새내기 파이썬

8장 튜플, 세트, 딕셔너리

학습 목표

- 리스트와 튜플의 차이점을 말할 수 있나요?
- 세트로 공통적인 요소를 삭제할 수 있나요?
- 딕셔너리로 영어 사전을 만들 수 있나요?
- 딕셔너리로 상품의 여러 속성을 저장할 수 있나요?
- 딕셔너리로 연락처 프로그램을 만들 수 있나요?

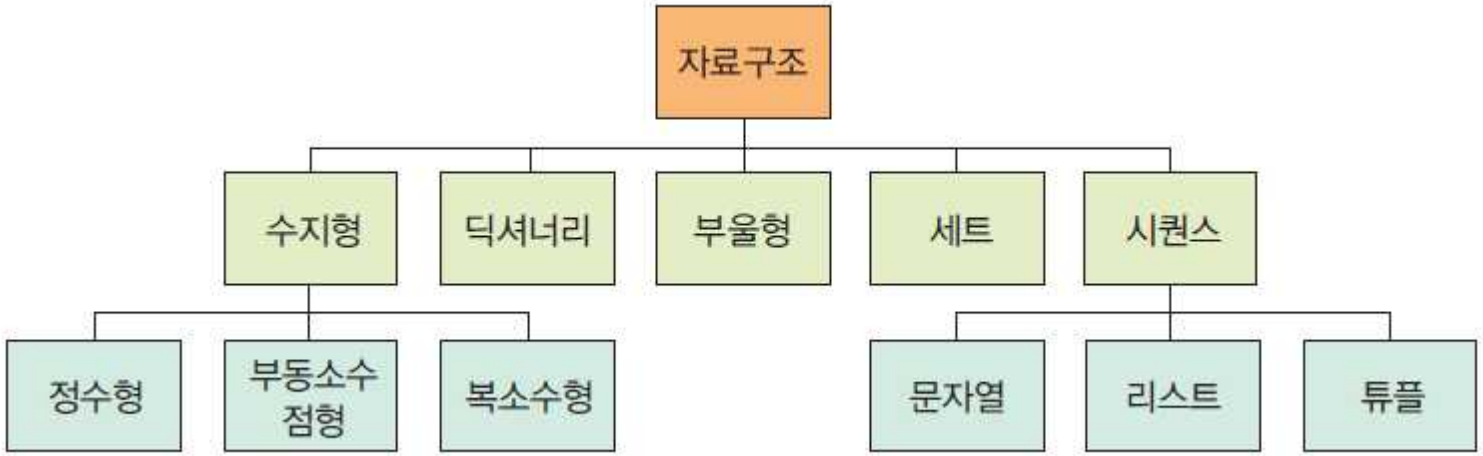
이번 장에서 만들 프로그램

```
첫 번째 문자열:Alice was beginning to get very tired of sitting by her sister on the
bank, and of having nothing to do
두 번째 문자열:Kim was beginning to get very tired of sitting by her sister on the
bank, and of having nothing to do
표절률 = 85.71428571428571
```

```
1. 연락처 추가
2. 연락처 삭제
3. 연락처 검색
4. 연락처 출력
5. 종료
메뉴 항목을 선택하시오: 1
이름: KIM
전화번호: 123-4567
```

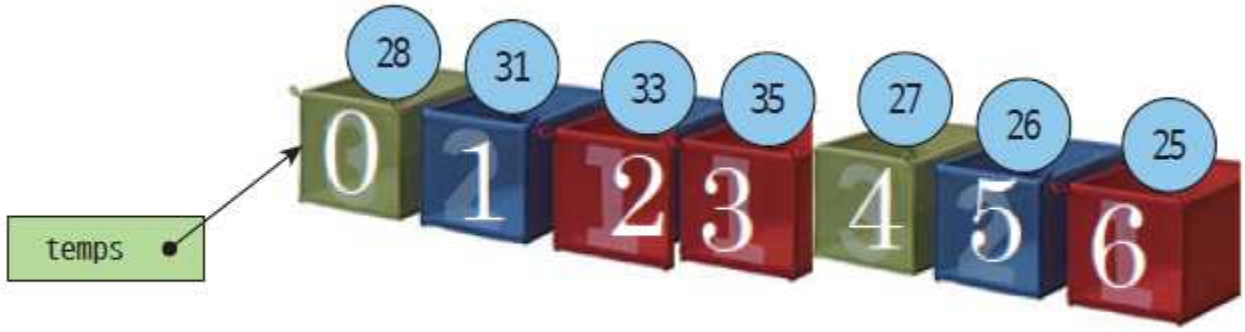
자료구조

- 자료들을 저장하는 여러 가지 구조들을 자료구**조(data structure),** 또는 데이터구조라 부른다.



시퀀스

- 시퀀스(**sequence**):
  - **요소(element)**로 구성
  - 요소 간에는 순서가 있다.
  - 시퀀스의 요소들은 번호가 붙여져 있다.
  - 내장 시퀀스**(str, bytes, bytearray, list, tuple, range)**

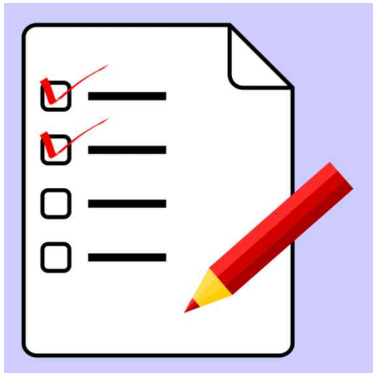


시퀀스

- 동일한 연산을 지원,
  - 인덱싱**(indexing)**, 슬라이싱**(slicing)**, 덧셈 연산**(adding)**, 곱셈 연산**(multiplying)**
- 내장함수 적용가능 : 시퀀스의 길이를 반환하는 len() 함수, 최대값과 최소값을 찾는 max()와 min() 함수

