# 파이썬 자료형 및 기본 자료형

- 파이썬 자료형
- 불린형(Boolean Type)
- 숫자형
  - 정수(Integer Type)
  - 실수(Floating-point Type)
- 문자열형(String Type)

# 파이썬 자료형

자료형(data type)은 다음과 같은 정보를 담고 있다.

- 해당 자료형이 가질 수 있는 가능한 값의 종류
- 해당 자료형이 참조하는 데이터(객체)가 컴퓨터 메모리에 저장되는 방식
- 해당 자료형으로 실행할 수 있는 수학적, 관계적, 논리적 명령의 종류

어떤 명령어들은 자료형에 따라 다른 결괏값을 돌려주기도 한다.

**+ 연산자의 경우**

- 정수끼리 더하면 정수의 합을 반환하지만
- 문자열끼리 더하면 문자열 결합을 반환한다.

In [ ]:
```
1 + 2
```

In [ ]:
```
'1' + '2'
```

*** 연산자의 경우**

- 정수끼리 곱하면 정수의 곱한 결과 값을 반환하지만
- 문자열을 곱하면 곱한만큼 반복한 문자열 결과 값을 반환한다

In [ ]:
```
2 * 5
```

In [ ]:
```
'2' * 5
```

자료형은 다음 중 하나의 속성을 가진다.

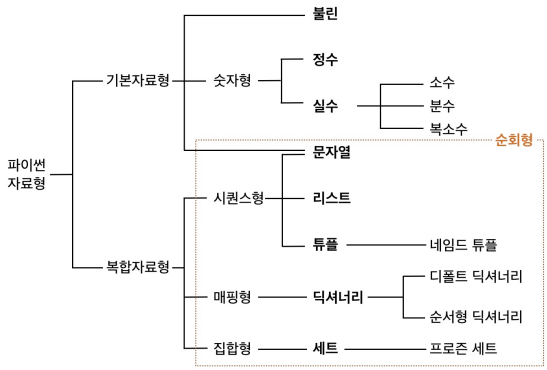
**불변성**(immutable) : 생성한 후 내용의 변경이 불가능하다.

- 예) 불린형, 정수형, 실수형, 문자열형, 튜플형 등

**가변성**(mutable) : 생성한 후 내용의 변경이 가능하다.

- 예) 리스트형, 세트형, 딕셔너리형 등

# 자료형의 종류




# 불린형(Boolean Type)

**불변자료형**(immutable)이다.

'거짓'(**False**)과 '참'(**True**) 두 가지 값 중 한 개만 취할 수 있는 논리 자료형이다.

#### 거짓(False)

In [ ]:

```
False
0
None
''      # 빈 문자열
[]      # 빈 리스트
()      # 빈 튜플
{}      # 빈 딕셔너리
set()   # 빈 세트
```

#### 참(True)

- 그 외 나머지(**True**, 0이 아닌 정수 등)

참, 거짓을 숫자로 표현하게 되면...

- **False**는 **0**으로
- **True**는 **1**로 표현한다.

불린 자료형은 주로 **논리 연산**과 **비교 연산**에서 사용한다.

#### 논리 연산

In [ ]:

```
not
and
or
```

| A | B | not A | A and B | A or B |
|---|---|---|---|---|
| true | true | false | true | true |
| true | false | false | false | true |
| false | true | true | false | true |
| false | false | true | false | false |

**비교 연산**(관계 연산)

In [ ]:

```
==    # 같다
!=    # 다르다
>     # 크다
<     # 작다
>=    # 크거나 같다
<=    # 작거나 크다
```

## 따라해보기

힌트: 참, 거짓을 숫자로 표현하게 되면 **False**는 **0**으로 **True**는 **1**로 표현한다.

In [ ]:

```
11 * False
```

In [ ]:

```
11.0 * False  # 실수와 연산하면 실숫값을 반환한다.
```

In [ ]:

```
11 + True
```

In [ ]:

```
11.0 + True   # 실수와 연산하면 실숫값을 반환한다.
```

In [ ]:

```
int(False)    # False를 정수로 변환한다.
```

In [ ]:

```
float(False)  # False를 실수로 변환한다.
```

In [ ]:

```
int(True)     # True를 정수로 변환한다.
```

In [ ]:

```
float(True)   # True를 실수로 변환한다.
```

# bool() 클래스

특정 데이터가 **True**인지 **False**인지를 검증할 때 사용한다.

