

6

6. 고급 자료형 (1) - 그 외

① 튜플 ② 딕셔너리 ③ 집합자료형

①

튜플(tuple) 자료형

◆ 리스트와 비슷하다. 여러 개의 자료를 저장한다.

- () 로 둘러싼다
- 값을 변경할 수 없다 (리스트는 가능)

```
>>> t1 = ( )
```

```
>>> t2 = (1, )
```

```
>>> t3 = (1, 2, 3)
```

```
>>> t4 = 1, 2, 3
```

```
>>> t5 = ('a', 'b', ('ab', 'cd'))
```

why?
 () 안 쓰면 주식에서
 하나일 때에는 , 를 붙인다. 우선순위를 정하는 ()라
 했갈릴 수 있기 때문!

()를 생략할 수 있다.

↳ 가능하면 괄호를 써주게 Better!

튜플이 1개 값으로
 들어간 것!

고정된 값을 계속
 사용할 때는 튜플!

변하지 않는 값을 사용할 때 튜플이다.

리스트가 기능이 더 많은데 리스트가 더 좋은 것 아닌가?

튜플 자료형

◆ 할 수 없는 것 *고정된 값이냐*

– 요소 삭제

>>> del t1[0] 첫 번째 요소 삭제

– 요소 값 변경

>>> t1[0] = 1 첫 번째 요소 값 변경

◆ 할 수 있는 것

– 인덱싱 >>> t1[0]

– 슬라이싱 >>> t1[1:]

– 튜플 더하기 >>> t3 = t1 + t2

– 튜플 곱하기 >>> t2 * 3 *t2가 3번 사용!*

>>> a = 1,2,3,4

>>> 끝에 5, 6를 추가하여 b에 넣고 싶다면? a.append?

>>> 끝에 7을 추가하여 c에 넣고 싶다면?

② 딕셔너리

리스트

인덱스	값
0	'Life'
1	'is'
2	'too'
3	'short'

딕셔너리

Key	Value
김연아	피겨스케이팅
류현진	야구
박지성	축구
박인비	골프

단어	뜻
----	---

- Key란 Value를 찾기 위한 수단 (순서는 의미가 없다) ^{***} *찾는 것이 1차적 목적 key-value가 묶여있음!*
- Value는 의미 있는 데이터
- Dictionary란 Key 와 Value의 세트로 이루어진 데이터
- Key와 Value의 세트를 구성한 후, 필요한 데이터를 찾기 위해 Key를 넣는다.

딕셔너리 생성

◆ Key와 Value의 묶음

- { } 를 사용한다. : 를 꼭 붙인다 (:가 없으면 뒤에 나오는 Set과 혼돈).
← 딕셔너리

```
>>> dic = {'name':'pey', 'phone':'01044445555', 'birth':'1010'}
```

Key	Value
name	pey
phone	01044445555
birth	1010

```
>>> a = {1:'hi'}  
>>> a = {'a':[1, 2, 3]}
```

← value에 리스트 가능!

Quiz) 딕셔너리 생성

실습1

◆ 다음의 딕셔너리를 생성하라.

- 여기는 주민센터이다. 많은 주민이 이 지역에 살고 있고, 주민센터 업무를 위하여 주민의 정보를 검색할 일이 종종 있다.
- 주민 검색은 주민등록번호로 진행하며, 주민의 이름, 주소, 전화번호를 확인한다.

Key	Value
'950102-1010101'	['홍길동', '흑석동 1-1', '010-4444-5555']
'961212-2211221'	['홍길녀', '흑석동 2-2', '010-8888-1111']

Q. 딕셔너리는 언제 사용할까?

- 무엇을 기준으로 Key를 정했는가
- Value는 어떤 자료형이어야 하는가

① Data가 많을 때!

② key로 Value 찾을 때!

↳ 빠르게 찾을 수 있음

딕셔너리 데이터 추가 삭제

◆ 추가

```
>>> a = {1:'a'}  
>>> a[2] = 'b'  
>>> a['name'] = 'pey'  
>>> a[3] = [1, 2, 3]
```

딕셔너리 a 생성, 1:'a'
2:'b' 추가
'name':'pey' 추가
3:[1,2,3] 추가

Key	Value
1	'a'
2	'b'
3	[1,2,3]
'name'	'pey'

◆ 삭제

```
>>> del a[1]
```

1:'a'행 삭제

딕셔너리의 활용

◆ 홍길동의 전화번호는?

```
>>> teldic['홍길동']
```

```
name = input('이름을? ')

```

```
print(teldic[name])
```

⇒ name으로 '홍길동'을 찾다면,
전화번호(Value)가 나옴!

teldic 딕셔너리

Key	Value
강감찬	010-3333-4444
김유신	010-3928-0391
이순신	010-8472-3092
홍길동	010-2727-0948

리스트를 이용한 딕셔너리의 구성

- ◆ 2개의 리스트로 딕셔너리를 만들 수도 있다.
- ◆ 2개보다는 1개가 간단하고, key로 직접 검색을 하는 것이 더 편하다. 그래서 딕셔너리가 존재한다.

