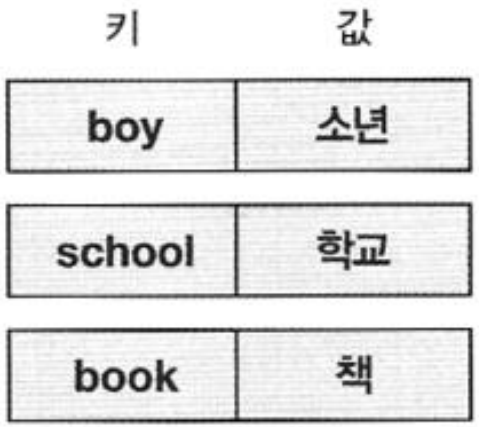
사전과 집합

사전

■ 키와 값의 쌍

```
{
   키1: 값1,
   키2: 값2,

   …
}
```




```
dic = {
    'boy': '소년',
    'school': '학교',
    'book':'책'
}
print(dic)
```

```
{'boy': '소년', 'school': '학교', 'book': '책'}
```

사전

❖ 키와 값의 쌍

```
dic = { 'boy': '소년', 'school': '학교', 'book':'책'}

print(dic['boy'])
print(dic['book'])
print(dic['girl'])
```

```
소년
책Traceback (most recent call last):
  File, line 5, in <module>
    print(dic['girl'])
KeyError: 'girl'
```

사전

❖ 키와 값의 쌍

```
dic = { 'boy': '소년', 'school': '학교', 'book':'책'}

print(dic.get('boy'))
print(dic.get('girl'))
print(dic.get('girl', '사전에 없는 단어입니다.'))
```

```
소년
None
사전에 없는 단어입니다.
```

```
dic = { 'boy': '소년', 'school': '학교', 'book':'책'}

if 'student' in dic:
    print('사전에 있는 단어입니다.')
else:
    print('이 단어는 사전에 없습니다.')
```

```
이 단어는 사전에 없습니다.
```

사전

❖ 사전 관리

- 사전[키]
  - 키의 값을 리턴, 키가 존재하지 않는 경우 예외 발생
- 사전.get(키 [, 기본값])
  - 키의 값을 리턴, 키가 존재하지 않는 경우, None 리턴, 키가 없을 때 리턴할 값 지정 가능
- .keys()
  - 키 목록 리턴
- .values()
  - 값 목록 리턴
- .items()
  - (키,값) 튜플 목록 리턴

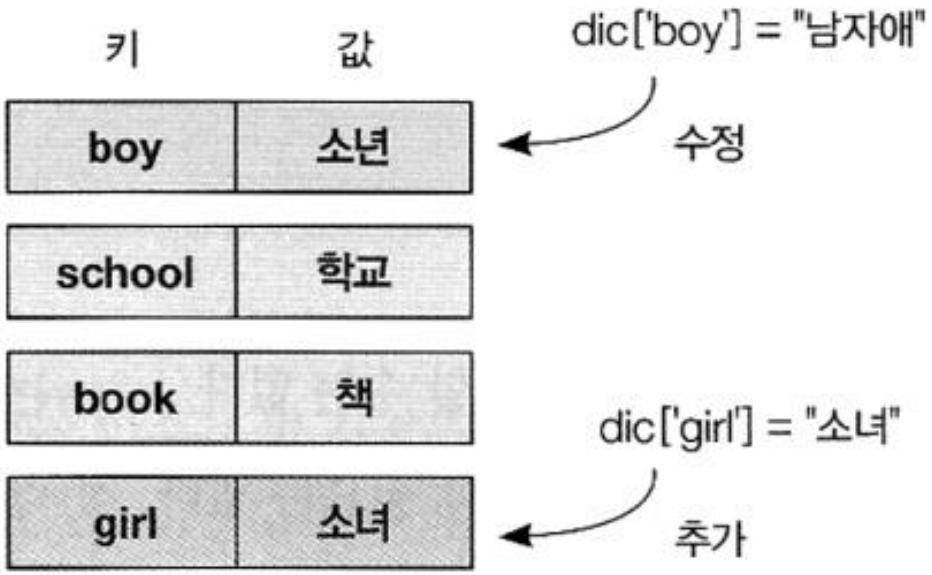
사전

❖ 사전 관리

```
dic = { 'boy': '소년', 'school': '학교', 'book':'책'}

dic['boy'] = '남자아이'    # 기존값 수정
dic['girl'] = '소녀'       # 새로운 엔트리 추가
del dic['book']            # 기존 엔트리 삭제
print(dic)
```

```
{'boy': '남자아이', 'school': '학교', 'girl': '소녀'}
```



