

기본 자료구조 이해

1. 표현식

2. 변수

3. 자료형

■ 표현식

● 연산자

연산자	연산	예	결과
+	더하기	$2 + 2$	4
-	빼기	$5 - 2$	3
*	곱하기	$3 * 5$	15
/	나누기	$22 / 8$	2.75
//	정수 나누기 / 나머지 버리기	$22 // 8$	2
%	나머지	$22 \% 8$	6
**	지수	$2 ** 3$	8

■ 표현식

● 연산자를 이용한 표현식 작성

$2 + 3 * 6$

20

$(2 + 3) * 6$

30

$48565878 * 578453$

28093077826734

$2 ** 8$

256

■ 표현식

● 연산자를 이용한 표현식 작성

```
23 / 7
```

```
3.2857142857142856
```

```
23 // 7
```

```
3
```

```
23 % 7
```

```
2
```

```
(5 - 1) * ((7 + 1) / (3 - 1))
```

```
16.0
```

■ 변수

- 변할 수 있는 값
- 값을 저장할 수 있는 메모리 상의 공간
- 변수를 사용하는 이유

- 재사용 가능
- 값에 이름을 부여하고 쉽게 사용할 수 있음

ex) 구글 key : AlzaSyAhVaeWRjyP71Hdd2IQBJb_rHjOcgvUU3M

애드몹 key : ca-app-pub-1251558083101982/7231656074

네이버 Client ID : n76OOSmkSve3Q5K2VsRy

- 반복적으로 등장하는 값을 쉽게 관리

■ 변수

● 변수 이름 규칙

- 빈칸이 없는 한 단어
- 글자, 숫자, 밑줄 기호로만 구성
- 숫자로 시작할 수 없음

유효한 변수 이름	유효하지 않은 변수 이름
current_balance	current-balance (하이픈 X)
currentBalance	current balance (빈칸)
account4	4account (숫자로 시작)
_42	42 (숫자로 시작)
TOTAL_SUM	TOTAL_\$UM (\$ - 특수문자)
hello	'hello' (' - 특수문자)

■ 변수

● 변수 사용

- var1 = 'Python'
- var2 = 12345
- var3 = ['a', 'b', 'c', 'd', 'e']
- data = { 'a' : 1, 'b' : 2, 'c' : 3 }
- a, b = ('python', 'variable') # 튜플 이용 a = 'python' b = 'variable'
- a, b = ['python', 'variable'] # 리스트 이용 a = 'python' b = 'variable'
- a = b = 1234 # a = 1234 b = 1234
- a, b = b, a # 두 변수의 값 바꾸기

● 변수 제거

```
var1 = 'Python'
del var1

var2 = 12345
del(var2)

var3 = ['a', 'b', 'c', 'd', 'e']
del(var3)
```

■ 자료형

● 숫자 (Number)

- 123, 1.23, -97, -3.14e10

● 문자 (String)

- 'A', 'Hello', '888', '-17'

● 논리 (Bool 또는 Boolean)

- True, False

● List : 수정 가능

- [1, 2, 3], ['H', 'e', 'l', 'l', 'o']

● Tuple : 수정 불가

- (1, 2, 3), ('H', 'e', 'l', 'l', 'o')

● Dictionary

- {'a' : 10, 'b' : 20}, {'seoul' : '서울', 'busan' : '부산'}

● Set

- {1, 2, 3}, {'H', 'e', 'l', 'l', 'o'}

■ 자료형 - 숫자

- 정수 (1, 10, -17)

- 실수 (1.414, -2.324)

- 지수 표현 (3.14e2, 1.34e-2)

- ex) $3.14e2 \Rightarrow 3.14 \times 10^2 \Rightarrow 3.14 \times 100 \Rightarrow 314.0$

- $1.34e-2 \Rightarrow 1.34 \times 10^{-2} \Rightarrow 1.34 \times 0.01 \Rightarrow 0.0134$

- 16진수

- $0x9 \Rightarrow 9, 0xA \Rightarrow 10, 0x10 \Rightarrow 16$

- 8진수

- $0o8 \Rightarrow 8, 0o10 \Rightarrow 8$

- 2진수

- $0b1 \Rightarrow 1, 0b10 \Rightarrow 2$

■ 자료형 - 숫자

● 숫자를 사용한 연산

사칙 연산

```
print(3 + 4)
print(3 - 4)
print(3 * 4)
print(3 / 4)
```

나머지 연산

```
print(3 % 4)
print(4 % 3)
```

소수점 버리기

```
print(4 // 3)
print(-4 // 3)
```

제곱 연산

```
print(2 ** 3)
print(3 ** 4)
```

자료형 변환

```
print( int(12.11) )
print( int("3") )
print( int(False) )
print( float(12) )
print( float("3") )
print( float(True) )
```

