

■ 자료형

- 숫자 (Number)

- 123, 1.23, -97, -3.14e10

- 문자 (String)

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

- 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'}

- 논리 (Bool 또는 Boolean)

- True, False

■ 자료형 - 숫자

- 정수 (1, 10, -17)

- 실수 (1.414, -2.324)

- 지수 표현 (3.14e10, 1.34e-10)

- ex) $3.14e10 \Rightarrow 3.14 \times 10^{10} \Rightarrow 3.14 \times 100 \Rightarrow 314$

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

- 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

● 사칙 연산

```
>>> 3 + 4
7
>>> 3 - 4
-1
>>> 3 * 4
12
>>> 3 / 4
0.75
```

● 제곱 연산 ($a ** b$) : a 의 b 제곱 연산 (a^b)

```
>>> 2 ** 3
8
>>> 3 ** 4
81
```

● 나머지 연산 ($a \% b$) : a 를 b 로 나눈 나머지

```
>>> 3 % 4
3
>>> 4 % 3
1
```

● 소수점 버리기

```
>>> 4 / 3
1.3333333333333333
>>> 4 // 3
1
>>> -4 / 3
-1.3333333333333333
>>> -4 // 3
-2
```

● 다른 자료형을 정수 또는 실수로 만들기

```
>>> int(12.11)
12
>>> int('3')
3
>>> int(False)
0
>>> float(12)
12.0
>>> float('3')
3.0
>>> float(True)
1.0
```

■ 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 W'PythonW''

- Hello "Python" 글자 출력

- 1) 'Hello "Python"'

- 2) "Hello W"PythonW""

- Hello

- 'Python'

- 1) """Hello

- 'Python'"""

- 2) '''Hello

- W'PythonW'''

코드	설명
<code>\n</code>	줄바꿈
<code>\t</code>	탭 (띄어쓰기)
<code>\\</code>	<code>\</code> 문자
<code>'\'</code>	단일 인용(') 문자
<code>"\"</code>	이중 인용(") 문자
<code>\\r, \\f, \\a, \\b, \\000</code>	

■ 자료형 - 문자

● 더하기 연산

```
>>> 'Hello' + ' Python'
'Hello Python'

>>> 'Hello' + ' ' + ' Python'
'Hello Python'
```

● 곱하기 연산

```
>>> 'Hello' * 3
'HelloHelloHello'

>>> 'Hello' * 2 + ' Python'
'HelloHello Python'
```

● Indexing, Slicing

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

```
>>> str = 'Hello Python'
>>> str[0]
'H'
>>> str[6]
'P'
>>> str[-1]
'n'
>>> str[-6]
'P'
>>> str[-12]
'H'
```

```
>>> str = 'Hello Python'
>>> str[0:]
'Hello Python'
>>> str[6:]
'Python'
>>> str[:6]
'Hello '
>>> str[:-4]
'Hello Py'
>>> str[6:-4]
'Py'
```

■ 자료형 - 문자

● Slicing

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

- 20120325 출력

```
>>> date = '2012 03 25'  
>>> date[0:4] + date[5:7] + date[8:]  
'20120325'
```

- 2012년03월25일 출력

```
>>> date = '2012 03 25'  
>>> date[0:4] + '년' + date[5:7] + '월' + date[8:] + '일'  
'2012년03월25일'
```

※ 특정 위치의 문자 변경은 불가

문자 자료형은 immutable한 자료형

```
>>> date[0] = '1'  
Traceback (most recent call last):  
  File "<pyshe11#63>", line 1, in <module>  
    date[0] = '1'  
TypeError: 'str' object does not support item assignment
```

■ 자료형 - 문자

● Formatting : 문자열의 특정 부분만 바꿔 사용

- 동적으로 입력되는 부분에 특수코드 사용
- 여러개의 값을 넣는 경우 () 와 , 를 이용하여 구분

```
>>> 'Python version %d' % 1
'Python version 1'

>>> 'Python version %d' % 3
'Python version 3'

>>> 'Python version %s' % 'latest'
'Python version latest'

>>> 'Python version %f' % 3.6
'Python version 3.600000'

>>> 'Python version %d%%' % 100
'Python version 100%'

>>> '%s version %d' % ('Python', 3)
'Python version 3'
```

코드	설명
%s	문자열 (string)
%c	문자 (character)
%d	정수 (integer)
%f	실수 (float)
%o	8진수
%x	16진수
%%	% 문자

