

3주

변수와 입력, 문자열 내장 함수



염 희 균

- 변수 만들기/사용하기
- 복합 대입 연산자
- 사용자 입력 : input()
- 문자열을 숫자로 바꾸기
- 숫자를 문자열로 바꾸기
- 다양한 문자열 함수
- 키워드로 정리하는 핵심 포인트
- 확인문제

시작하기 전에

[핵심 키워드] 변수 선언, 변수 할당, 변수 참조, input(), int(), float(), str()

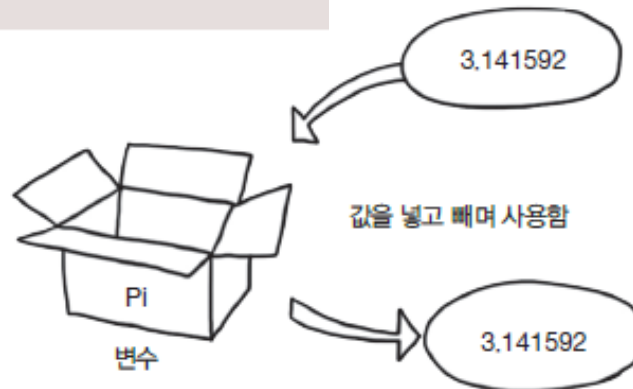
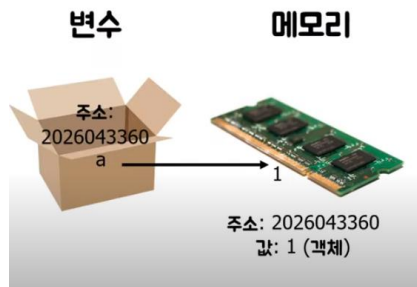
[핵심 포인트] 변수는 숫자뿐만이 아닌 모든 자료형을 의미하며, 파이썬에서 변수를 생성하는 것은 이를 사용하겠다고 선언하는 것이다. 변수에는 모든 자료형의 값을 저장할 수 있다.

시작하기 전에

- 변수

- 값을 저장할 때 사용하는 식별자
- 숫자뿐만 아니라 모든 자료형을 저장할 수 있음

```
>>> pi = 3.14159265  
>>> pi  
3.14159265
```



변수 만들기/사용하기

- 변수의 활용

- 변수를 선언하는 방법
 - 변수를 생성
- 변수에 값을 할당하는 방법
 - 변수에 값을 넣음
 - = 우변의 값을 좌변에 할당
- 변수를 참조하는 방법
 - 변수에서 값을 꺼냄
 - 변수 안에 있는 값을 사용

변수 = 값



값을 변수에 할당합니다.

변수 만들기/사용하기

- 변수를 참조

- 변수에 저장된 값을 출력

```
변수
```

- 변수에 저장된 값으로 연산

```
변수 + 변수
```

- 변수에 저장된 값을 출력

```
print(변수)
```

변수 만들기/사용하기

- 앞 예시에서 입력한 pi는 숫자 자료에 이름 붙인 것이기 때문에 숫자 연산 모두 수행할 수 있음

```
>>> pi = 3.14159265
>>> pi + 2
5.14159265
>>> pi - 2
1.1415926500000002
>>> pi * 2
6.2831853
>>> pi / 2
1.570796325
>>> pi % 2
1.1415926500000002
>>> pi * pi
9.869604378534024
```

변수 만들기/사용하기

- pi는 숫자 자료이므로 숫자와 문자열 연산은 불가능

```
pi + "문자열"
```

- 예시 - 원의 둘레와 넓이 구하기

```
01  # 변수 선언과 할당
02  pi = 3.14159265
03  r = 10
04
05  # 변수 참조
06  print("원주율 =", pi)
07  print("반지름 =", r)
08  print("원의 둘레 =", 2*pi*r)      # 원의 둘레
09  print("원의 넓이 =", pi*r*r)     # 원의 넓이
```

