Python

Jin Hyun Kim

2019

This Class

- “A Byte of Python”- Swaroop C H

In this class

- Why “python” in this course?
- Install compiler and IDE (PyCham)
- Lab 00 - “Hello World”

Why

- Simple
- Easy to Learn
- Free and Open Source
- High-level Language
- Portable (이식성)
- Object Oriented Language
- Extensible
- Embeddable
- Extensive Libraries

Lab 1

- print (Hello World)
    - Install python
    - Install “PyCham”
    - Create and configure the project
    - Run “python hello.py”

Basic

Basic

- Comments

```
print 'hello world' # Note that print is a statement
```

or:

```
# Note that print is a statement
print 'hello world'
```

Basic

- Constants
- Numbers - **5**, **1.23**, **52.3E-4**
- String - “This is a string”, ‘It’s a string’
- Multiple string - ‘ ‘ ‘ … ’ ’ ’, “ “ “ … ” ” ”

Format

```
age = 20
name = 'Swaroop'

print '{0} was {1} years old when he wrote this book'.format(name, age)
print 'Why is {0} playing with that python?'.format(name)
```

- A number in “{}” is optional

```
name + ' is ' + str(age) + ' years old'
```

Escape Sequences

- What if a string includes “‘”? `'What's your name?'`

'What\'s your name?'

- ‘\’ -> ‘\\’

- “\n” (newline)

```
'This is the first line\nThis is the second line'
```

- Ignore escape sequence

```
r"Newlines are indicated by \n"
```

Variable Identifiers

- The first character of the identifier must be a letter of the alphabet (uppercase ASCII or lowercase ASCII or Unicode character) or an underscore (_).

- The rest of the identifier name can consist of letters (uppercase ASCII or lowercase ASCII or Unicode character), underscores (_) or digits (0-9).

- Identifier names are case-sensitive. For example, ***myname*** and ***myName*** are not the same. Note the lowercase n in the former and the uppercase in the latter.

Basic

- Variables
- Objects

Lines

Logical vs Physical Lines

```
i = 5
print i
```

is effectively same as

```
i = 5;
print i;
```

which is also same as

```
i = 5; print i;
```

and same as

```
i = 5; print i
```

Lines

- A line can be broken by "\" in code

```
s = 'This is a string. \
This continues the string.'
print s
```

Output:

```
This is a string. This continues the string.
```

Indentation

- Whitespace at the beginning of the line, called **indentation**, is important

```
i = 5
# Error below! Notice a single space at the start of the line
 print 'Value is ', i
print 'I repeat, the value is ', i
```

- Statements should go together and have the same indentation

```
if True:
    print 'Yes, it is true'
```

Operators

- +, -, x, /
- ** (power): ex) 3 ** 4 = 3 * 3 * 3 * 3
- % (modulo): ex)3 % 2 = 1
- << (bit, shift left): ex) 2 << 2 = 8
- >> (bit, shift right): ex) 11 >> 1 = 5
- & (bit, And): ex) 5 & 3 = 1 (0101 & 0011 = 0001)
- | (bit, Or): ex) 5 | 3 = 7 (0101 & 0011 = 0111)

Operators

- ^ (bit, Xor): ex) 5 ^ 3 = 6 (101 ^ 011 = 110)
- ~ (bit, flip, Return -(x+1)) ex) ~5 = -6
- > (Logical “greater than”)
- < (Logical “less than”)
- <= (Logical “greater than or equal to”)
- >= (Logical “less than or equal to”)
- == (Logical “equal to”)

Operators

- != (Logical “not equal to”)
- not (Logical negation): ex) x = True ; not x
- and (Boolean logical “AND”)
- or (Boolean logical “OR”)