■ 자료형 - 문자

● Formatting : 문자열의 특정 부분만 바꿔 사용

- Format 코드와 숫자 함께 사용

```
>>> 'name %5s' % 'abc'
'name  abc'

>>> 'name %-5s' % 'abc'
'name abc  '

>>> 'name %5s' % 'abcdefg'
'name abcdefg'

>>> 'score %0.2f' % 91.1234
'score 91.12'

>>> 'score %10.2f' % 91.1234
'score      91.12'
```

코드	설명
%s	문자열 (string)
%c	문자 (character)
%d	정수 (integer)
%f	실수 (float)
%o	8진수
%x	16진수
%%	% 문자

■ 자료형 - 문자

● format 함수 이용

- {index}

```
>>> 'score {0}'.format(100)
'score 100'

>>> 'score {0}, {1}'.format(100, 80)
'score 100, 80'

>>> '{0} {1}'.format('score', 70)
'score 70'
```

- {name}

```
>>> '{lang} is easy'.format(lang='Python')
'Python is easy'

>>> '{lang} is easy, version {ver}'.format(ver=3.6, lang='Python')
'Python is easy, version 3.6'
```

- {index}, {name} 혼용 사용

```
>>> '{lang} is easy, version {0}.{1}'.format(3, 6, lang='Python')
'Python is easy, version 3.6'
```

■ 자료형 - 문자

● format 함수 이용

- 여백 및 정렬

```
>>> '{0:<10}'.format('1234')
'1234          '
>>> '{0:>10}'.format('1234')
'          1234'
>>> '{0:^10}'.format('1234')
'      1234      '
>>> '{0:공^10}'.format('1234')
'공공공1234공공공'
>>> '{0:!!^10}'.format('1234')
'!!!1234!!!'
```

- f 문자열 Formatting

```
>>> name = 'ggoreb'
>>> f'name is {name}'
'name is ggoreb'

>>> age = 10
>>> f'10년 후 나이 {age + 10}살'
'10년 후 나이 20살'

>>> f'{{ }}'
'{ }'
```

■ 자료형 - 문자

● 문자열 관련 함수

- count() : 특정 문자의 개수

```
>>> str = 'Life is too short, You need Python'

>>> str.count('i')
2

>>> str.count('e')
3

>>> str.count('Y')
1

>>> str.count(' ')
6

>>> str.count(' ', 5)
5

>>> str.count(' ', 5, 12)
2

>>> str.count('x')
0
```

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

● 문자열 관련 함수

- find() : 특정 문자의 위치

```
>>> str = 'Life is too short, You need Python'
```

```
>>> str.find('i')
```

```
1
```

```
>>> str.find(' ')
```

```
4
```

```
>>> str.find(' ', 5)
```

```
7
```

```
>>> str.find(' ', 8, 12)
```

```
11
```

```
>>> str.find('x')
```

```
-1
```

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

● 문자열 관련 함수

- index() : 특정 문자의 위치

```
>>> str = 'Life is too short, You need Python'
```

```
>>> str.index('i')
```

```
1
```

```
>>> str.index(' ')
```

```
4
```

```
>>> str.index(' ', 5)
```

```
7
```

```
>>> str.index(' ', 8, 12)
```

```
11
```

```
>>> str.index('x')
```

```
error
```

```
>>> str.index('x')
Traceback (most recent call last):
  File "<pyshell#34>", line 1, in <module>
    str.index('x')
ValueError: substring not found
```

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

● 문자열 관련 함수

- join() : 각 문자 사이에 문자 삽입

```
>>> '/'.join('가나다라')  
'가/나/다/라'
```

```
>>> str = '12345'  
>>> sep = ','
```

```
>>> sep.join(str)  
'1, 2, 3, 4, 5'
```

```
>>> ','.join('1')  
'1'
```

- upper() : 대문자로 변경

```
>>> str = 'Life is too short, You need Python'
```

```
>>> str.upper()  
'LIFE IS TOO SHORT, YOU NEED PYTHON'
```

- lower() : 소문자로 변경

```
>>> str = 'Life is too short, You need Python'
```

```
>>> str.lower()  
'life is too short, you need python'
```

■ 자료형 - 문자

● 문자열 관련 함수

- strip(), rstrip(), lstrip() : 공백 제거

```
>>> str = ' space '
```

```
>>> str.rstrip()  
' space'
```

```
>>> str.lstrip()  
'space '
```

```
>>> str.strip()  
'space'
```

