함수 복습

```
def subtract(a, b):
    x = a - b
    return x

y = subtract(b = 3, a = 7)
```

함수의 정의

함수의 호출

```
def avg(*args):
    sum = 0
    for n in args:
        sum += n
    return sum / len(args)

print( '평균: ', avg(1,3,5,7) )
```

args는 튜플

함수 복습

```
def printDic(**args):
    for s, t in args.items( ):
        print( s, ':', t )

printDic(a=20, b=30, c=50)
```

args는 딕셔너리

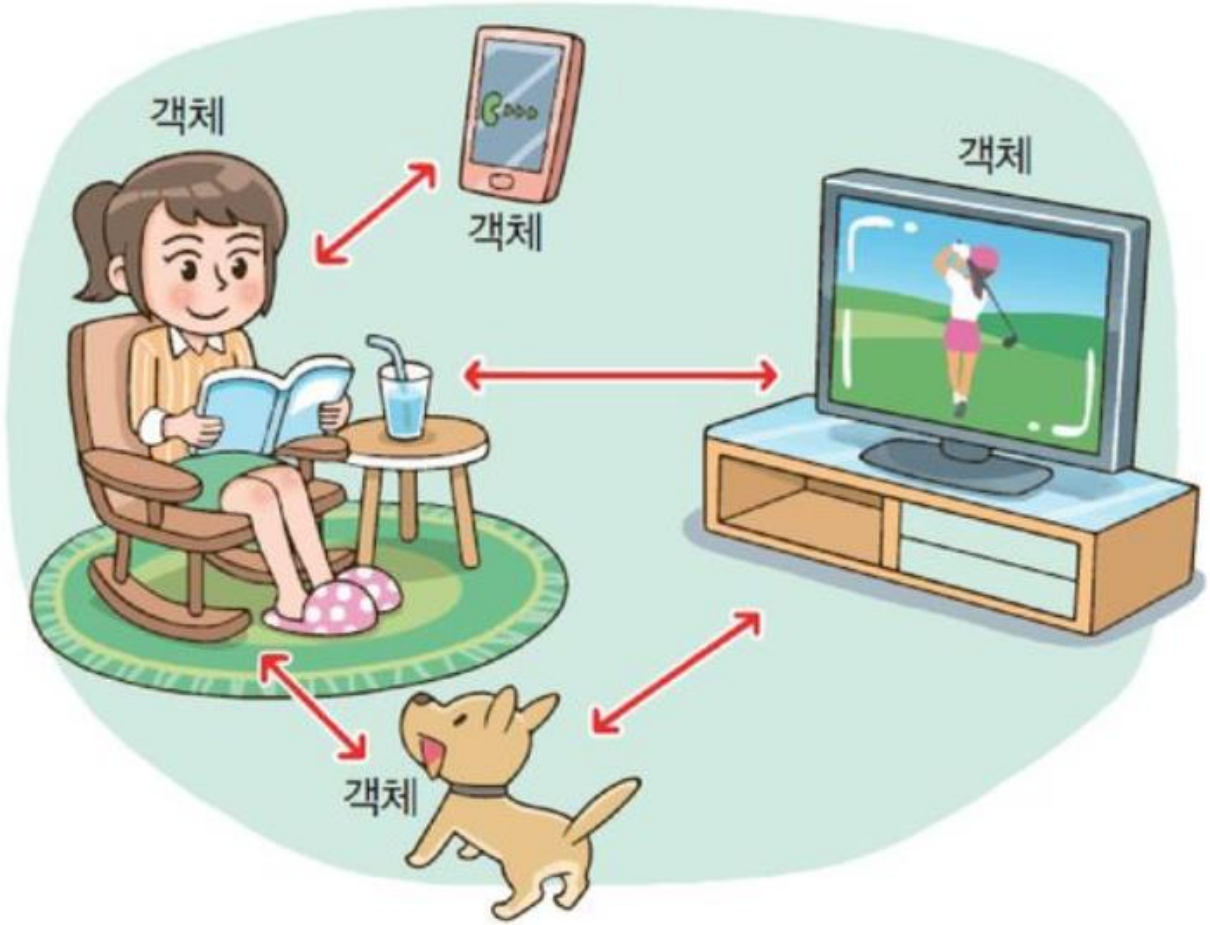
```
def printDic(a, b, c):
    print(a, b, c)

d = {'a':20, 'b':30, 'c':50}
printDic(**d)
```

딕셔너리가 a, b, c로 전개

딕셔너리

객체(object)