Assignments

- Shortened expression

```
a = 2
a *= 3
```

- Precedence (연산 우선 순위)
  - 2 + 3 * 4

Control Flow

If

```
number = 23
guess = int(raw_input('Enter an integer : '))

if guess == number:
    # New block starts here
    print '(but you do not win any prizes!)'
    # New block ends here
elif guess < number:
    # Another block
    print 'No, it is a little higher than that'
    # You can do whatever you want in a block ...
else:
    print 'No, it is a little lower than that'
    # you must have guessed > number to reach here

print 'Done'
# This last statement is always executed,
# after the if statement is executed.
```

While

```
number = 23
running = True

while running:
    guess = int(raw_input('Enter an integer : '))

    if guess == number:
        print 'Congratulations, you guessed it.'
        # this causes the while loop to stop
        running = False
    elif guess < number:
        print 'No, it is a little higher than that.'
    else:
        print 'No, it is a little lower than that.'
else:
    print 'The while loop is over.'
    # Do anything else you want to do here

print 'Done'
```

for

```
for i in range(1, 5):
    print i
else:
    print 'The for loop is over'
```

break

```
while True:
    s = raw_input('Enter something : ')
    if s == 'quit':
        break
    print 'Length of the string is', len(s)
print 'Done'
```

continue

```
while True:
    s = raw_input('Enter something : ')
    if s == 'quit':
        break
    if len(s) < 3:
        print 'Too small'
        continue
    print 'Input is of sufficient length'
    # Do other kinds of processing here...
```

Functions

Function

```
def say_hello():
    # block belonging to the function
    print 'hello world'
# End of function

say_hello() # call the function
say_hello() # call the function again
```

```
$ python function1.py
hello world
hello world
```

Parameters

```
def print_max(a, b):
    if a > b:
        print a, 'is maximum'
    elif a == b:
        print a, 'is equal to', b
    else:
        print b, 'is maximum'

# directly pass literal values
print_max(3, 4)

x = 5
y = 7

# pass variables as arguments
print_max(x, y)
```

Local Variables

```
x = 50

def func(x):
    print 'x is', x
    x = 2
    print 'Changed local x to', x

func(x)
print 'x is still', x
```

```
$ python function_local.py
x is 50
Changed local x to 2
x is still 50
```

