

Web/Python Programming

웹/파이썬 프로그래밍

Today

- Review functions (Ch. 3)
- Function parameters and return value
- Designing new functions – A recipe

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. It shows a function definition and two function calls. 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 parentheses and their contents `(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)`.

```
>>> 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  
>>> |
```

- 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
```


Let's use Editor Window from now on

- To better understand the return value of the function

Editor

```
def sum(a,b):  
    return a+b  
  
result = sum(3,4)
```

Shell

```
RESTART:  
C:/Users/jiyoung/AppD  
ata/Local/Programs/Py  
thon/Python35/Scripts  
/test01.py
```

Function with no parameters

- Function with no input
- Just execution

```
def say( ):  
    return 'Hello'
```

```
say( )
```

Function with no parameters

- Function with no input
- Just execution

```
def say():  
    return 'Hello'  
  
print(say())
```

Function with no parameters

- Function with no input
- Just execution

```
def say():  
    return 'Hello'
```

```
word = say()  
print(word)
```

Function with no return value

- Every function has only one return value
- If the return value is not defined in the function definition, the return value is `None`

```
def say():  
    print('Hello')
```

```
say()
```

Function with no return value

- Every function has only one return value
- If the return value is not defined in the function definition, the return value is `None`

```
def say():  
    print('Hello')  
  
print(say())
```


Function with no return value

- Every function has only one return value
- If the return value is not defined in the function definition, the return value is `None`

```
def say():  
    print('Hello')
```

```
word = say()  
print(word)
```

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

`Return the number of days between day1 and day2,`

2) Function description

→ `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
```

Birthday problem

- Which day of the week will a birthday fall upon, given what day of the week it is today and what day of the year the birthday is on?
- Final return value: Which day of the week will a birthday be?
 - Ex) The birthday is on Thursday!
- Given,
 - What day of the week it is today (Ex. Today is Thursday)
 - What day of the year it is today (Ex. Today is the 72nd day of the year)
 - What day of the year the birthday is on? (Ex. The birthday (Nov 1st) is the 306th day of the year)

Representation by numbers

- What day of the week

Day of the week	Number
Sunday	1
Monday	2
Tuesday	3
Wednesday	4
Thursday	5
Friday	6
Saturday	7

- What day of the year

Day of the year	Number
January 1 st	1
February 1 st	32
March 12 th	72
May 1 st	122
July 1 st	183
September 1 st	245
November 1 st	306
December 1 st	336
December 31 st	366

Problem design

- Three functions
- A function to get the day of the week of a birthday given today's day of the week, today's day of the year, and the birthday's day of the year
- A function to calculate the number between two days
- A function to get the day of the week after some days from another day of the week

Function design recipe

- Examples
 - Type contract
 - Header
 - Description
 - Body
 - Test
-
- `help(days_difference)`

```
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
```

FDR2: What day will it be in the future?

- Examples
- Type contract
- Header
- Description
- Body
- Test

```
def get_weekday(current_weekday, days_ahead):
```

```
    """(int, int) -> int
```

```
    Return which day of the week it will be days_ahead days from
    current_weekday.
```

```
    current_weekday is the current day of the week and is in
    the range 1-7, indicating whether today is Sunday (1),
    Monday (2), ..., Saturday (7).
```

```
    days_ahead is the number of days after today.
```

```
>>> get_weekday(3,2)
5
```

```
>>> get_weekday(6,1)
7
```

```
>>> get_weekday(7,1)
1
```

```
"""
```

```
>>> get_weekday(1,0)
1
```

```
>>> get_weekday(4,7)
4
```

```
>>> get_weekday(7,72)
2
```

```
    return (current_weekday + days_ahead)% 7
```

FDR3: What day is my birthday on?

- Examples
- Type contract
- Header
- Description
- Body
- Test

```
def get_birthday_weekday(current_weekday, current_day, birthday_day):
```

```
    """(int, int, int) -> int
```

```
    Return the day of the week it will be on birthday_day, given that
    the day of the week is current_weekday and
    the day of the year is current_day.
```

```
    current_weekday is the current day of the week and is in the range 1-7,
    indicating whether today is Sunday (1), Monday (2), ..., Saturday (7).
    current_day and birthday_day are both in the range 1-365.
```

```
>>> get_birthday_weekday(6,3,4)
```

```
7
```

```
>>> get_birthday_weekday(6,3,116)    % Day116->Apr25
```

```
7
```

```
>>> get_birthday_weekday(7,116,3)
```

```
6
```

```
    """
```

```
    days_diff = days_difference(current_day, birthday_day)
    return get_weekday(current_weekday, days_diff)
```

More information in your textbook

- Tracing function calls in Ch.3.5
- FDR examples explained in Ch.3.6
- Summary in Ch.3.10
- Exercises in Ch.3.11