실행결과

```
원주율 = 3.14159265
반지름 = 10
원의 둘레 = 62.831853
원의 넓이 = 314.159265
```


복합 대입 연산자

- 복합 대입 연산자
 - 기본 연산자와 = 연산자 함께 사용해 구성

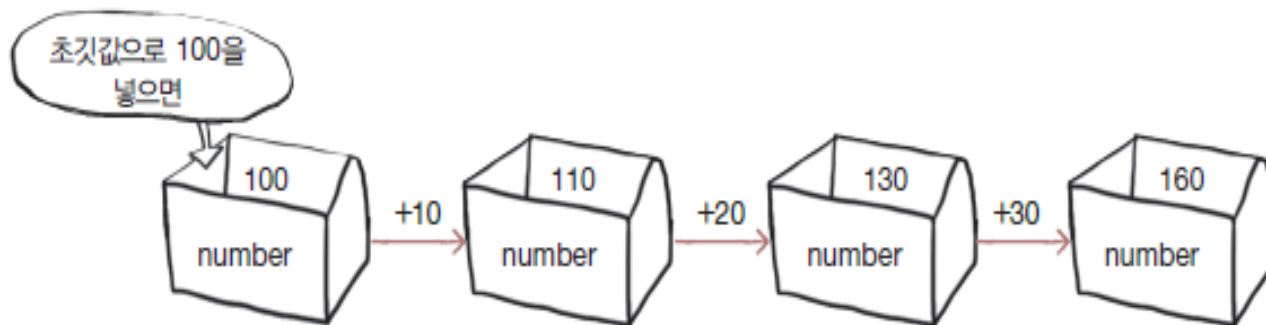
```
a += 10
```

연산자 이름	설명
+=	숫자 덧셈 후 대입
-=	숫자 뺄셈 후 대입
*=	숫자 곱셈 후 대입
/=	숫자 나눗셈 후 대입
%=	숫자의 나머지를 구한 후 대입
**=	숫자 제곱 후 대입

복합 대입 연산자

- 예시

```
>>> number = 100  
>>> number += 10  
>>> number += 20  
>>> number += 30  
>>> print("number:", number)  
number: 160
```



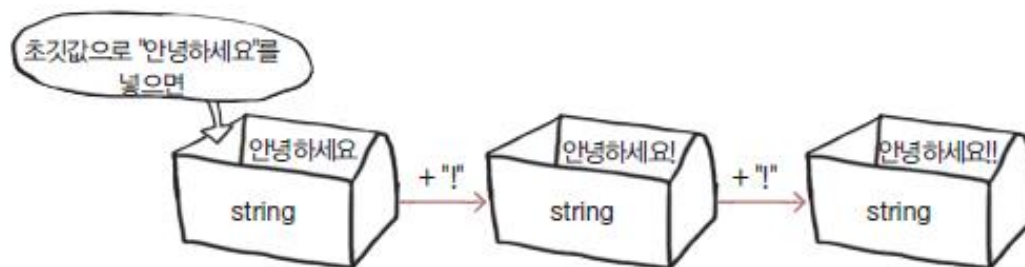
복합 대입 연산자

- 문자열 역시 복합 대입 연산자 사용 가능

연산자 이름	설명
<code>+=</code>	문자열 연결 후 대입
<code>*=</code>	문자열 반복 후 대입

- 예시

```
>>> string = "안녕하세요"
>>> string += "!"
>>> string += "!"
>>> print("string:", string)
string: 안녕하세요!!
```



사용자 입력 : input()

- input() 함수

- 명령 프롬프트에서 사용자로부터 데이터 입력받을 때 사용

- input() 함수로 사용자 입력받기

- 프롬프트 함수 : input 함수 괄호 안에 입력한 내용

```
>>> input("인사말을 입력하세요> ")
```

- 블록 (block) : 프로그램이 실행 중 잠시 멈추는 것

인사말을 입력하세요> | → 입력 대기를 알려주는 커서입니다. 커서는 프로그램에 따라 모양이 다를 수 있습니다.

