Python data structures

목차

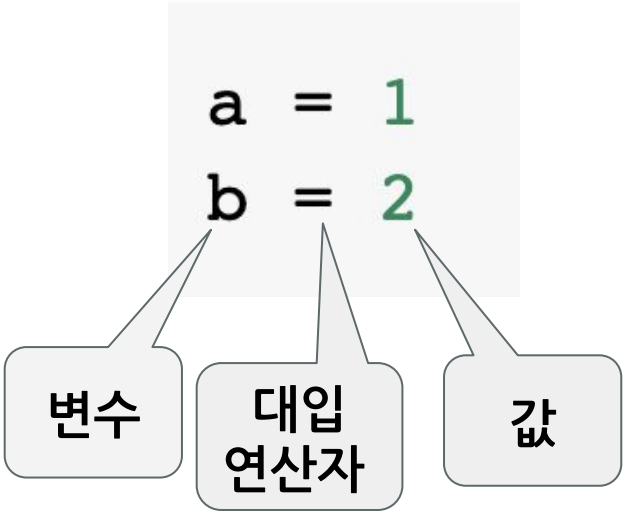
변수와 대입 연산자

변수 : 어떤 것의 이름(부록 참고)

대입 연산자 (=): 어떤 것에 변수라는 이름을 붙이는 것

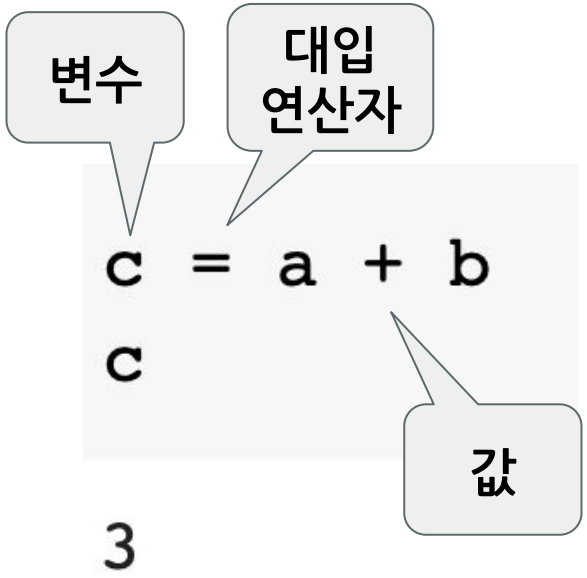
변수 = 값

변수와 대입 연산자



변수와 대입 연산자

```
a = 1
b = 2
```



Data Structures

숫자

정수

```
10
```

```
10
```

```
type(10)
```

```
int
```

실수

```
10.0
```

```
10.0
```

```
type(10.0)
```

```
float
```

숫자 형변환

float( )

정수를 실수로

```
float(10)
```

```
10.0
```

int( )

실수를 정수로

```
int(10.0)
```

```
10
```

숫자 연산자

덧셈 +

```
3 + 5
```

8

```
3.0 + 5
```

8.0

뺄셈 -

```
3 - 5
```

-2

```
3 - 5.0
```

-2.0

곱셈 *

```
3*5
```

15

```
3.0*5
```

15.0

나눗셈 /

```
10/3
```

3.3333333333333335

```
9/3
```

3.0

```
9.0/3
```

3.0

숫자 연산자

정수 나누기 //

나누고 정수
부분만 남기는
연산자

```
10//3
```

```
3
```

```
10.0//3
```

```
3.0
```

나머지 %

나머지 값을
구하는 연산자

```
10%3
```

```
1
```

```
10%3.0
```

```
1.0
```

숫자 연산자

제곱 **

```
10**3
```

```
1000
```

```
10.0**3
```

```
1000.0
```

숫자 연산자

연산자 우선순위 (부록 참고)

```
  ②   ①
1 + 3 * 2
```

7

```
  ①     ②
(1 + 3) * 2
```

8

숫자 연산자

복합 대입 연산자

기본 연산자(+, -, *, /, //, %, **) 와 대입 연산자(=)를 함께 사용하는 연산자

기본 연산자 적용 후 대입 연산자를 적용

```
a = 10
a = a + 5
a
```

```
15
```

```
a = 10
a += 5
a
```

```
15
```

숫자 연산자

복합 대입 연산자

```
a = 10
a -= 5
a
```

```
5
```

```
a = 10
a *= 5
a
```

```
50
```

```
a = 10
a /= 5
a
```

```
2.0
```

문자열

문자열 : 문자, 단어 등으로 구성된 문자들의 집합

- 불변(immutable): 문자열 자체를 변경할 수 없음.
- 시퀀스(sequence): 왼쪽부터 순서가 있음.