- replace() : 문자열 치환

```
>>> str = 'Life is too short, You need Python'
```

```
>>> str.replace('i', 'l')  
'L1fe 1s too short, You need Python'
```

```
>>> str.replace(',', ' ', 'Wn')  
'Life is too shortWnYou need Python'
```

```
>>> print(str.replace(',', ' ', 'Wn'))  
Life is too short  
You need Python
```

■ 자료형 - 문자

● 문자열 관련 함수

- `split()` : 특정 문자열을 기준으로 전체 문자열을 리스트로 변경

```
>>> str = 'Life is too short, You need Python'

>>> str.split()
['Life', 'is', 'too', 'short,', 'You', 'need', 'Python']

>>> str.split(' ')
['Life is too short', 'You need Python']

>>> str.split('o')
['Life is t', '', ' sh', 'rt, Y', 'u need Pyth', 'n']
```

● 다른 자료형을 문자열로 만들기

```
>>> str(123)
'123'

>>> str(False)
'False'

>>> str([1, 2, 3])
'[1, 2, 3]'

>>> str({'a', 'b', 'c'})
"{'a', 'b', 'c'}"
```


■ 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']

■ 자료형 - List

● 더하기 연산

```
>>> [1, 2, 3] + [4, 5]  
[1, 2, 3, 4, 5]
```

```
>>> list = [1, 2, 3]  
>>> list + ['a', 'b']  
[1, 2, 3, 'a', 'b']
```

● 곱하기 연산

```
>>> [1, 2, 3] * 3  
[1, 2, 3, 1, 2, 3, 1, 2, 3]
```

```
>>> list = [1, 2, 3]  
>>> list * 3  
[1, 2, 3, 1, 2, 3, 1, 2, 3]
```

● Indexing, Slicing

0	1	2	3	4
1	2	3	4	5

```
>>> list = [1, 2, 3, 4, 5]
```

```
>>> list[0]  
1
```

```
>>> list[4]  
5
```

```
>>> list[-1]  
5
```

```
>>> list[-3]  
3
```

```
>>> list = [1, 2, 3, 4, 5]
```

```
>>> list[0:]  
[1, 2, 3, 4, 5]
```

```
>>> list[2:]  
[3, 4, 5]
```

```
>>> list[: -2]  
[1, 2, 3]
```

```
>>> list[2: -1]  
[3, 4]
```

■ 자료형 - List

● 중첩 Slicing

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

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

```
>>> list[2:5]  
['c', [1, 2, 3], '가']
```

```
>>> list[3]  
[1, 2, 3]
```

```
>>> list[3][1]  
2
```

```
>>> list[3][:]  
[1, 2, 3]
```

```
>>> list[3][: -1]  
[1, 2]
```

■ 자료형 - List

● 중첩 Slicing (3중)

0	1			
'a'	0			1
	0	1	2	3
	'Life'	'is'	'too'	'short'
				2

```
>>> list = [ 'a', [ [ 'Life', 'is', 'too', 'short' ], 2 ] ]
```

```
>>> list[1]  
[['Life', 'is', 'too', 'short'], 2]
```

```
>>> list[1][1]  
2
```

```
>>> list[1][0]  
['Life', 'is', 'too', 'short']
```

```
>>> list[1][0][2]  
'too'
```

■ 자료형 - List

● List 요소 값 수정

```
>>> list = [1, 2, 3, 4, 5]

>>> list[0] = 10
[10, 2, 3, 4, 5]

>>> list[1] = list[1] * 2
[10, 4, 3, 4, 5]

>>> list[2:4] = [30, 40]
[10, 4, 30, 40, 5]

>>> list[:] = [1, 2, 3, 4, 5]
[1, 2, 3, 4, 5]

>>> list[2:3] = [] # 2번 요소 제거
[1, 2, 4, 5]

>>> del list[1] # 1번 요소 제거
[1, 4, 5]
```

● List 요소 값 추가

```
>>> list = [1]

>>> list = list + [2, 3]
[1, 2, 3]

>>> list = list + [['a', 'b']]
[1, 2, 3, ['a', 'b']]

>>> list += [4, 5, 6]
[1, 2, 3, ['a', 'b'], 4, 5, 6]
```

