[DSC4001-01] Python Programming for Data Science

Lecture 03: Python Basics 2

Hyeryung Jang
(hyeryung.jang@dgu.ac.kr)
AI Department, Dongguk University

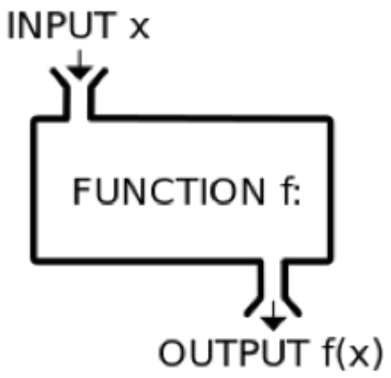
Syllabus: Today’s Topic

Week	Topics
1	Introduction to Data Science, Environment Set-up
2	Python Basics 1
3	**Python Basics 2**
4	Python for Data Analysis: NumPy
5	Python for Data Analysis: Pandas 1
6	Python for Data Analysis: Pandas 2
7	Python for Data Analysis: Web Crawling
8	Midterm Exam
9	Python for Data Visualization: Basics
10	Python for Data Visualization: Advanced
11	Machine Learning with Python: Supervised Learning
12	Machine Learning with Python: Unsupervised Learning
13	Machine Learning with Python: Recommender System
14	Project Presentation
15	Final Exam

Python Basics: Functions, File I/O

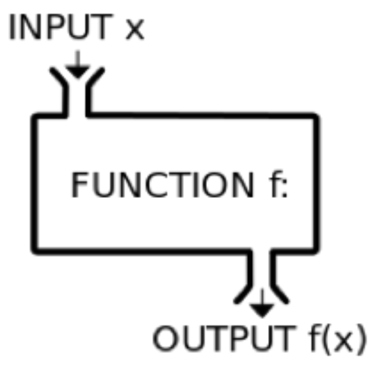
Functions (함수)

- In mathematics, a function is a binary relation between two sets



Functions (함수)

- In Python, a **function** (함수) is some reusable code that
  - takes **argument(s)** (x) as input
  - does some computation and then
  - returns a **result(s)** (y)
  - 반복적으로 수행하는 의미 있는 부분을 ' 함수화' 하여 사용



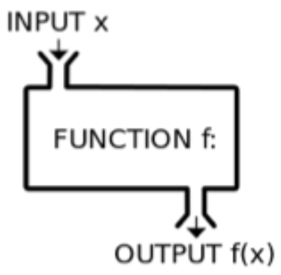
Functions

- Define a function

a, b

$f(a,b) = a + b$

$a + b$



```
# Define a function `plus()`
def plus(a,b):
  return a + b
```

- Use the keyword def to declare the function with the function name
- Add parameters to the function within the (), and end line with a colon :
- Add statements that the function should execute
- End the function with a return statement if necessary

```
def hello():
  print("Hello World")
  return
```

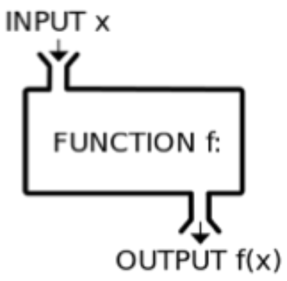
Functions

- Multiple inputs and/or outputs

```
# Define a function `plus()`
def plus(a,b):
  return a + b
```

```
result = plus(3,5)
```

```
result = plus(a=3, b=5)
```



  - Set default arguments

```
def plus(a, b=5):
  return a + b
```

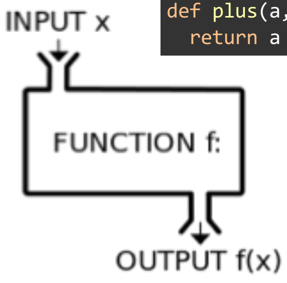
Functions

- Multiple inputs and/or outputs
  - Variable number of arguments: ***args** (tuple)

```python
def plus(*args):
  result = 0
  for i in args:
    result += i
  return result
```

  - plus(1,2,3)
  - plus(1,2,3,4,5)

```python
# Define a function `plus()`
def plus(a,b):
  return a + b
```

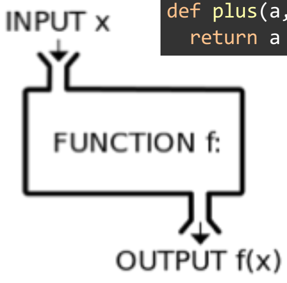


Functions

- Multiple inputs and/or outputs
    - Keyword Arguments: ****kwargs** (dictionary)

```python
def plus(**kwargs):
  result = 0
  for key in kwargs.keys():
    result += kwargs[key
  return result
```

```python
# Define a function `plus()`
def plus(a,b):
  return a + b
```



