

제3강 Functions

학습 목차

- 함수
- 변수 scope – local vs global
- 다중 리턴
- 다형성
- 예외 처리
- 주요 내장 함수들

함수(function)

- 어떤 특정한 일을 처리하는 기능을 모아놓은 것, 수학적인 함수도 구현 가능.
- 일반적으로 라이브러리, 모듈은 여러 개의 함수들로 구성됨.
- 함수의 이름은 그 함수의 기능을 정확히 나타내는 것이 좋음.

```
def 함수명(매개변수):  
    <수행할 문장1>  
    <수행할 문장2>  
    ...
```

함수 정의

함수 이름

인수: 외부에서 전달되는 값

```
def add(a, b):  
    sum = a + b  
    return sum
```

들여쓰기(indentation)

*** 매우 중요 ***

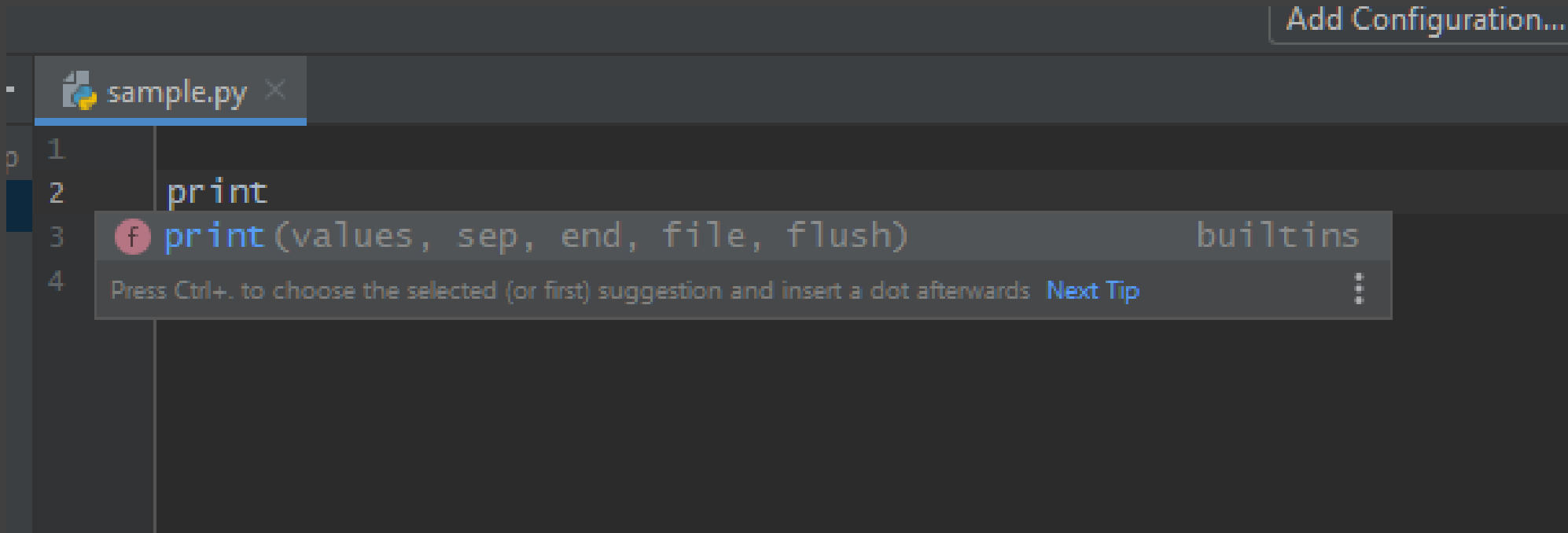
함수가 호출될 때 실행됨.

return 없는 함수값? None

```
>>> spam = print('Hello!')  
Hello!  
>>> None == spam  
True
```

키워드 인자(Keyword Arguments)

- 함수 호출 시, 인자를 선택해서 값을 넘겨주는 방식.
- 기본 방식 - Positional Arguments



```
print('Hello', end=' ')\nprint('World')
```

```
>>> print('cats', 'dogs', 'mice', sep=',')\ncats,dogs,mice
```

Local / Global Scope

- 함수 안에서 만들어진 변수는 Local Scope에 저장.
- Scope 은 변수를 담아놓는 컨테이너 박스. Scope 이 없어지면 변수도 사라짐.
- Global Scope 은 프로그램 실행 내내 유지됨.
- Global Scope의 code에서는 함수 내의 local variable 을 액세스할 수 없음.
- Local scope에서 다른 local scope 내의 variable을 액세스할 수 없음.
- Local Scope에서는 global scope의 variable 을 액세스할 수 있음.
 - global 문법 사용.
- Global scope과 local scope과 같은 이름의 변수도 사용 가능. 그러나, 해당 scope에 있는 것만 이용할 수 있음.