■ 자료형 - List

● List 관련 함수

- append() : 마지막 요소로 추가

```
>>> list = []  
  
>>> list.append(1)  
[1]  
  
>>> list.append('a')  
[1, 'a']  
  
>>> list.append(['가', '나'])  
[1, 'a', ['가', '나']]
```

- insert() : 지정 위치에 요소 삽입

```
>>> list = []  
  
>>> list.insert(0, 'a')  
['a']  
  
>>> list.insert(0, 'b')  
['b', 'a']  
  
>>> list.insert(1, ['가', '나'])  
['b', ['가', '나'], 'a']
```

- extend() : 마지막 요소로 추가

```
>>> list = []  
  
>>> list.extend(1)  
[1]  
  
>>> list.extend('a')  
[1, 'a']  
  
>>> list.extend(['가', '나'])  
[1, 'a', '가', '나']
```

■ 자료형 - List

● List 관련 함수

- remove() : 첫번째로 등장하는 지정 요소 제거

```
>>> list = ['a', 'b', 'c', 'd', 'e']
```

```
>>> list.remove('c')  
['a', 'b', 'd', 'e']
```

```
>>> list.remove('z')  
error
```

```
>>> list.remove('z')  
Traceback (most recent call last):  
  File "<pyshell#8>", line 1, in <module>  
    list.remove('z')  
ValueError: list.remove(x): x not in list
```

- pop() : 지정된 위치의 요소를 추출하면서 제거 (기본위치 : 마지막)

```
>>> list = ['a', 'b', 'c', 'd', 'e']
```

```
>>> list.pop() # ['a', 'b', 'c', 'd']  
'e'
```

```
>>> list.pop() # ['a', 'b', 'c']  
'd'
```

```
>>> list.pop(0) # ['b', 'c']  
'a'
```

■ 자료형 - List

● List 관련 함수

- sort() : 오름차순 정렬

```
>>> list = [3, 4, 2, 5, 1]
[3, 4, 2, 5, 1]

>>> list.sort()
[1, 2, 3, 4, 5]

>>> list = ['s', 'h', 'a', 'k', 'e']
['s', 'h', 'a', 'k', 'e']

>>> list.sort()
['a', 'e', 'h', 'k', 's']
```

- sort(reverse=True) : 내림차순 정렬

```
>>> list = [3, 4, 2, 5, 1]
[3, 4, 2, 5, 1]

>>> list.sort(reverse=True)
[5, 4, 3, 2, 1]

>>> list = ['s', 'h', 'a', 'k', 'e']
['s', 'h', 'a', 'k', 'e']

>>> list.sort(reverse=True)
['s', 'k', 'h', 'e', 'a']
```

- reverse() : 현재 상태의 반대로 정렬

```
>>> list = [3, 4, 2, 5, 1]
[3, 4, 2, 5, 1]

>>> list.reverse()
[1, 5, 2, 4, 3]

>>> list = ['s', 'h', 'a', 'k', 'e']
['s', 'h', 'a', 'k', 'e']

>>> list.reverse()
['e', 'k', 'a', 'h', 's']
```


■ 자료형 - List

● List 관련 함수

- index() : 요소의 위치 확인

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

```
>>> list.index('y')
1
```

```
>>> list.index('n')
5
```

```
>>> list.index('p')
error
```

```
>>> list.index('p')
Traceback (most recent call last):
  File "<pyshell#76>", line 1, in <module>
    list.index('p')
ValueError: 'p' is not in list
```

- count() : 요소의 개수 확인

```
>>> list1 # string → list
['L', 'i', 'f', 'e', ' ', 'i', 's', ' ', 't', 'o', 'o', ' ', 's', 'h', 'o', 'r', 't', ' ', ' ', 'Y', 'o', 'u', ' ', ' ', 'n', 'e', 'e', 'd', ' ', 'P', 'y', 't', 'h', 'o', 'n']
```

```
>>> list1.count('i')
2
```

```
>>> list1.count('e')
3
```

```
>>> list1.count(' ')
6
```

■ 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 쉼표(,)를 붙임
 - tuple3 = ('a', 'b', 'c', '가', '나', '다', 'ABC')
 - tuple4 = 1, 2, 3 ← 요소가 여러개인 경우 괄호 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

```
>>> tuple = (1, 2, 3)
>>> tuple[0] = 10
Traceback (most recent call last):
  File "<pyshell#52>", line 1, in <module>
    tuple[0] = 10
TypeError: 'tuple' object does not support item assignment
>>> del tuple[0]
Traceback (most recent call last):
  File "<pyshell#53>", line 1, in <module>
    del tuple[0]
TypeError: 'tuple' object doesn't support item deletion
```