- plus(a=1,b=2,c=3)
- plus(aa=1,bb=2,cc=3,dd=4,ee=5)

Functions

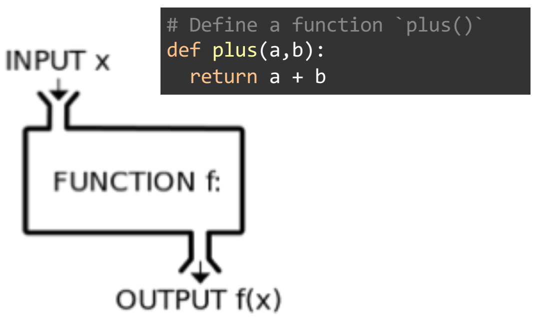
- Multiple inputs and/or outputs
  - Return multiple values as tuples

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

  - result = plus(3,5)

```
result = plus(3,5)
print(result)
(8, 3, 5)
```

```
result, aa, bb = plus(3,5)
print(result, aa, bb)
8 3 5
```



```
# Define a function `plus()`
def plus(a,b):
  return a + b
```

Reading and Writing Files

- Data could be from multiple sources like from databases, Excels, files, …
- How to open, read and write data into files, e.g., CSV, text files, … in Python

- File Open (파일 열기, 생성): open()

```
open(file, mode='r', buffering=-1, encoding=None,
errors=None, newline=None, closefd=True, opener=None)
```

```
f = open(filename, access_mode)
```

  - File path with the file name
  - FileNotFoundError
  - Access modes:
    - ‘r’: read-only mode
    - ‘w’: write-only mode
    - ‘a’: append mode
- File Close (파일 닫기): close()

Reading and Writing Files

- Reading from a file

```
"test_file.txt"

Line 1
Line 2
Line 3
Line 4
Line 5
```

- **read**([n])
  - Outputs the entire file if n is not given

```python
f = open("test_file.txt", 'r')
print(f.read())
```

```
Line 1
Line 2
Line 3
Line 4
Line 5
```

Reading and Writing Files

- Reading from a file

```
"test_file.txt"

Line 1
Line 2
Line 3
Line 4
Line 5
```

- **readline**([n])
  - Outputs at most n bytes of a single line of a file

```python
f = open("test_file.txt", 'r')
print(f.readline())
```

```
Line 1
```

Reading and Writing Files

- Reading from a file

```
"test_file.txt"

Line 1
Line 2
Line 3
Line 4
Line 5
```

- **readlines**()
  - Outputs a list of each line in the file

```
f = open("test_file.txt", 'r')
print(f.readlines())
```

```
[‘Line 1\n’, ‘Line 2\n’, ‘Line 3\n’, ‘Line 4\n’, ‘Line 5\n’]
```

Reading and Writing Files

- Writing to a file

- **write**(string)
  - Write a string to a file

```
f = open("new_test_file.txt", 'w')
f.write("Hello World!\n")
f.close()
```

```
"new_test_file.txt"

Hello World!
```

- **writelines**(list)

```
f = open("new_test_file.txt", 'w')
f.writelines(['Test Line 1\n', 'Test
Line 2\n'])
f.close()
```

```
"new_test_file.txt"

Test Line 1
Test Line 2
```

Reading and Writing Files

- Writing to a file

- Append mode
  - Use append mode ‘a’ to write to the existing file

```
f = open("new_test_file.txt", 'w')
f.write("Hello World!\n")
f.close()
```

“new_test_file.txt”

Hello World!

```
f = open("new_test_file.txt", ‘a')
f.writelines(['Test Line 1\n', 'Test Line 2\n'])
f.close()
```

“new_test_file.txt”

Hello World!
Test Line 1
Test Line 2

Python Basics: Module, Package, Unittest

Module

- Module
  - Defines classes, functions, variables, and other members for use in scripts that import it

```python
# in mod1.py

def plus(a,b):
  return a+b
```

```python
def plus(a,b):
  return a+b

result = plus(3,5)
print(result)
```

```python
from mod1 import plus

result = plus(3,5)
print(result)
```

Module

```
# in mod1.py

def plus(a,b):
  return a+b
```

- Import: 모듈 불러오기
- import <module_name>
  - import <module_name> as <alias>
- from <module_name> import <module_function>
- from <module_name> import *

```
import mod1

result = mod1.plus(3,5)
print(result)
```