## 따라해보기

In [ ]:

```
bool(0)
```

In [ ]:

```
bool(0.0)
```

In [ ]:

```
bool(1)
```

In [ ]:

```
bool(-1)
```

In [ ]:

```
bool(10)
```

In [ ]:

```
bool(0.1)
```

In [ ]:

```
bool(' ')
```

In [ ]:

```
bool([])
```

In [ ]:

```
bool([''])
```

In [ ]:

```
bool([1, 2])
```

In [ ]:

```
bool(())
```

# 비교 연산자

- **관계 연산자**라고도 한다.
- 연산자의 양쪽에 있는 값을 서로 비교한다.
  - 즉, 객체의 메모리 주소(객체참조)가 아닌 객체의 값을 비교한다.

- 주로 숫자 또는 문자열을 비교할 때 사용한다.
  - 문자열의 경우 유니코드 값을 기준으로 비교한다.
- 결과 값을 참(**True**) 또는 거짓(**False**) 형태로 반환하다.

프로그래밍에서 비교 연산자는 수학에서 사용될 때와 조금 다른 형태로 표현한다.

| 비교 연산자 | 설명 |
|---|---|
| x **==** y | ***x***와 ***y***가 같은가? |
| x **!=** y | ***x***와 ***y***가 다른가? |
| x **<** y | ***x***가 ***y***보다 작은가? |
| x **<=** y | ***x***가 ***y***보다 작거나 같은가? |
| x **>** y | ***x***가 ***y***보다 큰가? |
| x **>=** y | ***x***가 ***y***보다 크거나 같은가? |

## 따라해보기

In [ ]:

```
'a' == 'A'
```

In [ ]:

```
'a' < 'A'
```

In [ ]:

```
'a' <= 'z'
```

In [ ]:

```
5 != 5.0
```

In [ ]:

```
5 > 5.0
```

In [ ]:

```
5 >= 5.0
```

# 논리 연산자

- **불린 연산**에 사용한다.
- **최소평가**(short-circuit) 연산자다.
  - 즉, 두 번째 피연산자는 첫 번째 피연산자가 식의 값을 결정하기에 충분하지 않은 경우에만 실행되거나 평가된다.
  - 그래서 최소평가 연산은 속도가 빠르다.

| 문법 | 설명 |
|---|---|
| **not** ***x*** | ***x***의 반대를 반환한다. 즉, ***x***가 **False**인 경우 **True**를 반환하고 ***x***가 **True**이면 **False**를 반환한다. |
| ***x*** **and** ***y*** | ***x***와 ***y***가 모두 **True**여야만 **True**를 반환한다. 최소평가 연산자이기 때문에 ***y***는 ***x***가 **True**인 경우에만 계산한다. 둘 다 **True**여야 **True**를 반환하기 때문에 만약 ***x***가 **False**면 굳이 ***y***를 검사하지 않아도 결과가 **False**이기 때문에 ***y***를 검사하지 않고 바로 **False**를 반환한다. |
| ***x*** **or** ***y*** | ***x***와 ***y*** 둘 중 하나라도 **True**면 **True**를 반환한다. 최소평가 연산자이기 때문에 ***y***는 ***x***가 **False**인 경우에만 계산한다. 둘 중 하나만 **True**면 **True**를 반환하기 때문에 만약 ***x***가 **True**면 굳이 ***y***를 검사하지 않아도 결과가 **True**이기 때문에 ***y***를 검사하지 않고 바로 **True**를 반환한다. |

## 따라해보기

In [ ]:

```
# --- not
not True
```

In [ ]:

```
not False
```

In [ ]:

```
# --- and
# 좌변이 False라서 우변은 평가하지 않고 False를 반환한다.
False and True
```

In [ ]:

```
# 우변도 True인지 확인하기 위해 우변까지 평가한다.
True and True
```

In [ ]:

```
# --- or
# 좌변이 True라서 우변은 평가하지 않고 True를 반환하다.
True or False
```

In [ ]:

```
# 좌변이 False라서 우변도 평가한다.
False or False
```

***최소평가(short-circuit) 연산을 눈으로 확인할 수는 없을까?***

## 최소평가 연산

숫자를 사용해 최소평가(short-circuit) 연산을 증명해 보자.