❖ 파이썬은 객체지향 프로그래밍(OOP) 언어

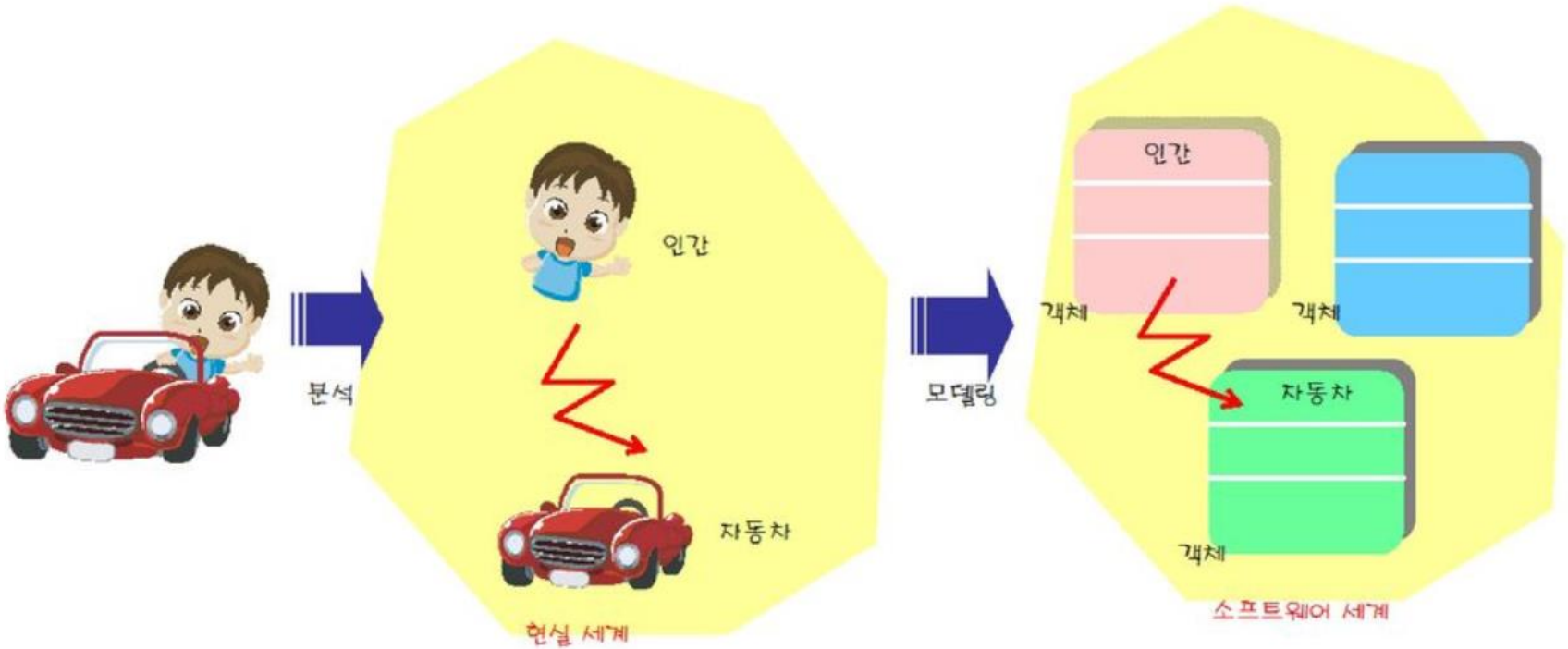


객체(object)

❖ 파이썬은 객체지향 프로그래밍(OOP) 언어

객체(object)

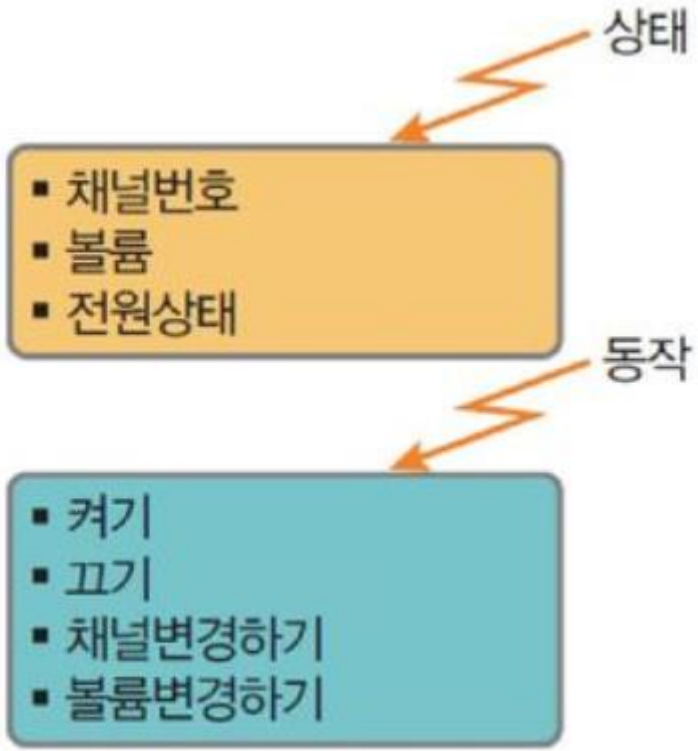
❖ 파이썬은 객체지향 프로그래밍(OOP) 언어



객체(object)

