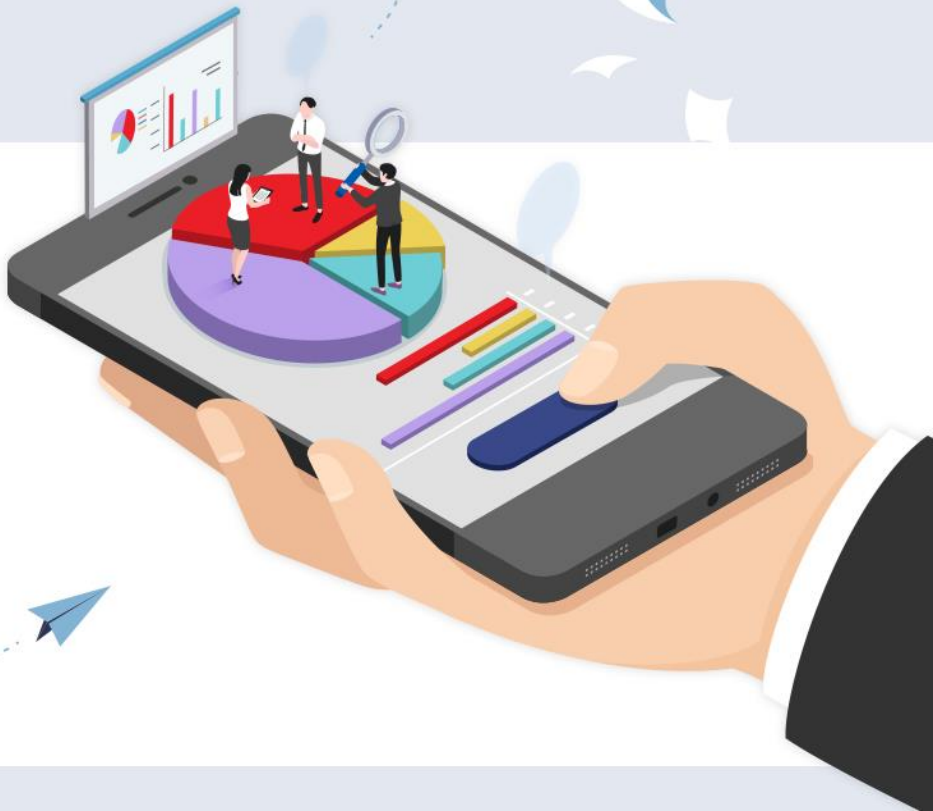
01

컬렉션 자료형 개념과 필요성



1 컬렉션 자료형 개념과 필요성

프로그래밍을 할 때 쓰이는
숫자, 문자열 등
자료 형태로 사용하는 모든 것

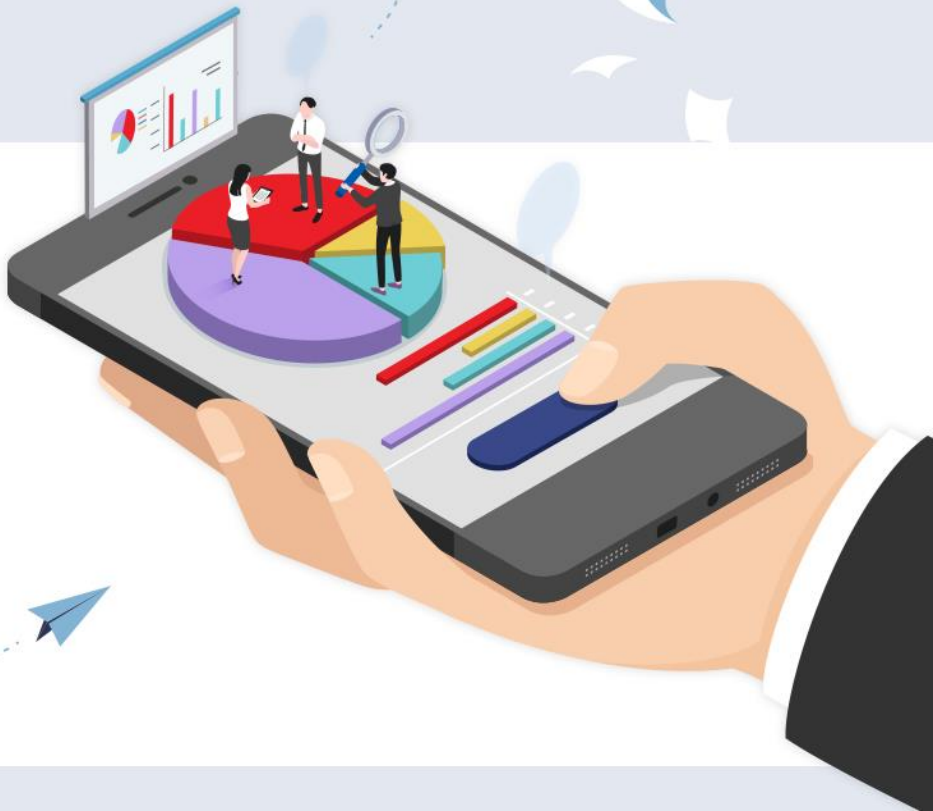
컬렉션 자료형

여러 개의 값을 하나의 변수에 담을 수 있으며
변수 안에 공간을 여러 개 가지며,
변수 안에 서로 다른 공간을 찾는 방법

1 컬렉션 자료형 개념과 필요성

컬렉션 자료형 종류	생성방법
리스트(List)	[ ]
튜플(Tuple)	( )
딕셔너리(Dictionary)	{키:값}
세트(Set)	{ }

02 리스트 자료형



리스트 자료형

- 파이썬 프로그래밍 언어 내에서 가장 많이 쓰이는 구조
- 많은 양의 데이터들을 한 번에 모아 효율적으로 처리 및 저장 가능

오직 하나의 리스트 변수를 이용하여 1,000명의 점수 데이터를 효율적으로 저장하고 처리하는 리스트

오직 하나의 데이터를 저장하기 때문에 1,000명의 점수 데이터를 저장하기 위해서는 1,000개의 변수가 필요한 기존의 변수

☑ 대괄호 [ ] 안에 서로 다른 자료형의 값을 콤마(,)로 구분해 하나 이상 저장할 수 있는 컬렉션 자료형

- 요소(Element)
  - 대괄호 [ ]에 넣는 자료
  - 요소들은 순서를 가지고 있고 인덱스를 사용하여 참조 가능

리스트 문법

리스트명 = [값1, 값2, 값3, ...]