문자열

문자열 생성

큰따옴표(“”) 또는 작은따옴표(‘’)로 생성

```
print('python')
```

```
python
```

```
print("python")
```

```
python
```

```
type('python')
```

```
str
```

```
type("python")
```

```
str
```

문자열

문자열 생성

문자열 안에 큰따옴표 넣고 싶을 때 작은따옴표 사용

문자열 안에 작은따옴표 넣고 싶을 때 큰따옴표 사용

```
print('"Readability counts."')
```

```
"Readability counts."
```

```
print("'Readability counts.'")
```

```
'Readability counts.'
```

문자열

문자열 생성

문자열 안에 따옴표 넣고 싶을 때 역슬래시 역슬래시(\) 기호 사용

```
string = "The Zen of Python, by Tim Peters. \"Beautiful is better than ugly.\""
string
```

```
'The Zen of Python, by Tim Peters. "Beautiful is better than ugly."'
```

문자열

문자열 생성

여러 줄 작성하고 싶을 때

“‘문장1

문장2”’

```
string = '''Simple is better than complex.
Complex is better than complicated.'''
print(string)
```

```
Simple is better than complex.
Complex is better than complicated.
```

```
string = """Simple is better than complex.
Complex is better than complicated."""
print(string)
```

```
Simple is better than complex.
Complex is better than complicated.
```

문자열

문자열 생성

여러 줄 작성하고 싶을 때

\n 이용

```
string = "Simple is better than complex.\nComplex is better than complicated."
print(string)
```

```
Simple is better than complex.
Complex is better than complicated.
```

문자열 연산자

문자열 연결 연산자 +

두 문자열을 연결하여 새로운 문자열 만들어 내는 것

문자열 + 문자열

문자열 연산자

문자열 연결 연산자 +

여러 문자열을 연결하여 새로운 문자열 만들어 내는 것

```
a = '문자열을'
b = '더해보겠습니다.'
print(a + ' ' + b)
```

```
문자열을 더해보겠습니다.
```

문자열 연산자

문자열 반복 연산자 *

물자열을 숫자와 * 연산자로 연결하면 문자열이 반복됨

문자열 * 숫자

문자열 연산자

문자열 반복 연산자 *

물자열을 숫자와 * 연산자로 연결하면 문자열이 반복됨

```
a = '안녕'
a * 3
```

```
'안녕안녕안녕'
```

문자열 연산자

문자열1 in 문자열2 문자열2에 문자열1이 있으면 True, 없으면 False

문자열1 not in 문자열2 문자열2에 문자열1이 없으면 True, 있으면 False

```
str1 = '커피'
str2 = '메뉴: 커피, 녹차, 과일주스'
```

```
str1 in str2
```

```
True
```

```
str1 not in str2
```

```
False
```

```
str3 = '딸기주스'
str3 in str2
```

```
False
```

```
str3 not in str2
```

```
True
```

문자열

문자 선택(인덱싱)

문자열 내부의 문자 하나를 선택하는 연산자

파이썬은 숫자를 0부터 세는 제로 인덱스 유형임

문자열[인덱스]

인덱스는 0부터 시작

문자열

문자 선택(인덱싱)

```
a = '좋은 하루 보내세요!'
a
```

```
'좋은 하루 보내세요!'
```

좋	은		하	루		보	내	세	요	!
a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[9]	a[10]

문자열

문자 선택(인덱싱)

```
a = '좋은 하루 보내세요!'
a
```

```
'좋은 하루 보내세요!'
```

좋	은		하	루		보	내	세	요	!
a[-11]	a[-10]	a[-9]	a[-8]	a[-7]	a[-6]	a[-5]	a[-4]	a[-3]	a[-2]	a[-1]

문자열

문자열 슬라이싱

문자열의 특정 범위를 선택하는 것

```
a = '오늘은 보람찬 하루였다.'
```

```
a[4:7]
```

```
'보람찬'
```

```
a[8:]
```

```
'하루였다.'
```

문자열[i : j]

문자열의 인덱스 i 부터 j-1까지 추출

i 또는 j가 문자열의 시작 또는 끝인 경우 생략 가능

문자열

문자열 슬라이싱

문자열의 특정 범위를 선택하는 것

```
b = '가나다라마바사아자차카타파하'
```

```
b[0:8:2]
```

```
'가다마사'
```

문자열[i : j : step]

문자열의 인덱스 i 부터 j-1까지 step씩 건너 뛰면서 추출
i 또는 j가 문자열의 시작 또는 끝인 경우 생략 가능
간격이 1인 경우 생략 가능

```
b[-5:-1:2]
```

```
'차타'
```

문자열

문자열 길이 구하기: len(문자열)

```
a = '좋은 하루 보내세요!'
len(a)
```

```
11
```

```
len('abc')
```

```
3
```

문자열 메서드

문자열.메서드명()

