

# 데이터 과학을 위한 파이썬 프로그래밍

2판



# Chapter 02

## 변수와 자료형



# 목차

1. 변수의 이해
2. 자료형과 기본 연산
3. 자료형 변환

# 학습목표

- 변수의 개념, 변수와 메모리의 관계에 대해 알아본다.
- 변수명을 선언하는 규칙에 대해 이해한다.
- 기본 자료형인 정수형, 실수형, 문자형, 불린형에 대해 학습한다.
- 사칙연산을 비롯한 간단한 연산을 수행한다.
- 자료형 간 변환하는 방법에 대해 알아보고, 자료형을 확인하는 방법을 학습한다.

# 01 변수의 이해

# 1. 변수와 값

```
>>> professor = "Sungchul Choi"
>>> print(professor)
Sungchul Choi
>>> a = 7
>>> b = 5
>>> print(a + b)
12
>>> a = 7
>>> b = 5
>>> print("a + b")
a + b
```

- **print( ) 함수:** 괄호 안의 내용을 화면에 출력하라는 뜻
- print(a + b)와 print("a + b")의 출력 결과가 다른 이유는 무엇일까?
- 이 코드를 실행하면 컴퓨터에서는 어떤 일이 일어날까?

# 1. 변수와 값

```
>>> professor = "Sungchul Choi"  
>>> print(professor)  
Sungchul Choi
```

- 첫 번째 줄의 `professor = "Sungchul Choi"`라는 코드는 무슨 뜻일까?
  - ☞ ④ `professor`에 `Sungchul Choi`를 넣어라.
- '='는 "`professor`라는 공간에 `Sungchul Choi`라는 글자를 넣어라."로 해석할 수 있음
  - ☞ `professor`라는 변수에 `Sungchul Choi`라는 값을 넣으라는 뜻.

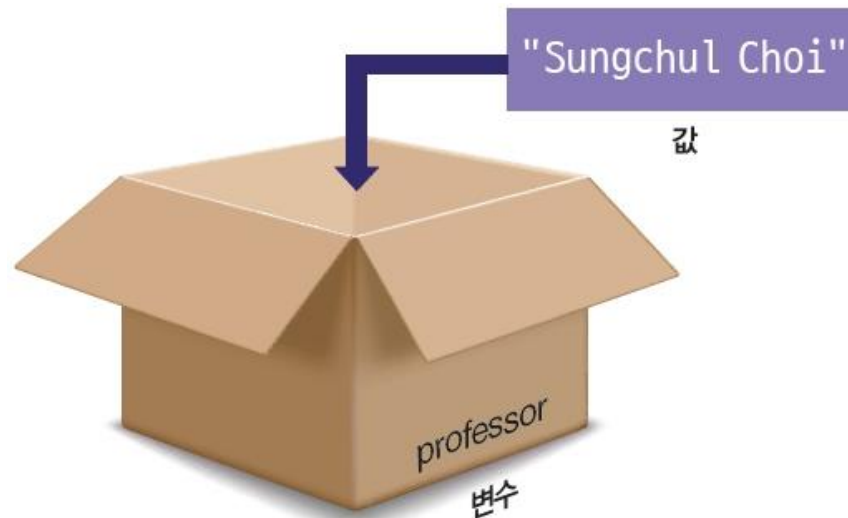


그림 2-1 변수와 값

# 1. 변수와 값

- print(a + b)와 print("a + b")의 차이는 무엇일까?

```
>>> a = 7
>>> b = 5
>>> print(a + b)
12
>>> a = 7
>>> b = 5
>>> print("a + b")
a + b
```

- a = 7과 b = 5 ➡ 변수 a와 변수 b에 각각 7과 5 를 넣으라는 뜻
- print(a + b)와 print("a + b")의 결과가 다른 결정적 차이는 따옴표(" ")의 사용 여부에 있음



# 1. 변수와 값

표 2-1 따옴표의 사용 여부에 따른 의미 차이

코드	의미
<code>print(a + b)</code>	a 변수에 있는 값과 b 변수에 있는 값을 더해 화면에 출력하라
<code>print("a + b")</code>	'a + b'라는 문자를 그대로 화면에 출력하라

- `print(a + b)`: a 변수에 있는 값과 b 변수에 있는 값을 더해 화면에 출력하라는 뜻
- `print("a + b")`: 파이썬 인터프리터는 따옴표 안의 문자를 하나의 값으로 보고 이 값을 그대로 화면에 출력하라는 의미.

