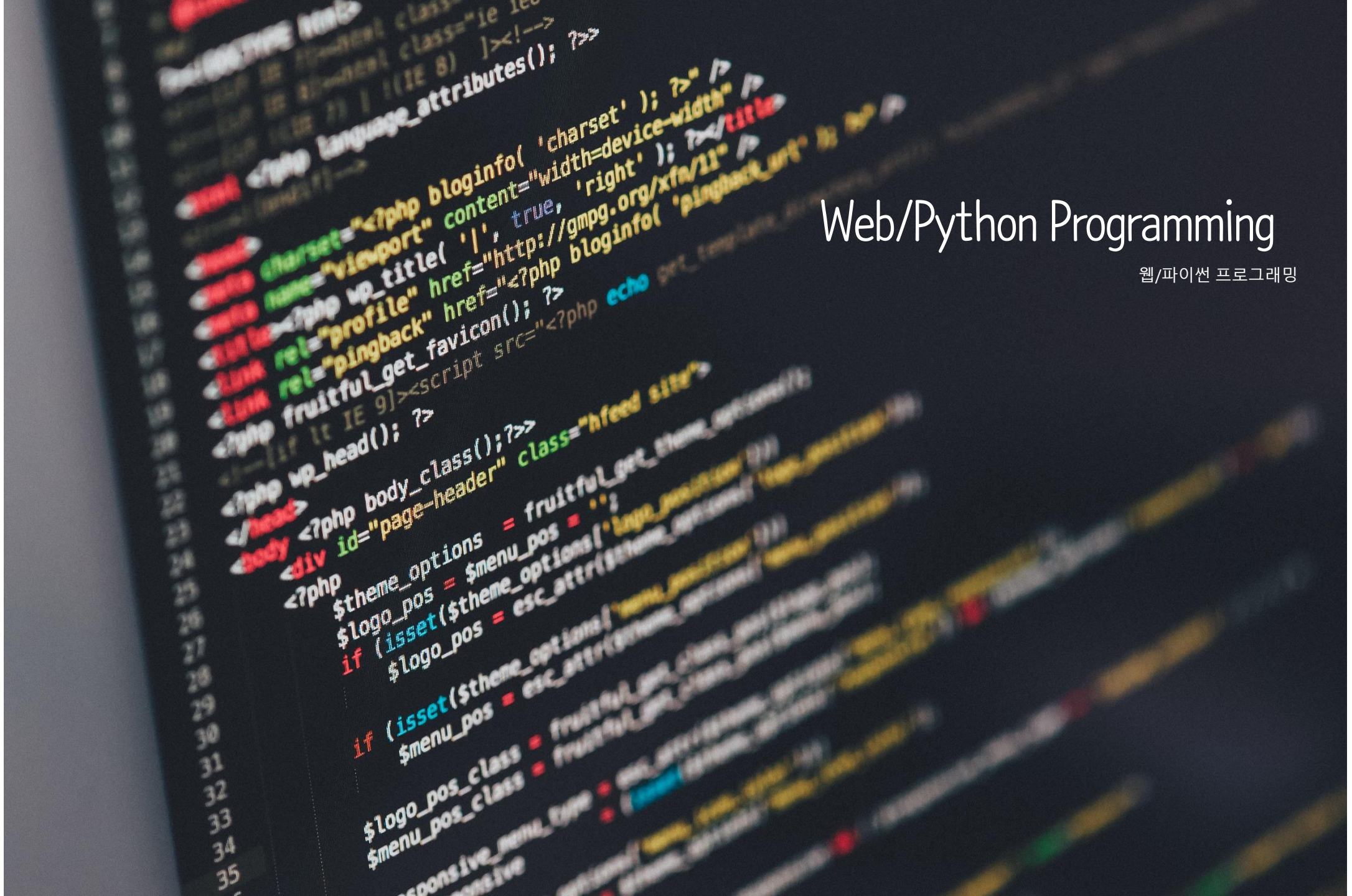


Web/Python Programming

웹/파이썬 프로그래밍

Today

- Modules
- Covers Chapter 6 of your textbook

Modules

- Mathematicians don’t prove every theorem from scratch.
- They build their proofs on the truths their predecessors have already established.