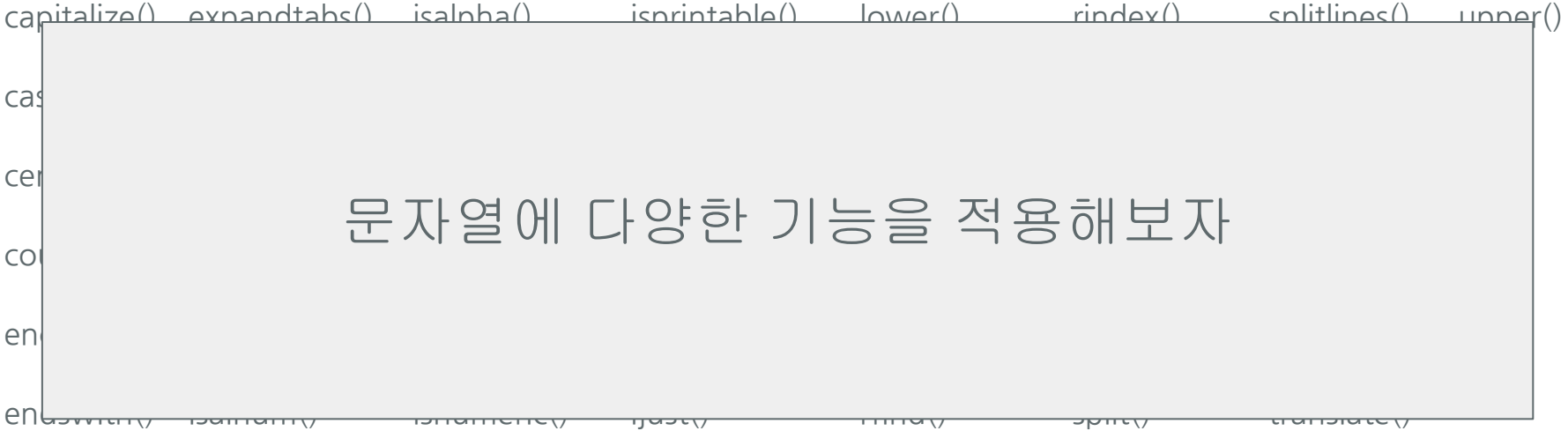
파이썬 문자열 메서드 참고 링크

capitalize()	expandtabs()	isalpha()	isprintable()	**lower()**	rindex()	splitlines()	**upper()**
casefold()	**find()**	isdecimal()	isspace()	**lstrip()**	rjust()	startswith()	zfill()
center()	**format()**	isdigit()	istitle()	maketrans()	rpartition()	**strip()**	
count()	format_map()	isidentifier()	isupper()	partition()	rsplit()	swapcase()	
encode()	index()	islower()	**join()**	**replace()**	**rstrip()**	title()	
endswith()	isalnum()	isnumeric()	ljust()	**rfind()**	**split()**	translate()	

문자열 메서드

문자열.메서드명()

파이썬 문자열 메서드 참고 링크



문자열 메서드

str.format()

```
'{}년 {}월 입니다.'.format(2020, 7)
```

```
'2020년 7월 입니다.'
```

```
'지난 달 평균 기온은 {:.1f}도 입니다.'.format(23.50)
```

```
'지난 달 평균 기온은 23.5도 입니다.'
```

```
'{} 주소는 {} {} {} {}입니다.'.format('국립중앙박물관','서울시', '용산구', '서빙고로', 137)
```

```
'국립중앙박물관 주소는 서울시 용산구 서빙고로 137입니다.'
```

문자열 메서드

str.lower(), str.upper()

str.lower(): 문자열의 알파벳을 소문자로 바꾸기

str.upper(): 문자열의 알파벳을 대문자로 바꾸기

```
'Hello'.lower()
```

```
'hello'
```

```
'Hello'.upper()
```

```
'HELLO'
```

문자열 메서드

공백 지우기

str.strip() 문자열의 양쪽에 있는 한 칸 이상의 연속된 공백을 모두 제거

str.rstrip() 문자열의 가장 오른쪽에 있는 한 칸 이상의 연속된 공백을 모두 제거

str.lstrip() 문자열의 가장 왼쪽에 있는 한 칸 이상의 연속된 공백을 모두 제거

문자열 메서드

공백 지우기

str.strip()

str.rstrip()

str.lstrip()

```
'   Readability counts.   '
```

```
'   Readability counts.   '
```

```
'   Readability counts.   '.strip()
```

```
'Readability counts.'
```

```
'   Readability counts.   '.rstrip()
```

```
'   Readability counts.'
```

```
'   Readability counts.   '.lstrip()
```

```
'Readability counts.   '
```

문자열 메서드

문자열 위치 찾기

str.find(sub_str) 문자열 왼쪽부터 찾아서 처음 등장하는 위치 반환.

str.rfind(sub_str) 문자열 오른쪽부터 찾아서 처음 등장하는 위치 반환.

찾는 문자열이 없는 경우 -1 반환

문자열 메서드

문자열 위치 찾기

str.find(sub_str), str.rfind(sub_str)

```
'안녕하세요? 네 안녕하세요.'.find('안녕')
```

```
0
```

```
'안녕하세요? 네 안녕하세요.'.rfind('안녕')
```

```
9
```

```
'안녕하세요? 네 안녕하세요.'.find('네네')
```

```
-1
```

```
'안녕하세요? 네 안녕하세요.'.find('네네')
```

```
-1
```

문자열 메서드

str.split(구분자) 구분자를 기준으로 하나의 문자열을 작은 문자열들의 리스트로 나눔. 구분자를 생략할 경우 공백 문자를 구분자로 사용.

```
string = '''남산 위에 저 소나무, 철갑을 두른 듯
바람 서리 불변함은 우리 기상일세.'''
```

```
string.split()
```

```
['남산', '위에', '저', '소나무,', '철갑을', '두른', '듯', '바람', '서리', '불변함은', '우리', '기상일세.']
```

```
string.split(',')
```

```
['남산 위에 저 소나무', ' 철갑을 두른 듯\n바람 서리 불변함은 우리 기상일세.']
```

문자열 메서드

str.join(sub_str) 문자열 sub_str의 각각의 문자 사이에 문자열 str을 삽입

```
' '.join('abcde')
```

```
'a b c d e'
```

```
'**'.join('abcde')
```

```
'a**b**c**d**e'
```

문자열 메서드

str.repalce(바꿀 문자, 대체할 문자, 횟수)

문자열 안의 특정한 값을 다른 값으로 횟수만큼 대체해줌.