■ 연습문제

- 23을 5로 나누었을 때의 몫과 나머지 구하기

- 16진수 FF의 10진수 값 구하기

- 8진수 33의 10진수 값 구하기

■ 연습문제

● 제시된 숫자의 각 자리 수 합 구하기

```
num = 215179
total = 0

# 코드 작성

print(total)
```

● 백의 자리 이하 숫자 버리기 (456 → 400 111 → 100)

```
num = 978
result = 0

# 코드 작성

print(result)
```

■ 자료형 - 문자

- 작은 따옴표(') 또는 큰 따옴표(")를 사용하여 표현
 - 'Hello', "Hello", '''Hello''', """"Hello""""
- 작은 따옴표와 큰 따옴표를 중복 사용
 - Hello 'Python' 글자 출력
 - 1) "Hello 'Python'"
 - 2) 'Hello ₩Python₩'
 - Hello "Python" 글자 출력
 - 1) 'Hello "Python"'
 - 2) "Hello ₩Python₩"

특수문자

코드	설명
₩n	줄바꿈
₩t	탭 (일정 간격 띄어쓰기)
₩₩	₩ 문자 표시
₩'	' (quote) 문자 표시
₩"	" (double quote) 문자 표시

■ 자료형 - 문자

● 문자를 사용한 연산

더하기 연산

```
print('Hello' + ' Python')  
print('Hello' + ' ' + 'Python')
```

곱하기 연산

```
print('Hello' * 3)  
print('Hello' * 2 + ' Python')
```

0	1	2	3	4	5	6	7	8	9	10	11
H	e	l	l	o		P	y	t	h	o	n

Indexing

```
text = 'Hello Python'  
print(text[0])  
print(text[6])  
print(text[-1])  
print(text[-6])  
print(text[-12])
```

■ 자료형 - 문자

● 문자를 사용한 연산

0	1	2	3	4	5	6	7	8	9	10	11
H	e	l	l	o		P	y	t	h	o	n

```
# Slicing
text = 'Hello Python'
print(text[0:])
print(text[6:])
print(text[:5])
print(text[:-4])
print(text[-6:-4])
```

0	1	2	3	4	5	6	7	8	9
2	0	1	2		0	3		2	5

```
# Slicing
date = '2012 03 25'
print( date[0:4] + '년' + date[5:7] + '월' + date[8:] + '일' )
```

■ 자료형 - 문자

● Formatting : 문자열의 특정 부분만 바꾸어 사용

1. % 기호 사용

2. {index} 또는 {name} 사용

3. f'문자열' 사용

■ 자료형 - 문자

● % 기호 사용

```
text = 'Python version %s'  
print(text % '3.9.12')
```

```
text = 'Python version %s.%s.%s'  
print(text % (3, 9, 12))
```

Formatting % 문자

코드	설명
%s	문자열
%c	문자
%d	정수
%f	실수
%o	8진수
%x	16진수
%%	% 문자

■ 자료형 - 문자

● {index} 또는 {name} 사용

```
text = 'Python version {}'  
print(text.format('3.9.12'))
```

```
text = 'Python version {0}.{1}.{2}'  
print(text.format(3, 9, 12))
```

```
text = 'Python version {a}.{b}.{c}'  
print(text.format(a=3, b=9, c=12))
```

■ 자료형 - 문자

● f'문자열' 사용

```
version = '3.9.12'
print(f'Python version {version}')

a = 3
b = 9
c = 12
print(f'Python version {a}.{b}.{c}')
```

■ 자료형 - 문자

● 문자 관련 함수

```
# count() - 특정 문자의 개수
text = 'Life is too short, You need Python'
print( text.count('i') )
print( text.count('e') )
print( text.count('Y') )
print( text.count(' ') )
print( text.count(' ', 5) )
print( text.count(' ', 5, 12) )
print( text.count('X') )
```

0	1	2	3	4	5	6	7	8	9
L	i	f	e		i	s		t	o
10	11	12	13	14	15	16	17	18	19
o		s	h	o	r	t	,		Y
20	21	22	23	24	25	26	27	28	29
o	u		n	e	e	d		P	y
30	31	32	33						
t	h	o	n						