사전

❖ 사전 관리

```
dic = { 'boy': '소년', 'school': '학교', 'book':'책'}

print(dic.keys())
print(dic.values())
print(dic.items())
```

```
dict_keys(['boy', 'school', 'book'])
dict_values(['소년', '학교', '책'])
dict_items([('boy', '소년'), ('school', '학교'), ('book', '책')])
```

```
dic = { 'boy': '소년', 'school': '학교', 'book':'책'}

keylist = dic.keys()
for key in keylist:
    print(key, dic[key])
```

```
boy 소년
school 학교
book 책
```

사전

❖ 사전 관리

```
li = list(dic.keys())
print(li)

li = list(dic)
print(li)
```

```
['boy', 'school', 'book']
['boy', 'school', 'book']
```

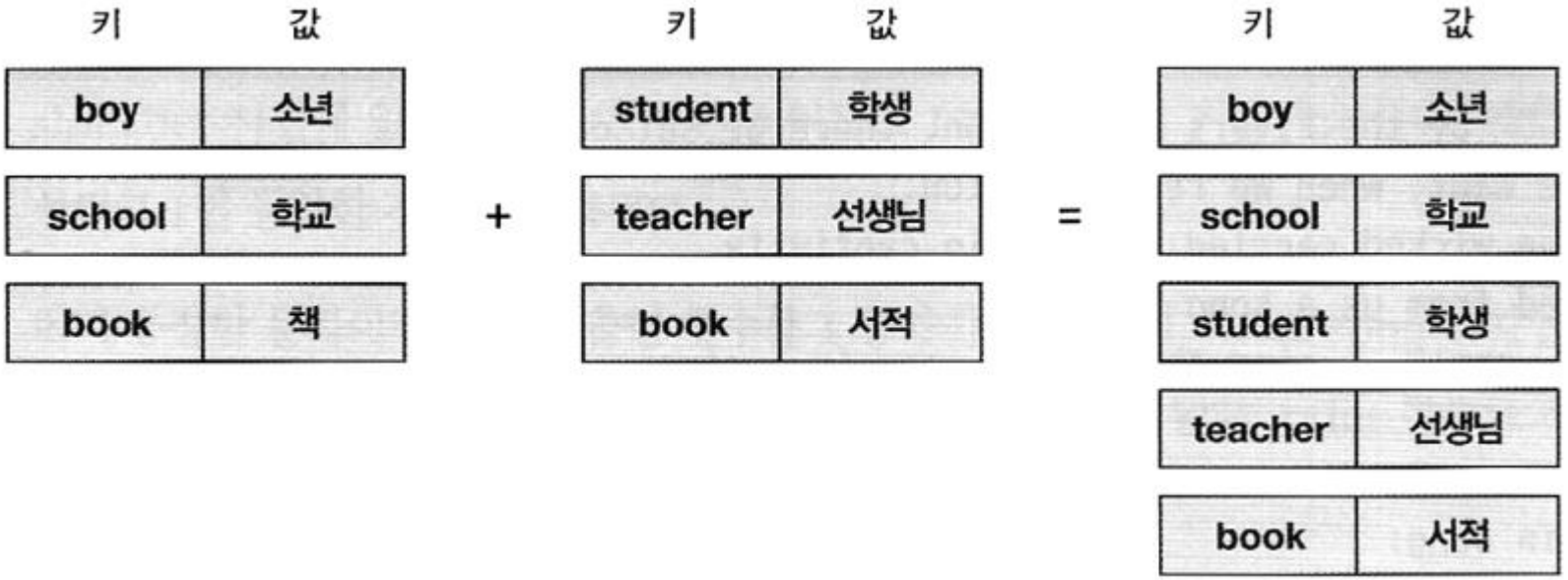
사전

❖ 사전 관리

```
dic = { 'boy': '소년', 'school': '학교', 'book':'책'}
dic2 = { 'student': '학생', 'teacher': '선생님', 'book':'서적'}

dic.update(dic2)
print(dic)
```

```
{'boy': '소년', 'school': '학교', 'book': '서적', 'student': '학생', 'teacher':
'선생님'}
```



사전

❖ 사전 관리

```
li = [
    ['boy', '소년'],
    ['school', '학교'],
    ['teacher', '선생님']
]

dic = dict(li)
print(dic)
```

```
{'boy': '소년', 'school': '학교', 'teacher': '선생님'}
```

사전

❖ 사전 활용

```
song = """by the rivers of babylon, there we sat down
yeah we wept, when we remember zion.
when the wicked carried us away in captivity
required from us a song
now how shall we sing the lord's song in a strange land"""

alphabet = dict()
for c in song:
    if c.isalpha() == False:
        continue

    c = c.lower()
    if c not in alphabet:
        alphabet[c] = 1
    else:
        alphabet[c] += 1

print(alphabet)
```

```
{'b': 4, 'y': 5, 't': 9, 'h': 9, 'e': 22, 'r': 12, 'i': 10, 'v': 2, 's': 10,
'o': 10, 'f': 2, 'a': 12, 'l': 5, 'n': 13, 'w': 12, 'd': 6, 'p': 2, 'm': 3,
'z': 1, 'c': 3, 'k': 1, 'u': 3, 'q': 1, 'g': 4}
```