## 2. 변수와 메모리

- 수학에서는 변수가 '변할 수 있는 수'라는 뜻이지만, 프로그래밍에서는 '어떠한 값을 저장하는 장소'라는 뜻으로 사용됨
- **메모리(memory):** 변수를 저장하는 물리적 장소이자 변수에 값이 저장되는 공간
- **메모리 주소:** 변수의 저장 위치로, 변수에 들어가는 값은 반드시 어떤 특정한 메모리 주소를 갖게 됨

**TIP** 스마트폰에서 64기가, 128기가와 같은 명칭이 바로 메모리 크기를 말하는 것인데, 이것이 프로그래밍할 때 저장 공간으로 사용되는 메모리를 의미함.

## 2. 변수와 메모리

여기서 잠깐! 컴퓨터의 구조: 폰 노이만 아키텍처

메모리, 변수, 값을 이해하기 위해 컴퓨터의 구조에 대해 알아야 한다. 현재 컴퓨터의 구조는 폰 노이만 아키텍처(Von Neumann architecture)에 기반을 둔다.

폰 노이만 아키텍처는 20세기 초반에 제시된 구조로, 컴퓨터가 어떤 프로그램을 실행시킬 때 실행하고자 하는 값을 컴퓨터의 메모리에 저장하고, 저장된 값을 순차적으로 CPU로 불러와 계산하는 방식을 말한다. 다시 말해 모든 컴퓨터에서는 값이 CPU로 가기 전에 반드시 메모리 공간에 저장되는데, 이 값을 CPU가 하나하나 돌아가면서 처리하는 구조가 오늘날 컴퓨터의 기본 구조인 폰 노이만 아키텍처이다.

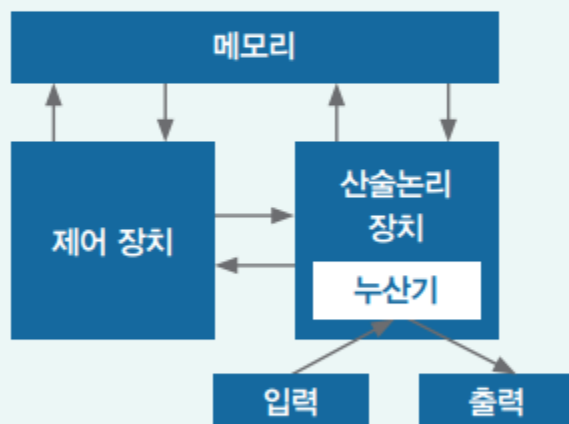


그림 2-2 폰 노이만 아키텍처

**TIP** 존 폰 노이만(John von Neumann) 폰 노이만 아키텍처를 제시했으며, 20세기 가장 뛰어난 천재 중 한 명으로 인정받고 있다. 폰 노이만은 컴퓨터 과학 분야뿐 아니라 수학, 경제학, 생물학 등 다양한 분야에서 천재성을 보인 학자이다. 대표적인 예로 오늘날 경제학에서 많은 사회현상을 설명하는 데 있어 가장 폭넓게 사용되는 게임 이론의 개념을 처음 제시한 사람이기도 하다. 또한, 생물학 분야에서도 DNA가 앞으로 발견될 것임을 예측하기도 하였다.

## 2. 변수와 메모리

- professor = "Sungchul Choi", a = 3, b = 7과 같은 변수를 선언하면 메모리 어딘가에 주소값을 할당받아 저장됨
  - RAM은 휘발성(volatile) 성질이 있어 컴퓨터 전원이 들어와 있는 동안에만 저장됨