3 리스트 인덱싱(Indexing) 및 슬라이싱(Slicing)

인덱싱(Indexing) 및 **슬라이싱(Slicing)**

무엇인가 '가리킨다'　　무엇인가 '잘라낸다'

3 리스트 인덱싱(Indexing) 및 슬라이싱(Slicing)

리스트 인덱싱(Indexing)

리스트 인덱스(Index) 구조

```
>>> a=[10,20,30,40]
```

인덱스(Index) →	0	1	2	3
	10	20	30	40
인덱스(Index) →	-4	-3	-2	-1

예 리스트 인덱싱(Indexing) 사용하기

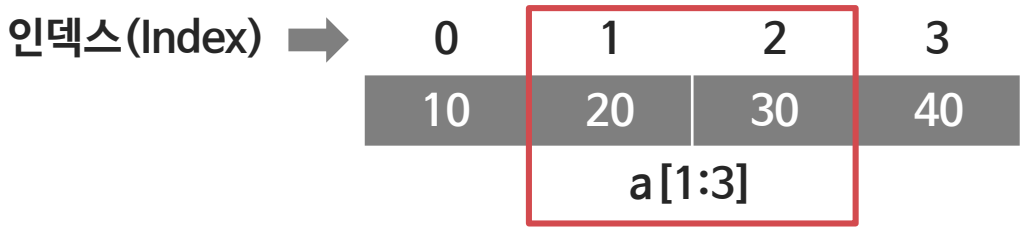
```
>>> a=[10,20,30,40]
>>> a
[10, 20, 30, 40]
>>> a[0] # 리스트 첫 번째 값을 가져온다.
10
>>> a[0]+a[1]
30
>>> a[-1] # 리스트 마지막 값을 가져온다.
40
>>> a[4]
Traceback (most recent call last):
File "<pyshell#21>", line 1, in <module>
a[4]
IndexError: list index out of range
```

3 리스트 인덱싱(Indexing) 및 슬라이싱(Slicing)

리스트 슬라이싱(Slicing)

리스트 슬라이싱(Slicing) 구조

```
>>> a=[10,20,30,40]
>>> a[1:3] #슬라이싱은 새로운 리스트를 반환한다.
[20,30]
```