■ 자료형 - Tuple

● 더하기 연산

```
>>> (1, 2, 3) + (4, 5)
(1, 2, 3, 4, 5)

>>> tuple = (1, 2, 3)
>>> tuple + (4, 5)
(1, 2, 3, 4, 5)

>>> tuple += (4, 5)
(1, 2, 3, 4, 5)
```

● 곱하기 연산

```
>>> (1, 2, 3) * 3
(1, 2, 3, 1, 2, 3, 1, 2, 3)

>>> tuple = (1, 2, 3)
>>> tuple * 3
(1, 2, 3, 1, 2, 3, 1, 2, 3)

>>> tuple *= 3
(1, 2, 3, 1, 2, 3, 1, 2, 3)
```

● Indexing, Slicing

0	1	2	3	4
1	2	3	4	5

```
>>> tuple = (1, 2, 3, 4, 5)

>>> tuple[0]
1

>>> tuple[4]
5

>>> tuple[-1]
5

>>> tuple[-3]
3
```

```
>>> tuple = (1, 2, 3, 4, 5)

>>> tuple[0:]
(1, 2, 3, 4, 5)

>>> tuple[2:]
(3, 4, 5)

>>> tuple[: -2]
(1, 2, 3)

>>> tuple[2: -1]
(3, 4)
```

■ 자료형 - Tuple

● Tuple 요소 값 수정 불가

```
>>> tuple = (1, 2, 3, 4, 5)
```

```
>>> tuple[0] = 10  
error
```

```
>>> tuple[0] = 10  
Traceback (most recent call last):  
  File "<pyshell#26>", line 1, in <module>  
    tuple[0] = 10  
TypeError: 'tuple' object does not support item assignment
```

```
>>> tuple[2:3] = ()  
error
```

```
>>> tuple[2:3] = ()  
Traceback (most recent call last):  
  File "<pyshell#27>", line 1, in <module>  
    tuple[2:3] = ()  
TypeError: 'tuple' object does not support item assignment
```

```
>>> del tuple[1]  
error
```

```
>>> del tuple[1]  
Traceback (most recent call last):  
  File "<pyshell#28>", line 1, in <module>  
    del tuple[1]  
TypeError: 'tuple' object doesn't support item deletion
```

● Tuple을 사용하는 이유

- 리스트에 비해 더 적은 메모리를 사용하고 속도도 빠름
- 수정이 안되므로 더 안정적, 프로그래머의 실수 방지

■ 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

● 더하기(+) 및 곱하기(*) 연산 불가

● Dictionary 요소 값 확인 / 추가 / 수정 / 삭제

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

```
>>> dict['b']
```

```
2
```

```
>>> dict['d'] = '추가'
```

```
{ 'a': 1, 'b': 2, 'c': 3, 'd': '추가' }
```

```
>>> dict['e'] = ('가', '나')
```

```
{ 'a': 1, 'b': 2, 'c': 3, 'd': '추가', 'e': ('가', '나') }
```

```
>>> dict['e'] = 5
```

```
{ 'a': 1, 'b': 2, 'c': 3, 'd': '추가', 'e': 5 }
```

```
>>> del dict['c']
```

```
{ 'a': 1, 'b': 2, 'd': '추가', 'e': 5 }
```

■ 자료형 - Dictionary

● 주의사항

- key는 고유하므로 중복으로 사용하면 이전 값은 사라지고 현재 값으로 변경

```
>>> dict = { 'name' : 'ggoreb' }
```

```
>>> dict['name'] = 'kim'  
{'name': 'kim'}
```

- key는 변경 불가, 변경이 가능한 값은 key로 사용 불가

```
>>> dict = { 'a' : 1, 1 : '가' } # O
```

```
>>> dict = { (1, 2, 3) : 'tuple' } # O
```

```
>>> dict = { [1, 2, 3] : 'list' } # X
```

```
>>> dict = { {1:'a'} : 'dict' } # X
```

■ 자료형 - Dictionary

● Dictionary 관련 함수

- keys() : key의 목록 확인 (≠ List)