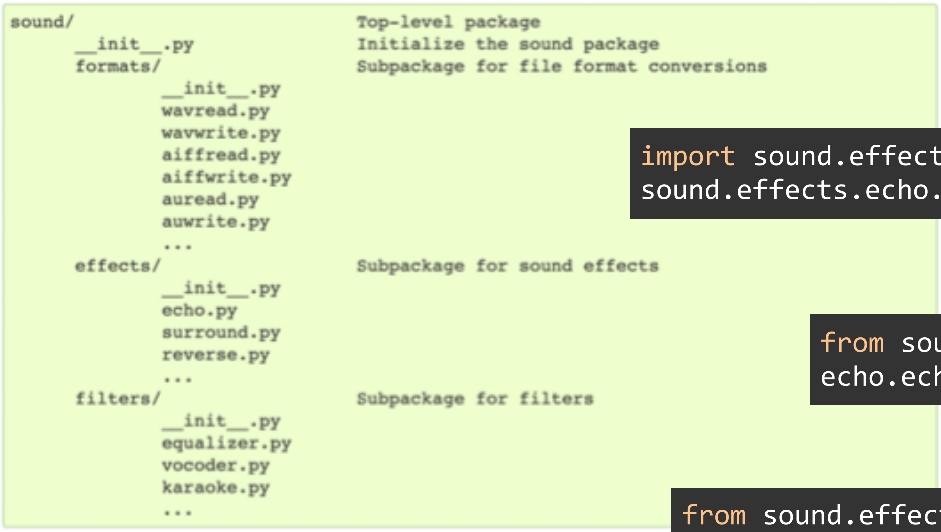
```
import mod1 as M1

result = M1.plus(3,5)
print(result)
```

```
from mod1 import plus

result = plus(3,5)
print(result)
```

```
from mod1 import *

result = plus(3,5)
print(result)
```

Package

- A collection of related modules
  - . 을 이용해 모듈 공간을 계층화 (구조화)
  - <package_name>.<module_name>



```
sound/                          Top-level package
      __init__.py               Initialize the sound package
      formats/                  Subpackage for file format conversions
              __init__.py
              wavread.py
              wavwrite.py
              aiffread.py
              aiffwrite.py
              auread.py
              auwrite.py
              ...
      effects/                  Subpackage for sound effects
              __init__.py
              echo.py
              surround.py
              reverse.py
              ...
      filters/                  Subpackage for filters
              __init__.py
              equalizer.py
              vocoder.py
              karaoke.py
              ...
```

```
import sound.effects.echo
sound.effects.echo.echofilter(input, output)
```

```
from sound.effects import echo
echo.echofilter(input, output)
```

```
from sound.effects.echo import echofilter
echofilter(input, output)
```

Python Standard Library

- Collection of modules and packages
  - That come bundled with every installation of Python
  - Don’t need to download them with PIP
  - E.g., csv, os.path, parser, …
  - https://docs.python.org/3/library/index.html

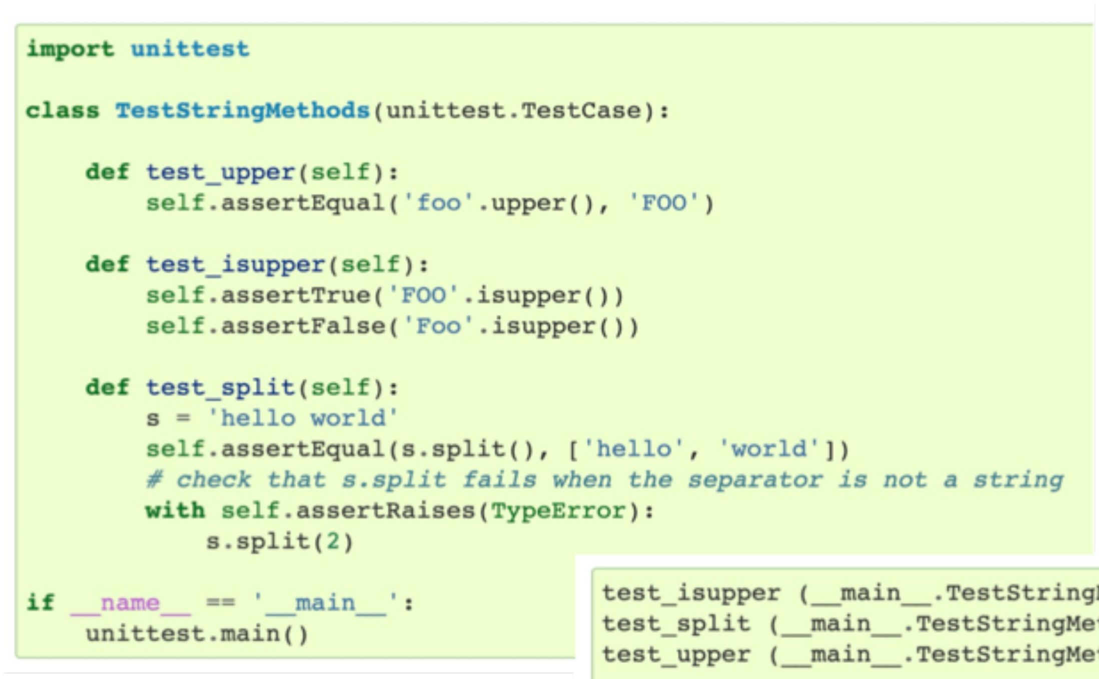
- Non-standard, but powerful libraries
  - NumPy, Pandas, Matplotlib, Scikit-Learn, …

