변수와 자료형 (2)

PDF	비어 있음
비고	기본 자료형
숫자	3
실습문제	비어 있음
실습문제답안	비어 있음

02. 기본 자료형

02-01. 숫자형 (Numeric)

02-01-01. 숫자 자료형의 종류

💡 파이썬에서는 정수, 실수, 복소수 등 숫자를 표현하는 자료형을 전부 숫자 자료형이라고 표현한다.

- int : 정수 값을 가지는 자료형

```
num1 = 1 num2 = -1 print(type(num1)) # <class 'int'> print(type(num2)) # <class 'int'>
```

- float : 실수(부동소수점) 값을 가지는 자료형

```
num3 = 3.14 num4 = 1.0 print(type(num3)) # <class 'float'> print(type(num4)) # <class 'float'>
```

- complex : 복소수 값을 가지는 자료형

```
num5 = 3 + 4j print(type(num5)) # <class 'complex'>
```

02-01-02. 숫자 자료형의 연산

- 숫자 자료형 간에는 기본적인 사칙연산과 파이썬에서만 가능한 특수연산을 할 수 있다.

1. 사칙연산 (+, -, ×, ÷) : +, -, *, /

```
num1 = 10 num2 = 3 print(num1 + num2) print(num1 - num2) print(num1 * num2) print(num1 / num2)
```

- 숫자 자료형 중에서도 정수형과 실수형을 연산하는 경우 정수형을 실수형으로 자동 변환하여, 결과는 실수형으로 반환된다.

```
num3 = 10 num4 = 0.1 print(num3 + num4) print(num3 - num4) print(num3 * num4) print(num3 / num4)
```

2. 특수연산
   a. % : 나누기 연산을 하고 나머지를 반환한다.

```
num1 = 11 num2 = 7 print(num1 % num2)
```

   b. // : 나누기 연산을 하고 몫을 반환한다.

```
num1 = 11 num2 = 7 print(num1 // num2)
```

   c. ** : 연산자의 좌항의 수를 밑, 우항의 수를 지수로 하여 제곱 연산한 결과를 반환한다.

```
base = 9 exponent = 2 print(base ** exponent)
```

02-02. 논리형 (Boolean)

02-02-01. 논리 자료형

💡 논리 자료형은 **참 또는 거짓**을 표현하는 자료형으로 **True, False**를 사용한다.

- 논리 자료형은 문자열이 아니므로 따옴표(‘’, “”)로 감싸서는 안되며, 첫 글자를 반드시 대문자로 해야만 한다.

```
bool1 = True bool2 = False print(type(bool1)) # <class 'bool'> print(type(bool2)) #
<class 'bool'>
```

- 또한 논리 자료형은 자료형 자체로 사용하기도 하지만 비교 연산, 논리 연산의 결과로 논리 자료형이 반환된다.

02-02-02. 논리 자료형의 연산

- 비교 연산 : ==, !=, >, <, >=, <=
  - == : 좌항과 우항의 값이 동일한지 비교하여, 같으면 True, 같지 않으면 False를 반환한다.
  - != : 좌항과 우항의 값이 같지 않은지 비교하여, 서로 다르면 True, 같으면 False를 반환한다.
  - >, < : 일반적으로 수학에서 사용하는 부등호와 같이 좌항과 우항의 크고 작음을 비교한다.
  - >=, <= : 같거나 크다 또는 같거나 작다를 비교한다.

```
print(1 == 3) # False print('ohgiraffers' != 'Ohgiraffers') # True print(100 > 99) #
True print(100 < 99) # False print(0 >= 1) # False print(0 <= 1) # True
```

> ✸ **! Tip !**
> 문자 자료형에서 소문자와 대문자는 같지 않다. 컴퓨터는 내부적으로 소문자와 대문자를 아스키코드표에 의해 해석하기 때문이다. (아스키코드표 상에서 대문자가 소문자보다 작은 수이다.)