- Programmers don’t write all of a program alone.
- They make use of the many lines of code that other programmers have written before.
- It’s very common and more productive.

Modules

- A module is a kind of object, which can contain functions and other variables.
- A module is a group of functions and variables defined within a single file.

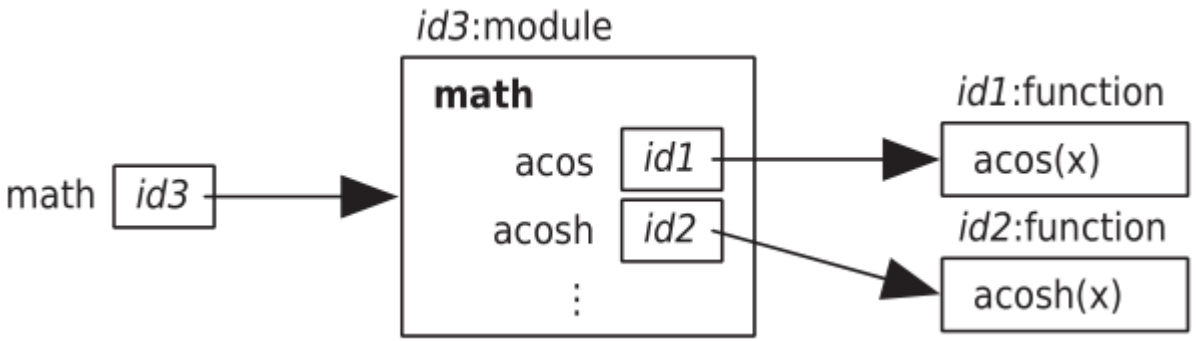
temperature.py

```
def convert_to_celsius(fahrenheit):
    """ (number) -> float

    Return the number of celsius degree
    equivalent to fahrenheit degrees

    >>> convert_to_celsius(212)
    100.0
    """

    return (fahrenheit - 32)*5/9

def convert_to_fahrenheit(celsius):
    """ (number) -> float

    Return the number of fahrenheit degree
    equivalent to celsius degrees

    >>> convert_to_celsius(100)
    212.0
    """

    return celsius*1.8 + 32
```

Import modules

```
>>> type(math)
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module
>
    type(math)
NameError: name 'math' is not defined
```

```
>>> import math
>>> type(math)
<class 'module'>
```




```
>>> help(math)
Help on built-in module math:

NAME
    math

DESCRIPTION
    This module is always available.  It
provides access to the
    mathematical functions defined by th
e C standard.

FUNCTIONS
    acos(...)
        acos(x)

        Return the arc cosine (measured
in radians) of x.

    acosh(...)
        acosh(x)

        Return the inverse hyperbolic co
sine of x.
```

How to use those functions?

```
>>> sqrt(9)
Traceback (most recent call last):
  File "<pyshell#5>", line 1, in <module
>
    sqrt(9)
NameError: name 'sqrt' is not defined
>>> math.sqrt(9)
3.0
```

The dot(.) is an operator, just like + and **

1) Look up the object that the variable to the left of the dot refers to.

2) In that object, find the name that occurs to the right of the dot.

Variables imported from modules

```
>>> import math
>>> math.pi
3.141592653589793
>>> radius = 5
>>> area = math.pi * radius **2
>>> area
78.53981633974483
>>> math.pi = 3
>>> area = math.pi * radius **2
>>> area
75
```

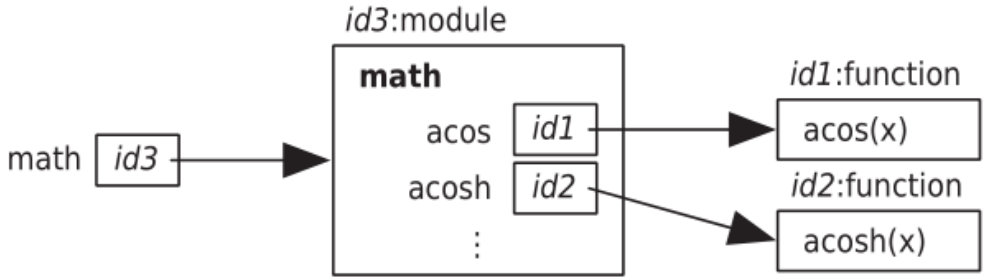
Don’t do this!!