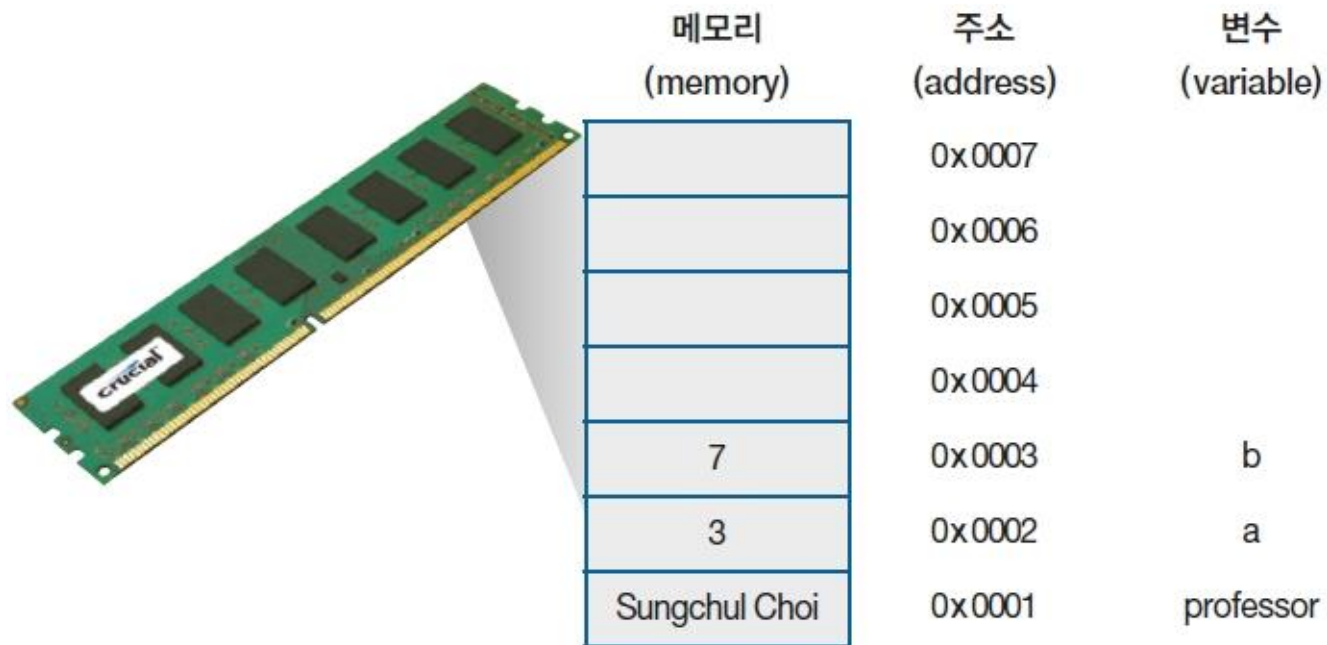


그림 2-3 메모리의 주소 할당

## 2. 변수와 메모리

- **변수:** 프로그램에서 특정한 값을 저장하는 공간의 이름
- **메모리 주소:** 변수에 값을 넣으라고 선언하는 순간 메모리 어딘가에 물리적 공간을 확보할 수 있도록 운영체제와 파이썬 인터프리터가 협력하여 메모리를 할당하여 저장하는 위치
- 예) `a = 3` : 'a라고 하는 메모리 공간에, a라고 하는 메모리 주소에, a라고 하는 변수에 3이라는 값을 넣어라.'

### 3. 변수명 선언

- 알파벳, 숫자, 밑줄( \_ )로 선언할 수 있음

예) `data = 0, _a12 = 2, _gg = 'afdf'`

- 변수명은 의미 있는 단어로 표기하는 것이 좋음

예) `professor_name = ' Sungchul Choi'`

- 변수명은 대소문자가 구분됨

예) ABC와 Abc는 다른 변수임

- 특별한 의미가 있는 예약어는 사용할 수 없음

예) `for`, `if`, `else` 등

**02**

# **자료형과 기본 연산**

# 1. 메모리 공간

## ■ 메모리 공간의 기본 개념

- 하드디스크(최근에는 SSD 사용)에 필요한 파일을 저장하면, 그 파일은 어느 정도의 저장 공간을 사용하게 됨
- 다시 말해 하나의 변수를 메모리에 저장할 때 그 변수의 크기 만큼 공간을 할당 받음
- 컴퓨터는 0과 1, 두 가지 정보만 저장할 수 있으므로 이진수를 사용함.
  - 비트(bit): 이진수 한 자리
  - 바이트(byte): 8개의 비트
  - 킬로바이트(kilobyte, KB): 1,024바이트
  - 메가바이트(megabyte, MB): 1,024킬로바이트

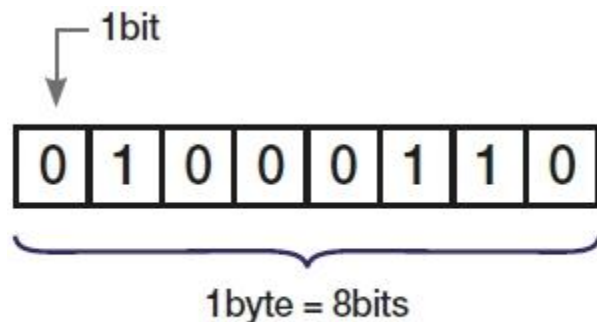



그림 2-4 비트(bit)와 바이트(byte)