- **Global Scope의 code에서는 함수 내의 local variable 을 액세스할 수 없음.**

```
def spam():  
    eggs = 31337  
  
spam()  
print(eggs)
```

Local scope에서 다른 local scope 내의 variable을 액세스할 수 없음.

```
def spam():  
    eggs = 99  
    bacon()  
    print(eggs)
```

```
def bacon():  
    ham = 101  
    eggs = 0
```

```
spam()
```

Local scope에서는 global scope 의 변수를 "READ"할 수 있음.

```
def spam():  
    print(eggs)  
eggs = 42  
spam()  
print(eggs)
```

같은 이름의 변수가 Local 및 Global Scope에 존재할 수 있음. 그러나 각자 독립적으로 존재.

```
def spam():
    eggs = 'spam local'
    print(eggs)    # prints 'spam local'

def bacon():
    eggs = 'bacon local'
    print(eggs)    # prints 'bacon local'
    spam()
    print(eggs)    # prints 'bacon local'

eggs = 'global'
bacon()
print(eggs)        # prints 'global'
```

Local scope에서는 global scope 의 변수를 “WRITE” 할 수 있음.
단, global 로 설정해야 함.

```
def spam():  
    global eggs  
    eggs = 'spam'  
  
eggs = 'global'  
spam()  
print(eggs)
```

퀴즈

```
def spam():  
    global eggs  
    eggs = 'spam'  
  
def bacon():  
    eggs = 'bacon'  
  
def ham():  
    print(eggs)  
  
eggs = 42  
spam()  
print(eggs)  
bacon()  
ham()
```

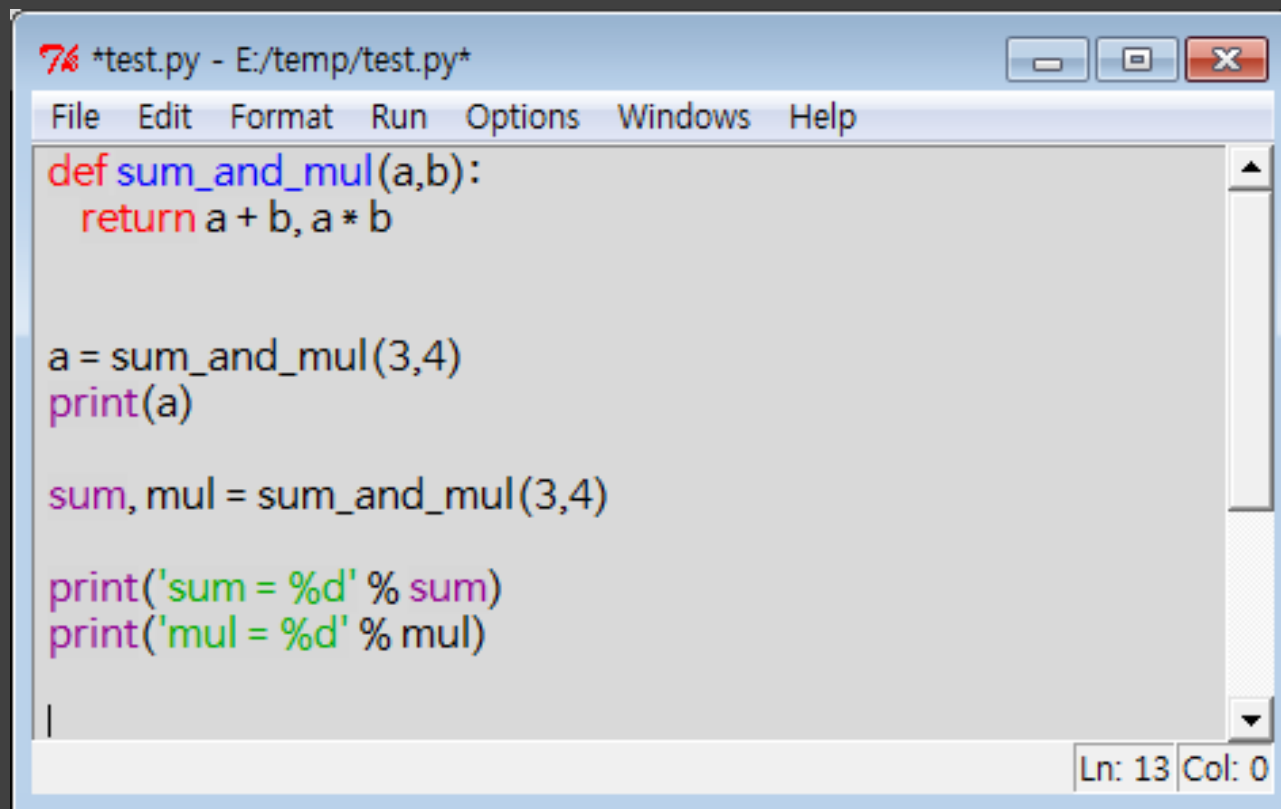
```
def spam():  
    print(eggs)
```

```
eggs = 'global'  
spam()
```

```
def spam():  
    print(eggs)  
    eggs = 'spam local'
```