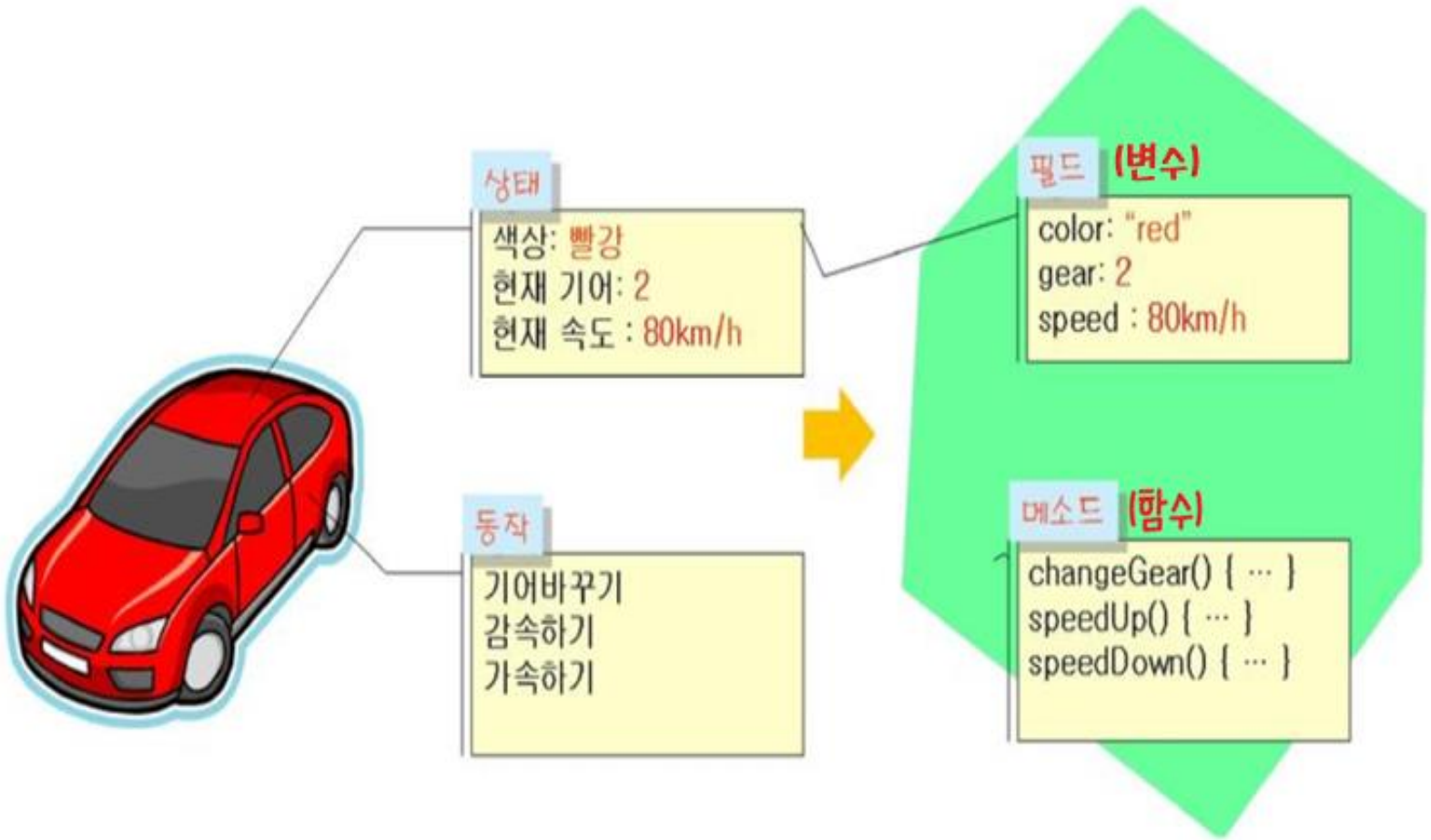
❖ 파이썬은 객체지향 프로그래밍(OOP) 언어

TV 객체



객체(object)

❖ 파이썬은 객체지향 프로그래밍(OOP) 언어



객체(object)

❖ 파이썬은 객체지향 프로그래밍(OOP) 언어

TV 객체

필드

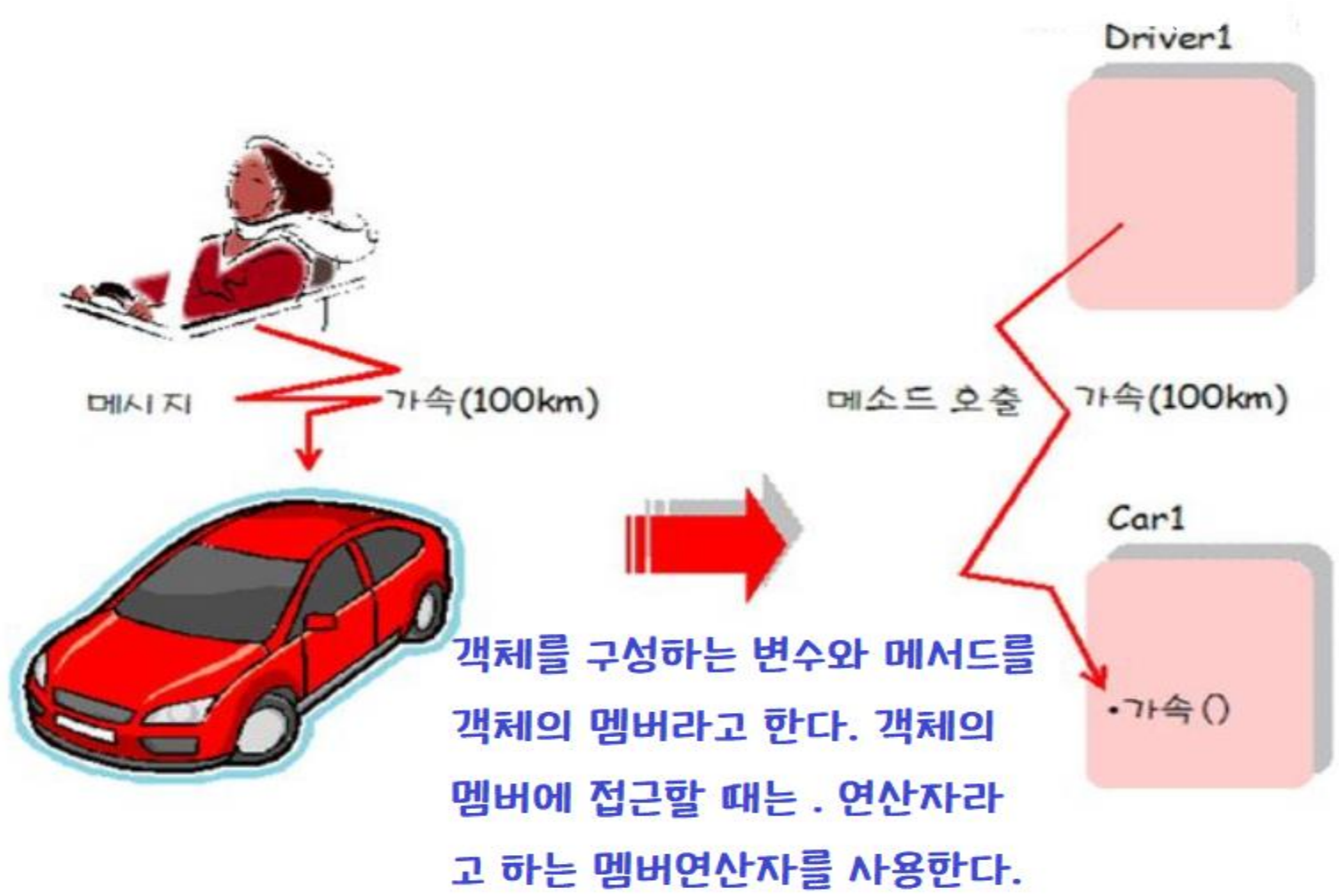
- int channelNO;
- int volume;
- bool on Off

메소드

- turnOn()
- turnOff()
- changeChannel()
- change Volume()

객체(object)

❖ 파이썬은 객체지향 프로그래밍(OOP) 언어



객체(object)

❖ 파이썬은 객체지향 프로그래밍(OOP) 언어

객체(Object) = 속성(Attribute) + 기능(Method)

= 변수(Field) + 함수(Function)

지금까지 학습한 문자열, 리스트, 튜플, 딕셔너리, 집합은 모두 콜렉션 객체들이다.