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

```
>>> dict.keys()  
dict_keys(['a', 'b', 'c', 'd', 'e'])
```

```
>>> list = list(dict.keys()) # key → List  
['a', 'b', 'c', 'd', 'e']
```

```
>>> for key in list:    # List로 만들어진 key 요소를 하나씩 추출  
    print(dict[key]) # 출력
```

```
1  
2  
3  
4  
5
```


■ 자료형 - Dictionary

● Dictionary 관련 함수

- values() : value의 목록 확인 (≠ List)

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

```
>>> dict.values()  
dict_values([1, 2, 3, 4, 5])
```

```
>>> list = list(dict.values()) # value → List  
[1, 2, 3, 4, 5]
```

```
>>> for value in list:    # List로 만들어진 value 요소를 하나씩 추출  
    print(value) # 출력
```

```
1  
2  
3  
4  
5
```

■ 자료형 – Dictionary

● Dictionary 관련 함수

– items() : key, value 목록 확인 (≠ List)

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

>>> dict.items()
dict_items([('a', 1), ('b', 2), ('c', 3), ('d', 4), ('e', 5)])

>>> t_list = list(dict.items()) # (key, value) → List
[('a', 1), ('b', 2), ('c', 3), ('d', 4), ('e', 5)]

>>> t_list[0]
('a', 1)

>>> t_list[0][0]
'a'

>>> t_list[0][1]
1
```

■ 자료형 – Dictionary

● Dictionary 관련 함수

– clear() : 모든 데이터 삭제

```
>>> dict = { 'a': 1, 'b': 2, 'c': 3, 'd': 4, 'e': 5 }  
  
>>> dict.clear()  
{}
```

– get() : key를 이용하여 value 조회

```
>>> dict = { 'name':'ggoreb', 'age':20, 'hobby':['당구', '배드민턴'] }  
  
>>> result = dict.get('id')  
>>> print(result)  
None  
  
>>> result = dict['id']  
error
```

```
>>> dict['id']  
Traceback (most recent call last):  
  File "<pyshe11#147>", line 1, in <module>  
    dict['id']  
KeyError: 'id'
```

– in : Dictionary 요소 중 해당하는 key가 있는지 확인

```
>>> dict = { 'name':'ggoreb', 'age':20, 'hobby':['당구', '배드민턴'] }  
  
>>> 'name' in dict  
True  
  
>>> 'address' in dict  
False
```

■ 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 글자씩 분리되어 집합 요소로 생성

– set4 = set(['a', 'b', 'c']) ← List

– set5 = set((1, 2, 3)) ← Tuple

– set6 = set({'a' : 1, 'b' : 2}) ← Dictionary, 각 key만 집합 요소로 생성

■ 자료형 - Set

- 중복 데이터를 허용하지 않고, 순서가 없음
- 순서가 없으므로 특정 값만 선택하여 조회하거나 수정 불가 (추가/삭제는 가능)

```
>>> s = set(['a', 'b', 1, 2, '가', '나', 'a', 1, '가'])  
{1, 2, 'b', '가', '나', 'a'}
```

```
>>> s[0] # 조회 불가  
error
```

```
>>> s[0] = 10 # 특정 요소 수정 불가  
error
```

```
>>> s += {3} # 연산자를 통한 값 추가 불가  
error
```

```
>>> s.add(3)  
{1, 2, 3, 'b', '가', '나', 'a'}
```

- 중복 자료를 제거하기 위한 필터 역할로도 사용

```
>>> li = [1, 2, 3, 4, 1, 2, 3]
```

```
>>> s = set(li) # List → Set  
{1, 2, 3, 4}
```

```
>>> li = list(s) # Set → List  
[1, 2, 3, 4]
```

```
>>> tu = (1, 2, 3, 4, 1, 2, 3)
```

```
>>> s = set(tu) # Tuple → Set  
{1, 2, 3, 4}
```

```
>>> tu = tuple(s) # Set → Tuple  
(1, 2, 3, 4)
```

■ 자료형 - Set

● 교집합 (intersection)

```
>>> a = set([1, 2, 3, 4, 5])
>>> b = set([3, 4, 5, 6, 7])

>>> a & b
{3, 4, 5}

>>> a.intersection(b)
{3, 4, 5}
```

● 합집합 (union)

```
>>> a = set([1, 2, 3, 4, 5])
>>> b = set([3, 4, 5, 6, 7])