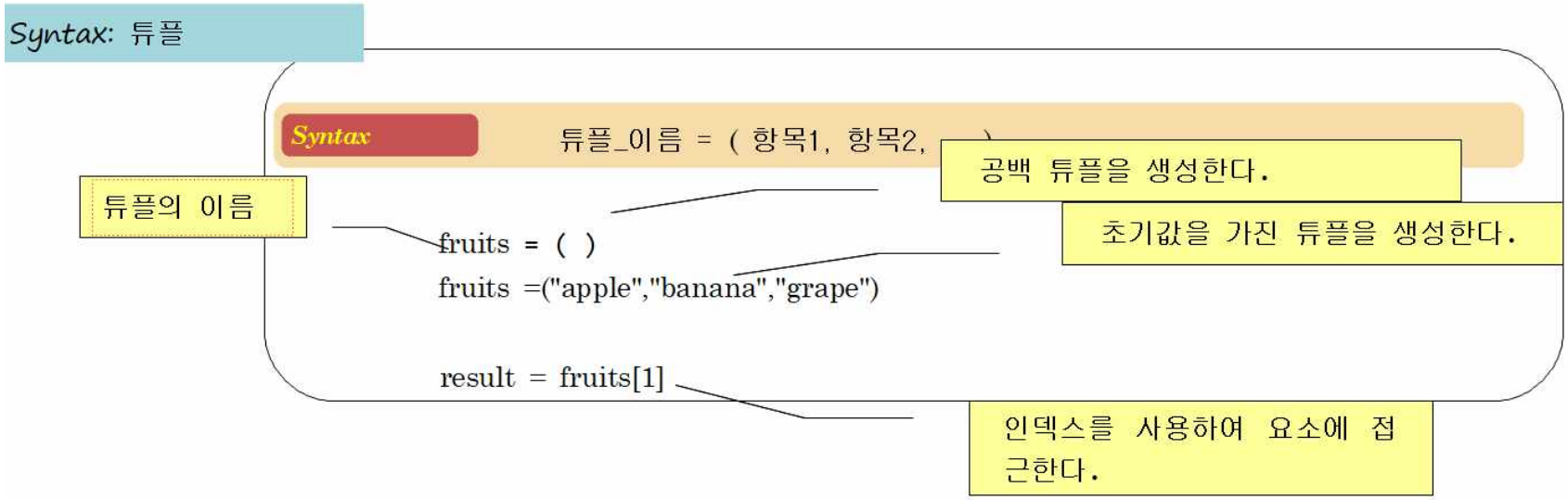
중간점검

1. 리스트는 시퀀스에 속하는가?
2. 시퀀스의 특징에는 어떤 것들이 있는가?



튜플

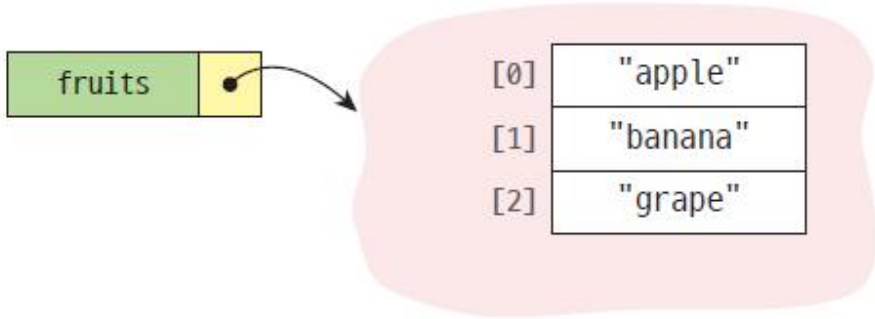
- 리스트와 튜플(tuple)은 아주 유사하다. 하지만 리스트와는 다르게 튜플은 변경이 불가능하다.



튜플 생성

```
fruits = ("apple", "banana", "grape")
```

```
fruits = "apple", "banana", "grape"
```



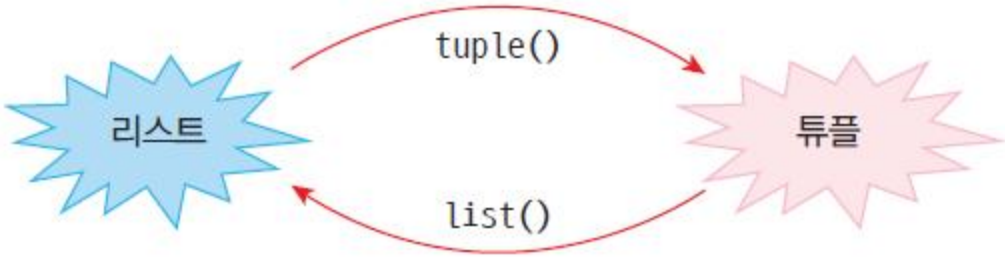
주의할 점

```
>>> single_tuple = ("apple",)   # 쉼표가 끝에 있어야 한다.
>>> single_tuple
("apple",)
>>> no_tuple = ("apple")                    # 쉼표가 없으면 튜플이 아니라 수식이 된다.
>>> no_tuple
"apple"
```

```
>>> fruits = ("apple", "banana", "grape")
>>> fruits[1]
banana


>>> fruits[1] = "pear"                      # 오류 발생!
TypeError: "tuple" object does not support item assignment
```