- 명령 프롬프트에서 글자 입력 후 [Enter] 클릭

```
인사말을 입력하세요> 안녕하세요 
```

```
'안녕하세요'
```

사용자 입력 : input()

- input 함수의 결과로 산출 (리턴값)
 - 다른 변수에 대입하여 사용 가능

```
>>> string = input("인사말을 입력하세요> ")
인사말을 입력하세요> 안녕하세요 [Enter]
>>> print(string)
안녕하세요
```

사용자 입력 : input()

- input() 함수의 입력 자료형
 - type() 함수로 자료형 알아보기

```
>>> print(type(string))  
<class 'str'>
```

```
>>> number = input("숫자를 입력하세요> ")  
숫자를 입력하세요> 12345   
>>> print(number)  
12345
```

```
>>> print(type(number))  
<class 'str'>
```

- input() 함수는 사용자가 무엇을 입력해도 결과는 무조건 문자열 자료형

사용자 입력 : input()

- 예시 - 입력 자료형 확인하기

```
01  # 입력을 받습니다.  
02  string = input("입력> ")  
03  
04  # 출력합니다.  
05  print("자료:", string)  
06  print("자료형:", type(string))
```

실행결과 1

```
입력> 52273 Enter  
자료: 52273  
자료형: <class 'str'>
```

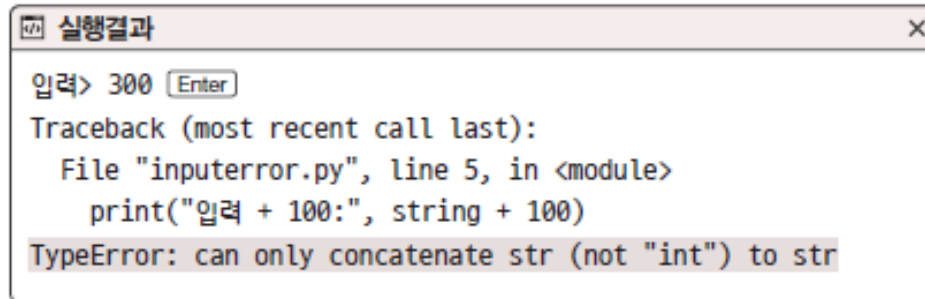
실행결과 2

```
입력> True Enter  
자료: True  
자료형: <class 'str'>
```

사용자 입력 : input()

- 예시 - 입력받고 더하기

```
01  # 입력을 받습니다.  
02  string = input("입력> ")  
03  
04  # 출력합니다.  
05  print("입력 + 100:", string + 100)
```



The screenshot shows a terminal window titled "실행결과" (Execution Result). It displays the user input "입력> 300" followed by an "Enter" button. Below this, a traceback message is shown: "Traceback (most recent call last):", "File \"inputerror.py\", line 5, in <module>", "print(\"입력 + 100:\", string + 100)", and the error message "TypeError: can only concatenate str (not \"int\") to str".

문자열을 숫자로 바꾸기

- 캐스트 (cast)
 - input() 함수의 입력 자료형은 항상 문자열이므로 입력받은 문자열을 숫자 연산에 활용하기 위해 숫자로 변환
- int() 함수
 - 문자열을 int 자료형으로 변환.
- float() 함수
 - 문자열을 float 자료형으로 변환

문자열을 숫자로 바꾸기

- 예시 - int() 함수 활용하기

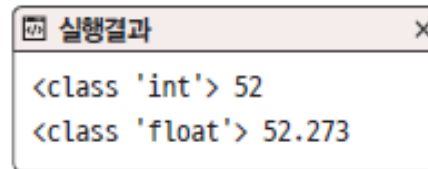
```
01 string_a = input("입력A> ")
02 int_a = int(string_a)
03
04 string_b = input("입력B> ")
05 int_b = int(string_b)
06
07 print("문자열 자료:", string_a + string_b)
08 print("숫자 자료:", int_a + int_b)
```