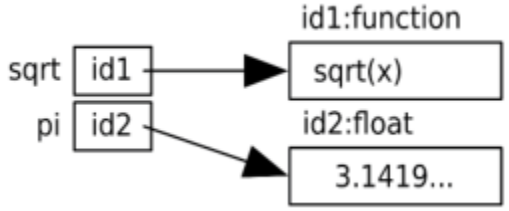
- It is a bad idea to change the value of a variable defined within the module (usually meant to be a constant value)
- However, it is possible in Python.

To avoid using the dot

```
>>> pi
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    pi
NameError: name 'pi' is not defined
>>> import math
>>> pi
Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    pi
NameError: name 'pi' is not defined
>>> math.pi
3.141592653589793
>>> from math import pi, sqrt
>>> pi
3.141592653589793
>>> sqrt(9)
3.0
>>> from math import *
>>>
```

Usually not a good idea





Python modules

- https://docs.python.org/release/3.3.0/py-modindex.html

Python Module Index

_ | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | z

_	
__future__	*Future statement definitions*
__main__	*The environment where the top-level script is run.*
_dummy_thread	*Drop-in replacement for the _thread module.*
_thread	*Low-level threading API.*
a	
abc	*Abstract base classes according to PEP 3119.*
aifc	*Read and write audio files in AIFF or AIFC format.*
argparse	*Command-line option and argument parsing library.*
array	*Space efficient arrays of uniformly typed numeric values.*
ast	*Abstract Syntax Tree classes and manipulation.*
asynchat	*Support for asynchronous command/response protocols.*
asyncore	*A base class for developing asynchronous socket handling services.*
atexit	*Register and execute cleanup functions.*
audioop	*Manipulate raw audio data.*
b	
base64	*RFC 3548: Base16, Base32, Base64 Data Encodings*
bdb	*Debugger framework.*
binascii	*Tools for converting between binary and various ASCII-encoded binary representations.*
binhex	*Encode and decode files in binhex4 format.*

Defining your own modules

Be careful with the saving directory (location)

temperature.py

```
def convert_to_celsius(fahrenheit):
    """ (number) -> float

    Return the number of celsius degree
    equivalent to fahrenheit degrees

    >>> convert_to_celsius(212)
    100.0
    """

    return (fahrenheit - 32)*5/9

def convert_to_fahrenheit(celsius):
    """ (number) -> float

    Return the number of fahrenheit degree
    equivalent to celsius degrees

    >>> convert_to_celsius(100)
    212.0
    """

    return celsius*1.8 + 32
```

```
>>> import temperature
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    import temperature
ImportError: No module named 'temperature'
>>> import temperature
>>> convert_to_celsius(212)
Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    convert_to_celsius(212)
NameError: name 'convert_to_celsius' is not defined
>>> temperature.convert_to_celsius(212)
100.0
>>> temperature.convert_to_fahrenheit(100)
212.0
```

What Happens During Import

- exp.py

```
print("this is experiment")
```

- Run exp.py (F5)
- Import the module exp.
- Import the module exp again.

```
>>>
================== RESTART: C:/Users/Jiyoung/Desktop/exp.py ==================
this is experiment
>>> import exp
this is experiment
>>> import exp
>>>
```

- Python executes modules as it imports them.
- Python loads modules only the first time they are imported.