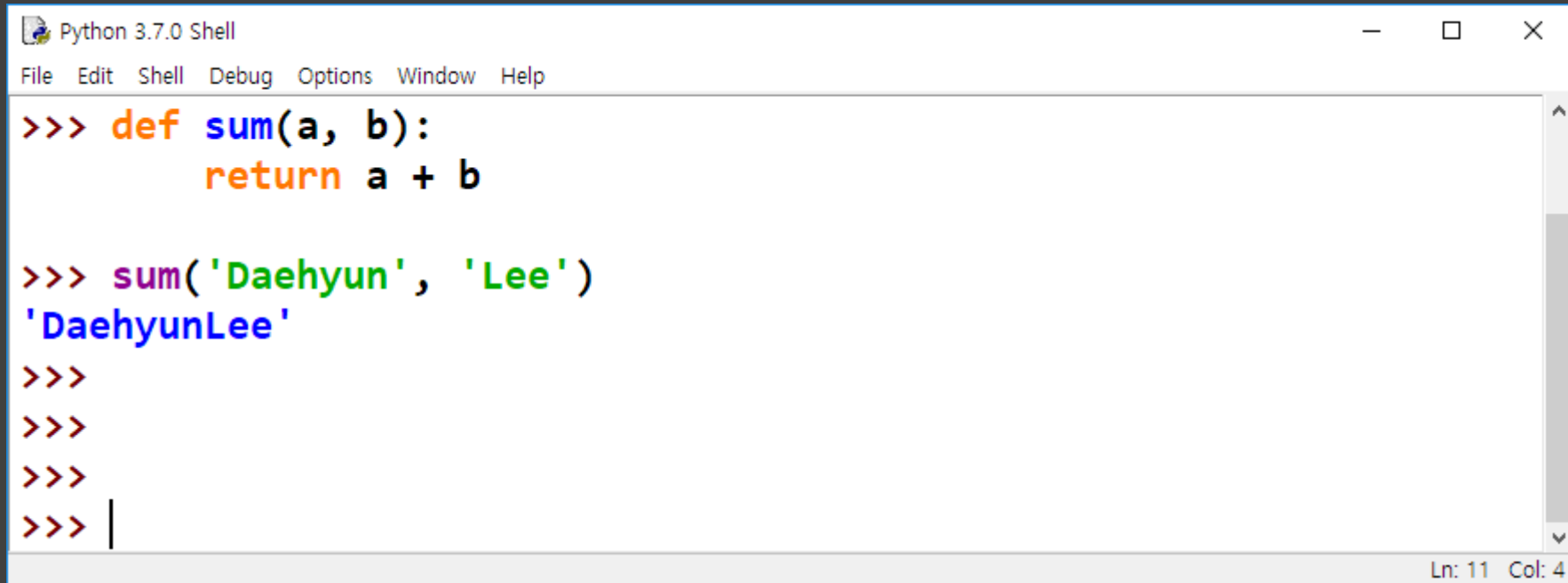
```
eggs = 'global'  
spam()
```

여러 개의 return 값 가능



```
*test.py - E:/temp/test.py*  
File Edit Format Run Options Windows Help  
def sum_and_mul(a,b):  
    return a + b, a * b  
  
a = sum_and_mul(3,4)  
print(a)  
  
sum, mul = sum_and_mul(3,4)  
  
print('sum = %d' % sum)  
print('mul = %d' % mul)  
  
Ln: 13 Col: 0
```


다형성(Polymorphism)



A screenshot of a Python 3.7.0 Shell window. The window has a title bar with the text 'Python 3.7.0 Shell' and standard window controls (minimize, maximize, close). Below the title bar is a menu bar with the following items: File, Edit, Shell, Debug, Options, Window, and Help. The main area of the window contains a Python code snippet demonstrating polymorphism. The code defines a function `sum(a, b)` that returns the concatenation of `a` and `b`. This function is then called with two different types of arguments: strings and integers. The output shows that the function works for both types, returning the concatenated string and the sum of the integers respectively. The code is as follows:

```
>>> def sum(a, b):  
    return a + b  
  
>>> sum('Daehyun', 'Lee')  
'DaehyunLee'  
>>>  
>>>  
>>>  
>>> |
```

The status bar at the bottom right of the window shows 'Ln: 11 Col: 4'.