# 1. 메모리 공간

여기서  잠깐! 컴퓨터가 이진수를 사용하는 이유

컴퓨터는 왜 이진수를 사용할까? 컴퓨터의 메모리는 실리콘으로 만든 반도체이다. 반도체의 가장 큰 특징은 어떤 자극을 주었을 때 전기가 통할 수 있어 전류의 흐름을 제어할 수 있다는 것이다. 이러한 성질을 이용해 반도체에 전류가 흐를 때 1, 흐르지 않을 때 0이라는 숫자로 표현할 수 있다. 따라서 메모리는 전류의 흐름을 이진수로 표현하는 것이다.

## 2. 기본 자료형

- 할당 받는 메모리 공간의 크기는 변수의 자료형(data type)에 의해 결정됨.

### 2.1 정수형 (integer type)

- 자연수를 포함해 0, 1, 2, -1, -2와 같이 값의 영역이 정수로 한정된 값
- 데이터를 선언할 때는 `data = 1`과 같은 방식으로 선언
- 파이썬의 인터프리터가 알아서 메모리 영역에 필요한 공간을 확보

### 2.2 실수형 (floating-point type)

- 10.2, 7.2와 같이 소수점이 포함된 값
- 실제로 값이 정수형이라도 9.0으로 입력하면 인터프리터는 실수형으로 해석

**TIP** **부동 소수점(floating-point):** 컴퓨터에서 실수를 표시하는 방법으로, 유효숫자와 소수점의 위치를 사용하여 실수를 표현

## 2. 기본 자료형

### 2.3 문자형 (string type)

- 값이 문자로 출력되는 자료형
- 파이썬에서는 보통 따옴표에 들어간 정보를 문자형 데이터라고 함.

### 2.4 불린형 (boolean type)

- 논리형이라고도 하며, 참(True) 또는 거짓(False)을 표현할 때 사용

표 2-2 기본 자료형

유형	자료형	설명	예	선언 형태
수치형	정수형	양수와 정수	1, 2, 3, 100, -9	data = 1
	실수형	소수점이 포함된 실수	10.2, -9.3, 9.0	data = 9.0
문자형	문자형	따옴표에 들어가 있는 문자형	abc, a20abc	data = 'abc'
논리형	불린형	참 또는 거짓	True, False	data = True

## 2. 기본 자료형

### 여기서 잠깐! 동적 타이핑

동적 타이핑(dynamic typing)은 변수의 메모리 공간을 확보하는 행위가 프로그램 실행 시점에서 발생하는 것을 뜻한다. 조금 어렵게 느낄 수 있는 개념이다. 일반적으로 C나 자바는 `int data = 8`과 같이 data라는 변수가 정수형이라고 사전에 선언한다. 그에 비해 파이썬은 `data = 8` 형태로 선언한다. 즉, data라는 변수의 자료형이 정수(integer)인지 실수(float)인지를 프로그래머가 아닌 인터프리터가 판단하는 것이다. 그리고 그것을 프로그램의 실행 시점에 동적으로 판단하는데, 이것을 파이썬 언어가 동적으로 자료형을 결정한다라고 말한다.

파이썬은 메모리 공간을 얼마나 사용할까? 앞서 이야기했듯이 변수에 있는 값을 메모리에 저장하기 위해서는 일정한 메모리 공간을 할당받아야 한다. 다른 언어들과 달리 파이썬은 매우 유연한 언어로, 할당받는 메모리 공간도 저장되는 값의 크기에 따라 동적으로 할당받을 수 있다.