튜플 <-> 리스트

```
>>> myList = [1, 2, 3, 4]
>>> myTuple = tuple(myList)          # tuple()는 튜플을 생성하는 함수이다.
>>> myTuple
(1, 2, 3, 4)
```




```
>>> myTuple = (1, 2, 3, 4)
>>> myList = list(myTuple)           # list()는 리스트를 생성하는 함수이다.
>>> myList
[1, 2, 3, 4]
```

튜플 추가 연산들

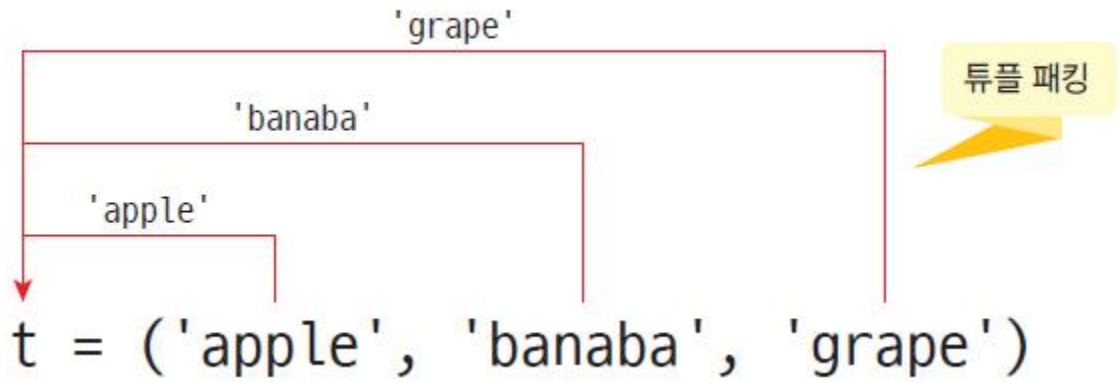
```
>>> fruits = ("apple", "banana", "grape")
>>> fruits += ("pear", "kiwi")
>>> fruits
("apple", "banana", "grape", "pear", "kiwi")
```

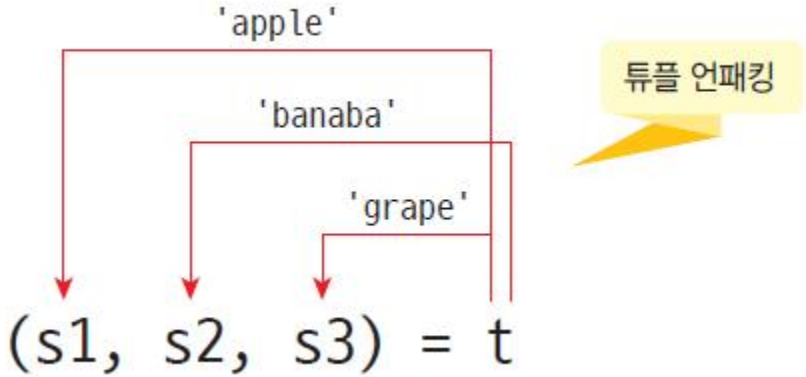
다른 튜플에 합치는 것은 가능

```
>>> numbers = [10, 20, 30]
>>> numbers += (40, 50)
>>> numbers
[10, 20, 30, 40, 50]
```

리스트에 튜플을 합치는 것은 가능

튜플 패킹과 언패킹


'grape'
'banaba'
'apple'
튜플 패킹
t = ('apple', 'banaba', 'grape')


'apple'
'banaba'
'grape'
튜플 언패킹
(s1, s2, s3) = t

예제

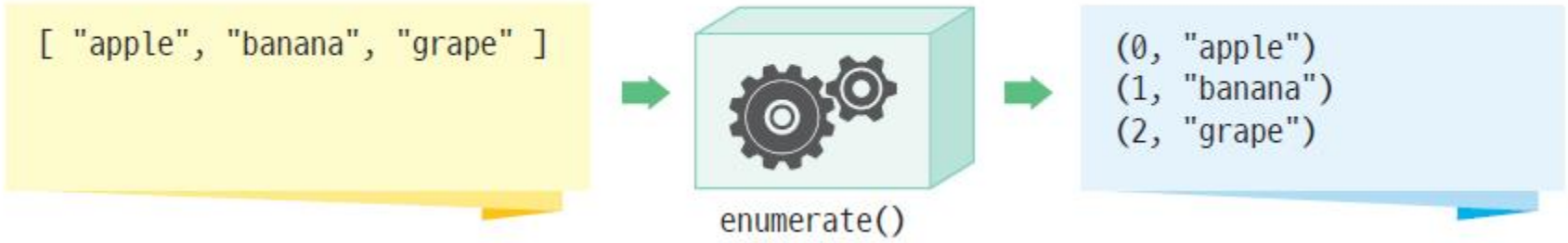
```
n1 = 10
n2 = 90
n1, n2 = (n2, n1)                    # (90 10)
```

튜플을 이용하여 데이터의 순서를 바꾼다.

```
n1, n2 = sub()
```

반환받는 것도 튜플을 통하여

enumerate() 사용하기




```
fruits =["apple","banana","grape"]
for index, value in enumerate(fruits):
        print(index, value)
```

```
0 apple
1 banana
2 grape
```