■ 자료형 - 문자

● 문자 관련 함수

find() - 특정 문자의 위치

```
text = 'Life is too short, You need Python'
```

```
print( text.find('i') )
```

```
print( text.find(' ') )
```

```
print( text.find(' ', 5) )
```

```
print( text.find(' ', 8, 12) )
```

```
print( text.find('X') )
```

0	1	2	3	4	5	6	7	8	9
L	i	f	e		i	s		t	o
10	11	12	13	14	15	16	17	18	19
o		s	h	o	r	t	,		Y
20	21	22	23	24	25	26	27	28	29
o	u		n	e	e	d		P	y
30	31	32	33						
t	h	o	n						

■ 자료형 - 문자

● 문자 관련 함수

```
# index() - 특정 문자의 위치
text = 'Life is too short, You need Python'
print( text.index('i') )
print( text.index(' ') )
print( text.index(' ', 5) )
print( text.index(' ', 8, 12) )
print( text.index('X') )
```

0	1	2	3	4	5	6	7	8	9
L	i	f	e		i	s		t	o
10	11	12	13	14	15	16	17	18	19
o		s	h	o	r	t	,		Y
20	21	22	23	24	25	26	27	28	29
o	u		n	e	e	d		P	y
30	31	32	33						
t	h	o	n						

■ 자료형 - 문자

● 문자 관련 함수

```
# strip(), rstrip(), lstrip() : 공백 제거
text = ' space '
print( text.rstrip() )
print( text.lstrip() )
print( text.strip() )
```

```
# replace() : 문자열 치환
text = 'Life is too short, You need Python'
print( text.replace('i', '1') )
print( text.replace(',', '\n') )
```

■ 자료형 - 문자

● 문자 관련 함수

```
# maketrans() : 문자열 치환 (ASCII 코드값으로 테이블 생성)
text = 'Life is too short, You need Python'
table = str.maketrans('i', '1')
print(table)
```

```
{105: 49}
```

```
# translate() : Dictionary 형태의 테이블에 따라 문자열 치환
text = 'Life is too short, You need Python'
table = str.maketrans('i', '1')
trans = text.translate(table)
print(trans)
```

```
Llfe 1s too short, You need Python
```


■ 자료형 - 문자

● 문자 관련 함수

```
# split() : 특정 문자열을 기준으로 전체 문자열을 리스트로 변경
text = 'Life is too short, You need Python'
print( text.split(' ') )
print( text.split(',') )
print( text.split('o') )
```

```
# join() : 리스트를 문자열로 변경
items = ['Life', 'is', 'too', 'short,', 'You', 'need', 'Python']
print( ' '.join(items) )
```

```
# str() : 다른 자료형을 문자열로 변환
print( str(123) )
print( str(False) )
print( str([1, 2, 3]) )
print( str({'a': 1, 'b': 2, 'c': 3}) )
```

■ 연습문제

- "Life" is too short, You need 'Python' 출력하기

- 아래와 같은 문자열에서 "short"라는 문자열이 시작되는 위치(인덱스) 구하기

```
s = 'Life is too short, You need Python'
```

- 공백 문자를 기준으로 문자열을 리스트로 변환하기

```
s = 'Life is too short, You need Python'
```

■ 자료형 - 논리

- 참(True) 또는 거짓(False)을 나타내는 자료형
- 제어문의 조건을 표현할 때 주로 사용
- Bool 사용
 - b = True
 - isFile = False
 - isDirectory = False
- Bool 함수

```
print( bool(0) )  
print( bool(1) )  
  
print( bool('') )  
print( bool('a') )  
  
print( bool([]) )  
print( bool([1, 2]) )  
  
print( bool({}) )  
print( bool({'a': 1, 'b': 2}) )
```

■ 자료형 - List

- 여러개의 자료를 하나로 묶어 사용할 수 있는 자료형
- 대괄호([])로 표현하고 쉼표(,)로 각각의 요소를 구분
 - 리스트명 = [요소1, 요소2, 요소3, ... , 요소N]
- List 사용
 - list1 = [] 또는 list1 = list()
 - list2 = [1, 2, 3, 4, 5]
 - list3 = ['a', 'b', 'c', '가', '나', '다', 'ABC']
 - list4 = [1, 2, 3, '가', '나', '다', False]
 - list5 = ['a', 'b', 'c', [1, 2, 3], '가', '나', '다']
- List 함수

```
print( list('123') )  
  
print( list((1, 2, 3)) )  
  
print( list({1, 2, 3}) )  
  
print( list({'a': 1, 'b': 2, 'c': 3}) ) # [a, b, c]
```