>>> a | b
{1, 2, 3, 4, 5, 6, 7}

>>> a.union(b)
{1, 2, 3, 4, 5, 6, 7}
```

● 차집합 (difference)

```
>>> a = set([1, 2, 3, 4, 5])
>>> b = set([3, 4, 5, 6, 7])

>>> a - b
{1, 2}

>>> a.difference(b)
{1, 2}

>>> b - a
{6, 7}

>>> b.difference(a)
{6, 7}
```

■ 자료형 - Set

● Set 관련 함수

- add() : 하나의 값 추가

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

>>> s.add(4)
{1, 2, 3, 4}
>>> s.add('a')
{1, 2, 3, 4, 'a'}

>>> s.add('가')
{1, 2, 3, 4, '가', 'a'}
```

- update() : 여러개의 값 추가

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

>>> s.update([4, 5]) # List
{1, 2, 3, 4, 5}

>>> s.update((6, 7, 8)) # Tuple
{1, 2, 3, 4, 5, 6, 7, 8}
```

- remove() : 지정 값 제거

```
>>> s = {'a', 'b', 'c', 'd', 'e'}

>>> s.remove('a')
{'d', 'b', 'c', 'e'}

>>> s.remove(0)
error
```

■ 자료형 - 논리 (Boolean)

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

● Bool 연산

```
>>> bool(1)
True
>>> bool(0)
False

>>> bool('')
False
>>> bool('a')
True

>>> bool([1, 2])
True
>>> bool([])
False
```

● 여러 자료형에서의 Bool

Type	Value	Bool
String	'Python'	True
	''	False
Number	1	True
	0	False
List	[1, 2]	True
	[]	False
Tuple	(1, 2)	True
	()	False
Dictionary, Set	{1, 2} or {'a':1}	True
	{}	False
None	None	False

■ 변수

- 변할 수 있는 값
- 객체를 가리키는 것
- 변수를 사용하는 이유

- 재사용 가능

- 값에 이름(label)을 부여하고 쉽게 사용 가능

ex) 구글 key : AlzaSyAhVaeWRjyP71Hdd2IQBJb_rHjOcgvUU3M

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

네이버 Client ID : n76OOSmkSve3Q5K2VsRy

- 반복적으로 등장하는 값을 쉽게 관리

■ 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 (Garbage collection)

```
>>> var1 = 'Python'
>>> del var1

>>> var2 = 12345
>>> del(var2)

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

■ 변수

● 각 자료형의 변수 사용

mutable

- List

```
>>> list1 = [1, 2, 3]

>>> list2 = list1
>>> list2[0] = 10 # list2 요소 수정

>>> list1
[10, 2, 3]
```

- Dictionary

```
>>> dict1 = {'a':1, 'b':2}

>>> dict2 = dict1
>>> dict2['c'] = 3 # dict2 요소 추가

>>> dict1
{'a': 1, 'b': 2, 'c': 3}
```

- Set

```
>>> set1 = {1, 2, 3}

>>> set2 = set1
>>> set2.add(4) # set2 요소 추가

>>> set1
{1, 2, 3, 4}
```

immutable

- Tuple

```
>>> tuple1 = (1, 2, 3)

>>> tuple2 = tuple1
>>> tuple2 += (4, ) #tuple2 요소 추가

>>> tuple1
(1, 2, 3)
```

- Number

```
>>> num1 = 10

>>> num2 = num1
>>> num2 = 20 # num2 값 수정

>>> num1
10
```

- String

```
>>> str1 = 'hello'

>>> str2 = str1
>>> str2 = str2 + ' bye' # str2 값 수정

>>> str1
'hello'
```

■ 변수

● 요소 복사

- [:]

```
>>> list1 = [1, 2, 3]
```

```
>>> list2 = list1
```

```
>>> list3 = list1[:]
```

```
>>> list1[0] = 100
```

```
>>> list2  
[100, 2, 3]
```

```
>>> list3  
[1, 2, 3]
```

```
>>> list1 is list2  
True
```

```
>>> list1 is list3  
False
```

- copy 모듈

```
>>> import copy
```

```
>>> list1 = [1, 2, 3]
```

```
>>> list2 = list1
```

```
>>> list3 = copy.copy(list1)
```

```
>>> list1[0] = 100
```

```
>>> list2  
[100, 2, 3]
```

```
>>> list3  
[1, 2, 3]
```

```
>>> list1 is list2  
True
```

```
>>> list1 is list3  
False
```