튜플의 장점

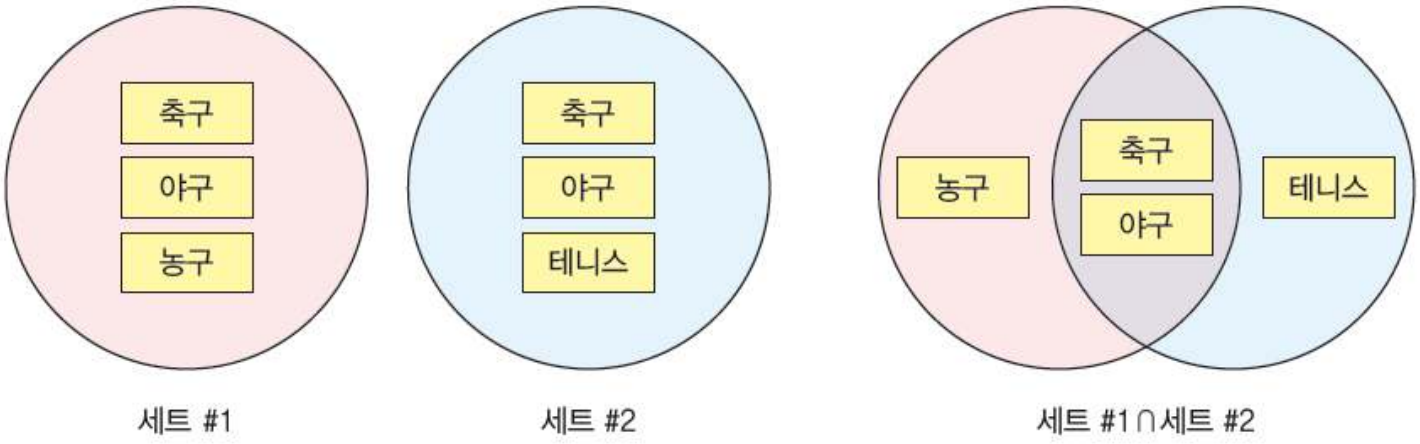
	리스트	튜플
문법	항목을 [ ]으로 감싼다.	항목을 ( )으로 감싼다.
변경여부	변경 가능한 객체	변경 불가능한 객체
메소드	약 46개의 메소드 지원	약 33개의 메소드 지원
용도	딕셔너리에서 키로 이용할 수 없다.	딕셔너리에서 키로 이용할 수 있다.

중간점검

1. 리스트와 튜플의 다른 점은 무엇인가?
2. 리스트를 튜플로 바꾸려면 어떤 함수를 사용하는가?
3. 패킹과 언패킹을 설명해보자.
4. enumerate() 함수가 하는 역할은 무엇인가?

세트

- 세트(set)는 우리가 수학에서 배웠던 집합이다. 세트는 고유한 값들을 저장하는 자료구조라고 할 수 있다.
- 리스트와는 다르게 세트의 요소는 특정 순서로 저장되지 않으며 위치별로 액세스할 수 없다



세트 생성하기

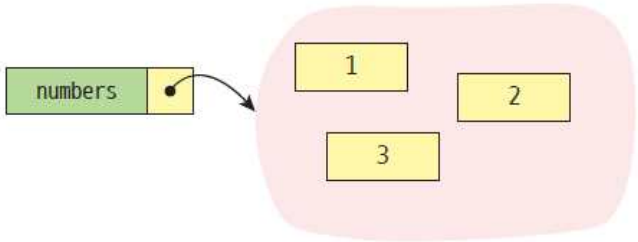
Syntax: 세트

Syntax 세트_이름 = { 항목1, 항목2, 항목3, ... }

```
numbers = {1, 2, 3}
values = set( )
```

초기화된 세트를 생성한다.

공백 세트를 생성한다.

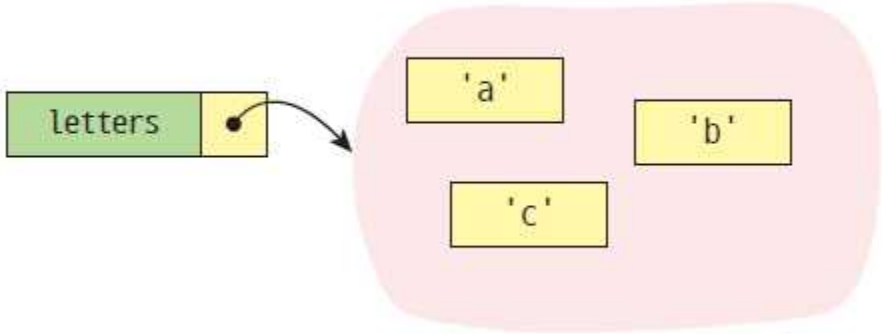


리스트<-> 세트

```
numbers =set([1,2,3,1,2,3])
print(numbers)
```

```
{ 1, 2, 3}
```

```
letters = set("abc")
```



세트의 연산

- all(), any(), enumerate(), len(), max(), min(), sorted(), sum() 사용 가능

```
fruits ={"apple","banana","grape"}
size =len(fruits)                                    # size는 3이 된다.
```

```
fruits = { "apple", "banana", "grape" }
if "apple" in fruits:
        print("집합 안에 apple이 있습니다.")
```

```
집합 안에 apple이 있습니다.
```

세트의 연산

```
fruits ={"apple","banana","grape"}
for x in fruits:
        print(x, end=" ")
```

```
grape banana apple
```

```
fruits ={"apple","banana","grape"}
for x in sorted(fruits):
        print(x, end=" ")
```