- 논리 연산 : AND, OR, NOT
  - and : 좌항과 우항의 비교 연산 결과 또는 논리 자료형이 전부 True여야만 True의 결과를 반환한다. 하나라도 False이면 False를 반환한다.

```
print(1 != 3 and 100 > 99 and True) # True print(1 != 3 and 100 > 99 and 100 < 99) # False
```

  - or : 좌항과 우항의 비교 연산 결과 또는 논리 자료형 중 하나라도 True이면 True의 결과를 반환한다. 전부 False이면 False를 반환한다.

```
print(1 != 3 or 100 > 99 or True) # True print(1 != 3 or 100 > 99 or 100 < 99) # True print(1 == 3 or False or 100 < 99) # False
```

  - not : 우항에 있는 비교 연산 결과 또는 논리 자료형을 뒤집는다. 즉, True가 오면 False를, False가 오면 True를 반환한다.

```
print(not'ohgiraffers' == 'Ohgiraffers') # True print(not True) # False
```

02-03. 문자형 (String)

02-03-01. 문자형 선언

- 큰따옴표(" ") 또는 작은따옴표(' ')를 사용하여 문자형 자료(리터럴)을 선언할 수 있다.

```
my_name = 'deer'
```

- print() 함수의 매개변수로 삼중 따옴표를 사용하면 문자열 여러 줄을 전달하여 출력할 수 있다.

```
print(""" 세상에서 제일 긴 게 뭘까요? 기차? 기린의 목? 코끼리의 코? 전부 아닙니다. 여러분의 무한한 가능성입니다^^ """)
```

02-03-02. 문자형 연산

- 문자열 이어 붙이기 : +
  - 문자형 간 + 연산자를 이용하여 이어 붙이기를 할 수 있다.

```
greeting = "반갑습니" name = "다람쥐" print(greeting + name) # 반갑습니다람쥐
print(greeting + "다. " + name + "입니다!") # 반갑습니다. 다람쥐입니다!
```

```
# 다른 자료형과 연산 불가 (자동 형 변환 불가) # print(900 + "살 " + name + "이올시다.")
# 아래처럼 다른 자료형은 문자형으로 형 변환 이후 연산 가능 print(str(900) + "살 " + name + "이올시다.")
```

- 문자열 반복하기 : *
  - 문자형에 * 연산을 하면 문자형을 반복할 수 있다.

```
subject = "python" print("무슨 일이든 시작 전에 세 번만 외쳐보자. " + ("나는 " + subject + "을 잘할 수 있다! ") * 3) # 무슨 일이든 시작 전에 세 번만 외쳐보자. 나는 python을 잘할 수 있다! 나는 python을 잘할 수 있다! 나는 python을 잘할 수 있다!
```

02-03-03. 문자형의 메서드

1. replace(old, new) : 문자열을 치환하는 메서드이다.

```
enroll_date = '2025/06/30' rep_enroll_date = enroll_date.replace("/", "-") print(rep_
enroll_date) # 2025-06-30
```

2. strip([chars])
   a. 인자는 제거할 문자 집합을 지정하는 문자열이다.
   b. 문자열의 선행과 후행에서 해당 문자가 제거된 문자열의 복사본을 반환한다.
   c. 인자를 생략하면 공백을 제거한다.

```
origin = 'ohgiraffers' with_white_space = ' oh giraffers ' # 인자 생략 == 공백 제거 pr
int(with_white_space.strip()) # oh giraffers # ' o'까지 제거 print(with_white_space.s
trip(' o')) # h giraffers # 'os' 제거 print(origin.strip('os')) # hgiraffer # lstrip
() - 선행만 제거 print(origin.lstrip('os')) # hgiraffers # rstrip() - 후행만 제거 prin
t(origin.rstrip('os')) # ohgiraffer
```

3. 대소문자 관련 메서드
   a. upper() : 대문자로 변경한다.
   b. lower() : 소문자로 변경한다.
   c. capitalize() : 첫 글자만 대문자로 변경한다.
   d. swapcase() : 대문자는 소문자로, 소문자는 대문자로 변경한다.
   e. title() : 단어의 첫글자를 대문자로 변경한다.

```
origin_str = 'hELLO wORLD!' print(origin_str.upper()) # HELLO WORLD! print(origin_st
r.lower()) # hello world! print(origin_str.capitalize()) # Hello world! print(origin_
str.swapcase()) # Hello World! print(origin_str.title()) # Hello World!
```