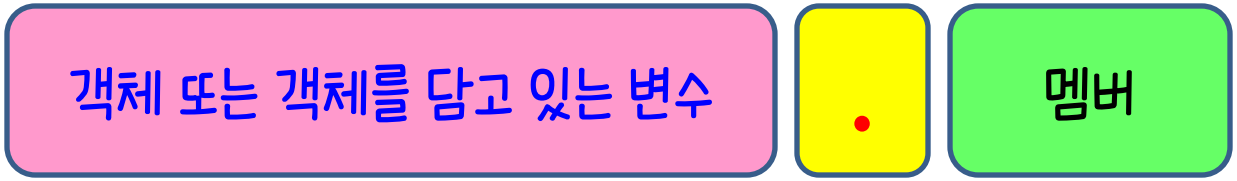
. 연산자를 사용하여 각 객체가 가지고 있는 함수(메서드) 호출이 가능하다.

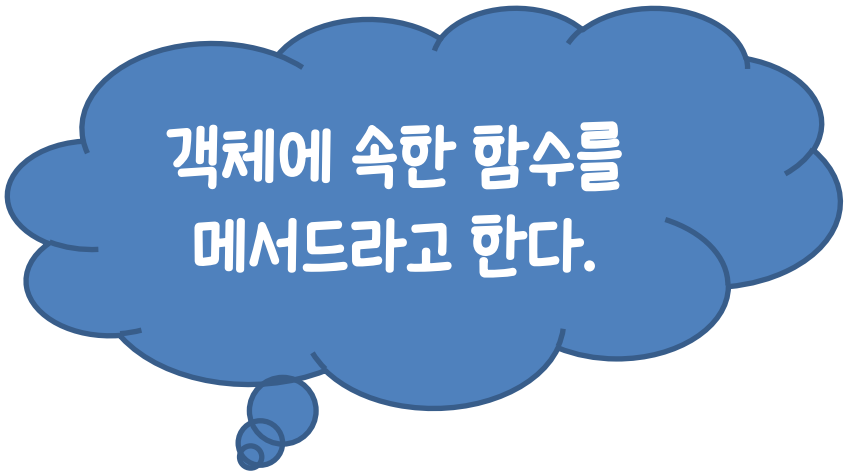
"abc".upper(), [4,15,2,30,4].count(4), {1:100, 2:88, 3:90}.get(3)

객체(object)

❖ 파이썬은 객체지향 프로그래밍(OOP) 언어

다른 객체의 역할이 필요할 때 다음 형식으로 다른 객체의 메서드 또는 변수를 사용



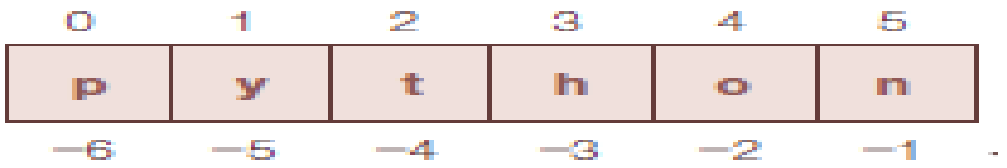


Contents

❖ 목차

1. 문자열 분리

❖ 첨자



- 문자의 위치
- 대괄호와 첨자 적어 문자열 구성하는 개별 문자 읽음
- 앞뒤 양쪽에서 읽을 수 있음

index

```
s = "python"
print(s[2])
print(s[-2])
```

실행결과

```
t
o
```

- for문으로 순회하여 개별 문자 순서대로 꺼냄

stringindex

```
s = "python"
for c in s:
    print(c, end = ',')
```

실행결과

```
p,y,t,h,o,n,
```

1. 문자열 분리

❖ 슬라이스

- 범위 지정하여 부분 문자열을 추출
- [begin:end:step]
  - 시작, 끝, 중간값을 지정

slice

```
s = "python"
print(s[2:5])
print(s[3:])
print(s[:4])
print(s[2:-2])
```

실행결과

```
tho
hon
pyth
th
```

1. 문자열 분리

- 일정 형식 가진 문자열에서 원하는 정보만 추출할 수 있음

slice2

```
file = "20171224-104830.jpg"
print("촬영 날짜 : " + file[4:6] + "월 " + file[6:8] + "일")
print("촬영 시간 : " + file[9:11] + "시 " + file[11:13] + "분")
print("확장자 : " + file[-3:])
```

실행결과

```
촬영 날짜 : 12월 24일
촬영 시간 : 10시 48분
확장자 : jpg
```

2. 문자열 메서드

❖ 메서드

- 클래스에 소속된 함수
- 객체에 대해 특화된 작업 수행

❖ 검색

- len 함수
  - 문자열의 길이를 조사

find

```
s = "python programming"
print(len(s))
print(s.find('o'))
print(s.rfind('o'))
print(s.index('r'))
print(s.count('n'))
```

실행결과

```
18
4
9
8
2
```

2. 문자열 메서드

- find 메서드
  - 인수로 지정한 문자 또는 부분 문자열의 위치 조사
- rfind 메서드
  - 뒤에서 검색 시작
- count 메서드
  - 특정 문자 개수