■ 자료형 - List

● List를 사용한 연산

더하기 연산

```
list1 = [1, 2, 3] + [4, 5]  
print(list1)
```

```
list2 = [1, 2, 3]  
list2 = list2 + ['a', 'b']  
print(list2)
```

곱하기 연산

```
list1 = [1, 2, 3] * 3  
print(list1)
```

```
list2 = [1, 2, 3]  
list2 = list2 * 3  
print(list2)
```

■ 자료형 - List

● List를 사용한 연산

0	1	2	3	4	5	6	7	8	9	10	11
H	e	l	l	o		P	y	t	h	o	n

```
# indexing
```

```
items = ['H', 'e', 'l', 'l', 'o', ' ', 'P', 'y', 't', 'h', 'o', 'n']  
print(items[0])  
print(items[6])  
print(items[-1])  
print(items[-8])
```

```
# slicing
```

```
items = ['H', 'e', 'l', 'l', 'o', ' ', 'P', 'y', 't', 'h', 'o', 'n']  
print(items[0:])  
print(items[6:])  
print(items[:5])  
print(items[:-4])  
print(items[-6:-4])
```

■ 자료형 - List

● List를 사용한 연산

0	1	2	3			4	5	6
'a'	'b'	'c'	0	1	2	'가'	'나'	'다'
			1	2	3			

중첩 리스트

```
items = ['a', 'b', 'c', [1, 2, 3], '가', '나', '다']
```

```
print(items[2:5])
```

```
print(items[3])
```

```
print(items[3][1])
```

```
print(items[3][:])
```

```
print(items[3][: -1])
```

■ 자료형 - List

● List를 사용한 연산

0	1				2
'a'	0	1	2		'가'
	1	2	0	1	
			'Life'	'is'	

중첩 리스트

```
items = ['a', [1, 2, ['Life', 'is', 'too', 'short']], '가']  
print(items[1])  
print(items[1][1])  
print(items[1][2])  
print(items[1][2][2])
```


■ 자료형 - List

● 요소 추가

```
# List 요소 추가
```

```
items = [1]
```

```
items = items + [2, 3]
```

```
print(items)
```

```
items = items + [['a', 'b']]
```

```
print(items)
```

```
items[3] = items[3] + ['c']
```

```
print(items)
```

```
items.append('가')
```

```
print(items)
```

■ 자료형 - List

● 요소 수정

```
# List 요소 수정
items = [1, 2, 3, 4, 5]
items[0] = 10
print(items)

items[1] = items[1] * 2
print(items)

items[2:4] = [30, 40]
print(items)

items[:] = ['a', 'b', 'c']
print(items)
```

■ 자료형 - List

● 요소 삭제

```
# List 요소 삭제
items = [1, 1, '삭제', 3, 3, 4, 5]
del items[2]
print(items)

items.remove(3) # 첫번째 등장 요소 제거
print(items)

last = items.pop() # 마지막 요소 (삭제 + 조회)
print(last, items)
```

■ 자료형 - List

● List 관련 함수

```
# count() : 특정 요소의 개수
```

```
items = [1, 1, 2, 3, 3, 4, 4, 4]
```

```
print( items.count(4) )
```

```
print( items.count(3))
```

```
# index() : 특정 요소의 위치
```

```
items = ['P', 'y', 't', 'h', 'o', 'n']
```

```
print( items.index('y') )
```

```
print( items.index('n') )
```

```
print( items.index('X') )
```

■ 자료형 - List

● List 관련 함수

```
items = [3, 4, 2, 5, 1]
```

```
# sort() : 정렬 (오름차순)
```

```
items.sort()  
print(items)
```

```
# sort(reverse=True) : 정렬 (내림차순)
```

```
items.sort(reverse=True)  
print(items)
```

```
# reverse() : 현재 상태의 반대로 정렬 (인덱스 기준)
```

```
items = [3, 4, 2, 5, 1]  
items.reverse()  
print(items)
```

```
items = ['s', 'h', 'a', 'k', 'e']  
items.reverse()  
print(items)
```

■ 연습문제

- 중첩 리스트 person에서 인덱싱을 사용하여 아래와 같이 출력하기

```
person = [  
    'ggoreb',  
    20,  
    ['서울', '관악구', '신림동'],  
    ['대전', '서구', '둔산동']  
]
```