횟수 생략시 1번.

```
string = '나는 자주 까먹어서 자주 복습합니다.'
```

```
string.replace('자주', '종종')
```

```
'나는 종종 까먹어서 종종 복습합니다.'
```

```
string.replace('자주', '종종', 2)
```

```
'나는 종종 까먹어서 종종 복습합니다.'
```

불

불(bool)

- 참(True)과 거짓(False)를 나타내는 자료형
- True, False 두가지 값만 있음

```
type(True)
```

```
bool
```

```
type(False)
```

```
bool
```

불

조건문의 반환값으로 사용

조건문 조건문이란 참과 거짓을 판단하는 문장

```
10 < 20
```

```
True
```

```
10 == 20
```

```
False
```

```
10 >= 20
```

```
False
```

```
10 != 20
```

```
True
```

비교 연산자	설명
x < y	x가 y보다 작다
x > y	x가 y보다 크다
x == y	x와 y가 같다
x != y	x와 y가 같지 않다
x >= y	x가 y보다 작거나 같다
x <= y	x가 y보다 작거나 같다

불

불 연산

논리 연산자

```
a = True
b = True
print(a and b)
print(a or b)
```

```
True
True
```

```
a = True
b = False
print(a and b)
print(a or b)
```

```
False
True
```

```
a = False
b = False
print(a and b)
print(a or b)
```

```
False
False
```

논리 연산자	설명
x and y	x, y 모두 참이면 참 그 외에는 거짓
x or y	x, y 둘 중 하나라도 참이면 참 그 외에는 거짓
not x	불을 반대로 전환 x가 거짓이면 not x는 참 x가 참이면 not x는 거짓

불

불 연산

논리 연산자

```
a = True
not a
```

```
False
```

```
b = False
not b
```

```
True
```

논리 연산자	설명
x and y	x, y 모두 참이면 참 그 외에는 거짓
x or y	x, y 둘 중 하나라도 참이면 참 그 외에는 거짓
not x	불을 반대로 전환 x가 거짓이면 not x는 참 x가 참이면 not x는 거짓

리스트

리스트(list)

- **모든 것의 시퀀스** 요소가 무엇이든 가능. 순서가 있음.
- **변경 가능(mutable)** 요소 삽입, 수정, 삭제 가능

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

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

```
[print, 'print 함수']
```

```
[<function print>, 'print 함수']
```

리스트

리스트 만들기 []

```
empty_list = []
empty_list
```

```
[]
```

```
type(empty_list)
```

```
list
```

```
weekdays = ['월', '화', '수', '목', '금']
weekdays
```

```
['월', '화', '수', '목', '금']
```

```
type(weekdays)
```

```
list
```

```
list_a = [10, 20, 30, ['a', 'b'], 40]
list_a
```

```
[10, 20, 30, ['a', 'b'], 40]
```

```
type(list_a)
```

```
list
```

리스트

리스트 인덱싱

리스트[인덱스]

인덱스는 0부터 시작
마이너스 인덱싱도 가능

```
city = ['서울', '인천', '부산', '대구', '대전', '광주', '울산']
```

```
city[0]
```

```
'서울'
```

```
city[-2]
```

```
'광주'
```

리스트

리스트 인덱싱

리스트[i][j]

리스트의 가장 바깥쪽 괄호부터 인덱싱

```
city = [['서울', '인천', '부산', '대구', '대전', '광주', '울산', '세종'],
        ['경기도', '강원도', '충청도', '전라도', '경상도', '제주']]
```

```
city[0]
```

```
['서울', '인천', '부산', '대구', '대전', '광주', '울산', '세종']
```

```
city[0][0]
```

```
'서울'
```

리스트

리스트 슬라이싱

리스트[i : j]

리스트[i : j : step]

리스트의 인덱스 i 부터 j-1까지 step 간격으로 추출
i 또는 j가 리스트의 시작 또는 끝인 경우 생략 가능
step이 1인 경우 생략 가능

```
food = ['한식', '양식', '중식', '일식']
```

```
food[0:2]
```

```
['한식', '양식']
```

```
food[::2]
```

```
['한식', '중식']
```

리스트

리스트 요소 변경하기

```
food = ['한식', '양식', '중식', '일식']
```

```
food[-1] = '분식'
food
```

```
['한식', '양식', '중식', '분식']
```

```
food[0:2] = ['Korean', 'western']
food
```

```
['Korean', 'western', '중식', '분식']
```

리스트

len(list) 리스트 길이 구하기

```
food = ['한식', '양식', '중식', '일식']
len(food)
```

```
4
```

```
empty_list = []
len(empty_list)
```

```
0
```