Global Variables

- Local

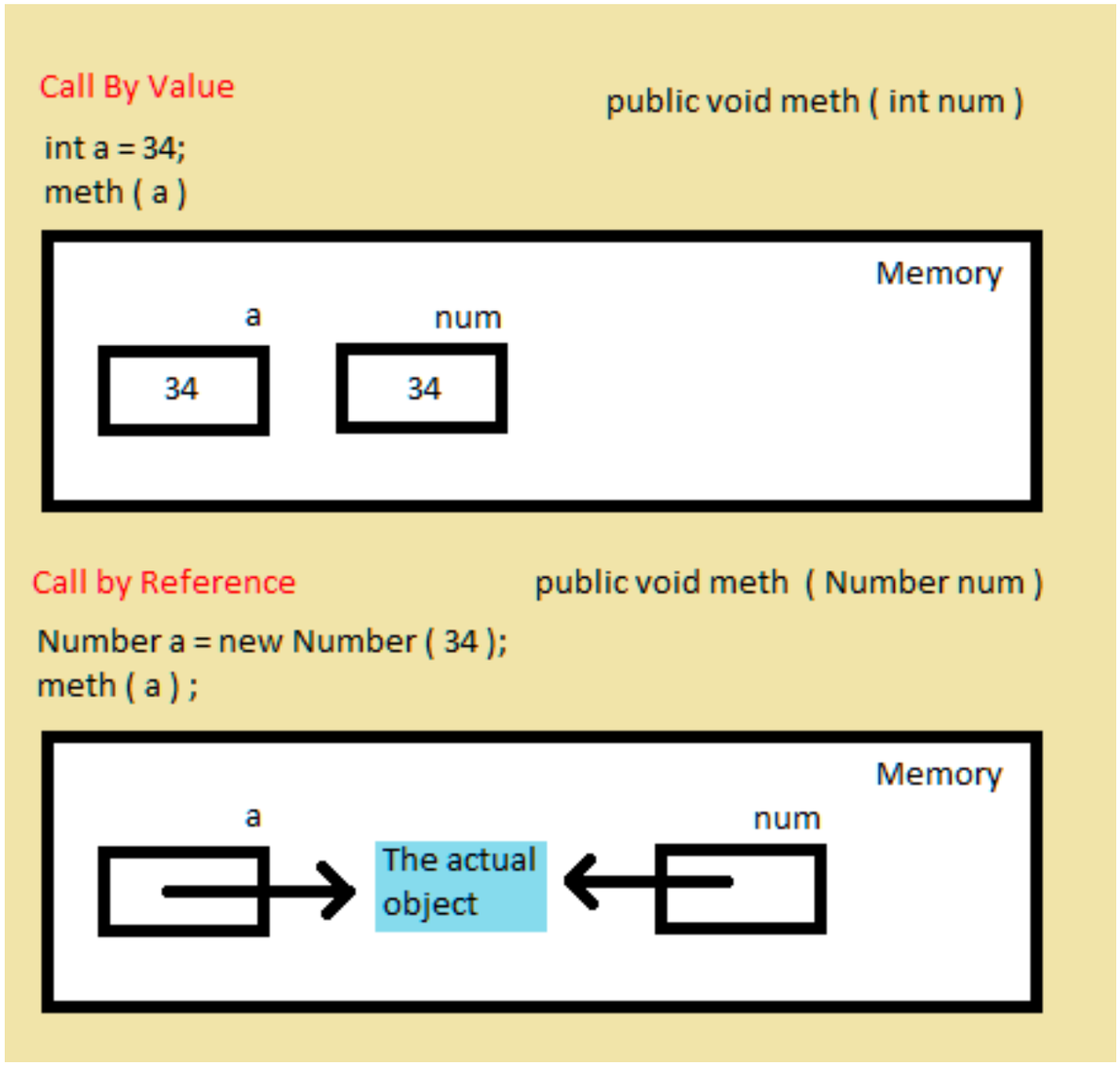
```
x = 50

def func(x):
    print 'x is', x
    x = 2
    print 'Changed local x to', x

func(x)
print 'x is still', x
```

- Global

```
x = 50

def func():
    global x

    print 'x is', x
    x = 2
    print 'Changed global x to', x

func()
print 'Value of x is', x
```

```
$ python function_global.py
x is 50
Changed global x to 2
Value of x is 2
```

Call by Ref. / Val. / …



http://www.javawithus.com/tutorial/call-by-value-and-call-by-reference

Call by … (Python)

- According to variable types
  - Immutable object
    - Call by Value
  - Mutable Object
    - Pass object reference

https://code13.tistory.com/214

```python
def dataCalc1(data):
    data = data + 1
def dataCalc2(data):
    data[0] = data[0] + 1
def main():
    data1 = 1
    data2 = [1]
    # Call by Value
    dataCalc1(data1);
    print 'data1 : ', data1
    # Call by Reference
    dataCalc2(data2);
    print 'data2 : ', data2

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

결과

data1 : 1

data2 : [2]

Function Parameters

- Default value

Positional args

Keyword args

```
def say(message, times=1):
    print message * times

say('Hello')
say('World', 5)
```

```
$ python function_default.py
Hello
WorldWorldWorldWorldWorld
```

```
def func(a, b=5, c=10):
    print 'a is', a, 'and b is', b, 'and c is', c

func(3, 7)
func(25, c=24)
func(c=50, a=100)
```

```
$ python function_keyword.py
a is 3 and b is 7 and c is 10
a is 25 and b is 5 and c is 24
a is 100 and b is 5 and c is 50
```

VarArgs
(Variable + Arguments)

- Positional args