문자열을 숫자로 바꾸기

- 예시 - int() 함수와 float() 함수 활용하기

```
01 output_a = int("52")
02 output_b = float("52.273")
03
04 print(type(output_a), output_a)
05 print(type(output_b), output_b)
```



실행결과

```
<class 'int'> 52
<class 'float'> 52.273
```

- 예시 - int() 함수와 float 함수 조합하기

```
01 input_a = float(input("첫 번째 숫자> "))
02 input_b = float(input("두 번째 숫자> "))
03
04 print("덧셈 결과:", input_a + input_b)
05 print("뺄셈 결과:", input_a - input_b)
06 print("곱셈 결과:", input_a * input_b)
07 print("나눗셈 결과:", input_a / input_b)
```




실행결과

```
첫 번째 숫자> 273 Enter
두 번째 숫자> 52 Enter
덧셈 결과: 325.0
뺄셈 결과: 221.0
곱셈 결과: 14196.0
나눗셈 결과: 5.25
```

문자열을 숫자로 바꾸기

- ValueError 예외
 - 변환할 수 없는 것을 변환하려 할 경우
 - 숫자가 아닌 것을 숫자로 변환하려 할 경우

```
int("안녕하세요")  
float("안녕하세요")
```

 오류

```
Traceback (most recent call last):  
  File "intconvert.py", line 2, in <module>  
    int_a = int(string_a)  
ValueError: invalid literal for int() with base 10: '안녕하세요'
```

문자열을 숫자로 바꾸기

- 소수점이 있는 숫자 형식의 문자열을 int() 함수로 변환하려 할 때

```
int("52.273")
```

[오류]

```
Traceback (most recent call last):
```

```
File "intconvert.py", line 2, in <module>
```

```
    int_a = int(string_a)
```

```
ValueError: invalid literal for int() with base 10: '52.273'
```

숫자를 문자열로 바꾸기

- `str()` 함수
 - 숫자를 문자열로 변환

`str(다른 자료형)`

```
01 output_a = str(52)
02 output_b = str(52.273)
03 print(type(output_a), output_a)
04 print(type(output_b), output_b)
```

실행결과

```
<class 'str'> 52
<class 'str'> 52.273
```

키워드로 정리하는 핵심 포인트

- **변수 선언** : 변수를 생성하는 것을 의미
- **변수 할당** : 변수에 값을 넣는 것을 의미
- **변수 참조** : 변수에서 값을 꺼내는 것
- **input() 함수** : 명령 프롬프트에서 사용자로부터 데이터 입력 받음
- **int() 함수** : 문자열을 int 자료형으로 변환
- **float 함수** : 문자열을 float 자료형으로 변환
- **str() 함수** : 숫자를 문자열로 변환

확인문제

- 변수에 값을 할당하기 위한 구문입니다. 빈칸에 알맞은 기호를 쓰세요.

변수 이름 값

- 숫자에 적용할 수 있는 복합 대입 연산자입니다. 왼쪽 연산자 항목에 알맞은 기호를 써 보세요.

연산자	내용
<input type="text"/>	숫자 덧셈 후 대입
<input type="text"/>	숫자 뺄셈 후 대입
<input type="text"/>	숫자 곱셈 후 대입
<input type="text"/>	숫자 나눗셈 후 대입
<input type="text"/>	숫자 나머지 구한 후 대입
<input type="text"/>	숫자 제곱 후 대입

확인문제

- 다음 코드는 inch 단위의 자료를 입력 받아 cm를 구하는 예제입니다. 빈 칸에 알맞은 내용을 넣어 코드를 완성해 주세요. (1inch = 2.54cm)

```
str_input =  ("숫자 입력> ")  
num_input =  (str_input)  
  
print()  
print(num_input, "inch")  
print((num_input * 2.54), "cm")
```