리스트

“.join(list) 리스트를 문자열로 반환

```
j = ['Joins', 'the', 'elements', 'of', 'an', 'iterable',
    'to', 'the', 'end', 'of', 'the', 'string']
' '.join(j)
```

```
'Joins the elements of an iterable to the end of the string'
```

리스트 연산자

연결 연산자 +

```
list_a = [1, 2, 3]
list_b = [10, 20]
```

```
list_a + list_b
```

```
[1, 2, 3, 10, 20]
```

```
list_a + list_b + list_a
```

```
[1, 2, 3, 10, 20, 1, 2, 3]
```

반복 연산자 *

```
x = [10, 20]
x*3
```

```
[10, 20, 10, 20, 10, 20]
```

리스트 연산자

요소 in 리스트 리스트에 요소가 있으면 True, 없으면 False

요소 not in 리스트 리스트에 요소가 없으면 True, 있으면 False

```
stocks = ['GOOGL', 'MSFT', 'AAPL', 'AMZN', 'TSLA']
```

```
'EBAY' in stocks
```

```
False
```

```
'EBAY' not in stocks
```

```
True
```

리스트 메서드

list.append(요소)

리스트의 끝에 항목 추가하기

```
x = [1, 2, 3]
x.append(4)
x
```

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

list1.extend(list2)

list1에 list2를 병합.

```
x = [1, 2, 3]
x.extend([4, 5, 6])
x
```

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

리스트 메서드

list.insert(i, x) list의 인덱스 i 위치에 x를 삽입

```
x = [10, 20, 30, 40, 50]
x.insert(3, 1000)
x
```

```
[10, 20, 30, 1000, 40, 50]
```

```
x.insert(-2, 2000)
x
```

```
[10, 20, 30, 1000, 2000, 40, 50]
```

리스트 메서드

리스트 요소 제거하기

- **인덱스로 제거하기**

  del list(i) 리스트의 인덱스 i 요소를 제거

  list.pop(i) 리스트의 인덱스 i 요소를 반환하고 제거함

- **값으로 제거하기**

  list.remove(v) 리스트에서 첫번째로 나오는 v를 삭제

리스트 메서드

리스트 요소 제거하기

list.pop(i) 리스트의 인덱스 i 요소를 반환하고 제거함

```
stocks = ['GOOGL', 'MSFT', 'AAPL', 'AMZN', 'TSLA']
```

```
stocks.pop(1)
```

```
'MSFT'
```

```
stocks
```

```
['GOOGL', 'AAPL', 'AMZN', 'TSLA']
```

리스트 메서드

리스트 요소 제거하기

del list[i] 리스트의 인덱스 i 요소를 제거함

```
stocks = ['GOOGL', 'MSFT', 'AAPL', 'AMZN', 'TSLA']
del stocks[2]
```

```
stocks
```

```
['GOOGL', 'MSFT', 'AMZN', 'TSLA']
```

```
del stocks[-2]
```

```
stocks
```

```
['GOOGL', 'MSFT', 'TSLA']
```

리스트 메서드

리스트 요소 제거하기

list.remove(v) 리스트에서 첫번째로 나오는 v를 삭제

```
list_a = ['a', 'b', 'c', 'd', 'a', 'b', 'c', 'd']
list_a.remove('a')
list_a
```

```
['b', 'c', 'd', 'a', 'b', 'c', 'd']
```

리스트 메서드

list.count(x) 리스트 안에 x가 몇 개 있는지

```
list_a = ['김', '이', '박', '나', '박', '한', '김', '이', '김', '최']
list_a.count('박')
```

```
2
```

```
list_a.count('배')
```

```
0
```

리스트 메서드

list.sort() 리스트 자체를 내부적으로 정렬

sorted(list) 리스트의 정렬된 복사본을 반환

```
a = [10, 20, 30, 10, 20, -10, 0, -10]
a.sort()
a
```

```
[-10, -10, 0, 10, 10, 20, 20, 30]
```

```
a = [10, 20, 30, 10, 20, -10, 0, -10]
sorted(a)
```

```
[-10, -10, 0, 10, 10, 20, 20, 30]
```

튜플

튜플(tuple)

- **모든 것의 시퀀스** 요소가 무엇이든 가능. 순서가 있음.
- **변경 불가능(immutable)** 튜플을 정의한 후에는 요소 삽입, 수정, 삭제 불가능

튜플

리스트 vs 튜플

가장 큰 차이는 값을 변화시킬 수 있는가 여부

프로그램이 실행되는 동안 값이 변하지 않기 바라면 튜플 사용

값을 변화시킬 경우 리스트 사용

튜플

튜플 만들기 (), tuple()

```
empty_tuple = ()
empty_tuple
```

```
()
```

```
type(empty_tuple)
```

```
tuple
```

```
t1 = 'a',
t2 = ('a',)
print(t1, t2)
print(type(t1), type(t2))
```

```
('a',) ('a',)
<class 'tuple'> <class 'tuple'>
```

```
t1 = 'a', 'b', 'c'
t2 = ('a', 'b', 'c')
print(t1, t2)
print(type(t1), type(t2))
```

```
('a', 'b', 'c') ('a', 'b', 'c')
<class 'tuple'> <class 'tuple'>
```

요소가 하나일 때 마지막에 꼭 콤마 (,) 붙이기

튜플

튜플 인덱싱

문자열, 리스트와 마찬가지로 인덱싱이 가능

튜플[인덱스]