Key	Value
강감찬	010-3333-4444
김유신	010-3928-0391
이순신	010-8472-3092
홍길동	010-2727-0948

teledic['홍길동']

name

index	Value
0	강감찬
1	김유신
2	이순신
3	홍길동

phone

index	Value
0	010-3333-4444
1	010-3928-0391
2	010-8472-3092
3	010-2727-0948

phone[name.index('홍길동')]
[이것보다는 딕셔너리를 사용하길 Better!]

딕셔너리의 특성 ① 고유성 ② 불변성

◆ Key는 고유해야 한다.

- 고유하지 않아도 에러가 나는 것은 아니다. 데이터의 신뢰도가 떨어질 뿐

↳ 동일한 key가 있다면, 제일 처음의 key에만 응답!

그래서 최대한 고유의 key가 사용되도록 하는 것이 좋다!!

```
>>> a = {1:'a', 1:'b'}
```

```
>>> a
```

하나의 결과만 나온다. => 신뢰도 하락

기존데이터 날리고

○ 변경된 data 추가하기

◆ Key는 변하지 않아야 한다.

주의

Key를 바꾸고 싶다면?

- 리스트는 Key로 쓸 수 없다. 튜플은 가능하다. 왜? key는 변경할 수 없기 때문에! 바꾸는 작업 불가!

Key	Value
[강감찬, 90년생] X	010-3333-4444
[김유신, 93년생] X	010-3928-0391
[이순신, 94년생] X	010-8472-3092
[홍길동, 97년생] X	010-2727-0948

Key	Value
(강감찬, 90년생) ○	010-3333-4444
(김유신, 93년생) ○	010-3928-0391
(이순신, 94년생) ○	010-8472-3092
(홍길동, 97년생) ○	010-2727-0948

딕셔너리 관련 함수

Key/Value 뽑는 것의 응용.
why? 너무 다양하니까!!!

◆ Key 리스트 만들기(Key만 뽑기)


>>> a.keys() 딕셔너리의 Key만 추출하여 dict_keys 를 만든다.

dict_keys는 리스트가 아니다.

>>> list(a.keys()) 리스트로 바꾸려면  정렬해서 나열 안함.

◆ Value 리스트 만들기

>>> a.values() 딕셔너리의 value만 추출하여 dict_values 객체를 만든다.

 in 뒤에는 리스트가 있어야함!

```
for i in list(a.keys( ) ) :  
    print( i, a[i] )
```

⇒ dict을 차례대로 화면에
표시하고 싶다고 하면!

요청자의
의미를
100% 이해
하세요!

생성한 리스트들은 반복문에서 바로 사용 가능하다.(반복문 참고)

딕셔너리 관련 함수

◆ 딕셔너리 비우기

```
>>> a.clear()
```

◆ Key로 Value 얻기

①

```
>>> a.get('name')
```

 ②

```
= a['name']
```

차이점은?

```
>>> a.get('name', 'anonymous')
```

못 찾으면 이 것을 반납

① `a.get('name')`

: 없는거 넣으면 none이 나온
→ 실패시 프로그램이 죽지않고 자명하게 넣어줌

② `a['name']`

: 안에 key가 없으면 프로그램 중단

어떻게든
프로그램이
돌아가게 해야함

→ 못찾으면 anonymous로
나오는것!

◆ Key 존재여부 확인

```
>>> 'name' in a
```

① True / False로
나옴!

딕셔너리에 'name' Key가 있느냐
(for에 쓰인 in 과 다르다)

딕셔너리의 용도

- ◆ 검색할 키(비교적 단순)와 데이터 부분(복잡)으로 나뉘어져 있는 경우
- ◆ 데이터가 아주 많은 경우 *사용!*
 - 데이터가 적으면 검색의 큰 의미가 없다.
 - 딕셔너리는 지정된 키를 빨리 찾는 저장 방식을 쓴다.

실습 2

- ◆ 다음 딕셔너리의 내용을 변수에 넣은 후 이름을 입력 받고 전화번호를 출력하는 프로그램을 작성하라.

Key	Value
강감찬	010-3333-4444
김유신	010-3928-0391
이순신	010-8472-3092
홍길동	010-2727-0948

이름은? 홍길동

홍길동의 전화번호는 010-2727-0948

이름은? 강감찬

강감찬의 전화번호는 010-3333-4444

이름은? 김승태

김승태는 없어요.