2. 문자열 메서드

❖ 조사

■ in 구문

- 특정 문자 유무 여부 조사

in

```
s = "python programming"
print('a' in s)
print('z' in s)
print('pro' in s)
print('x' not in s)
```

실행결과

```
True
False
True
True
```

2. 문자열 메서드

함수	설명
isalpha	모든 문자가 알파벳인지 조사한다.
islower	모든 문자가 소문자인지 조사한다.
isupper	모든 문자가 대문자인지 조사한다.
isspace	모든 문자가 공백인지 조사한다.
isalnum	모든 문자가 알파벳 또는 숫자인지 조사한다.
isdecimal	모든 문자가 숫자인지 조사한다.
isdigit	모든 문자가 숫자인지 조사한다.
isnumeric	모든 문자가 숫자인지 조사한다.
isidentifier	명칭으로 쓸 수 있는 문자로만 구성되어 있는지 조사한다.
isprintable	인쇄 가능한 문자로만 구성되어 있는지 조사한다.

2. 문자열 메서드

❖ 변경

■ lower 메서드 / upper 메서드

- 각기 영문자를 전부 소문자 / 대문자로 바꿈

lower

```
s = "Good morning. my love KIM."
print(s.lower())
print(s.upper())
print(s)

print(s.swapcase())
print(s.capitalize())
print(s.title())
```

실행결과

```
good morning. my love kim.
GOOD MORNING. MY LOVE KIM.
Good morning. my love KIM.
gOOD MORNING. MY LOVE kim.
Good morning. my love kim.
Good Morning. My Love Kim.
```

2. 문자열 메서드

- 문자열 자체를 변경하는 것은 아님

- lstrip / rstrip / strip 메서드
  - 왼쪽 / 오른쪽 / 양측 공백을 제거

strip

```
s = "  angel   "
print(s + "님")
print(s.lstrip() + "님")
print(s.rstrip() + "님")
print(s.strip() + "님")
```

실행결과

```
  angel   님
angel   님
  angel님
angel님
```

2. 문자열 메서드

❖ 분할

■ split 메서드

- 구분자를 기준으로 문자열을 분할

split

```
s = "짜장 짬뽕 탕슉"
print(s.split())

s2 = "서울->대전->대구->부산"
city = s2.split("->")
print(city)
for c in city:
    print(c, "찍고", end = ' ')
```

실행결과

```
['짜장', '짬뽕', '탕슉']
['서울', '대전', '대구', '부산']
서울 찍고 대전 찍고 대구 찍고 부산 찍고
```

■ splitlines 메서드

- 개행 문자나 파일 구분자 등 기준으로 문자열 잘라 리스트로 만듦

2. 문자열 메서드

join 메서드

- 문자열의 각 문자 사이에 다른 문자열 끼워넣음

join

```
s = "._."
print(s.join("대한민국"))
```

실행결과

```
대._.한._.민._.국
```

2. 문자열 메서드

❖ 대체

■ replace 메서드

- 특정 문자열을 찾아 다른 문자열로 대체
- 첫 번째 인수 : 검색할 문자열 지정
- 두 번째 인수 : 바꿀 문자열 지정

replace

```
s = "독도는 일본땅이다. 대마도도 일본땅이다."
print(s)
print(s.replace("일본", "한국"))
```

실행결과

```
독도는 일본땅이다. 대마도도 일본땅이다.
독도는 한국땅이다. 대마도도 한국땅이다.
```

■ just 메서드

- 문자열을 특정 폭에 맞추어 정렬

2. 문자열 메서드

- center 메서드
  - 중앙 정렬

center

```
traveler = """강나루 건너서\n밀밭 길을\n\n구름에 달 가듯이\n가는 나그네\n
길은 외줄기\n남도 삼백리\n\n술 익는 마을마다\n타는 저녁놀\n
구름에 달 가듯이\n가는 나그네"""
poet = traveler.splitlines()
for line in poet:
    print(line.center(30))
```

실행결과

```
            강나루 건너서
             밀밭 길을

          구름에 달 가듯이
             가는 나그네

             길은 외줄기
             남도 삼백리

          술 익는 마을마다
             타는 저녁놀

          구름에 달 가듯이
             가는 나그네
```

3. 포맷팅

❖ 포맷팅

- 문자열 사이사이에 다른 정보 삽입하여 조립하는 기법

stringcat

```
price = 500
print("궁금하면 " + str(price) + "원!")
```

실행결과 궁금하면 500원!

format % values

format

```
price = 500
print("궁금하면 %d원!" % price)
```

3. 포맷팅

표식	설명
%d	정수
%f	실수
%s	문자열
%c	문자 하나
%h	16진수
%o	8진수
%%	% 문자

format2

```
month = 8
day = 15
anni = "광복절"
print("%d월 %d일은 %s이다." % (month, day, anni))
```

실행결과

```
8월 15일은 광복절이다.
```

3. 포맷팅

- 서식
  - %[-]폭[.유효자리수]서식

align

```
price = [30, 13500, 2000]
for p in price:
    print("가격:%d원" % p)
for p in price:
    print("가격:%7d원" % p)
```

실행결과

```
가격:30원
가격:13500원
가격:2000원
가격:     30원
가격:  13500원
가격:   2000원
```

3. 포맷팅

- 폭에 – 지정하여 왼쪽 정렬

numalign

```
price = [30, 13500, 2000]
for p in price:
    print("가격:%-7d원" % p)
```

실행결과

```
가격:30     원
가격:13500  원
가격:2000   원
```

- . 기호로 소수점 이하 표시 자리수 설정

precision

```
pie = 3.14159265
print("%10f" % pie)
print("%10.8f" % pie)
print("%10.5f" % pie)
print("%10.2f" % pie)
```