- 리스트 안에서 범위를 지정하여 원하는 요소들을 선택하는 연산
- 리스트명[시작:끝]와 같은 형식을 사용
- 시작요소부터 (끝-1) 인덱스에 있는 요소까지 선택

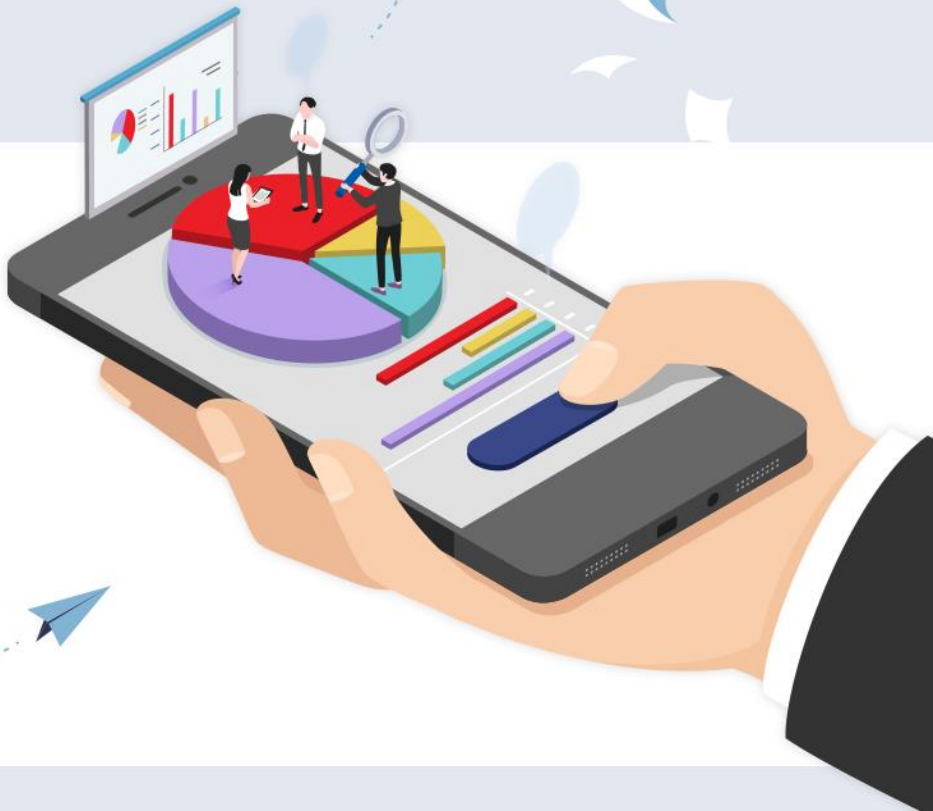
4 리스트 조작함수

- [x] 리스트변수 이름 뒤에 마침표(.)를 붙인 다음 함수 이름을 사용

4 리스트 조작함수

함수	설명	사용법
append()	리스트에 요소를 마지막 위치에 새로 추가	리스트.append(값)
insert()	리스트의 해당 위치에 요소를 새로 삽입	리스트.insert(위치,값)
sort()	오름차순정렬 내림차순정렬	리스트.sort() 리스트.sort(reverse=True)
count()	해당 요소의 개수를 반환	리스트.count(찾을값)
pop()	리스트 제일 뒤의 항목을 빼내고, 빼낸 항목은 삭제 제거할 위치에 있는 요소를 제거	리스트.pop() 리스트.pop(위치)
remove()	해당 요소를 찾아 삭제	리스트.remove(삭제할값)