인덱스는 0부터 시작

마이너스 인덱싱도 가능

```
prime_5 = (2, 3, 5, 7, 11)
```

```
2
```

```
prime_5[0]
```

```
2
```

```
prime_5[-2]
```

```
7
```

```
tp = ((100, 200), 300, (400, (500, 600)))
tp[2][1][0]
```

```
500
```

튜플

튜플 슬라이싱

문자열, 리스트와 마찬가지로 슬라이싱이 가능

튜플[i : j]

튜플[i : j : step]

튜플의 인덱스 i 부터 j-1까지 step 간격으로 추출
i 또는 j가 튜플의 시작 또는 끝인 경우 생략 가능
step이 1인 경우 생략 가능

```
t = ('a', 'b', 'c', 'd', 'e')
t[1:3]
```

```
('b', 'c')
```

```
t[:3]
```

```
('a', 'b', 'c')
```

```
t[3:]
```

```
('d', 'e')
```

```
t[0:4:2]
```

```
('a', 'c')
```

튜플

튜플의 요소를

삭제하거나

변경하려고 할 때

```
t = ('a', 'b', 'c', 'd', 'e')
```

```
del t[0]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-326-542f9bb8f15e> in <module>()
----> 1 del t[0]

TypeError: 'tuple' object doesn't support item deletion
```

```
t[0] = 'x'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-327-715bd1e5d49f> in <module>()
----> 1 t[0] = 'x'

TypeError: 'tuple' object does not support item assignment
```

튜플

튜플 연산자

+

*

```
t1 = ('a', 'b', 'c', 'd', 'e')
t2 = ('f', 'g')
```

```
t1 + t2
t1
```

```
('a', 'b', 'c', 'd', 'e', 'f', 'g')
```

```
t2*3
```

```
('f', 'g', 'f', 'g', 'f', 'g')
```

튜플

in, not in

```
t = ('a', 'b', 'c', 'd', 'e')
```

```
'a' in t
```

```
True
```

```
'b' not in t
```

```
False
```

```
'k' in t
```

```
False
```

```
'k' not in t
```

```
True
```

len(튜플)

```
t = ('a', 'b', 'c', 'd', 'e')
len(t)
```

```
5
```

```
t = ((1, 2, 3, 4), (5, 6, 7))
len(t)
```

```
2
```

딕셔너리

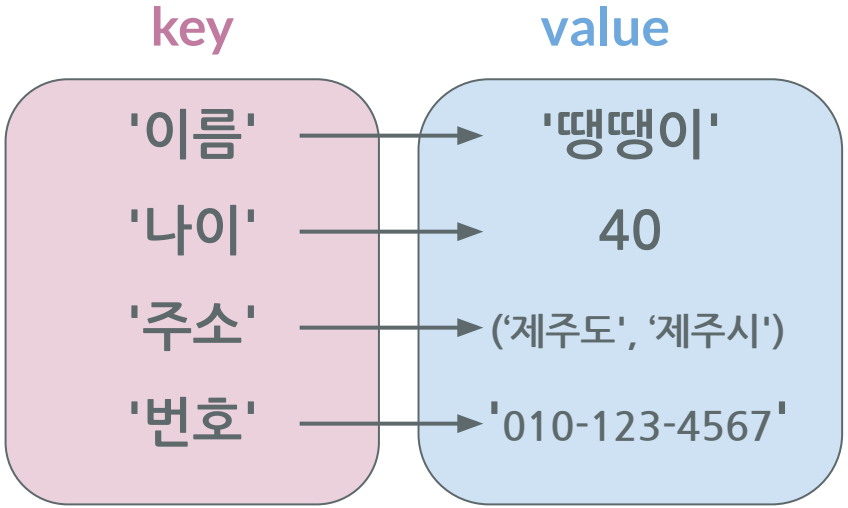
딕셔너리(Dictionary)

- **key-value** 시퀀셜한 리스트나 튜플는 인덱스를 통해 값을 찾을 수 있지만 딕셔너리는 key를 통해 value를 얻음
- **변경 가능** 키-값 요소를 추가, 삭제, 수정 할 수 있음

딕셔너리

딕셔너리 예시

```
dic = {'이름':'땡땡이',
       '나이':40,
       '주소':('제주도', '제주시'),
       '번호':'010-123-4567'}
```



딕셔너리

딕셔너리 만들기 { }

중괄호 안에 콤마로 구분된 **키:밸류** 쌍을 지정

키: 중복 불가능. 불변형인 데이터 타입만 가능.

```
empty_dict = {}
empty_dict
```

```
{}
```

```
dic1 = {'사과':[1, 2], '딸기':[3, 4], '배':[5, 6]}
dic2 = {True:100, False:-100}
dic3 = {(0,0):100, (0, 1):200, (1, 0):100 ,(1, 1):0}
```

딕셔너리

항목 얻기

인덱스가 아니라 키를 사용해서 밸류 얻음

키가 존재하지 않는 경우 에러

딕셔너리[키]

```
dic = {'사과':40000, '딸기':12000, '배':50000}
```

```
dic['사과']
```

```
40000
```

```
dic['딸기']
```

```
12000
```

```
dic['배']
```

```
50000
```

