Python 과학 프로그래밍 기초

6. 함수 (2)

한림대학교 소프트웨어융합대학

박섭형

2021년 1학기

[1]:
```
lst = [1, 3, 5]
sum(lst)
```

[1]: 9

[2]:
```
help(sum)
```

```
Help on built-in function sum in module builtins:

sum(iterable, /, start=0)
    Return the sum of a 'start' value (default: 0) plus an iterable of numbers

    When the iterable is empty, return the start value.
    This function is intended specifically for use with numeric values and may
    reject non-numeric types.
```

[3]:
```
sum(lst, 10)
```

[3]: 19

[4]:
```
sum(lst, start=10)
```

[4]: 19

[5]:
```
def my_sum(iterable, init=0):
    total = init
```

```
    for a in iterable:
        total += a
    return total
```

[6]:
```
my_sum(lst)
```

[6]: 9

[7]:
```
my_sum(lst, 10)
```

[7]: 19

[8]:
```
my_sum(lst, init=10)
```

[8]: 19

[9]:
```
def my_sum_2(lst, *, init=0):
    total = init
    for a in lst:
        total += a
    return total
```

[10]:
```
my_sum_2(lst)
```

[10]: 9

[11]:
```
my_sum_2(lst, init=10)
```

[11]: 19

[12]:
```
my_sum_2(lst, 10)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-12-6cfc123a4776> in <module>
----> 1 my_sum_2(lst, 10)

TypeError: my_sum_2() takes 1 positional argument but 2 were given
```

가변 길이 Argument

인수의 최댓값을 반환하는 함수

[13]:
```
max([5, 3], [4, 5], [1, 3])
```

[13]: [5, 3]

[14]:
```
max((2, 3), (4, 5))
```

[14]: (4, 5)

[15]:
```
max([2, 3])
```

[15]: 3

[16]:
```
max(2, 3)
```

[16]: 3

[17]:
```
max(2, 5, 3)
```

[17]: 5

[18]:
```
def my_max(a, b):
    if a > b:
        return a
    else:
        return b

print(my_max(3, 5))
```

```
5
```

[19]:
```
def dummy(a, b):
    print(a)
    print(b)

dummy(3, 5)
```

```
3
5
```

함수의 positional parameter에 packing operator *를 이용하는 방법

[20]:
```
def dummy(a, *b):
    print(a)
    print(b)

dummy(3, 5)
```

```
3
(5,)
```

[21]:
```
def dummy(a, *b):
    print(a)
    print(b)

dummy(3, 5, 8, 5, 9)
```

```
3
(5, 8, 5, 9)
```

[22]:
```
def my_max(*iterator):
    if len(iterator) < 1:
        return None
    else:
        _max = iterator[0]
        for e in iterator[1:]:
            if e > _max:
                _max = e
        return _max

print(f"The maximum value is {my_max()}")
print(f"The maximum value is {my_max(3)}")
print(f"The maximum value is {my_max(3, 5)}")
print(f"The maximum value is {my_max(3, 5, 6)}")
print(f"The maximum value is {my_max(3, 5, 6, 7)}")
```

```
The maximum value is None
The maximum value is 3
```

```
The maximum value is 5
The maximum value is 6
The maximum value is 7
```

변수의 Scope

- 변수에 접근할 수 있는 코딩 영역
- local scope
  - 함수 내에서 생성된 변수의 scope는 그 함수 내부이다.
- global scope
  - Main 바디에서 생성된 변수의 scope는 메인 바디뿐 아니라 모든 함수 내에서 접근 가능하다.
- Built-in scope

[23]:
```
import turtle
t = turtle.Turtle()

def forward_and_right(distance):
    t.forward(distance)
    t.right(90)

def draw_a_rectangle(side):
    """     side
    parameter:
      - side:
    """
    for _ in range(4):
        forward_and_right(side)

draw_a_rectangle(100)
draw_a_rectangle(120)



turtle.exitonclick()
```

[24]:
```
def add_two_variables(x, y):
    z = x + y
```

```
    print(x, y, z,  )
    print(id(x), id(y), z, a, b)

    return z

# main body
a = 1000
b = 2000
print(id(a), id(b))
c = add_two_variables(a, b)
print(z)
```

```
2477291899920 2477291897424
1000 2000 3000
2477291899920 2477291897424 3000 1000 2000
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-24-8e518ac12c51> in <module>
     11 print(id(a), id(b))
     12 c = add_two_variables(a, b)
---> 13 print(z)

NameError: name 'z' is not defined
```

[25]:
```
def add_two(x, y):
    # local scope of add_twp function
    z = x + y            # z:     (local variable)
    print(x, y, z, a, b) # a, b:     (global variable)
    return z

# main body --- global scope
a = 1000
b = 2000
c = add_two(a, b)
```

```
print(c)
#print(z)
```

```
1000 2000 3000 1000 2000
3000
```

[26]:
```
def add_two():
    global a
    z = x + y
    a = a + 1
    print(x, y, z, w, a, b)
    return z

# main body
x = 5000
y = 6000
w = 7000
a = 1000
b = 2000
c = add_two()
print(x, y, a, b, c)
#print(z)
```

```
5000 6000 11000 7000 1001 2000
5000 6000 1001 2000 11000
```

Scope을 기준으로 한 변수의 분류

- 전역 변수 (global variable)
- 지역 변수 (local variable)
- 비지역 변수 (nonlocal variable)

[27]:
```
def outer_function(x, y):
    """ x**2 + y + 10
    """
    def add_two(x, y):
        nonlocal u
```

```
        u = u + 1
        z = x + y
        print("In add_two:", x, y, z, u)
        return z
    x = x**2
    u = 10
    y = y + u
    v = add_two(x, y)
    print("In outer_function:", x, y, u, v)
    return v

# main body
x = 2
y = 6
z = 7
u = 1000
res = outer_function(x, y)
print("In main body:", x, y, z, u, res)
```

```
In add_two: 4 16 20 11
In outer_function: 4 16 11 20
In main body: 2 6 7 1000 20
```