실행결과 1

숫자 입력> 1

1.0 inch
2.54 cm

실행결과 2

숫자 입력> 26

26.0 inch
66.04 cm

확인문제

- 원의 반지름을 입력 받아 원의 둘레와 넓이를 구하는 코드입니다. 빈칸에 알맞은 내용을 넣어 코드를 완성해 주세요.
 - 둘레 : $2 * \text{원주율} * \text{반지름}$

```
str_input =  ("원의 반지름 입력> ")
num_input =  (str_input)
print()
print("반지름: ", num_input)
print("둘레: ", 2 * 3.14 * )
print("넓이: ", 3.14 *  ** 2)
```

실행결과 1

원의 반지름 입력> 2

반지름: 2.0
둘레: 12.56
넓이: 12.56

실행결과 2

원의 반지름 입력> 5

반지름: 5.0
둘레: 31.400000000000002
넓이: 78.5

3주

숫자와 문자열의 다양한 기능

시작하기 전에


[핵심 키워드] format(), upper(), lower(), strip(), find(), in연산자, split()

[핵심 포인트] 함수는 영어로 function, 즉 사람 또는 사물의 기능이라는 뜻을 가진 단어와 동음이의어다. 지금까지 살펴본 숫자나 문자열과 같은 자료도 컴퓨터에서는 하나의 사물처럼 취급되기에 내부적으로 여러 기능을 가지고 있다.

시작하기 전에

- 문자열 뒤에 **마침표** 입력해 보면 자동 완성 기능으로 다양한 자체 기능들이 제시됨

```
3  format_b = "파이썬 열공하여 첫 연봉 {}만 원 만들기".format(5000)
4  format_c = "{} {} {}".format(1, 2, 3)
5  format_d = "{} {} {}".format(1, 2, 3)
6  format_e = "{} {} {}".format(1, 2, 3)
7  format_f = "{} {} {}".format(1, 2, 3)
8
9  # 출력하기
10 print(format_a)
11 print(format_b)
12 print(format_c)
13 print(format_d)
14 print(format_e)
15 print(format_f)
```



문자열의 format() 함수

- format() 함수로 숫자를 문자열로 변환
 - 중괄호 포함한 문자열 뒤에 마침표 찍고 format() 함수 사용하되, 중괄호 개수와 format 함수 안 매개변수의 개수는 반드시 같아야 함
 - 문자열의 중괄호 기호가 format() 함수 괄호 안의 매개변수로 차례로 대치되면서 숫자가 문자열이 됨

```
"{}".format(10)
```

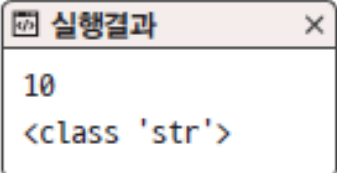
```
"{} {}".format(10, 20)
```

```
"{} {} {} {} {}".format(101, 202, 303, 404, 505)
```

문자열의 format() 함수

- 예시 - format() 함수로 숫자를 문자열로 변환하기

```
01  # format() 함수로 숫자를 문자열로 변환하기
02  string_a = "{}".format(10)
03
04  # 출력하기
05  print(string_a)
06  print(type(string_a))
```

A small window titled "실행결과" (Execution Result) with a close button. It displays the output of the Python code: the number 10 on the first line and its type, <class 'str'>, on the second line.

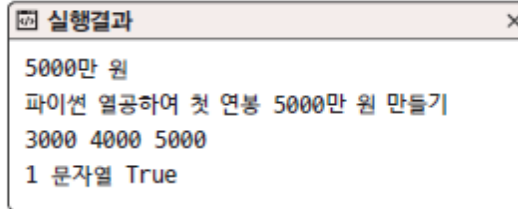
실행결과 ×

```
10
<class 'str'>
```

문자열의 format() 함수

- 예시 - format() 함수의 다양한 형태

```
01  # format() 함수로 숫자를 문자열로 변환하기
02  format_a = "{}만 원".format(5000)
03  format_b = "파이썬 열공하여 첫 연봉 {}만 원 만들기 ".format(5000)
04  format_c = "{} {} {}".format(3000, 4000, 5000)
05  format_d = "{} {} {}".format(1, "문자열", True)
06
07  # 출력하기
08  print(format_a)
09  print(format_b)
10  print(format_c)
11  print(format_d)
```