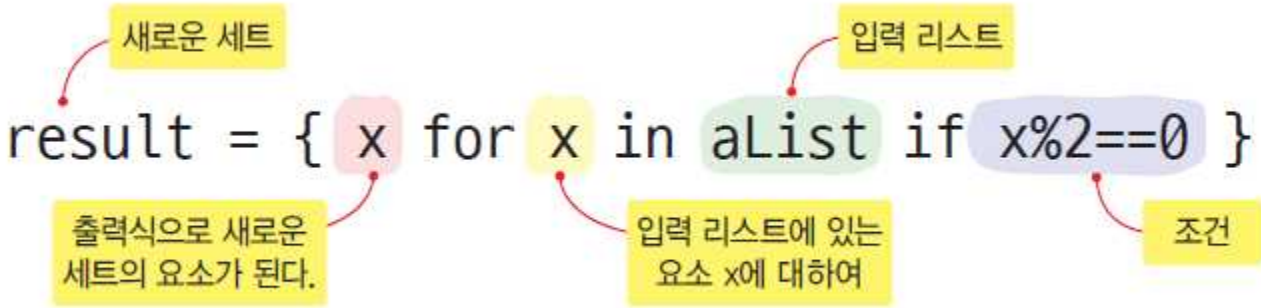
```
apple banana grape
```

세트에 요소 추가하기

```
fruits ={"apple","banana","grape"}
fruits.add("kiwi")



# 요소를 삭제할 때는 remove() 메소드를 사용할 수 있다.
fruits ={"apple","banana","grape", "kiwi"}
fruits.remove("kiwi")



#세트의 모든 요소를 삭제하려면 clear() 메소드를 사용한다.
fruits ={"apple","banana","grape","kiwi"}
fruits.clear()                    # 공백 세트가 된다.
```

삭제하는 요소가 없으면 예외 발생, 예외 발생시키지 않으려면 discard() 사용!

세트 함축 연산




```
aList  =[1,2,3,4,5,1,2 ]
result ={ x for x in aList if x%2==0 }
print(result)
```

```
{2, 4}
```

부분 집합 연산

```
A ={"apple","banana","grape"}
B ={"apple","banana","grape","kiwi"}

if A < B :                          # 또는 A.issubset(B) :
        print("A는 B의 부분 집합입니다.")
```

```
A는 B의 부분 집합입니다.
```

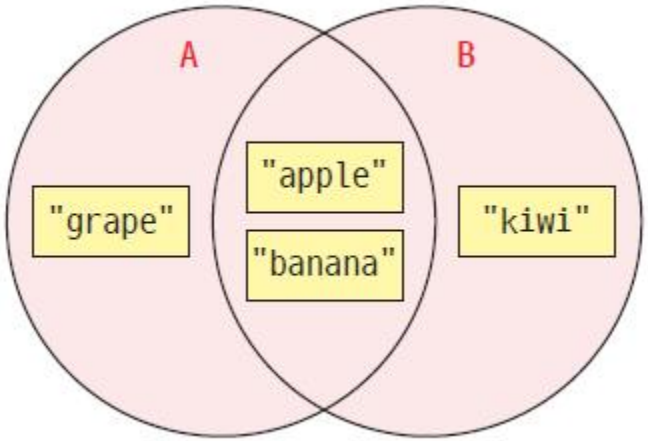
```
A ={"apple","banana","grape"}
B ={"apple","banana","grape","kiwi"}
if A == B :
        print("A와 B는 같습니다.")
else :
        print("A와 B는 같지 않습니다.")
```

```
A와 B는 같지 않습니다.
```

합집합

```
A = {"apple", "banana", "grape"}
B = {"apple", "banana", "kiwi"}

C = A | B                          # 또는 C = A.union(B)
print(C)
```

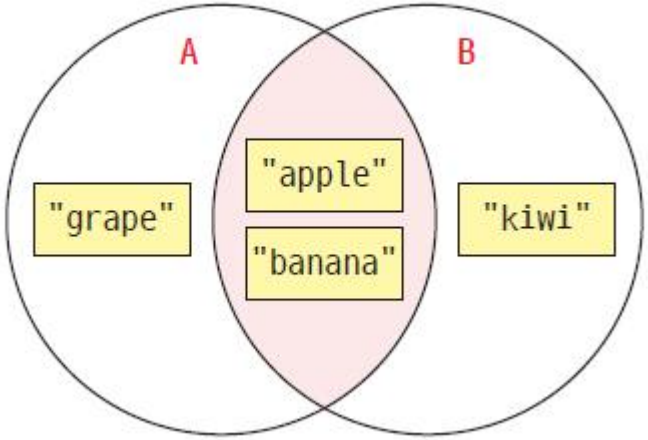
```
{'banana', 'grape', 'apple', 'kiwi'}
```



교집합

```
A = {"apple", "banana", "grape"}
B = {"apple", "banana", "kiwi"}

C = A & B                          # 또는 C = A.intersection(B)
print(C)
```

```
{'banana', 'apple'}
```

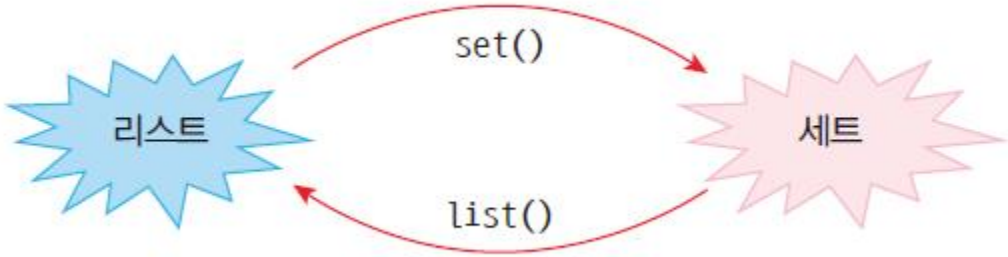


차집합

```
A = {"apple", "banana", "grape"}
B = {"apple", "banana", "kiwi"}

C = A - B                          # 또는 C = A.difference(B)
print(C)
```

```
{'grape'}
```

리스트 <-> 세트




```
>>> list1 =[1,2,3,4,5,1,2,4 ]
>>> len(set(list1))
5
```

서로 다른 정수는 몇 개나 있을까?

```
>>> list1 =[1,2,3,4,5 ]
>>> list2 =[3,4,5,6,7 ]
>>> set(list1)&set(list2)
{3, 4, 5}
```

공통적인 정수는무엇일까?