Exception Handling

```
def spam(divideBy):  
    return 42 / divideBy  
  
print(spam(2))  
print(spam(12))  
print(spam(0))  
print(spam(1))
```

```
def spam(divideBy):  
    try:  
        return 42 / divideBy  
    except ZeroDivisionError:  
        print('Error: Invalid argument.')
```



```
print(spam(2))  
print(spam(12))  
print(spam(0))  
print(spam(1))
```

```
def spam(divideBy):  
    return 42 / divideBy  
  
try:  
    print(spam(2))  
    print(spam(12))  
    print(spam(0))  
    print(spam(1))  
except ZeroDivisionError:  
    print('Error: Invalid argument.')
```

		내장 함수		
<code>abs()</code>	<code>delattr()</code>	<code>hash()</code>	<code>memoryview()</code>	<code>set()</code>
<code>all()</code>	<code>dict()</code>	<code>help()</code>	<code>min()</code>	<code>setattr()</code>
<code>any()</code>	<code>dir()</code>	<code>hex()</code>	<code>next()</code>	<code>slice()</code>
<code>ascii()</code>	<code>divmod()</code>	<code>id()</code>	<code>object()</code>	<code>sorted()</code>
<code>bin()</code>	<code>enumerate()</code>	<code>input()</code>	<code>oct()</code>	<code>staticmethod()</code>
<code>bool()</code>	<code>eval()</code>	<code>int()</code>	<code>open()</code>	<code>str()</code>
<code>breakpoint()</code>	<code>exec()</code>	<code>isinstance()</code>	<code>ord()</code>	<code>sum()</code>
<code>bytearray()</code>	<code>filter()</code>	<code>issubclass()</code>	<code>pow()</code>	<code>super()</code>
<code>bytes()</code>	<code>float()</code>	<code>iter()</code>	<code>print()</code>	<code>tuple()</code>
<code>callable()</code>	<code>format()</code>	<code>len()</code>	<code>property()</code>	<code>type()</code>
<code>chr()</code>	<code>frozenset()</code>	<code>list()</code>	<code>range()</code>	<code>vars()</code>
<code>classmethod()</code>	<code>getattr()</code>	<code>locals()</code>	<code>repr()</code>	<code>zip()</code>
<code>compile()</code>	<code>globals()</code>	<code>map()</code>	<code>reversed()</code>	<code>__import__()</code>
<code>complex()</code>	<code>hasattr()</code>	<code>max()</code>	<code>round()</code>	

any(), all()

- 집합 데이터에 대한 조건을 확인

```
import random

ages = [random.randint(1,100) for n in range(20)]

if any(i < 18 for i in ages):
    print('모두 다 성인임.')
else:
    print('미성년자 있음.')

print('모두 다 성인임.') if all(i >= 18 for i in ages) else print('모두 다 성인임.')
```

filter()

- 특정 조건을 만족하는 데이터만을 골라낼 수 있음.

```
numbers = [random.randint(0, 99) for i in range(20)]

def mul3_filter(n):
    return n % 3 == 0
result = list(filter(mul3_filter, numbers))

result = list(filter(lambda n: n % 3 == 0, numbers))
```

map()

- 집합 데이터들을 한꺼번에 변환

```
# Define a function to square a number
def square(x):
    return x ** 2

# Create a list of numbers
numbers = [1, 2, 3, 4, 5]

# Use map() to apply the square() function to each element of the list
squares = map(square, numbers)

# Convert the map object to a list
squares_list = list(squares)

# Print the result
print(squares_list)
```


max,min,sum

- 집합적 데이터에 대해서 잘 적용됨.

```
min(1, 2, 3, 4)
```

```
max([100, 2, 2, 3])
```

```
sum([n for n in range(1, 100+1)])
```

zip()

- 배열 요소들을 차례로 결합
- 한번 사용하면 소진됨.

```
names = ["Alice", "Bob", "Charlie"]
ages = [25, 30, 35]
heights = [180.5, 172.1, 185.3]
zipped = zip(names, ages, heights)
print(list(zipped))
print(list(zipped))
```