02-03-04. 문자형 포맷팅

1. % 포맷팅 : 변수 포맷을 사용하여 문자열에 변수 값을 삽입할 수 있다. (오래된 방식)
   - 변수 포맷 종류
     - %s : 문자열
     - %c : 문자
     - %d : 정수
     - %f : 실수

```
x = 10 print("x is %d" %x) # x is 10 y = "code" print("y is %s" % y) # y is code
```

2. format() : 타입 명시 없이 format() 함수를 이용할 수도 있다.

```
{인덱스0}, {인덱스1} = {인덱스2}.format(a,b,a*b)
```

- 단, 이때 이때 더 큰 범위로만 자동 형 변환된다.

```
x, y = 10, "code" print("x is {0}".format(x)) # x is 10 print("x is {0} y is {1}".for
mat(x,y)) # x is 10 y is code print("x is {new_x} and y is {new_y}".format(new_x=x, n
ew_y=y)) # x is 10 and y is code
```

3. f-string : 변수를 활용하여 문자열을 구성할 때, 따옴표 앞에 f를 붙이고 따옴표 안에서 중괄호{} 안에 변수명을 넣어 활용할 수 있다.

```
teacher_name = "다람쥐" print(f"안녕하세요. 오늘부터 여러분과 함께 공부할 {teacher_nam
e} 강사입니다~!") # 안녕하세요. 오늘부터 여러분과 함께 공부할 다람쥐 강사입니다~!
```

02-04. 주석

02-04-01. 주석의 종류

- 주석이란 컴퓨터가 해석하지 않는, 무시하고 지나치는 구문을 의미한다.
  - 한 줄 주석 # : # 이후로 오는 문자를 컴퓨터가 해석하지 않는다.

```
이렇게 그냥 작성하면 에러가 발생합니다. # SyntaxError 발생
```

```
# 주석은 해석하지 않으므로 에러가 발생하지 않습니다.
```

```
print('이건 해석하구요') # print('이건 해석하지 않습니다') print('여기는 해석합니
다.') # 여기부터는 해석하지 않습니다.
```

  - 여러 줄 주석 '''~''', """~""" : 삼중 따옴표 시작부터 삼중 따옴표가 끝나는 구간까지의 내용을 컴퓨터가 해석하지 않는다.

```
''' 삼중 따옴표를 사용하면 구간에 걸친 주석이 적용됩니다. print('그래서') print('여기
도') print('출력되지 않습니다.') ''' print('여기만 출력됩니다.')
```

```
""" 작은 따옴표여도 큰 따옴표여도 똑같습니다. print('그래서') print('여기도')
print('출력되지 않습니다.') """ print('여기만 출력됩니다.')
```

02-05. 형 변환

02-05-01. 형 변환이란

- 서로 다른 자료형 간 연산 등을 위하여 다른 자료형으로 타입을 변환하는 것이다.

02-05-02. 형 변환의 종류

1. 암시적 형 변환

```
print(True + 3) # 4 print(3 + 5.0) # 8.0 # print(3 + "5") # 암시적 변환 불가
```

2. 명시적 형 변환
   - int(자료형) : 문자형이나 숫자형 중 실수를 정수으로 형 변환하는 함수
     - 단, 형식에 맞는 문자열만 정수로 변환 가능하다.

     ```
     print(int('3') + 4) # 7 # print(int("3.5") + 2) # 문자열의 형태가 정수형이 아니므로 형 변환 불가
     ```

   - float(자료형) : 문자형이나 숫자형 중 정수를 실수로 형 변환하는 함수
     - 단, 형식에 맞는 문자열만 실수로 변환 가능하다.

     ```
     print(float('3')) # 3.0
     ```

   - str(자료형) : 숫자형, 리스트, 튜플, 딕셔너리 등의 자료형을 문자로 형 변환하는 함수
     - 모든 자료형이 문자열로 형 변환 가능하다.