```
def save_ranking(*args):
    print(args)
save_ranking('ming', 'alice', 'tom', 'wilson', 'roy')
# ('ming', 'alice', 'tom', 'wilson', 'roy')
```

- Positional args + Keyword args

```
def save_ranking(*args, **kwargs):
    print(args)
    print(kwargs)
save_ranking('ming', 'alice', 'tom', fourth='wilson', fifth='roy')
# ('ming', 'alice', 'tom')
# {'fourth': 'wilson', 'fifth': 'roy'}
```

- Keyword args

```
def save_ranking(**kwargs):
    print(kwargs)
save_ranking(first='ming', second='alice', fourth='wilson', third='tom', fifth='roy')
# {'first': 'ming', 'second': 'alice', 'fourth': 'wilson', 'third': 'tom', 'fifth': 'roy'}
```

https://mingrammer.com/understanding-the-asterisk-of-python/

Return

```
def maximum(x, y):
    if x > y:
        return x
    elif x == y:
        return 'The numbers are equal'
    else:
        return y

print maximum(2, 3)
```

```
def some_function():
    pass
```

- Return a computation result by a function

Return

```
#!/usr/bin/env python3

def getPerson():
    name = "Leona"
    age = 35
    country = "UK"
    return name,age,country

name,age,country = getPerson()
print(name)
print(age)
print(country)
```

- Python allows multiple outputs to be returned

Module

```
import sys

print('The command line arguments are:')
for i in sys.argv:
    print i

print '\n\nThe PYTHONPATH is', sys.path, '\n'
```

- A module imported by “import” includes methods and data
- A module is defined and achieved as its file name, i.e., a module is given in its file name
- ***Run …***

from … import

```
from math import sqrt
print "Square root of 16 is", sqrt(16)
```

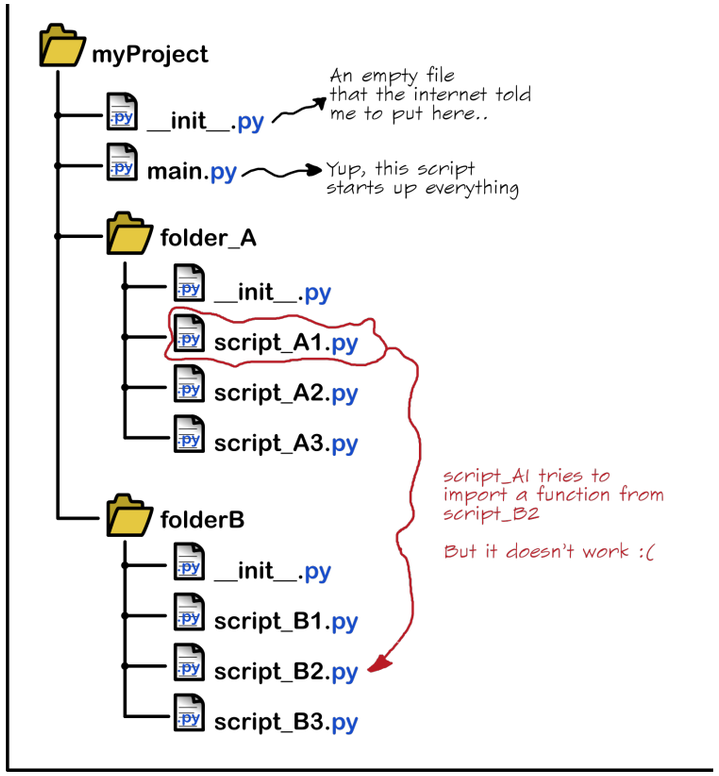
import math
math.squre(16)

name in Module

```
if __name__ == '__main__':
    print 'This program is being run by itself'
else:
    print 'I am being imported from another module'
```

- Keyword “name” tell you the name of module
- It is used to indicate where you are

Module



m_eat.py

```
name = "홍길동"

def cook():
    print("요리하다.")

def eat():
    print("먹다.")
```