세트 연산 정리

연산	설명
set()	공백 세트 생성
set(seq)	시퀀스에서 요소를 꺼내서 세트를 만든다.
s1 = { e1, e2, e3, ...}	초기값이 있는 세트는 중괄호로 만든다.
len(s1)	세트에 있는 요소의 수
e in s1	e가 세트 안에 있는지 여부
add(e)	e를 세트에 추가한다.
remove(e) discard(e)	e를 세트에서 삭제한다.
clear()	세트의 모든 요소를 삭제한다.
s1.issubset(s2)	부분 집합인지를 검사한다.
s1 == s2 s1 != s2	동일한 집합인지를 검사한다.
s1.union(s2) s1 | s2	합집합
s1.intersection(s2) s1 & s2	교집합
s1.difference(s2) s1 - s2	차집합

중간점검

1. 리스트와 세트의 차이점은 무엇인가?
2. 세트에 저장된 항목에 접근할 때 인덱스를 사용할 수 있는가?
3. 세트 A와 세트 B의 교집합을 계산하는 수식을 만들어보자.
4. 세트에 항목을 추가하는 함수는?

Lab: 간단한 표절 검사 프로그램

- 사용자로부터 2개의 문자열을 받아서 두 문자열이 얼마나 유사한지를 측정하는 프로그램을 작성해보자.

```
첫 번째 문자열:Alice was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do
두 번째 문자열:Kim was beginning to get very tired of sitting by her sister on the bank, and of having nothing to do

표절률 = 85.71428571428571
```

Solution:

```
s1=input("첫 번째 문자열:")
s2=input("두 번째 문자열:")

s1 = s1.lower()                    # 문자열을 소문자로 만드는 메소드, 9장 참조
s2 = s2.lower()
list1 = s1.split(" ")              # 문자열을 단어로 분리하는 메소드, 9장 참조
list2 = s2.split(" ")

total = len(list1)                 # 단어의 개수
n = len(list(set(list1)&set(list2)))          # 세트 교집합으로 공통 단어 개수 계산

print(f"\n표절률 = {100*n/total}")
```

Lab: 중복되지 않은 단어의 개수 세기

- 중복되지 않은 단어의 개수 세기

입력 텍스트: I have a dream that one day every valley shall be exalted and every hill and mountain shall be made low

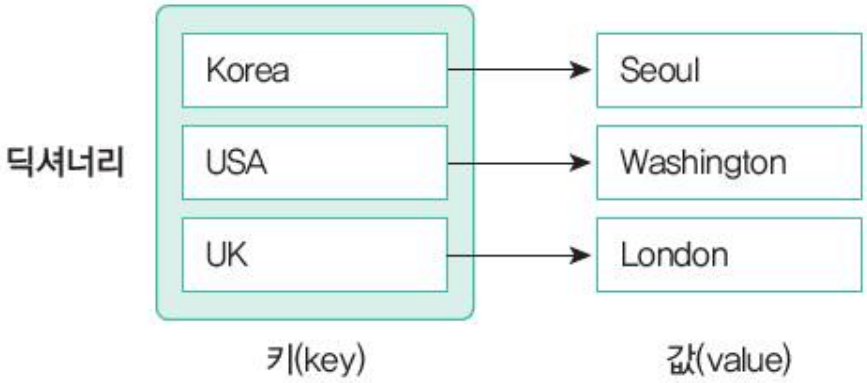
사용된 단어의 개수= 17
{"be", "and", "shall", "low", "have", "made", "one", "exalted", "every", "mountain", "I", "that", "valley", "hill", "day", "a", "dream"}

Solution:

```
txt = input("입력 텍스트: ")
words = txt.split(" ")
unique = set(words)# 집합으로 만들면 자동적으로 중복을 제거한다.

print("사용된 단어의 개수= ", len(unique))
print(unique)
```

딕셔너리

- 딕셔너리(dictionary)도 값을 저장하는 자료구조이다. 하지만 딕셔너리에는 값(value)과 관련된 키(key)도 저장된다.



딕셔너리 생성

Syntax: 딕셔너리

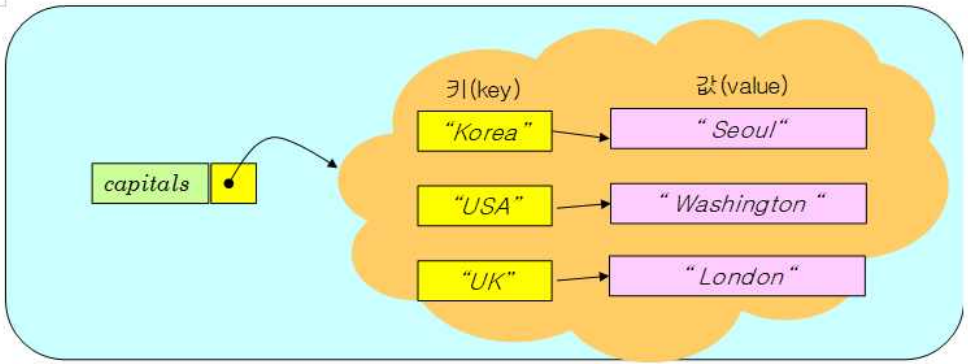
공백 딕셔너리를 생성한다.