딕셔너리

항목 얻기

dict.get(key)

키가 존재하지 않는 경우 None

```
dic = {'사과':40000, '딸기':12000, '배':50000}
```

```
dic.get('사과')
```

```
40000
```

```
dic.get('배')
```

```
#키가 존재하지 않는 경우
dic.get('포도')
```

```
#키가 존재하지 않는 경우
dic['포도']
```

```
---------------------------------------------------------
KeyError                                  Traceba
<ipython-input-31-5a9fdea98633> in <module>()
      1 dic = {'사과':40000, '딸기':12000, '배':500(
----> 2 dic['포도']

KeyError: '포도'
```

딕셔너리

항목 변경/추가하기 딕셔너리[키] = 밸류

```
webtoon = {'유미의 세포들': '이동거', '아홉수 우리들': '수박양', '위대한 방옥숙': ('매미','희세')}
```

```
# 기존에 있는 키
webtoon['유미의 세포들'] = '이동건'
print(webtoon)
```

```
{'유미의 세포들': '이동건', '아홉수 우리들': '수박양', '위대한 방옥숙': ('매미', '희세')}
```

```
# 기존에 없던 키
webtoon['고래별'] = '나윤희'
print(webtoon)
```

```
{'유미의 세포들': '이동건', '아홉수 우리들': '수박양', '위대한 방옥숙': ('매미', '희세'), '고래별': '나윤희'}
```

딕셔너리

딕셔너리 결합하기

dict.update(dict2)

```
webtoon
```

```
{'고래별': '나윤희',
 '라일락 200%': '이나영',
 '아홉수 우리들': '수박양',
 '위대한 방옥숙': ('매미', '희세'),
 '유미의 세포들': '이동건'}
```

```
webtoon.update({'1인용 기분': '윤파랑',
                'Ho!': '억수씨',
                '노곤하개': '홍끼'})
webtoon
```

```
{'1인용 기분': '윤파랑',
 'Ho!': '억수씨',
 '고래별': '나윤희',
 '노곤하개': '홍끼',
 '라일락 200%': '이나영',
 '아홉수 우리들': '수박양',
 '위대한 방옥숙': ('매미', '희세'),
 '유미의 세포들': '이동건'}
```

딕셔너리

항목 삭제하기/전체 삭제하기

del dict[key]

dict.clear()

```
webtoon
```

```
{'1인용 기분': '윤파랑',
 'Ho!': '억수씨',
 '고래별': '나윤희',
 '노곤하개': '홍끼',
 '라일락 200%': '이나영',
 '아홉수 우리들': '수박양',
 '위대한 방옥숙': ('매미', '희세'),
 '유미의 세포들': '이동건'}
```

```
del webtoon['위대한 방옥숙']
webtoon
```

```
{'1인용 기분': '윤파랑',
 'Ho!': '억수씨',
 '고래별': '나윤희',
 '노곤하개': '홍끼',
 '라일락 200%': '이나영',
 '아홉수 우리들': '수박양',
 '유미의 세포들': '이동건'}
```

```
webtoon.clear()
webtoon
```

```
{}
```

딕셔너리

key in dict 키가 딕셔너리에 있는지(True) 없는지(False)

```
webtoon = {'유미의 세포들': '이동건', '아홉수 우리들': '수박양', '고래별': '나윤희'}
```

```
'유미의 세포들' in webtoon
```

```
True
```

```
'1인용 기분' in webtoon
```

```
False
```

```
'이동건' in webtoon
```

```
False
```

딕셔너리

dict.items()
모든 키-밸류 얻기

dict.keys()
모든 키 얻기

dict.values()
모든 밸류 얻기

```
webtoon = {'유미의 세포들': '이동건', '아홉수 우리들': '수박양', '고래별': '나윤희'}
```

```
webtoon.items()
```

```
dict_items([('유미의 세포들', '이동건'), ('아홉수 우리들', '수박양'), ('고래별', '나윤희')])
```

```
webtoon.keys()
```

```
dict_keys(['유미의 세포들', '아홉수 우리들', '고래별'])
```

```
webtoon.values()
```

```
dict_values(['이동건', '수박양', '나윤희'])
```

셋

셋(set) 딕셔너리와 거의 동일하지만 셋은 값을 가지지 않음

- **중복을 허용하지 않음**
- **변경 가능**
- **순서가 없음** 순서가 없어서 인덱싱으로 값을 얻을 수 없음. 저장된 값을 인덱싱으로 접근하고 싶은 경우 튜플이나 리스트로 변환 후 접근

셋

셋 만들기 { }

중괄호 안에 콤마로 구분된 하나 이상의 요소 넣기.