실행결과

```
5000만 원
파이썬 열공하여 첫 연봉 5000만 원 만들기
3000 4000 5000
1 문자열 True
```

- format_a : 중괄호 옆에 다른 문자열 넣음
- format_b : 중괄호 앞뒤로 다른 문자열 넣음
- format_c : 매개변수 여러 개 넣음

문자열의 format() 함수

- IndexError 예외
 - 종괄호 기호의 개수가 format() 함수의 매개변수 개수보다 많은 경우

```
>>> "{} {}".format(1, 2, 3, 4, 5)
'1 2'
>>> "{} {} {}".format(1, 2)
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    "{} {} {}".format(1, 2)
IndexError: tuple index out of range
```

format() 함수의 다양한 기능

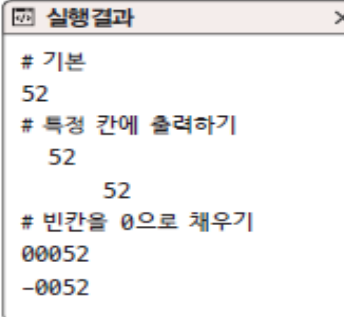
- 정수 출력의 다양한 형태
 - 예시 - 정수를 특정 칸에 출력하기

```
01  # 정수
02  output_a = "{:d}".format(52)
03
04  # 특정 칸에 출력하기
05  output_b = "{:5d}".format(52)      # 5칸
06  output_c = "{:10d}".format(52)     # 10칸
07
08  # 빈칸을 0으로 채우기
09  output_d = "{:05d}".format(52)     # 양수
10  output_e = "{:05d}".format(-52)    # 음수
11
12  print("# 기본")
13  print(output_a)
14  print("# 특정 칸에 출력하기")
15  print(output_b)
16  print(output_c)
17  print("# 빈칸을 0으로 채우기")
18  print(output_d)
19  print(output_e)
```

output_a : {:d}를 사용하여 int 자료형 정수 출력한
다는 것을 직접 지정

output_b, output_c : 특정 칸에 맞춰서 숫자를 출력
하는 형태

output_d, output_e : 빈칸을 0으로 채우는 형태

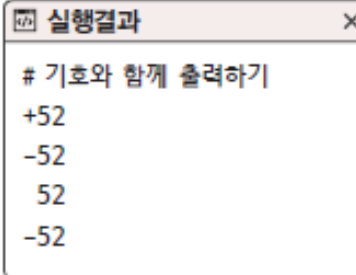


```
실행결과
# 기본
52
# 특정 칸에 출력하기
52
52
# 빈칸을 0으로 채우기
00052
-0052
```

format() 함수의 다양한 기능

- 예시 - 기호 붙여 출력하기

```
01  # 기호와 함께 출력하기
02  output_f = "{:+d}".format(52)  # 양수
03  output_g = "{:+d}".format(-52) # 음수
04  output_h = "{: d}".format(52)  # 양수: 기호 부분 공백
05  output_i = "{: d}".format(-52) # 음수: 기호 부분 공백
06
07  print("# 기호와 함께 출력하기")
08  print(output_f)
09  print(output_g)
10  print(output_h)
11  print(output_i)
```



