

Web/Python Programming

웹/파이썬 프로그래밍

```
23 <?php language_attributes(); ?>
24 <?php bloginfo( 'charset' ); ?>
25 <?php wp_title( '|', true, 'right' ); ?>
26 <?php wp_title( 'profile' href="http://gmpg.org/xfn/11" ?>
27 <?php fruitful_get_favicon(); ?>
28 <?php fruitful_get_theme_options(); ?>
29 <?php fruitful_get_theme_options(); ?>
30 <?php fruitful_get_theme_options(); ?>
31 <?php fruitful_get_theme_options(); ?>
32 <?php fruitful_get_theme_options(); ?>
33 <?php fruitful_get_theme_options(); ?>
34 <?php fruitful_get_theme_options(); ?>
35 <?php fruitful_get_theme_options(); ?>
```

Today

- Review functions (Ch. 3)
- Designing new functions – A recipe

Question

```
>>> 0.1+0.2
0.30000000000000004
>>> round(4.5)
4
>>>
```



- Read: <https://docs.python.org/3.7/tutorial/floatingpoint.html>
- Floating-point numbers are represented in computer hardware as base 2 (binary) fractions.
- Decimal fraction vs. binary fraction

Functions

- In mathematics

$$y = f(x) = x^2 + 3x + 2$$

$$f(2) = 12$$

$$z = f(x, y) = x^2y + 4x + 1$$

$$f(2, 3) = 21$$

- Python build-in functions

```
>>> abs(-9)
```

```
0
```

```
>>> pow(3, 2)
```

```
9
```

```
>>> round(4.3)
```

```
4
```

```
>>> pow(abs(-2), round(4.3))
```

```
16
```

```
>>> round(-3.5)
```

```
-4
```

Typecast

- Functions that convert from one type to another

```
>>> int(34.6)
```

```
34
```

```
>>> int(-4.3)
```

```
-4
```

```
>>> float(21)
```

```
21.0
```

help()

```
>>> help(abs)
Help on built-in function abs in module builtins:
```

```
abs(x, /)
    Return the absolute value of the argument.
```

```
>>> help(pow)
Help on built-in function pow in module builtins:
```

```
pow(x, y, z=None, /)
    Equivalent to x**y (with two arguments) or x**y % z (with three arguments)
```

Some types, such as ints, are able to use a more efficient algorithm when invoked using the three argument form.

```
>>> help(round)
Help on built-in function round in module builtins:
```

```
round(...)
    round(number[, ndigits]) -> number
```

Round a number to a given precision in decimal digits (default 0 digits). This returns an int when called with one argument, otherwise the same type as the number. ndigits may be negative.

```
>>>
```

```
>>> pow(2,4)
```

```
16
```

```
>>> pow(2,4,3)
```

```
1
```

```
>>> round(3.141592)
```

```
3
```

```
>>> round(3.141592,2)
```

```
3.14
```

Defining your own functions

```
>>> convert_to_celsius(212)
100.0
```

```
>>> convert_to_celsius(212)
Traceback (most recent call last):
  File "<pyshell#48>", line 1, in <module>
    convert_to_celsius(212)
NameError: name 'convert_to_celsius' is not defined
```

```
>>> def convert_to_celsius (fahrenheit):
    return (fahrenheit - 32) * 5/9
```

Local variables

The diagram illustrates the components of a Python function definition and its usage. It shows a code block with the following content:

```
>>> def quadratic(a,b,c,x):  
    first = a * x ** 2  
    second = b * x  
    third = c  
    return first + second + third  
  
>>> quadratic (2,3,4,2)  
18  
  
>>> quadratic (2,3,4,1.0)  
9.0  
>>> |
```

Annotations with arrows point to specific parts of the code:

- Function name:** Points to the word `quadratic` in the function definition.
- Function parameters:** Points to the parameters `(a,b,c,x)` in the function definition.
- Function header:** Points to the entire first line of the function definition: `>>> def quadratic(a,b,c,x):`.
- Function body:** A bracket indicates the lines of code inside the function: `first = a * x ** 2`, `second = b * x`, `third = c`, and `return first + second + third`.
- Function call:** Points to the first call: `>>> quadratic (2,3,4,2)`.
- Function call:** Points to the second call: `>>> quadratic (2,3,4,1.0)`.

- Local variables are created within a function
- `first`, `second`, `third` are local variables of the function `quadratic`
- Function parameters are also local variables

Errors

- Number of parameters
- Redefinition is ok
- Local variables

```
>>> def quadratic(a,b,c,x):
    first = a * x ** 2
    second = b * x
    third = c
    return first + second + third

>>> quadratic (2,3,4,2)
18
>>> quadratic (2,3,4,1.0)
9.0
>>> quadratic ( 2,3,4)
Traceback (most recent call last):
  File "<pyshell#68>", line 1, in <module>
    quadratic ( 2,3,4)
TypeError: quadratic() missing 1 required positional argument: 'x'

>>> def quadratic (a,b,x):
    first = a * x **2
    second = b * x
    return first + second

>>> quadratic ( 2,3,4,2)
Traceback (most recent call last):
  File "<pyshell#75>", line 1, in <module>
    quadratic ( 2,3,4,2)
TypeError: quadratic() takes 3 positional arguments but 4 were given

>>> quadratic(2,3,2)
14
>>> first
Traceback (most recent call last):
  File "<pyshell#77>", line 1, in <module>
    first
NameError: name 'first' is not defined
```

Designing a new function

- Writing a good essay

- A topic
- Background material
- An outline
- Filling in the outline with details

- Writing a good function

- An idea
- A name
- Parameters
- A return value
- Function body (the details)

Docstring

- Documentation string
- For humans to read
 - For yourself
 - For co-workers
 - For sharing

```
def days_difference(day1, day2):  
    """ (int, int) -> int
```

```
    Return the number of days between day1 and day2,  
    which are both in the range 1-365  
    (thus indicating the day of the year).
```

```
>>> days_difference(200, 224)  
24  
>>> days_difference(50, 50)  
0  
>>> days_difference(100, 99)  
-1  
"""  
return day2 - day1
```

Docstring

Function header → `def days_difference(day1, day2):`

Start of the docstring → `""" (int, int) -> int`

1) Types of (Parameters) -> Return value

2) Function description

Return the number of days between day1 and day2,
which are both in the range 1-365
(thus indicating the day of the year).

3) Example calls

`>>> days_difference(200, 224)`

`24`

`>>> days_difference(50, 50)`

`0`

`>>> days_difference(100, 99)`

`-1`

End of the docstring

`"""`

Function body

`return day2 - day1`

Function design recipe

■ Examples

- What arguments/parameters to give
- What information it will return
- Pick a function name

■ Type contract

- Types of parameters and return value

■ Header

- Give parameters names

■ Description

■ Body

■ Test

```
def days_difference(day1, day2):  
    """ (int, int) -> int
```

Return the number of days between day1 and day2,
which are both in the range 1-365
(thus indicating the day of the year).

```
>>> days_difference(200, 224)  
24  
>>> days_difference(50, 50)  
0  
>>> days_difference(100, 99)  
-1  
"""  
return day2 - day1
```