빈 셋은 set()으로 생성

```
empty_set = set()
empty_set
```

```
set()
```

```
type(empty_set)
```

```
set
```

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

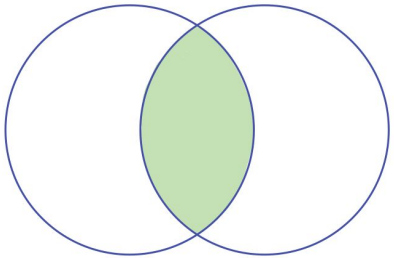
```
{1, 2, 3}
```

```
set2 = {'a', 1, 'b', 'c', 2}
set2
```

```
{1, 2, 'a', 'b', 'c'}
```

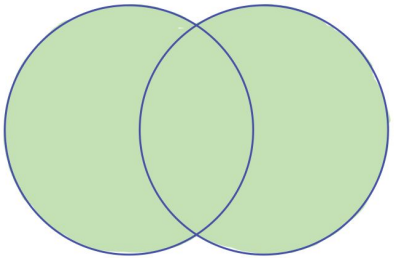
셋

교집합



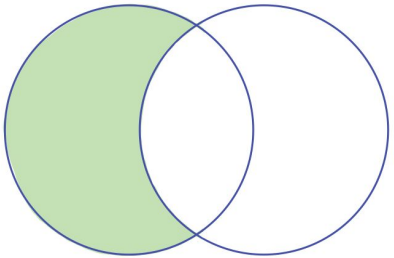
s1 & s2
s1.intersection(s2)

합집합



s1 | s2
s1.union(s2)

차집합



s1 - s2
zs1.difference(s2)

셋

값 추가하기

set.add() 값 1개 추가

set.update() 값 여러개 추가

```
s = {'001', '002'}
```

```
s.add('003')
s
```

```
{'001', '002', '003'}
```

```
s.update(['004', '005', '006'])
s
```

```
{'001', '002', '003', '004', '005', '006'}
```

셋

값 제거하기

set.remove()

키가 존재하지 않는 경우

에러 발생

```
s.remove('006')
```

```
s.remove('007')
```

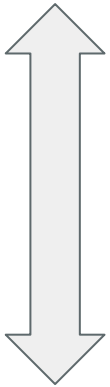
```
---------------------------------------
KeyError
<ipython-input-26-48e4b2498c0f> in <mo
----> 1 s.remove('007')

KeyError: '007'
```

Appendix

연산자 우선 순위

우선 순위 낮음



우선 순위 높음

Operator	Description
`:=`	Assignment expression
`lambda`	Lambda expression
`if - else`	Conditional expression
`or`	Boolean OR
`and`	Boolean AND
`not x`	Boolean NOT
`in`, `not in`, `is`, `is not`, `<`, `<=`, `>`, `>=`, `!=`, `==`	Comparisons, including membership tests and identity tests
`|`	Bitwise OR
`^`	Bitwise XOR
`&`	Bitwise AND
`<<`, `>>`	Shifts
`+`, `-`	Addition and subtraction
`*`, `@`, `/`, `//`, `%`	Multiplication, matrix multiplication, division, floor division, remainder [5]
`+x`, `-x`, `~x`	Positive, negative, bitwise NOT
`**`	Exponentiation [6]
`await x`	Await expression
`x[index]`, `x[index:index]`, `x(arguments...)`, `x.attribute`	Subscription, slicing, call, attribute reference
`(expressions...)`, `[expressions...]`, `{key: value...}`, `{expressions...}`	Binding or parenthesized expression, list display, dictionary display, set display

출처 - 파이썬 공식 문서

변수명 짓기

- 키워드를 사용하면 안됨
- 대소문자 구분됨
- 특수 문자는 언더바(_)만 가능
- 숫자로 시작하면 안됨
- 공백을 포함할 수 없음

변수명 짓기 - 키워드

키워드 검사하는 방법

```
import keyword
```

```
keyword.iskeyword('if')
```

```
True
```

```
keyword.iskeyword('a')
```

```
False
```

```
False      await      else       import     pass
None       break      except     in         raise
True       class      finally    is         return
and        continue   for        lambda     try
as         def        from       nonlocal   while
assert     del        global     not        with
async      elif       if         or         yield
```

출처 - 파이썬 공식 문서

파이썬 데이터 타입 분류

가변형: 리스트, 딕셔너리, 집합

불변형: 불, 숫자, 문자열, 튜플

자료형의 참과 거짓

bool() 참과 거짓을 식별

```
print(bool(None))
```

```
False
```

```
print(bool(0))
print(bool(1))
```

```
False
True
```

```
print(bool(0.0))
print(bool(1.0))
```

```
False
True
```

```
print(bool(''))
print(bool('abc'))
```

```
False
True
```

False로 간주되는 요소	False
Null	None
정수 0	0
부동소수점수 0	0.0
빈 문자열	''
빈 리스트	[]
빈 튜플	()
빈 딕셔너리	{}
빈 셋	set()

자료형의 참과 거짓

bool() 참과 거짓을 식별

```
print(bool([]))
print(bool(['a','b','c']))
```

```
False
True
```

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

```
False
True
```

```
print(bool(()))
print(bool(('a','b','c')))
```

```
False
True
```

```
print(bool(set()))
print(bool({'a','b','c'}))
```

```
False
True
```

False로 간주되는 요소	False
Null	None
정수 0	0
부동소수점수 0	0.0
빈 문자열	''
빈 리스트	[]
빈 튜플	()
빈 딕셔너리	{}
빈 셋	set()

Reference

파이썬 공식 문서
러닝 파이썬
혼자 공부하는 파이썬
생활코딩 파이썬
처음 시작하는 파이썬
점프 투 파이썬