실행결과

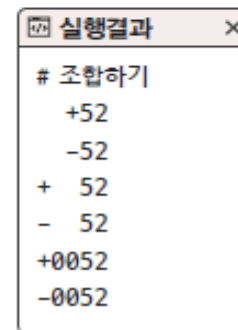
```
# 기호와 함께 출력하기
+52
-52
 52
-52
```

- `{:+d}` 앞에 + 기호 추가하면 양수의 경우 + 붙여줌
- `{: d}`처럼 앞에 공백두면 양수의 경우 기호 위치를 공백으로 비워줌

format() 함수의 다양한 기능

- 예시 - 조합

```
01  # 조합하기
02  output_h = "{:+5d}".format(52)      # 기호를 뒤로 밀기: 양수
03  output_i = "{:+5d}".format(-52)     # 기호를 뒤로 밀기: 음수
04  output_j = "{:=+5d}".format(52)     # 기호를 앞으로 밀기: 양수
05  output_k = "{:=+5d}".format(-52)    # 기호를 앞으로 밀기: 음수
06  output_l = "{:05d}".format(52)      # 0으로 채우기: 양수
07  output_m = "{:05d}".format(-52)     # 0으로 채우기: 음수
08
09  print("# 조합하기")
10  print(output_h)
11  print(output_i)
12  print(output_j)
13  print(output_k)
14  print(output_l)
15  print(output_m)
```



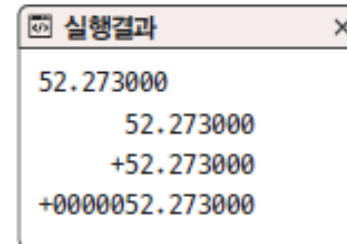
실행결과

```
# 조합하기
+52
-52
+ 52
- 52
+0052
-0052
```

format() 함수의 다양한 기능

- 부동 소수점 출력의 다양한 형태
 - 예시 - float 자료형 기본

```
01 output_a = "{:f}".format(52.273)
02 output_b = "{:15f}".format(52.273)    # 15칸 만들기
03 output_c = "{:+15f}".format(52.273)    # 15칸에 부호 추가하기
04 output_d = "{:+015f}".format(52.273)    # 15칸에 부호 추가하고 0으로 채우기
05
06 print(output_a)
07 print(output_b)
08 print(output_c)
09 print(output_d)
```



실행결과

```
52.273000
52.273000
+52.273000
+0000052.273000
```

format() 함수의 다양한 기능

- 예시 - 소수점 아래 자릿수 지정하기

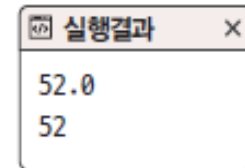
```
01 output_a="{:15.3f}".format(52.273)
02 output_b="{:15.2f}".format(52.273)
03 output_c="{:15.1f}".format(52.273)
04
05 print(output_a)
06 print(output_b)
07 print(output_c)
```

실행결과	
	52.273
	52.27
	52.3

format() 함수의 다양한 기능

- 의미 없는 소수점 제거하기
 - 예시 - { :g}

```
01 output_a = 52.0
02 output_b = "{:g}".format(output_a)
03 print(output_a)
04 print(output_b)
```



실행결과
52.0
52

대소문자 바꾸기 : upper()와 lower()

- upper() 함수
 - 문자의 알파벳을 대문자로 바꿈
- lower() 함수
 - 문자의 알파벳을 소문자로 바꿈

```
>>> a = "Hello Python Programming...!"  
>>> a.upper()  
'HELLO PYTHON PROGRAMMING...!'
```

```
>>> a.lower()  
'hello python programming...!'
```


문자열 양옆의 공백 제거하기: strip()

- strip() 함수
 - 문자열 양옆의 공백을 제거
- lstrip() 함수
 - 왼쪽의 공백을 제거
- rstrip() 함수
 - 오른쪽의 공백을 제거

문자열 양옆의 공백 제거하기: strip()

- 의도하지 않은 줄바꿈 등의 제거

```
>>> input_a = """
안녕하세요
문자열의 함수를 알아봅니다
"""
```

```
>>> print(input_a)
```

```

안녕하세요
문자열 함수를 알아봅니다
```

```
>>> print(input_a.strip())
```

```
안녕하세요
문자열 함수를 알아봅니다
```

문자열의 구성 파악하기 : isOO()