실행결과

```
3.141593
3.14159265
   3.14159
      3.14
```

3. 포맷팅

stname = "유니코"; stage=20; stavg=95.869

```
print(stname+"학생은 나이가 "+str(stage)+"이고 평균은 "+str(stavg)+"입니다")

print(stname, "학생은 나이가 ", stage, "이고 평균은 ", stavg, "입니다", sep="")

print("%s학생은 나이가 %d이고 평균은 %.2f입니다" % (stname, stage, stavg))

print("{}학생은 나이가 {}이고 평균은 {:.2f}입니다".format(stname, stage, stavg))

print(f"{stname}학생은 나이가 {stage}이고 평균은 {stavg:.2f}입니다")
```

1. string cat
2. format % values
3. '{ }'.format() → 2.7, 3.0
4. f-Strings → 3.6

3. 포맷팅

stname = "유니코"; stage=20; stavg=95.869

```
v1 = stname+"학생은 나이가 "+str(stage)+"이고 평균은 "+str(stavg)+"입니다"

v2 = "%s학생은 나이가 %d이고 평균은 %.2f입니다" % (stname, stage, stavg)

v3 = "{}학생은 나이가 {}이고 평균은 {:.2f}입니다".format(stname, stage, stavg)

v4 = f"{stname}학생은 나이가 {stage}이고 평균은 {stavg:.2f}입니다"
```

1. string cat
2. format % values
3. '{ }'.format() → 2.7, 3.0
4. f-Strings → 3.6

Contents

❖ 목차

1. 리스트

❖ 자료의 집합

- 리스트는 여러 개 값을 집합적으로 저장
- 요소 (Element)
  - 리스트에 소속되는 각각의 값
  - 리스트에는 주로 같은 타입 요소를 모음

```
score = [ 88, 95, 70, 100, 99 ]
name = [ "최상미", "이한승", "김기남" ]
```

listscore

```
score = [ 88, 95, 70, 100, 99 ]
sum = 0
for s in score:
    sum += s
print("총점 : ", sum)
print("평균 : ", sum / len(score))
```

실행결과

```
총점 :  452
평균 :  90.4
```

1. 리스트

❖ 리스트의 요소

- 개별요소 읽기 : 대괄호 안에 읽고자 하는 요소의 순서값 적음

```
score = [ 88, 95, 70, 100, 99 ]
print(score[0])     # 88
print(score[2])     # 70
print(score[-1])    # 99
```

- 요소 분리 : 범위 지정
  - [begin:end:step]

listslice

```
nums = [0,1,2,3,4,5,6,7,8,9]
print(nums[2:5])        # 2~5까지
print(nums[:4])         # 처음부터 4까지
print(nums[6:])         # 6에서 끝까지
print(nums[1:7:2])      # 1~7까지 하나씩 건너뛰며
```

실행결과

```
[2, 3, 4]
[0, 1, 2, 3]
[6, 7, 8, 9]
[1, 3, 5]
```

1. 리스트

- 대괄호에 첨자 지정할 수 있음

listassign

```
score = [ 88, 95, 70, 100, 99 ]
print(score[2])     # 70
score[2] = 55       # 값 변경
print(score[2])     # 55
```

실행결과

```
70
55
```

```
for i in range(len(score)) :
    print(score[i])

for d in score :
    print(d)
```

listreplace

```
nums = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
nums[2:5] = [20, 30, 40]
print(nums)
nums[6:8] = [90, 91, 92, 93, 94]
print(nums)
```

실행결과

```
[0, 1, 20, 30, 40, 5, 6, 7, 8, 9]
[0, 1, 20, 30, 40, 5, 90, 91, 92, 93, 94, 8, 9]
```

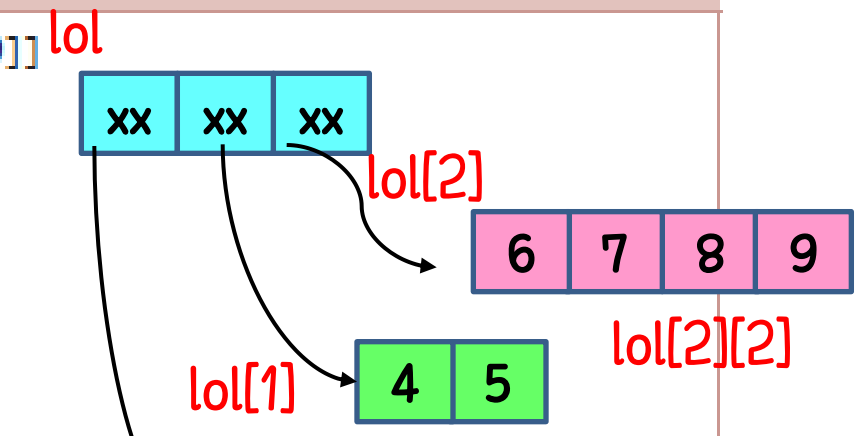
1. 리스트

❖ 이중 리스트

- 리스트의 요소로 리스트 넣어 중첩할 수 있음

nestlist

```
lol = [ [1, 2, 3], [4, 5], [6, 7, 8, 9]]
print(lol[0])
print(lol[2][1])

for sub in lol:
    for item in sub:
        print(item, end=' ')
    print()
```



실행결과

```
[1, 2, 3]
7
1 2 3
4 5
6 7 8 9
```

- 이중 리스트 순회하여 최종 값 읽으려면 루프도 이중으로 하여야 함

❖ 리스트 컴프리헨션 (List Comprehension) – 지능형 리스트

[값에 대한 수식 for 변수 in 대상 if 조건]

listcomp

```
nums = [n * 2 for n in range(1, 11)]
for i in nums:
    print(i, end = ', ')
```

실행결과

```
2, 4, 6, 8, 10, 12, 14, 16, 18, 20,
```

```
nums = []
for n in range(1, 11):
    nums.append(n * 2)
```

2. 리스트 관리

❖ 삽입

- append : 인수로 전달한 요소를 리스트 끝에 추가
- insert : 삽입할 위치와 요소값을 전달받아 리스트 중간에 삽입

listinsert

```
nums = [1, 2, 3, 4]
nums.append(5)
nums.insert(2, 99)
print(nums)
```