03 튜플 자료형



튜플 자료형

- 한번 저장된 값은 수정할 수 없는 자료형
- 읽기 전용의 데이터를 저장할 때 유용하게 사용

프로그램이 실행되는 동안
데이터가 항상
변하지 않아야 한다면
튜플 사용

수시로 데이터를
변화시켜야 할 경우라면
리스트 사용

2 문법

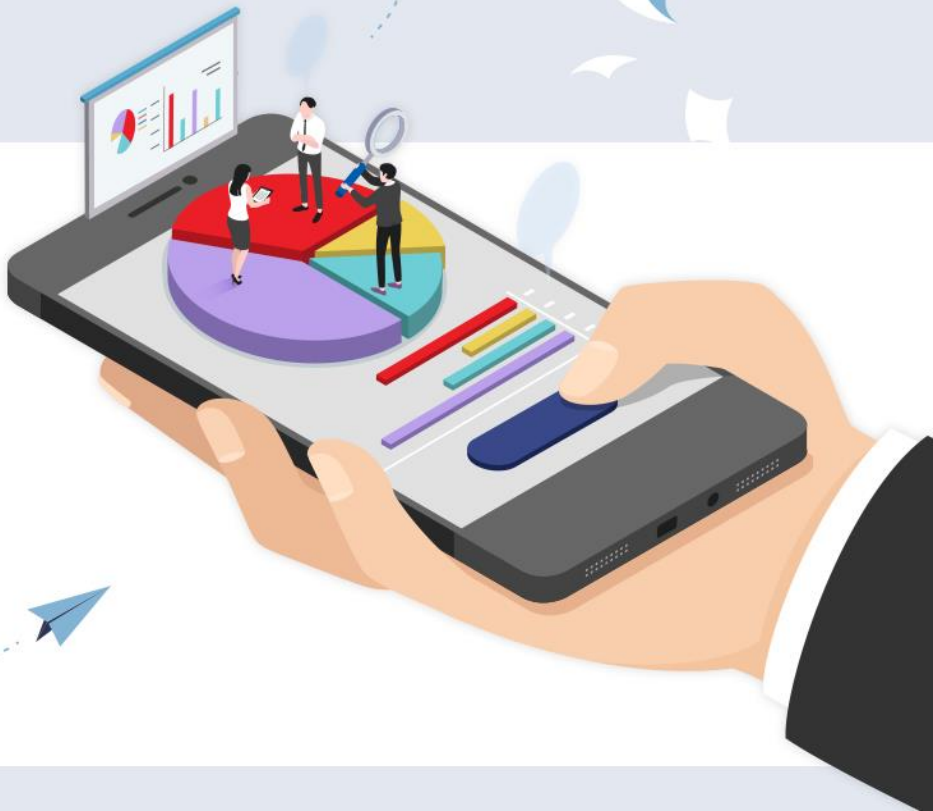
- ( ) 안에 서로 다른 자료형의 값을 콤마(,)로 구분해 하나 이상 저장할 수 있는 컬렉션 자료형

- 0부터 시작하는 인덱스를 이용해 접근할 수 있고 한 번 저장된 요소는 변경할 수 없음

튜플 문법

튜플명 = (값1, 값2, 값3, ...)

04 딕셔너리 자료형



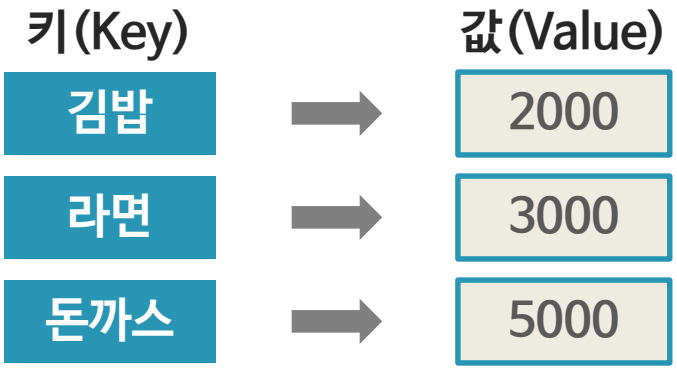
☑ 딕셔너리(dictionary)는 한글로 표현하면 '사전'

- 한글, 영어 사전들은 가나다순 알파벳순으로 정렬
- 순서가 없는 컬렉션 자료형으로 각각의 요소는 key:value 형태로 저장
- 리스트나 튜플처럼 index에 의해 해당 요소를 찾지 않고 key를 통해 value를 획득

- ☑ {}안에 키:값 형식의 항목을 콤마(,)로 구분해 하나 이상 저장할 수 있는 컬렉션 자료형

- ☑ 키를 먼저 지정하고 :(콜론)을 붙여서 값을 표현
  - ▪ 키와 값은 1:1 대응관계

딕셔너리 구조



딕셔너리 문법

딕셔너리명 = {key1:value1, key2:value2, key3:value3,...}

3 딕셔너리 요소에 접근하기

☑ 딕셔너리를 생성할 때는 중괄호를 사용하지만,
딕셔너리 요소에 접근할 때는 리스트처럼 딕셔너리 뒤에
대괄호[ ]를 입력하고 내부에 인덱스 대신 키를 입력