[ggoreb]신림동/둔산동

- 리스트 요소 정렬하기 (오름차순, 내림차순)

```
ch_list = ['p', 'y', 't', 'h', 'o', 'n']
```

['h', 'n', 'o', 'p', 't', 'y']
['y', 't', 'p', 'o', 'n', 'h']

■ 자료형 - Tuple

- 여러개의 자료를 하나로 묶어 사용할 수 있는 자료형 (≡ List)
- 소괄호(())로 표현하고 쉼표(,)로 각각의 요소를 구분
 - 튜플명 = (요소1, 요소2, 요소3, ... , 요소N)
- Tuple 사용
 - tuple1 = () 또는 tuple1 = tuple()
 - tuple2 = ('a',) ← 요소가 한개인 경우 반드시 쉼 표 (,) 를 붙임
 - tuple3 = ('a', 'b', 'c', '가', '나', '다', 'ABC')
 - tuple4 = 1, 2, 3 ← 요소가 여러개인 경우 괄 호 생략 가능
 - tuple5 = ('a', 'b', 'c', (1, 2, 3), ('가', '나'))
- List 와의 차이점
 - List : 요소 추가/수정/삭제 가능
 - Tuple : 요소 수정/삭제 불가

■ 자료형 - Dictionary

- 여러개의 자료를 하나로 묶어 사용할 수 있는 자료형
- 중괄호({})로 표현하고 쉼표(,)로 각각의 요소를 구분
- List / Tuple 과는 다르게 index가 아닌 key를 사용
 - 딕셔너리명 = { 키1:값1, 키2:값2, 키3:값3, ... , 키N:값N }
- Dictionary 사용
 - dict1 = { } 또는 dict1 = dict()
 - dict2 = { 'name' : 'ggoreb' }
 - dict3 = { 'name' : 'ggoreb', 'age' : 20 }
 - dict4 = { 'name' : 'ggoreb', 'age' : 20, 'hobby' : ['당구', '배드민턴'] }
 - dict5 = { 'name':'ggoreb', 'age':20, 'hobby':['당구', '배드민턴'], 123:456 }

key	value
'name'	'ggoreb'
'age'	20
'hobby'	['당구', '배드민턴']
123	456

■ 자료형 - Dictionary

● 요소 추가

```
# Dictionary 요소 추가
dic = { 'a' : 1, 'b' : 2, 'c' : 3 }

dic['d'] = '추가'
print(dic)

dic['e'] = ('가', '나')
print(dic)

dic[3] = 4 # 숫자 key
print(dic)

dic[(4, )] = 5 # Tuple key
print(dic)

dic[True] = 6 # Bool key
print(dic)
```

■ 자료형 - Dictionary

● 요소 수정

```
# Dictionary 요소 수정
```

```
dic = { 'a' : 1, 'b' : 2, 'c' : 3 }
```

```
dic['a'] = '100'
```

```
print(dic)
```

```
dic['b'] = dic['b'] * 2
```

```
print(dic)
```

```
dic.update({'c': 10, 'd': 1000}) # 수정 + 추가
```

```
print(dic)
```

■ 자료형 - Dictionary

● 요소 삭제

```
# Dictionary 요소 삭제
dic = { 'a' : 1, 'b' : 2, 'c' : 3 }

del dic['a']
print(dic)

dic.clear()
print(dic)
```

■ 자료형 - Dictionary

● Dictionary 관련 함수

```
# keys() : key의 목록 확인
```

```
dic = { 'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5 }
```

```
print( dic.keys() )
```

```
keys = list(dic.keys()) # key → List
```

```
for key in keys : # List로 만들어진 key 요소를 하나씩 추출
```

```
    print(key, dic[key]) # 출력
```

```
# values() : value의 목록 확인
```

```
dic = { 'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5 }
```

```
print( dic.values() )
```

```
items = list(dic.values()) # value → List
```

```
for value in items: # List로 만들어진 value 요소를 하나씩 추출
```

```
    print(value) # 출력
```

■ 자료형 - Dictionary

● Dictionary 관련 함수

```
# items() : key, value의 목록 확인
dic = { 'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5 }
print( dic.items() )

items = list(dic.items()) # value → List
for key, value in items: # (key, value) 요소를 하나씩 추출
    print(key, value) # 출력
```

```
# clear() : 모든 데이터 삭제
dic = { 'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5 }
print( dic.clear() )
```