Unittest

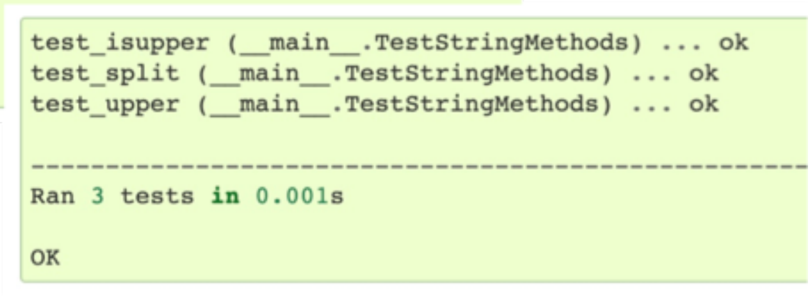
- Testing framework
  - 개발된 output이 요구사항과 부합하는 지 검증하는 작업
  - 개별 코드 단위가 예상대로 작동하는 지 확인하는 반복 가능한 활동

- Unittest
  - https://docs.python.org/ko/3/library/unittest.html
  - assertEqual, assertTrue, assertFalse, assertIn, …

Unittest



```
import unittest

class TestStringMethods(unittest.TestCase):

    def test_upper(self):
        self.assertEqual('foo'.upper(), 'FOO')

    def test_isupper(self):
        self.assertTrue('FOO'.isupper())
        self.assertFalse('Foo'.isupper())

    def test_split(self):
        s = 'hello world'
        self.assertEqual(s.split(), ['hello', 'world'])
        # check that s.split fails when the separator is not a string
        with self.assertRaises(TypeError):
            s.split(2)

if __name__ == '__main__':
    unittest.main()
```

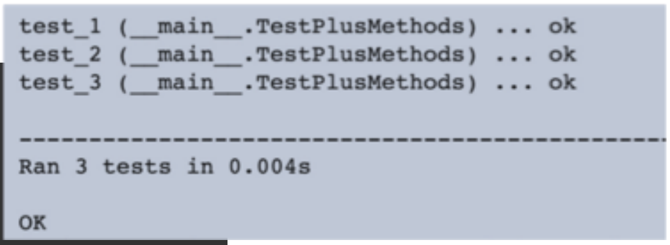


```
test_isupper (__main__.TestStringMethods) ... ok
test_split (__main__.TestStringMethods) ... ok
test_upper (__main__.TestStringMethods) ... ok

----------------------------------------------------
Ran 3 tests in 0.001s

OK
```

Unittest

- Test the function ‘plus’ outputs correct values

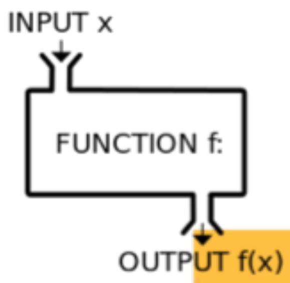
```python
import unittest
from mod1 import *

class TestPlusMethods(unittest.TestCase):

  def test_1(self):
      self.assertEqual(plus(1,2), 3)

  def test_2(self):
      self.assertEqual(plus(12,23), 35)

  def test_3(self):
      self.assertEqual([plus(2,3), plus(2,4)], [5,6])

unittest.main(argv=[''], verbosity=2, exit=False)
```

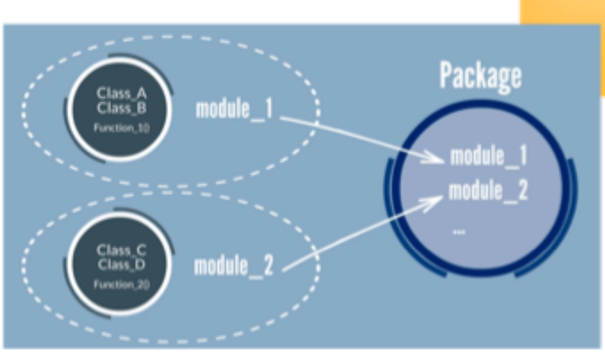


```
test_1 (__main__.TestPlusMethods) ... ok
test_2 (__main__.TestPlusMethods) ... ok
test_3 (__main__.TestPlusMethods) ... ok

----------------------------------------------
Ran 3 tests in 0.004s

OK
```

In this lesson, you have learned:

- Python Basics:
- Functions
- File operations
- Modules, Packages, Libraries
- Test





Thank you!

Any Questions?

hyeryung.jang@dgu.ac.kr