## 2. 기본 자료형

- 다음 코드를 파이썬 셸에 입력하여 실제 값이 화면에 출력되는지 확인

```
>>> a = 1          # 정수형
>>> b = 1          # 정수형
>>> print(a, b)
1 1
>>> a = 1.5        # 실수형
>>> b = 3.5        # 실수형
>>> print(a, b)
1.5 3.5
>>> a = "ABC"      # 문자형
>>> b = "101010"   # 문자형
>>> print(a, b)
ABC 101010
>>> a = True       # 불린형
>>> b = False      # 불린형
>>> print(a, b)
True False
```

### 3. 간단한 연산

- 기본적으로 연산은 연산자와 피연산자로 구분할 수 있음.
  - 연산자:  $+$ ,  $-$ ,  $*$ ,  $/$  기호
  - 피연산자: 연산자에 의해 계산되는 숫자

## 3. 간단한 연산

### 3.1 사칙연산

- 덧셈 연산자: 덧셈 기호(+)
- 뺄셈 연산자: 뺄셈 기호(-)
- 곱셈 연산자: 별표 기호(\*)
- 나눗셈 연산자: 빗금 기호(/ )

```
>>> 25 + 30
55
>>> 30 - 12
18
>>> 50 * 3
150
>>> 30 / 5
6.0
```

## 3. 간단한 연산

### 3.2 제곱승

- 파이썬에서 제곱승을 구하는 연산자는 2개의 별표(\*\*)임.

```
>>> print(3 * 3 * 3 * 3 * 3)    # 3을 다섯 번 곱함
243
>>> print(3 ** 5)               # 3의 5승
243
```



## 3. 간단한 연산

### 3.3 나눗셈의 몫과 나머지

- 파이썬에서 몫을 반환하는 연산자는 2개의 빗금 기호(//)이며, 나머지 연산자는 백분율 기호 (%)임.

```
>>> print(7 // 2)      # 7 나누기 2의 몫
3
>>> print(7 % 2)      # 7 나누기 2의 나머지
1
```

## 3. 간단한 연산

### 3.4 증가 연산과 감소 연산

- C나 자바에서는 `a++` 또는 `a--`로 표현하고 변수 `a`에 들어가 있는 값을 1만큼 증가 또는 감소하라는 의미

```
>>> a = 1                                # 변수 a에 1을 할당
>>> a = a + 1                            # a에 1를 더한 후 그 값을 다시 a에 할당
>>> print(a) # a 출력
2
>>> a += 1                               # a 증가 연산
>>> print(a)                             # a 출력
3
>>> a = a - 1                            # a에서 1을 뺀 후 그 값을 다시 a에 할당
>>> a -= 1                               # a 감소 연산
>>> print(a)                             # a 출력
1
```

# **03**

## **자료형 변환**

# 1. 정수형과 실수형 간 변환

- 변수의 자료형은 float( ) 함수나 int( ) 함수를 사용하면 간단히 변환 가능

```
>>> a = 10                                # a 변수에 정수 데이터 10을 할당
>>> print(a)                             # a가 정수형으로 출력
10
>>> a = float(10)                         # a를 실수형으로 변환 / 정수형인 경우 int()
>>> print(a)                             # a를 출력
10.0                                     # a가 실수형으로 출력
```

- a = 10이라고 선언하면, a는 정수형으로 선언됨
- 실수형으로 변환해주는 float( ) 함수를 사용하면 기존 a값인 10을 10.0으로 변경함

# 1. 정수형과 실수형 간 변환

- b값을 3으로 선언하고, 기존 a값(10)을 b값(3)으로 나눈 코드

```
>>> a = 10          # a에 정수 데이터 10 할당
>>> b = 3           # b에 정수 데이터 3 할당
>>> print(a / b)    # 실수형으로 a 나누기 b를 출력
3.3333333333333335  # 실수형 결과값 출력
```

- 정수형 변수 2개를 나눗셈하면 결과는 3.3333333...이 나옴
- A = 10.0이라고 해도 결과값은 똑같음

# 1. 정수형과 실수형 간 변환

- 실수형을 정수형으로 변환해주는 int( ) 함수의 코드

```
>>> a = int(10.7)
>>> b = int(10.3)
```

- `a = int(10.7)`, `b = int(10.3)`에서 10.7과 10.3은 실수형임.

- 이 두 수를 정수형으로 변환한 후 더한 결과 확인하기

```
>>> print(a + b)           # 정수형 a와 b의 합을 출력
20
>>> print(a)              # 정수형 a값 출력
10
>>> print(b)              # 정수형 b값 출력
10
```

- 수학에서 정수형으로 바꾼다고 하면 반올림이 일어나지만, 여기에서는 내림이 발생.

# 1. 정수형과 실수형 간 변환

## TIP 함수

- 수학에서  $f(x)$ 와 같은 꼴을 함수라고 하며,  $x$ 에 어떤 값이 들어가면 그 함수가 가진 수식에 따라 변환되어 나옴

예) 함수 ' $f(x) = 2x + 4$ '라고 한다면  $f(2)$ 일 때 결과값은 8이 나옴.