■ 자료형 - Dictionary

● Dictionary 관련 함수

```
# get() : key를 이용하여 value 조회
dic = {
    'name': 'ggoreb',
    'age': 20,
    'hobby': ['당구', '배드민턴']
}
print( dic.get('id') )
print( dic['id'] ) # key error
```

```
# in : 지정된 key가 존재하는지 확인
dic = {
    'name': 'ggoreb',
    'age': 20,
    'hobby': ['당구', '배드민턴']
}
print( 'name' in dic )
print( 'address' in dic )
```

■ 연습문제

● 데이터

```
number = {  
    '서울': '02', '대전': '042', '부산': '051',  
    '광주': '062', '제주': '064'  
}
```

1. '부산'에 해당하는 value 출력
2. '천안' key가 존재하는지 확인
3. key 리스트 생성
4. value 리스트 생성

■ 연습문제

● 인디언식 이름 짓기

1. 자신의 태어난 년도 맨 끝자리와 월, 일에 해당하는 수식어를 조합
2. 태어난 년도는 색이나 성격, 월은 주어, 일은 술어를 표현

```
year = {  
    0: '시끄러운 또는 말 많은',  
    1: '푸른',  
    ...  
    9: '욕심많은'  
}  
month = {  
    1: '늑대', 2: '태양', ... 12: '바람'  
}  
day = {  
    1: '와(과) 함께 춤을',  
    2: '의 기상',  
    ...  
    30: '의 혼',  
    31: '은(는) 말이 없다',  
}
```

결과 예)

birthday = '1971-08-07'

푸른 달빛의 환생

■ 자료형 - Set

- 여러개의 자료를 하나로 묶어 사용할 수 있는 자료형
- 중괄호({})로 표현하고 쉼표(,)로 각각의 요소를 구분
- 겉모습은 Dictionary처럼 보이지만 내부 요소는 List와 유사
- 각 요소는 순서가 없으며, 중복값을 허용하지 않음
 - Set명 = { 값1, 값2, 값3, ... , 값N }
- Set 사용
 - set1 = set() set1 = {} (X)
 - set2 = { 1, 2, 3 }
 - set3 = set('Python') ← 한 글자 씩 분리되어 집 합 요 소로 생성
 - set4 = set(['a', 'b', 'c']) ← List
 - set5 = set((1, 2, 3)) ← Tuple
 - set6 = set({'a' : 1, 'b' : 2}) ← Dictionary, 각 key만 집합요소로 생성

■ 자료형 - Set

● 요소 추가

```
# Set 요소 추가
s = set(['a', 'b', 1, 2, '가', '나', 'a', 1, '가'])
s.add(3)
print(s)
s.update(['z', 'y', 'x'])
print(s)

s[0] # 조회 불가
s += {3} # 연산자를 통한 값 추가 불가
```

● 요소 삭제

```
# Set 요소 삭제
s = set(['a', 'b', 1, 2, '가', '나', 'a', 1, '가'])
print(s.pop()) # 마지막 요소 (삭제 + 조회)
s.remove(2)
print(s)
s.clear()
print(s)
```

■ 자료형 - Set

● Set 관련 함수

교집합

```
a = set([1, 2, 3, 4, 5])  
b = set([3, 4, 5, 6, 7])  
print(a & b)  
print(a.intersection(b))
```

합집합

```
a = set([1, 2, 3, 4, 5])  
b = set([3, 4, 5, 6, 7])  
print(a | b)  
print(a.union(b))
```

차집합

```
a = set([1, 2, 3, 4, 5])  
b = set([3, 4, 5, 6, 7])  
print(a - b)  
print(a.difference(b))
```

■ 자료형 - Set

- 중복 자료를 제거하기 위한 필터 역할로도 사용

```
# List 중복 제거
```

```
items = [1, 2, 3, 4, 1, 2, 3]
```

```
s = set(items)
```

```
print(s)
```

```
items2 = list(s)
```

```
print(items2)
```

```
# Tuple 중복 제거
```

```
items = (1, 2, 3, 4, 1, 2, 3)
```

```
s = set(items)
```

```
print(s)
```

```
items2 = tuple(s)
```

```
print(items2)
```

■ 연습문제

- 두 개의 집합에서 공통 요소만 출력하기 (교집합)

```
lotto_num1 = {6, 12, 17, 21, 34, 37}  
lotto_num2 = {6, 12, 19, 24, 34, 41}
```

- Set을 활용하여 List 요소의 중복 데이터 제거하기

```
num_list = [9, 7, 11, 26, 15, 9, 15, 26]
```