실행결과

```
[1, 2, 99, 3, 4, 5]
```

2. 리스트 관리

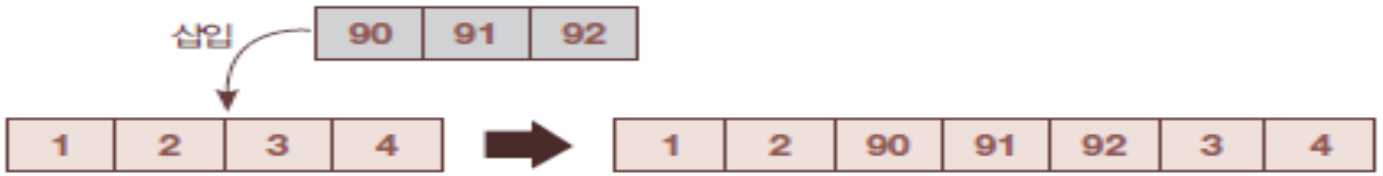
- 범위에 리스트 대입하여 여러 요소 한꺼번에 삽입 가능

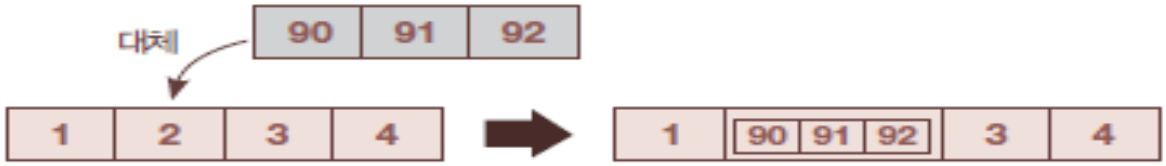
insertrange

```
nums = [1, 2, 3, 4]
nums[2:2] = [90, 91, 92]
print(nums)

nums = [1, 2, 3, 4]
nums[2] = [90, 91, 92]
print(nums)
```

실행결과

```
[1, 2, 90, 91, 92, 3, 4]
[1, 2, [90, 91, 92], 4]
```





2. 리스트 관리

❖ 삭제

- 대상 선택 방법에 따라 다른 메서드 사용

listremove

```
score = [ 88, 95, 70, 100, 99, 80, 78, 50 ]
score.remove(100)
print(score)
del(score[2])
print(score)
score[1:4] = []
print(score)
```

실행결과

```
[88, 95, 70, 99, 80, 78, 50]
[88, 95, 99, 80, 78, 50]
[88, 78, 50]
```

- remove : 인수로 전달받은 요소값 찾아 삭제
- del : 순서값 지정하여 삭제
- clear : 리스트 모든 요소 삭제
- 빈 리스트 대입 : 일정 범위 요소 다수 삭제

2. 리스트 관리

❖ 검색

- index : 특정 요소 위치 찾음
- count : 특정 요소값의 개수 조사

listindex

```
score = [ 88, 95, 70, 100, 99, 80, 78, 50 ]
perfect = score.index(100)
print("만점 받은 학생은 " + str(perfect) + "번입니다.")
pernum = score.count(100)
print("만점자 수는 " + str(pernum) + "명입니다")
```

실행결과

```
만점 받은 학생은 3번입니다.
만점자 수는 1명입니다
```

- min / max : 리스트 요소 중 최소값 / 최대값 찾음
- in / not in : 특정 요소 유무 여부 검사

2. 리스트 관리

❖ 정렬

- 요소를 크기순으로 재배열
- sort : 리스트 정렬하며 요소 순서 조정. 리스트 자체 수정
- reverse : 요소 순서 반대로

sort

```
score = [ 88, 95, 70, 100, 99 ]
score.sort()
print(score)
score.reverse()
print(score)
```

실행결과

```
[70, 88, 95, 99, 100]
[100, 99, 95, 88, 70]
```

- key : 정렬 시 요소 비교할 키 추출
- sorted : 리스트 그대로 두고 정렬된 새로운 리스트 만들어 리턴

3. 튜플

❖ 불변 자료 집합

- 튜플은 초기화한 후 편집할 수 없다는 점에서 리스트와 차이
- 소괄호 사용하여 정의

tuplescore

```
score = ( 88, 95, 70, 100, 99 )
sum = 0
for s in score:
    sum += s
print("총점 : ", sum)
print("평균 : ", sum / len(score))
```

실행결과

```
총점 :  452
평균 :  90.4
```

- print : 튜플 출력 시 소괄호 함께 출력하여 리스트 아님을 나타냄
- 정의할 때에는 소괄호 없이 값만 나열해도 무관함
  - 요소 하나밖에 없는 경우에는 값 다음에 콤마 찍어 튜플임을 표시

3. 튜플

❖ 튜플로 가능한 일

- + / * 연산자 사용하여 연결 및 반복

tupleop

```
tu = 1, 2, 3, 4, 5
print(tu[3])          # 가능
print(tu[1:4])        # 가능
print(tu + (6, 7))    # 가능
print(tu * 2)         # 가능
tu[1] = 100           # 불가능
del tu[1]             # 불가능
```

실행결과

```
4
(2, 3, 4)
(1, 2, 3, 4, 5, 6, 7)
(1, 2, 3, 4, 5, 1, 2, 3, 4, 5)
Traceback (most recent call last):
  File "C:/Python/test.py", line 6, in <module>
    tuple[1] = 100          # 불가능
TypeError: 'tuple' object does not support item assignment
```

- 요소를 변경하거나 삭제할 수는 없음

3. 튜플

- 여러 개의 변수에 값을 한꺼번에 대입
  - 좌변에 변수 목록, 우변에 튜플을 대입

unpacking

```
tu = "이순신", "김유신", "강감찬"
lee, kim, kang = tu
print(lee)
print(kim)
print(kang)
```