Syntax 딕셔너리_이름 = { 키1:값1, 키2:값2, 키3:값3, ... }

```
capitals = { }
capitals = { “Korea”: ”Seoul” , ”USA”: "Washington", "UK": ”London” } # ②
```

값

키



항목 탐색하기

```
>>> capitals ={"Korea":"Seoul","USA":"Washington","UK":"London"}
>>> print( capitals["Korea"])
Seoul

>>> print( capitals[“France”] )
...
KeyError: "France"

>>> print( capitals.get("France", "해당 키가 없습니다." ) )
해당 키가 없습니다.
```

프로그램이 오류로 중단되지 않게
하려면 이렇게 해야 함!

항목 포함 여부

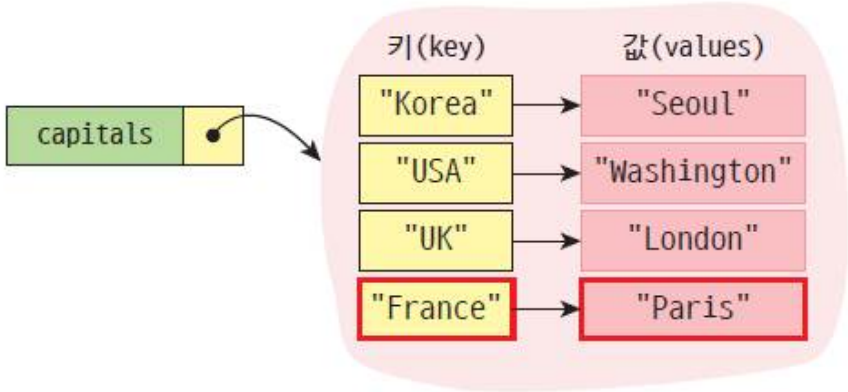
```
if "France" in capitals :
        print( "딕셔너리에 포함됨" )
else:
        print( "딕셔너리에 포함되지 않음" )
```

```
딕셔너리에 포함되지 않음
```

항목 추가하기

```
capitals ={}
capitals["Korea"]="Seoul"
capitals["USA"]="Washington"
capitals["UK"]="London"
capitals["France"]="Paris"
print(capitals)
```

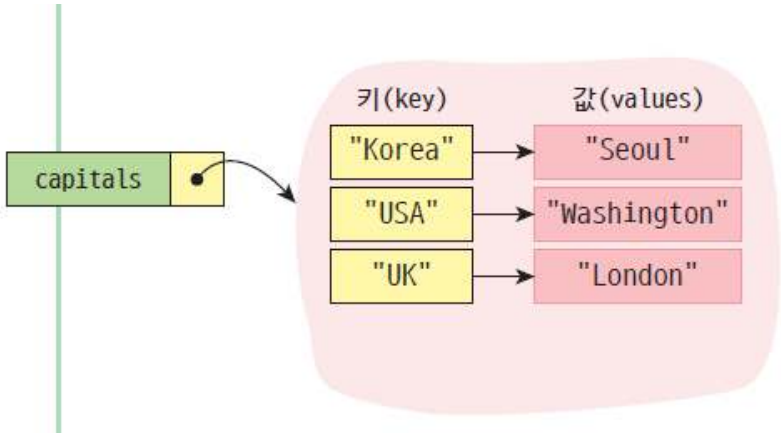
```
{'Korea': 'Seoul', 'USA': 'Washington', 'UK': 'London', 'France': 'Paris'}
```

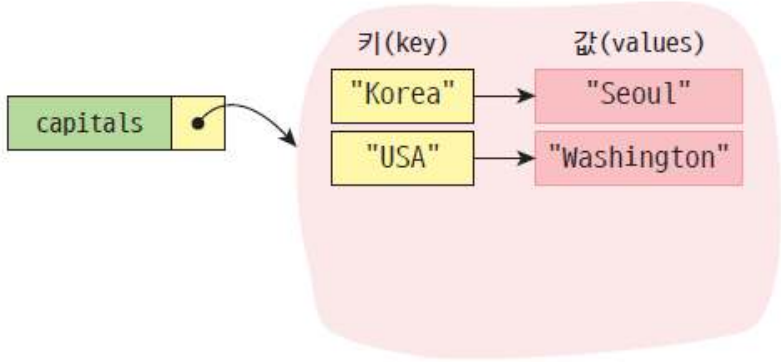


항목 삭제하기

```
city = capitals.pop("UK")
```

만약 주어진 키를 가진 항목이 없으면
KeyError 예외가 발생한다






```
if "UK" in capitals :
        capitals.pop("UK")
```

항목 방문하기

```
capitals ={"Korea":"Seoul","USA":"Washington","UK":"London"}
for key in capitals :
        print( key,":", capitals[key])
```

```
Korea : Seoul
USA : Washington
UK : London
```

```
capitals ={"Korea":"Seoul","USA":"Washington","UK":"London"}
for key, value in capitals.items():
                print( key,":", value )
```

```
Korea : Seoul
USA : Washington
UK : London
```