- 문자열이 소문자로만, 알파벳으로만, 혹은 숫자로만 구성되어 있는지 확인
 - `isalnum()`: 문자열이 알파벳 또는 숫자로만 구성되어 있는지 확인합니다.
 - `isalpha()`: 문자열이 알파벳으로만 구성되어 있는지 확인합니다.
 - `isidentifier()`: 문자열이 식별자로 사용할 수 있는 것인지 확인합니다.
 - `isdecimal()`: 문자열이 정수 형태인지 확인합니다.
 - `isdigit()`: 문자열이 숫자로 인식될 수 있는 것인지 확인합니다.
 - `isspace()`: 문자열이 공백으로만 구성되어 있는지 확인합니다.
 - `islower()`: 문자열이 소문자로만 구성되어 있는지 확인합니다.
 - `isupper()`: 문자열이 대문자로만 구성되어 있는지 확인합니다.

문자열의 구성 파악하기 : is00()

- 불 (boolean)
 - 출력이 True 혹은 False로 나오는 것

```
>>> print("TrainA10".isalnum())
True
>>> print("10".isdigit())
True
```

문자열 찾기: find()와 rfind()

- find()
 - 왼쪽부터 찾아서 처음 등장하는 위치 찾을
- rfind()
 - 오른쪽부터 찾아서 처음 등장하는 위치 찾을

```
>>> output_a = "안녕안녕하세요".find("안녕")
>>> print(output_a)
0
```

```
>>> output_b = "안녕안녕하세요".rfind("안녕")
>>> print(output_b)
2
```

문자열과 in 연산자

- in 연산자

- 문자열 내부에 어떤 문자열이 있는지 확인할 때 사용
- 결과는 True(맞다), False(아니다)로 출력

```
>>> print("안녕" in "안녕하세요")  
True
```

```
>>> print("잘자" in "안녕하세요")  
False
```

문자열 자르기 : split()

- split() 함수
 - 문자열을 특정한 문자로 자름

```
>>> a = "10 20 30 40 50".split(" ")  
>>> print(a)  
['10', '20', '30', '40', '50']
```

- 실행 결과는 리스트 (list)로 출력

키워드로 정리하는 핵심 포인트

- **format() 함수** : 숫자와 문자열을 다양한 형태로 출력
- **upper() 및 lower() 함수** : 문자열의 알파벳을 대문자 혹은 소문자로 변경
- **strip() 함수** : 문자열 양옆의 공백 제거
- **find() 함수** : 문자열 내부에 특정 문자가 어디에 위치하는지 찾을 때 사용
- **in 연산자** : 문자열 내부에 어떤 문자열이 있는지 확인할 때 사용
- **split() 함수** : 문자열을 특정한 문자로 자를 때 사용

확인문제

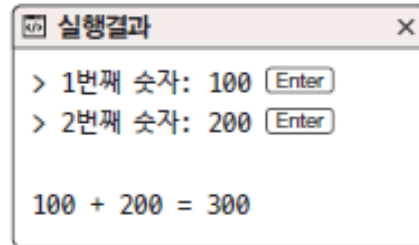
- 함수와 그 기능을 연결해 보세요.

- | | | |
|-----------|---|------------------------|
| ① split() | • | • ㉑ 문자열을 소문자로 변환합니다. |
| ② upper() | • | • ㉒ 문자열을 대문자로 변환합니다. |
| ③ lower() | • | • ㉓ 문자열 양옆의 공백을 제거합니다. |
| ④ strip() | • | • ㉔ 문자열을 특정 문자로 자릅니다. |

- 다음 코드의 빈칸을 채워서 실행결과처럼 출력해 보세요.

```
a = input("> 1번째 숫자: ")
b = input("> 2번째 숫자: ")
print()

print("{} + {} = {}".format( , ))
```



```
실행결과 x
> 1번째 숫자: 100 Enter
> 2번째 숫자: 200 Enter

100 + 200 = 300
```