



숫자와 문자열의 다양한 기능

목차

- 시작하기 전에
- 문자열의 `format()` 함수
- `format()` 함수의 다양한 기능
- 대소문자 바꾸기 : `upper()`와 `lower()`
- 문자열 양 옆의 공백 제거하기: `strip()`
- 문자열의 구성 파악하기 : `isOO()`
- 문자열 찾기: `find()`와 `rfind()`
- 문자열 자르기 : `split()`
- 키워드로 정리하는 핵심 포인트
- 확인문제

[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.

- 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)
```

capitalize
casefold
center
count
encode
endswith
expandtabs
find
format
format_map
index
isalnum

- 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

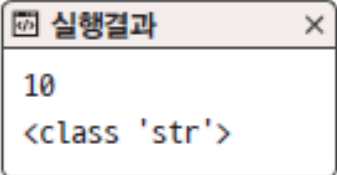
```
"{}".format(10)
```

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

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

- 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))
```

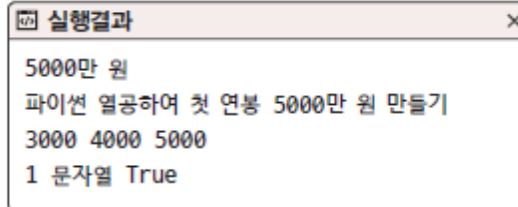


실행결과

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

– 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

- IndexError 예외
 - 중괄호 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
```

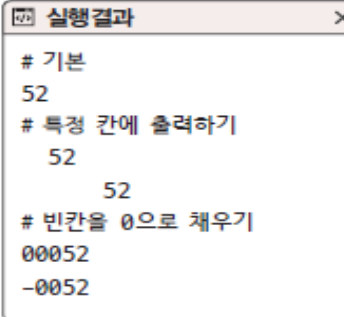

- 정수 출력의 다양한 형태
 - 예시 - 정수를 특정 칸에 출력하기

```
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 출력한다는 것을 직접 지정

output_b, output_c : 특정 칸에 맞춰서 숫자를 출력하는 형태

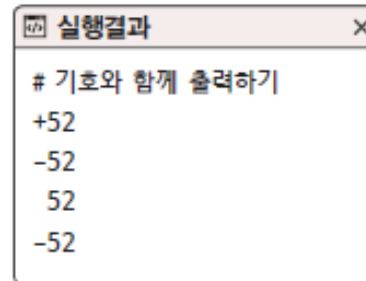
output_d, output_e : 빈칸을 0으로 채우는 형태



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

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}처럼 앞에 공백두면 양수의 경우 기호 위치를 공백으로 비워줌

- 예시 - 조합

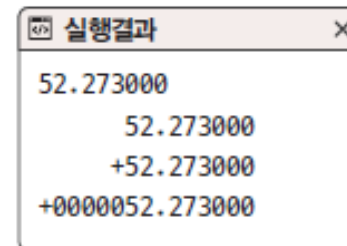
```
01  # 조합하기
02  output_h = "{:+5d}".format(52)      # 기호를 뒤로 밀기: 양수
03  output_i = "{:+5d}".format(-52)     # 기호를 뒤로 밀기: 음수
04  output_j = "{:=+5d}".format(52)     # 기호를 앞으로 밀기: 양수
05  output_k = "{:=+5d}".format(-52)    # 기호를 앞으로 밀기: 음수
06  output_l = "{:05d}".format(52)      # 0으로 채우기: 양수
07  output_m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

- 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
```

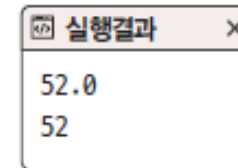
예시 - 소수점 아래 자릿수 지정하기

```
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

- 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...!'
```

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

- 의도하지 않은 줄바꿈 등의 제거

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

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

```

    안녕하세요
문자열 함수를 알아보니다
```

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

```
안녕하세요
문자열 함수를 알아보니다
```

- 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.

- 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 찾기
- rfind()
 - 오른쪽부터 찾아서 처음 등장하는 위치 찾기

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

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

- 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
```

- split() 함수
 - 문자열을 특정한 문자로 자름

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

- **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

- 함수와 그 기능을 연결해 보세요.

① split() •

② upper() •

③ lower() •

④ strip() •

• ㉠ 문자열을 소문자로 변환합니다.

• ㉡ 문자열을 대문자로 변환합니다.

• ㉢ 문자열 양옆의 공백을 제거합니다.

• ㉣ 문자열을 특정 문자로 자릅니다.

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

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

실행결과

> 1번째 숫자: 100

> 2번째 숫자: 200

100 + 200 = 300