### 따라해보기

In [ ]:

```
# --- and
# 좌변이 False(0)라서 우변은 평가하지 않고 False(0)를 반환한다.
0 and 5
```

In [ ]:

```
# 우변도 True인지 확인하기 위해 우변까지 평가한 후 True(5)를 반환한다.
1 and 5
```

In [ ]:

```
# --- or
# 좌변이 True(5)라서 우변은 평가하지 않고 True(5)를 반환하다.
5 or 1
```

In [ ]:

```
# 좌변이 False(0)라서 우변도 확인해서 True(-1)를 반환한다.
0 or -1
```

***한 번에 여러 개의 논리 연산자를 사용할 수 없을까?***

가능하다. 하지만 논리 연산자 사이에 우선 순위가 있다.

아래 코드를 실행한 결과는?

In [ ]:

```
False and not True or True
```

논리 연산자는 왼쪽에서 오른쪽으로 평가되지 않는다. 즉, 연산 우선 순위가 있다.

## 논리 연산자 우선 순위

1. **not**이 제일 먼저 평가된다.
2. **and**는 그 다음에 평가된다.
3. **or**는 마지막으로 평가된다.

# 멤버십 연산자

**in** / **not in** 연산자

- 특정 항목이 속해 있는지 존재 여부를 확인한다.
- 나중에 복합자료형의 리스트에서 상세히 다룬다.

## 따라해보기

In [ ]:

```
2 in [1, 2, 3]
```

In [ ]:

```
1 in ['1', '2', '3']
```

In [ ]:

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

In [ ]:

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

# 숫자형(Numeric Data Types)

- 일반적으로 수치를 나타내고 수치 연산을 수행하기 위해 사용하는 모든 자료형을 일컫는다.
- **정수**(integer)와 **실수**(float)는 숫자 자료형 중 가장 많이 사용하는 자료형이다.
- **불변자료형**(immutable)이다.

# 정수(Integer Type)

- 소수점이 없는 숫자의 조합으로 이루어지며 음수도 포함한다.
  - 숫자(digits) : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- 표현법
  - 파이썬에서는 **int**로 표현된다.
  - 기본적으로 10진수를 사용하지만 2진수, 8진수, 16진수로도 표현할 수 있다.

## 정수 예시

In [ ]:

```
11
```

In [ ]:

```
+123
```

In [ ]:

```
-5
```

In [ ]:

```
0
```

## 정수 연산

### 더하기, 빼기, 곱하기

In [ ]:

```
9 + 5  # 더하기 연산
```

In [ ]:

```
9 - 5  # 빼기 연산
```

In [ ]:

```
9 * 5  # 곱하기 연산
```

### 나누기

In [ ]:

```
9 / 5    # 나누기 연산
```

**나누기 연산**의 결괏값은 항상 실수(**float**)이기 때문에 '실수 나누기 연산'이라고도 부른다.

**몫 구하기 연산**은 두 수를 나눈 몫을 구한다.

- 두 수를 나눈 후 몫의 값을 정수로 반환하기 때문에 '정수 나누기 연산'이라고도 부른다.

In [ ]:

```
9 // 5   # 몫 구하기 연산
```

**나머지 구하기 연산**은 두 수를 나눴을 때 나머지 값을 구한다.

In [ ]:

```
9 % 5    # 나머지 구하기 연산
```

### 거듭제곱

In [ ]:

```
2 ** 3  # 거듭제곱 연산(2 ** 3 == 2 x 2 x 2)
```

## 정수 연산의 특징

숫자(피연산자) 중 하나라도 실수면 결괏값은 실수다.

In [ ]:

```
1 + 2.0 + 3 + 4 + 5
```

In [ ]:

```
3.0 - 3
```

In [ ]:

```
1 * 1.0
```

## 산술 연산자 우선 순위

#### 산술 연산자 우선 순위

1. 괄호 **()**
2. 지수 **
3. 곱셈, 나눗셈 *, /
4. 덧셈, 뺄셈 +, -

## 증강 할당 연산자 : x *operator*= y

- 모든 이진 산술 연산자(+. -. *, /, //, %, **)는 각 연산자에 대응하는 증강 할당 연산자가 있다.
- 증강 할당 연산자(+=, -=, *=, /=, //=, %=, **=)를 사용하면 일부 표현식을 줄여 쓸 수 있다.
- 즉, 일반식 <***x = x operator y***>를 <***x operator= y***> 형식으로 간편하게 표현할 수 있다.
- 예를 들어, **x = x + y** 형식의 표현식이면 **x += y** 형식으로 표현할 수 있다.