실행결과

```
이순신
김유신
강감찬
```

3. 튜플

- 두 개 이상 값을 반환
  - 내부에 요소 포함하는 튜플 사용

tworeturn

```
import time

def gettime():
    now = time.localtime()
    return now.tm_hour, now.tm_min

result = gettime()
print("지금은 %d시 %d분입니다" % (result[0], result[1]))
```

실행결과 지금은 5시 26분입니다

  - import : 모듈 기능 사용 명령

  - divmod : 나눗셈의 몫과 나머지를 튜플로 묶어 리턴

Contents

❖ 목차

1. 사전

❖ 사전 (Dictionary)

- 키와 값의 쌍을 저장하는 대용량 자료구조
- 맵 / 연관배열
- 중괄호 안에 키:값 형태로 콤마로 구분하여 나열

dic

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

실행결과

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

- 빠른 검색 가능
  - [키]

dicread

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

실행결과

```
소년
책
```

1. 사전

- 찾는 키가 없을 경우 예외 발생
  - 예외 처리 구문
  - get 메서드

dicget

```
dic = { 'boy':'소년', 'school':'학교', 'book':'책' }
print(dic.get('student'))
print(dic.get('student', '사전에 없는 단어입니다.'))
```

실행결과

```
None
사전에 없는 단어입니다.
```

- 특정 키 검색 시에는 in 구문 사용

1. 사전

❖ 사전 관리

- 실행 중 삽입, 삭제, 수정 등 편집 가능

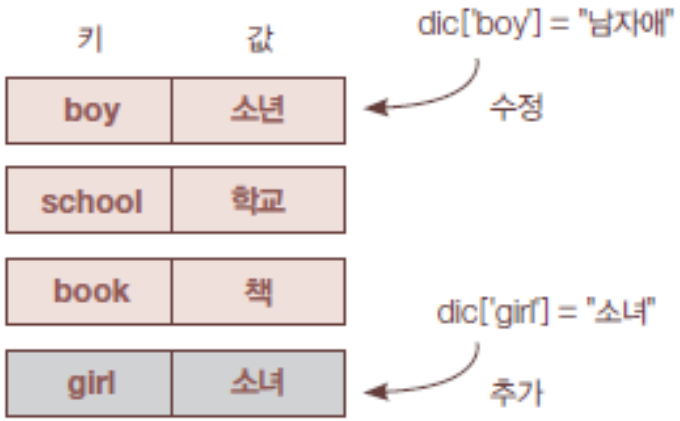
dicchange

```
dic = { 'boy':'소년', 'School':'학교', 'book':'책' }
dic['boy'] = '남자애'
dic['girl'] = '소녀'
del dic['book']
print(dic)
```

실행결과

```
{'boy': '남자애', 'girl': '소녀', 'School': '학교'}
```

- 사전[키]
  - 키의 존재 여부에 따라 동작 다름
    - **존재할 경우 : 기존 값의 변경**
    - **존재하지 않을 경우 : 키를 추가**



1. 사전

- del
  - 해당 키를 찾아 값과 함께 삭제
- keys / values 메서드
  - 사전의 키 / 값 목록 얻음

keys

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

실행결과

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

1. 사전

- dict_* 객체
  - 리스트처럼 순회하여 값 순서대로 읽음

keylist

```
dic = { 'boy':'소년', 'school':'학교', 'book':'책' }
keylist = dic.keys()
for key in keylist:
    print(key)
```

실행결과

```
boy
school
book
```

- update 메서드
  - 두 개 사전을 병합
- dict 함수
  - 빈 사전 만들거나 다른 자료형을 사전으로 변환

1. 사전

❖ 사전의 활용

- Ex) 노래 가사의 특정 알파벳 출현 횟수

alphanum

```
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)
```

실행결과

```
{'b': 4, 'y': 5, 't': 9, 'h': 9, 'e': 22, 'r': 12, 'i': 10, 'v':
2, 's': 10, 'o': 10, .....
```

1. 사전

❖ 사전 컴프리헨션 (Dictionary Comprehension) – 지능형 사전

{ 키와 값에 대한 수식 for 변수 in 대상 if 조건}

```
score_dict = {t[0]: t[1] for t in score_tuples}
print(score_dict)

score_dict = {k : v for k, v in score_tuples}
print(score_dict)

score_dict = {k : v for k, v in score_tuples if len(k) > 5}
print(score_dict)
```

2. 집합

❖ 집합 정의

- 여러 가지 값의 모임

set

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

실행결과

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

- set() 함수
  - 빈 집합 만들거나 다른 컬렉션을 집합형으로 변환

set2

```
print(set("sanghyung"))
print(set([12, 34, 56, 78]))
print(set(("신지희", "한주완", "김태륜")))
print(set({'boy':'소년', 'School':'학교', 'book':'책'}))
print(set())
```

실행결과

```
{'u', 'n', 'a', 's', 'y', 'h', 'g'}
{56, 34, 12, 78}
{'김태륜', '신지희', '한주완'}
{'boy', 'School', 'book'}
set()
```

집합

❖ 집합 컴프리헨션 (Set Comprehension) – 지능형 집합

[값에 대한 수식 for 변수 in 대상 if 조건]

```
a = {i for i in range(1, 101) if i % 3 == 0}
b = {i for i in range(1, 101) if i % 5 == 0}


listv = [dan * num for dan in range(1, 10) for num in range(1, 10)]
setv =  { dan * num for dan in range(1, 10) for num in range(1, 10)}
```

2. 집합

- 인수 없이 set() 함수 호출
  - **공집합 만들기**

- add 메서드
  - 집합에 원소 추가
- update 메서드
  - 집합끼리 결합하여 합집합 만들기
  - 중복 허용되지 않음에 유의

2. 집합

❖ 집합 연산

- 연산을 통한 집합 간 조합

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

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

2. 집합

setop

```
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)
```

실행결과

```
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}
```

객체 정리