- 키를 이용해 값을 읽어올 수 있음
- 요소를 추가할 때 동일키가 없으면 새로운 요소를
추가하고, 동일키가 있으면 저장된 요소를 변경함

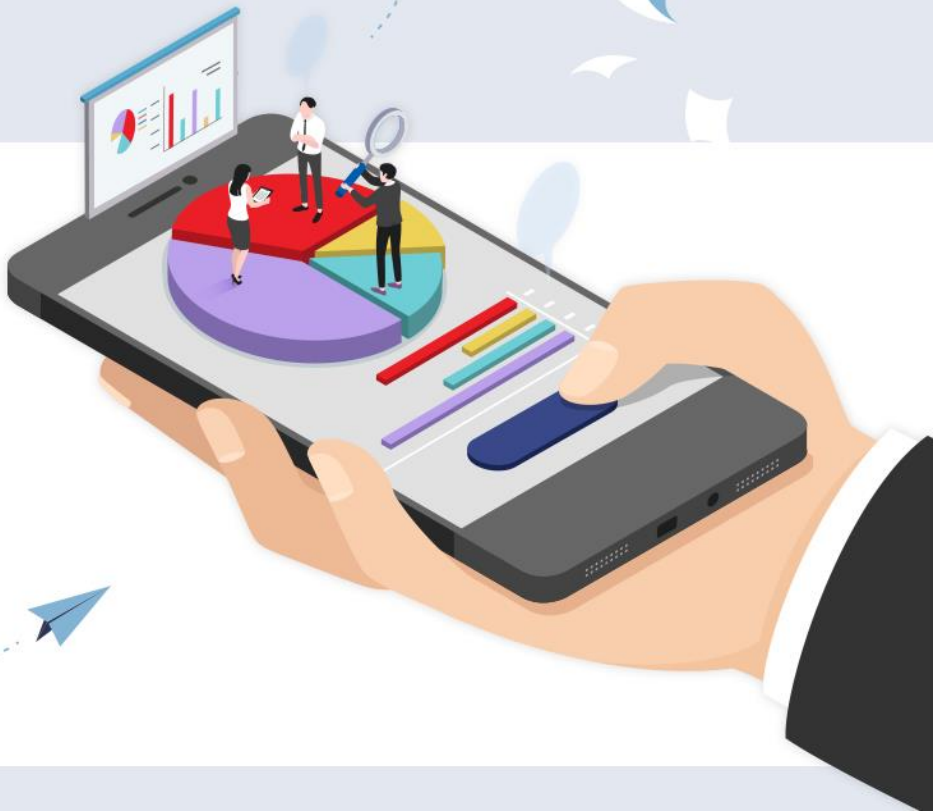
예 딕셔너리 요소에 접근

```
>>> menu={'김밥':2000,'라면':3000}
>>> menu['김밥']
2000
>>> menu['라면']=3500
>>> menu['라면']
3500
>>> menu['어묵']=1000
>>> menu
{'김밥': 2000, '라면': 3500, '어묵': 1000}
>>> menu['떡볶이']
Traceback (most recent call last):
File "<pyshell#13>", line 1, in <module>
menu['떡볶이']
KeyError: '떡볶이'
```

4 딕셔너리 조작함수

함수	설명	사용법
get()	항목접근하기	딕셔너리.get(key)
pop() del ()	항목 꺼내고 삭제하기 항목삭제하기	딕셔너리.pop(key) del(딕셔너리 [key])
items()	딕셔너리에 저장된 항목	딕셔너리.items()
keys()	딕셔너리에 저장된 키	딕셔너리.keys()
values()	딕셔너리에 저장된 값	딕셔너리.values()

05 세트 자료형



세트 (Set) 자료형

집합에 관련된 것을 쉽게 처리하기 위해 만든 자료형으로 중복을 허용하지 않는 컬렉션 자료형

순서가 없기 때문에 인덱싱으로 값을 얻을 수 없음 (리스트나 튜플은 순서가 있기 때문에 가능)

☑ 특징

- 수학의 집합과 같음
- 중복되지 않은 요소들로 구성됨
- 세트 간의 순서가 없음

세트 문법

세트명 = { }

세트 연산

```
>>> A={10,20,30}
>>> B={20,40}
>>> A & B # 교집합
{20}
>>> A.intersection(B) # 교집합
{20}
>>> A | B # 합집합
{20, 40, 10, 30}
>>> A.union(B) # 합집합
{20, 40, 10, 30}
>>> A - B # 차집합
{10, 30}
>>> A.difference(B) # 차집합
{10, 30}
```