사전

❖ 사전 활용

```
key = list(alphabet.keys())
key.sort()
for c in key:
    num = alphabet[c]
    print(c, '=>', num)
```

```
a => 12
b => 4
c => 3
d => 6
:
w => 12
y => 5
z => 1
```

사전

❖ 사전 활용

```
for code in range(ord('a'), ord('z')+1):
    c = chr(code)
    num = alphabet.get(c, 0)
    print(c, '=>', num)
```

```
a => 12
b => 4
c => 3
d => 6
:
w => 12
x => 0
y => 5
z => 1
```

집합

❖ 집합 정의

- { 값1, 값2, ... }
- 값의 중복을 허용하지 않음

- set(다른 시퀀스)
  - 집합 변환 함수
- .add(값)
  - 집합에 값 추가, 이미 값이 있으면 추가하지 않음
- .remove(값)
  - 집합에서 값을 제거, 값이 없는 경우 예외 발생

```
asia = {'korea', 'china', 'japan', 'korea'}
print(asia)
```

```
{'china', 'japan', 'korea'}
```

집합

❖ 집합 정의

```
print(set('aaabbbccc'))
print(set([12, 34, 56, 78]))
print(set(('홍길동', '고길동','둘리')))
print(set({'boy': '소년', 'school': '학교', 'book': '책'}))  # 사전의 키 목록을 집합으로 변환
print(set())
```

```
{'a', 'c', 'b'}
{56, 34, 12, 78}
{'홍길동', '둘리', '고길동'}
{'book', 'boy', 'school'}
set()
```

```
asia = {'korea', 'china', 'japan'}
asia.add('vietnam')
asia.add('korea')
asia.remove('japan')
print(asia)
```

```
{'vietnam', 'china', 'korea'}
```

집합

❖ 집합 연산

연산	기호	메서드	설명
합집합	|	union	두 집합의 모든 원소
교집합	&	intersection	두 집합 모두에 있는 원소
차집합	-	difference	왼쪽 집합의 원소 중 오른쪽 집합의 원소를 뺀 것
배타적 차집합	^	symmetric_difference	한쪽 집합에만 있는 원소의 합

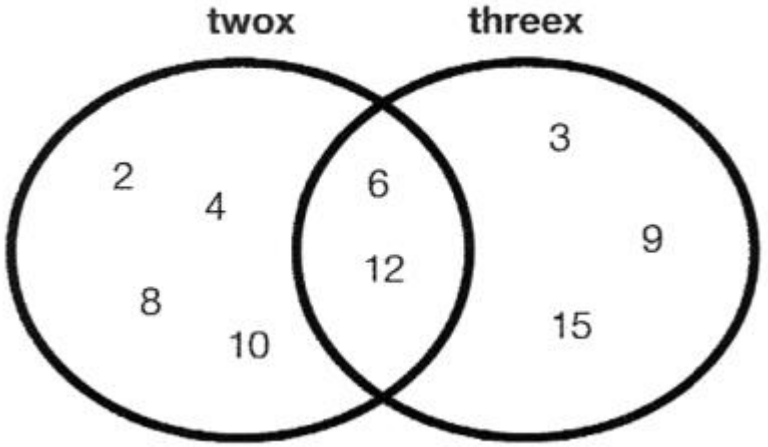
집합

❖ 집합 연산

```
twox = { 2, 4, 6, 8, 10, 12 }
threex = { 3, 6, 9, 12, 15 }

print("교집합", twox & threex)
print("합집합", twox | threex)
print("차집합", twox - threex)
print("차집합", threex - twox)
print("배타적 차집합", twox ^ threex)
```

```
교집합 {12, 6}
합집합 {2, 3, 4, 6, 8, 9, 10, 12, 15}
차집합 {8, 2, 10, 4}
차집합 {9, 3, 15}
배타적 차집합 {2, 3, 4, 8, 9, 10, 15}
```



집합

❖ 집합 연산

연산	기호	메서드	설명
부분집합	<=	issubset	왼쪽이 오른쪽의 부분집합인지 조사한다.
진성 부분집합	<		부분집합이면서 여분의 원소가 더 있음
포함집합	>=	issuperset	왼쪽이 오른쪽 집합을 포함하는지 조사한다.
진성 포함집합	>		포함집합이면서 여분의 원소가 더 있음

집합

❖ 집합 연산

```
mammal = { '코끼리', '고릴라', '사자', '고래', '사람', '원숭이', '개'}
primate = {'사람', '원숭이', '고릴라'}

if '사자' in mammal:
    print('사자는 포유류입니다.')
else:
    print('사자는 포유류가 아닙니다')

print(primate <= mammal)
print(primate < mammal)
print(primate <= primate)
print(primate < primate)
```

```
사자는 포유류입니다.
True
True
True
False
```