importlib.reload

exp.py

```
print("this is experiment")
```

- What if we edit the module?
- Your edit won't have any effect until you restart the shell or call importlib.reload

```
>>> import exp
this is experiment
>>> import exp
>>> import importlib
>>> importlib.reload(exp)
this is experiment
<module 'exp' from 'C:\\Users\\jiyou\\AppData\
\Local\\Programs\\Python\\Python37\\exp.py'>
```

Two ways of running a python module

- Run the code directly (pressing F5)
- Run indirectly (imported by another module)
  - Both files should be saved in the same directory

- Sometimes, we want to write code that should only be run when the module is run directly and not when the module is imported.

```
print("this is experiment")

if run_directly:
    print("I am the main program")
else:
    print("Another module is importing me")
```

__name__

```
>>> __name__
'__main__'
>>> exp.__name__
'exp'
>>>
```

```
print("this is experiment")

if __name__=="__main__":
    print("I am the main program")
else:
    print("Another module is importing me")
```

Import math

```
>>> import math
>>> a = 0
>>> b = 30
>>> c = 45
>>> d = 60
>>> e = 90
>>> math.sin(b)
>>> math.sin(math.radians(b))
>>> math.degrees(math.asin(0.5))
```

```
>>> math.pi
>>> math.inf
>>> math.e
>>> math.exp(1)
>>> math.log(math.e)
>>> math.log(math.exp(2))
>>> math.log10(10.0)
>>> math.log10(100.0)
```

Import random

```
>>> import random
>>> x1 = random.random()
    # generates a random floating-point number in range [0,1)
>>> x2 = random.uniform(a,b)
    # generates a random floating-point number in range [a,b)
>>> x3 = random.randrange(stop)
    # chooses an integer in the range [0,stop)
>>> x4 = random.randrange(start,stop)
    # chooses an integer in the range [start,stop)
>>> x5 = random.randrange(start,stop,step)
    # chooses an integer in the range [start,start+step, start+2*step,…,stop)
>>> x5 = random.randint(start,stop)
    # chooses an integer in the range [start, stop] including both end points
```

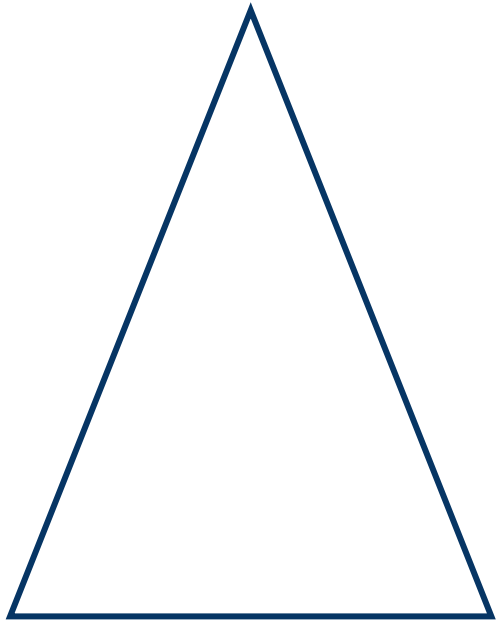
Graphic exercise

```
>>> import turtle
>>> t = turtle.Pen()
>>> t.forward(50)
>>> t.left(90)
>>> t.forward(50)
>>> t.left(90)
>>> t.forward(50)
>>> t.left(90)
>>> t.forward(50)
>>> t.left(90)
```

```
>>> t.reset()
>>> t.backward(100)
>>> t.up()
>>> t.right(90)
>>> t.forward(20)
>>> t.left(90)
>>> t.down()
>>> t.forward(100)

>>> t.clear()
```

Graphic exercise

- Draw a isosceles triangle (이등변삼각형)



- Draw a box with open corners (size is not important)

Modules - summary

- A module is a kind of object, which can contain functions and other variables.
- A module is a collection of functions and variables defined within a single file.
- Math module and random module are useful
- You can make your own module
- Variable __name__ is created by Python and can be used to specify that some code should only run when the module is run directly and not when the module is imported.
- Exercises in Ch.6.6