기타 연산

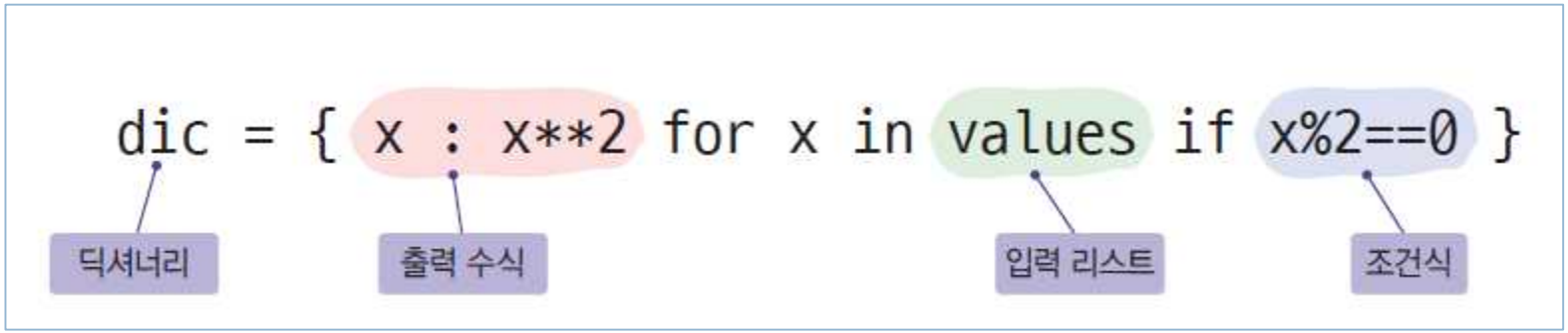
```
capitals ={"Korea":"Seoul","USA":"Washington","UK":"London"}
print( capitals.keys())
print( capitals.values())
```

```
dict_keys(['Korea', 'USA', 'UK'])
dict_values(['Seoul', 'Washington', 'London'])
```

```
for key in sorted( capitals.keys()):
        print(key, end=" ")
```

```
Korea UK USA
```

딕셔너리 함축




```
values =[1,2,3,4,5,6]

dic ={ x : x**2 for x in values if x%2==0 }
print(dic)
```

```
{2: 4, 4: 16, 6: 36}
```

딕셔너리 메소드

연산	설명
d = dict()	공백 딕셔너리를 생성한다.
$d=\{k_1:v_1,k_2:v_2, \ldots,k_n:v_n\}$	초기값으로 딕셔너리를 생성한다.
len(d)	딕셔너리에 저장된 항목의 개수를 반환한다.
k in d	k가 딕셔너리 d 안에 있는지 여부를 반환한다.
k not in d	k가 딕셔너리 d 안에 없으면 True를 반환한다.
d[key] = value	딕셔너리에 키와 값을 저장한다.
v = d[key]	딕셔너리에서 key에 해당되는 값을 반환한다.
d.get(key, default)	주어진 키를 가지고 값을 찾는다. 만약 없으면 default 값이 반환된다.
d.pop(key)	항목을 삭제한다.
d.values()	딕셔너리 안의 모든 값의 시퀀스를 반환한다.
d.keys()	딕셔너리 안의 모든 키의 시퀀스를 반환한다.
d.items()	딕셔너리 안의 모든 (키, 값)을 반환한다.

중간점검

1. 공백 딕셔너리를 생성하는 명령문을 만들어보자.
2. 딕셔너리에 존재하는 모든 키를 방문하는 코드를 작성해보자.
3. 딕셔너리 d에 (k, v)를 저장하는 명령문을 만들어보자.

Lab: 영한 사전

```
단어를 입력하시오: one
하나
```

```
단어를 입력하시오: two
둘
```

Solution:

```
english_dict ={}                                  # 공백 딕셔너리를 생성한다.

english_dict["one"]="하나"                        # 딕셔너리에 단어와 의미를 추가한다.
english_dict["two"]="둘'"
english_dict["three"]="셋"

word =input("단어를 입력하시오: ");
print (english_dict[word])
```

Lab: 학생 성적 처리

```
score_dic =   {
    "Kim":[99,83,95],
    "Lee":[68,45,78],
    "Choi":[25,56,69]
}
```

```
Kim 의 평균성적= 92.33333333333333
Lee 의 평균성적= 63.666666666666664
Choi 의 평균성적= 50.0
```

Solution:

```
score_dic =   {
    "Kim":[99,83,95],
    "Lee":[68,45,78],
    "Choi":[25,56,69]
}

for name, scores in  score_dic.items():
     print(name,"의 평균성적=",sum(scores)/len(scores))
```

Mini Project: 주소록 작성

1. 연락처 추가
2. 연락처 삭제
3. 연락처 검색
4. 연락처 출력
5. 종료

메뉴 항목을 선택하시오: 1
이름: KIM
전화번호: 123-4567

1. 연락처 추가
2. 연락처 삭제
3. 연락처 검색
4. 연락처 출력
5. 종료

메뉴 항목을 선택하시오: 4
KIM 의 번화번호: 123-4567
...

이번 장에서 배운 것

- 튜플은 변경 불가능한 항목들을 모아둔 곳이다.
- 튜플의 항목은 어떤 것이든 가능하다.
- ()을 이용하여 공백 튜플을 만들 수 있다.

- 딕셔너리는 키와 값으로 이루어진다.
- 딕셔너리에서 [] 연산자를 사용하여 키와 관련된 값을 액세스할 수 있다.
- 딕셔너리에서 [] 연산자를 사용하여 새 항목을 추가하거나 수정할 수 있다.
- 딕셔너리에서 pop 메소드를 사용하여 항목을 제거한다.

- 세트는 고유한 값들을 저장한다.
- 세트는 set() 함수를 사용하여 생성할 수 있다.
- 세트의 add() 메소드를 사용하여 새 요소를 추가할 수 있다.
- 세트의 remove() 메소드를 사용하여 세트에서 요소를 제거한다.

Q & A