 항목들의 순서를 반대로 함. [::-1]는 역순으로 출력하고 원본데이터는 바꾸지 않는다
sort(reverse=False)	리스트의 항목들을 정렬. reverse 속성을 True로 하면 내림차순으로 정렬
copy()	복제된 새로운 객체를 생성
=	주소를 복사해 같은 객체를 참조

2. 튜플(tuple)

• 튜플은 소괄호( )를 이용해 만든다.

• 읽기 전용

- 속도가 빨라 수정이 필요 없는 배열 형태의 데이터 타입에 사용

- 데이터를 수정할 수 없기 때문에 제공되는 함수가 많지 않다.

tupleData = ( )	튜플을 만든다.
len()	튜플의 항목 수를 반환한다.
min() / max()	최댓값 / 최솟값
count(value)	value의 개수 반환
index(value, position	position 위치 이후에서 value가 있는 인덱스 반환

3. 딕셔너리

• 키(key)와 값(value)의 쌍으로 구성된 자료 구조

• 중괄호 { }를 이용

• 키 (key)

- 중복이 없이 유일한 값

- 리스트 타입을 사용할 수 없지만 튜플 타입은 사용할 수 있다.

• 값은 중복이 가능하며 모든 타입이 가능하다.

• 인덱스를 이용한 데이터 참조는 지원하지 않는다.

• 키 목록에 없는 데이터를 사용하여 참조하면 에러가 발생

dictData = {"key" : "value",…}	딕셔너리를 만든다
len()	항목의 수를 반환한다.
items()	각 항목들을 (key, value) 형식의 튜플로 반환한다.
keys()	키를 반환한다.
values()	값을 반환한다.

4. 셋

- 순서가 정해지지 않고, 중복을 허용하지 않는 집합
- { }를 이용하여 정의

5. enumerate

- 반복자(iterator) 또는 순서(sequence) 객체를 인수로 받는다.
- enumerate(iter)라고 사용했을 경우, iter 객체를 (0, iter[0]), (1, iter[1]), (2, iter[2]), ... 형식으로 반환