### 따라해보기

In [ ]:

```
x = y = 9
```

In [ ]:

```
# 숫자형 변수 x의 값을 바꾼다.
x = x + 2
x  # 이제 x는 9가 아니다.
```

In [ ]:

```
# 더 간단한 방법은 증강 할당 연산자를 사용하는 것이다.
y += 2
y  # 이제 y는 9가 아니다.
```

## 정리 : 산술 연산자

| 문법 | 설명 |
| --- | --- |
| `x + y` | ▸ *x*와 *y*를 더함; [`x = x + y`]는 [`x += y`]와 같음; *x*, *y* 중 하나가 실수면 실수를 반환 |
| `x - y` | ▸ *x*에서 *y*를 뺌; [`x = x - y`]는 [`x -= y`]와 같음; *x*, *y* 중 하나가 실수면 실수를 반환 |
| `x * y` | ▸ *x*에 *y*를 곱함; [`x = x * y`]는 [`x *= y`]와 같음; *x*, *y* 중 하나가 실수면 실수를 반환 |
| `x / y` | ▸ *x*를 *y*로 나눔; [`x = x / y`]는 [`x /= y`]와 같음<br>▸ 항상 실수(`float`, 부동소수점) 값을 반환<br>- *x*나 *y*가 복소수일 경우 복소수형(`complex`) 값을 반환 |
| `x // y` | ▸ *x*를 *y*로 나누고 소수점 이하는 버림(floor division); [`x = x // y`]는 [`x //= y`]와 같음<br>▸ 소수점 이하를 버리기 때문에, 둘 다 정수인 경우에 정수(`int`) 값을 반환 |
| `x % y` | ▸ *x*를 *y*로 나눴을 때 나머지 값을 반환; [`x = x % y`]는 [`x %= y`]와 같음 |
| `x ** y` | ▸ *x*의 *y* 제곱 값을 반환; [`x = x ** y`]는 [`x **= y`]와 같음; *x*, *y* 중 하나가 실수면 실수를 반환 |
| `-x` | ▸ 부호 반전<br>- *x*가 `0`이 아닌 경우 부호를 반전 시킴<br>- *x*가 `0`이면 아무런 동작도 하지 않음 |
| `+x` | ▸ 아무런 동작도 하지 않음; 가끔 코드의 의미를 명확히 할 필요가 있을 때 사용 |

## 정수형 변환

#### int(객체)

- ***객체***를 정수로 변환한다.
- 변환을 실패하면 **ValueError** 예외가 발생한다.
- 변환한 숫자의 소수점 이하는 버린다.
  - 즉, 반올림을 하지 않고 절삭한다.

### 따라해보기

In [ ]:
```
int()
```

In [ ]:
```
int(10)
```

In [ ]:
```
int('  -10   ')
```

In [ ]:
```
int(12.9999)
```

In [ ]:
```
int('12.999')
```

***오류가 나지 않게 형변환을 할 수 있는 방법은 없을까?***

In [ ]:
```
int(float('12.999'))
```

위와 같이 문자열 안에 숫자가 아닌 마침표가 있다면 실수(**float**)로 형변환을 한 후에 정수로 형변환을 해야 오류가 나지 않는다.

# 실수(Floating-Point Numbers)

- 소수점이 있는 숫자의 조합으로 이루어지며 음수도 포함한다.
  - 숫자(digits) : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- 파이썬에서는 **float**로 표현하며 **부동소수점수**(floating-point number)라고도 한다.
- 소수점 또는 지수 표현법을 사용한다.
  - 지수 표현법은 숫자, **e**, 숫자에 곱할 10진법의 지수를 이용하는 표현이다.