집합 자료형 ^③

그냥 있다고
알아놓으세요~

◆ 집합을 처리하기 위한 자료형

- set 키워드로 생성

```
>>> s1 = {1,2,3}
```

```
>>> s2 = set([1,2,3])
```

```
>>> s3 = set("Hello")
```

집합 자료형은 {} 로 표현

!

◆ 집합 자료형의 특징

- 중복을 허용하지 않는다.
- 순서가 없다.
- 인덱싱이 불가능하다.
- 인덱싱을 하려면 리스트나 튜플로 변환 후에 해야 한다.

```
>>> lst1 = list(s1)
```

```
>>> lst1[0]
```

이걸에 변환하면
튜플/리스트 쓰겠지...?

◆ 교집합, 합집합, 차집합

– 교집합 연산자 & 또는 함수 intersection

```
>>> s1 & s2
```

```
>>> s1.intersection(s2)
```

– 합집합 연산자 | 또는 함수 union

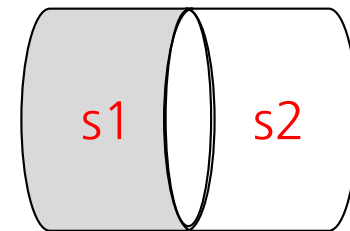
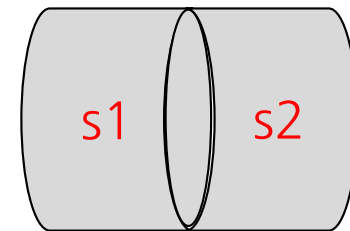
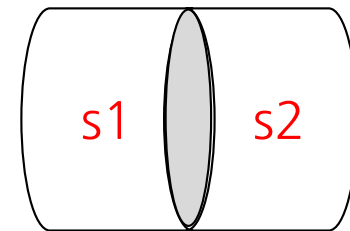
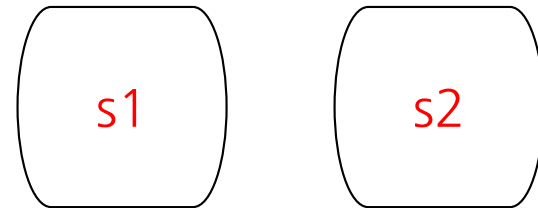
```
>>> s1 | s2
```

```
>>> s1.union(s2)
```

– 차집합 연산자 - 또는 함수 difference

```
>>> s1 - s2
```

```
>>> s1.difference(s2)
```



집합 자료형 관련 함수

잘못된
pass!

◆ 원소 하나 추가하기 - add

```
>>> s1.add(4)
```

union 으로도 가능

◆ 원소 여러 개 추가하기 - update

```
>>> s1.update([4,5,6])
```

update 함수는 1개만 입력을 받으므로
[]로 묶어주어 s1에 공급해줘야 함!

◆ 특정 원소 제거하기 - remove

```
>>> s1.remove(2)
```

◆ 왜 이런 다양한 자료형을 알아야 하나 *○ 사용자가 다양한 자료형을 필요로 하기 때문에!*

- 수많은 데이터가 있다. 이 데이터는 컴퓨터에 옮겨야 고속의 처리와 분석이 가능하다.
- 컴퓨터에 어떻게 저장하느냐에 따라 효과적으로 사용할 수도 있고 그렇지 않을 수도 있다.
- 여기서 나열한 자료형들은 수십 년간 컴퓨터에서 이용한 자료형을 분석하여 빈번하게 이용된 것들을 뽑아 만든 것이다.

◆ 우리가 할 일

- 자료를 어디에 어떻게 저장할 것인가
- 그 자료 저장 형태는 우리의 요구에 부합하는가
- 더 효과적인 자료 저장 형태/자료 사용 방법은 무엇인가

◆ 활용

- 데이터 분석 모듈에 대량의 데이터 공급 용도

◆ 언제 다음의 자료형을 사용할까?

- 리스트
 - 보편적인 다수의 데이터
- 튜플
 - 바뀌지 않는 상수, 하나의 데이터에 포함된 여러 개의 값
- 딕셔너리
 - 다수의 데이터와 빠른 검색
- 집합

◆ 1에서 100000 사이의 숫자 중

- 27과 86의 최소공배수를 화면에 표시하라.
- for 문을 생각하지 않아도 된다. hint : range, set, sort

◆ 이름을 넣으면 전화번호와 주소를 찾는 프로그램

◆ 실행 예

- 이름? 홍길동
- 주소 : 서울시 동작구 흑석로 84
- 전화번호 : 02-820-0001
- 이름? 심청
- 새로운 데이터입니다.
- 주소를 입력하세요 : 동해 물속 용왕로
- 전화번호를 입력하세요 : 02-1111-1111
- 자료를 어떤 식으로 저장하면 좋을까?