- 프로그래밍에서 함수는 어떤 값이 입력되면 그 함수가 프로그래밍되어 있는 형태로 결과값이 변형되어 출력됨

### 여기서 잠깐! 형 변환을 하지 않아도 형 변환이 일어나는 경우

'10 / 3'처럼 별도의 형 변환을 하지 않아도 자연스럽게 자료형이 변환되는 경우가 있다. 이것도 역시 파이썬의 대표적인 특징인 동적 타이핑 때문에 나타나는 현상 중 하나이다. 이러한 현상은 값의 크기를 비교할 때도 나타난다. 대표적인 예로 1은 정수형이고 True는 불린형인데, 이것을 ' $1 == \text{True}$ '라고 입력하면 결과는 True로 출력된다. 또한, 아무것도 넣지 않은 "" 같은 문자열을 불린형과 비교하면 False로 인식된다. 모두 파이썬의 특징에 의해 나타나는 현상이므로 기억해야 한다.

## 2. 숫자형과 문자형 간 변환

- 문자형으로 선언된 값도 정수형이나 실수형으로 변환할 수 있으며 물론 그 반대도 가능

```
>>> a = '76.3'          # a에 문자형 76.3을 할당 → 문자열을 의미
>>> b = float(a)        # a를 실수형으로 변환 후 b에 할당
>>> print(a)            # a값 출력
76.3
>>> print(b)            # b값 출력
76.3
>>> print(a + b)        # a와 b를 더함 → 문자열과 숫자열의 덧셈이 불가능하여 에러 발생
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: can only concatenate str (nor "float") to str
```

- a는 '76.3'이라고 문자형으로 선언하고, 다음으로 b = float(a)로, a를 실수형으로 변환한 값을 b에 할당함. 각각의 값을 화면에 출력하면 두 값 모두 76.3이 출력됨.
- 둘을 더하면 에러 발생 🖱 숫자형 데이터와 문자형 데이터는 기본 연산이 되지 않음.



## 2. 숫자형과 문자형 간 변환

- 두 변수를 더하기 위해서는 두 변수의 자료형을 통일해야 함

```
>>> a = float(a)           # a를 실수형으로 변환 후 a에 할당
>>> b = a                  # 실수형 a값을 b에 할당
>>> print(a + b)           # 두 실수형을 더한 후 출력
152.6
```

- a의 값을 실수형으로 변환한 후 실수형 a값을 b에 할당하고, 다시 덧셈 연산을 실행하면 두 수가 더해진 채로 화면에 출력됨

## 2. 숫자형과 문자형 간 변환

- 이번에는 a, b를 문자형으로 바꾼 후 덧셈을 실행해보자.

```
>>> a = str(a)           # 실수형 a값을 문자형으로 변환 후 a에 할당
>>> b = str(b)           # 실수형 b값을 문자형으로 변환 후 b에 할당
>>> print(a + b)         # 두 값을 더한 후 출력
76.376.3                 # 문자형 간 덧셈은 문자열 간 단순 연결
```

- 위 코드와 같이 str( ) 함수는 기존의 정수형이나 실수형을 문자열로 바꿔줌.
- 문자형 간의 덧셈은 숫자 연산이 아닌 단순 붙이기(concatenate)가 일어남.

### 3. 자료형 확인하기

- 변수의 자료형을 알기 위해서는 type( ) 함수 사용

```
>>> a = int(10.3)          # a는 정수형으로 10.3을 할당
>>> b = float(10.3)        # b는 실수형으로 10.3을 할당
>>> c = str(10.3)          # c는 문자형으로 10.3을 할당
>>>
>>> type(a)                # a의 타입을 출력
<class 'int'>
>>> type(b)                # b의 타입을 출력
<class 'float'>
>>> type(c)                # c의 타입을 출력
<class 'str'>
```

- '10.3'이라는 값을 각각 정수형, 실수형, 문자형으로 a, b, c 변수에 저장한 후 print( ) 함수를 사용해 화면에 출력하면 각각 다른 값이 나옴
- 별도로 print( ) 함수를 사용하지 않고 각 자료형을 확인하기 위해서는 type( ) 함수를 사용하면 되는데, 변수별로 정수형은 'int', 실수형은 'float', 문자형은 'str'로 출력

# Thank you!