## 실수 예시

In [ ]:

```
0.0
```

In [ ]:

```
.5
```

In [ ]:

```
4.7
```

In [ ]:

```
-3.5
```

In [ ]:

```
7.9e-4
```

## 실수 연산

In [ ]:

```
3 + 7.0     # 더하기 연산
```

In [ ]:

```
.0 - .2     # 빼기 연산
```

In [ ]:

```
5 * 1.2     # 곱하기 연산
```

In [ ]:

```
3 * 7.0e+1  # 곱하기 연산
```

In [ ]:

```
3.6 / 2     # 나누기 연산
```

In [ ]:

```
3.6 // 2    # 몫 구하기 연산
```

In [ ]:

```
3.6 % 2     # 나머지 구하기 연산
```

In [ ]:

```
2.0 ** 3    # 거듭제곱 연산(2.0 ** 3 == 2.0 x 2.0 x 2.0)
```

## 실수형 변환

#### float(객체)

- ***객체***를 실수로 변환한다.
- 변환을 실패하면 **ValueError** 예외가 발생한다.

### 따라해보기

In [ ]:

```
float()
```

In [ ]:

```
float(123)
```

In [ ]:

```
float('  -12.345')
```

In [ ]:

```
float('1e3')
```

In [ ]:

```
float('-Infinity')
```

In [ ]:

```
float('3.14') / float(-.1)
```

In [ ]:

```
float('False')
```

In [ ]:

```
float(False)
```

# 정수와 실수에서 자주 사용하는 함수

**abs(x)**

- ***x***의 절댓값을 반환한다.

In [ ]:

```
# 5의 절댓값을 구한다.
abs(5)
```

In [ ]:

```
# -12의 절댓값을 구한다.
abs(-12)
```

**divmod(x, y)**

- ***x***를 ***y***로 나눴을 때의 '몫'과 '나머지', 총 두 개의 정수를 반환한다.

In [ ]:

```
# 몫과 나머지를 구한다.
divmod(13, 7)
```

In [ ]:

```
# 몫과 나머지를 따로 변수에 할당해서 처리할 수 있다.
x, y = divmod(13, 7)
```

In [ ]:

```
x  # 몫의 값을 확인한다.
```

In [ ]:

```
y  # 나머지의 값을 확인한다.
```

**pow(x, y)**

- ***x***를 ***y***번 곱한 값을 구한다
- 즉, **x ** y**와 같다.

In [ ]:

```
pow(2, 3)  # 2 x 2 x 2
```

**pow()** 함수의 전달인자가 앞의 예처럼 두 개가 아니라 세 개면 전혀 다른 결괏값을 계산한다.

***pow(x, y, z)***

- ***x***를 ***y***번 곱한 값을 구한 후 ***z***로 나누어 나머지 값을 구한다.
- 즉, ***(x** y) % z***와 같다.

In [ ]:

```
pow(2, 3, 5)
```

**round(x, n)**

- 반올림한 결과를 반환한다.
- ***n***이 양의 정수면 ***x***를 소수점 ***n***자리로 반올림하여 반환하며,
- ***n***이 음의 정수면 ***x***를 ***n***의 절댓값 자리에서 반올림하여 반환한다.

- 만약 ***n***이 생략되거나 **0**이면, 소수점 **0**자리로 반올림하기 때문에 소수점 이하를 모두 반올림하는 것과 같다.
- **round()** 함수가 반환하는 값의 자료형은 ***x***와 같다.
  - 즉, ***x***가 정수면 정수를 반환하고 실수면 실수를 반환한다.

In [ ]:

```
round(1.23546, 1)
```

In [ ]:

```
round(1.23546, 2)
```

In [ ]:

```
round(1.23546, 3)
```

In [ ]:

```
round(123546, -2)
```

In [ ]:

```
round(123546, -3)
```

In [ ]:

```
round(123546.789, -3)
```

# 정리 : 정수와 실수에서 자주 사용하는 함수

| 문법 | 설명 |
|---|---|
| abs(x) | ▸ x의 절대 값을 반환 |
| divmod(x,y) | ▸ x를 y로 나눴을 때의 '몫'과 '나머지', 총 2개의 정수를 반환 |
| pow(x,y) | ▸ x ** y와 같음 |
| pow(x,y,z) | ▸ (x ** y) % z와 같음 |
| round(x,n) | ▸ n이 양의 정수면 x를 소수점 n자리로 반올림하여 반환<br>▸ n이 음의 정수면 x를 n자리에서 반올림하여 반환<br>▸ 반환되는 값의 자료형은 x와 같음 |