초심자를 위한
Python

2주차

spy@securityfact.com

Index

1. Module / Class / Object

2. 상속과 합성

3. self / is-a / has-a

spy@securityfact.com

Module / Class / Object

spy@securityfact.com

Module

함수 / 변수가 포함된 Python File

‘import’하여 사용

모듈 안의 함수/변수에 . 연산자로 접근

spy@securityfact.com

모듈사용 예제코드 작성

** 바탕화면에 따로 폴더를 만들어서 다음의 코드들을 함께 넣어주세요.*

module_ex.py

```
# -*- coding : utf-8 -*-

im_in_module="I'm variable in module.“

def printer (content):
    print("Running Printer...")
    print("**** Printing Result ****")
    print(content)
    print("# End of File")

def call_me ():
    print("Hello! Why are you calling me?")
```

모듈사용 예제코드 작성

using_module.py

```
# -*- coding : utf-8 -*-
import module_ex

variable="I don't like python. ₩nDo you like python?“

module_ex.printer(variable)

module_ex.call_me()

print(module_ex.im_in_module)
```

모듈사용 예제코드 실행

```
PS C:₩Users₩Jiyoon₩Desktop₩For_Git₩F-Shield python₩week 2> py module_ex.py
PS C:₩Users₩Jiyoon₩Desktop₩For_Git₩F-Shield python₩week 2> py using_module.py
Running Printer...
**** Printing Result ****
I don't like python.
Do you like python?
# End of File
Hello! Why are you calling me?
I'm variable in module.
```

Class

함수 / 자료를 묶어서 다룸

. 연산자로 접근가능

여러 객체 생성 가능

spy@securityfact.com

클래스사용 예제코드 작성

** 임의의 폴더 안에 다음의 코드들을 함께 넣어주세요.*

class_ex.py

①
```
class example:
    def __init__(self, username):
        self.username=username
        print("Your name is %s"%username)
        print("₩n")

    def textViewer(self, sentence):
        print("Hi %s"%self.username)
        print("***** Text Viewer *****")
        print(sentence)
        print("***** End Of Text *****")
        print("₩n")
```

②
```
class new_example:
    def __init__(self, username):
        self.username=username

    def printName(self):
        print("%s, you can print your name₩n“
              %self.username)
```

코드 작성 순서 : ① -> ②

클래스사용 예제코드 작성

using_class.py

```
from class_ex import example
from class_ex import new_example

A=example("atom")
B=example("Bob")

sentence_A="class! class! help me!“
A.textViewer(sentence_A)

sentence_B="major lazer“
B.textViewer(sentence_B)

C=new_example("Jerry")
D=new_example("Tom")
C.printName()
D.printName()
```

클래스사용 예제코드 실행

```
PS C:₩Users₩Jiyoon₩Desktop₩For_Git₩F-Shield python₩week 2> py .₩class_ex.py
PS C:₩Users₩Jiyoon₩Desktop₩For_Git₩F-Shield python₩week 2> py .₩using_class.py
Your name is atom


Your name is Bob


Hi atom
***** Text Viewer *****
class! class! help me!
***** End Of Text *****


Hi Bob
***** Text Viewer *****
major lazer
***** End Of Text *****


Jerry, you can print your name

Tom, you can print your name
```

Object

Class가 형상화 된 것

인스턴스화 된 클래스

spy@securityfact.com

상속과 합성

spy@securityfact.com

상속

한 클래스가 다른 클래스의 특성을
자식이 부모에게 물려받듯 상속!

한 클래스가 부모 클래스에서
부모 기능의 대부분 or 전체를 가져옴

OverLoading 불가능 (OverRiding 가능)

여러 부모를 두는 '다중 상속' 권장 X

class [클래스 이름] (상속받을 부모 클래스 이름) :
    클래스 내용 …

상속 예제코드 작성

inherit_ex.py

①
```
class Mother():
    def normalMethod(self):
        print("I'm your father. -Darth Vader")

    def MotherPower(self):
        print("Power!!!!!!!!!!!!!!!!")

class Son(Mother):
    def normalMethod(self, name):
        print("I'm your son. -%s"%name)

    def OneOfKind(self):
        print("But.. You're not the only one")
```

②
```
A=Mother()
A.normalMethod()
A.MotherPower()

B=Son()
B.normalMethod("Kim")
B.MotherPower()
B.OneOfKind()
```

코드 작성 순서 : ① -> ②

상속 예제코드 실행

```
PS C:₩Users₩Jiyoon₩Desktop₩For_Git₩F-Shield python₩week 2> py .₩inherit_ex.py
*****Testing Mother class*****
I'm your father. -Darth Vader
Power!!!!!!!!!!!!!!!!


******Testing Son class******
I'm your son. -Kim
Power!!!!!!!!!!!!!!!!
But.. You're not the only one
```

합성

차가 바퀴를 가지듯,
한 클래스를 다른 클래스의 일부로서 합성!

spy@securityfact.com

합성 예제코드 작성

inherit_ex.py

①
```
class other(object):
    def overried(self):
        print("other override")

    def implicit(self):
        print("other implicit")

    def altered(self):
        print("other altered")
```

②
```
class child(object):
    def __init__(self):
        self.other=other( )
    def implicit(self):
        self.other.implicit( )
    def override(self):
        print("child   override")
    def altered(self):
        print("child, before other altered")
        self.other.altered( )
        print("child, after other altered")

son=child( )
son.implicit( )
son.override( )
son.altered( )
```

코드 작성 순서 : ① -> ②

합성 예제코드 실행

```
PS C:₩Users₩Jiyoon₩Desktop₩For_Git₩F-Shield python₩week 2> py .₩cor
other implicit
child  override
child, before other altered
other altered
child, after other altered
```

```
>>> class mother(object):
        def altered(self):
            print("mother altered")


>>> class child(mother):
        def altered(self):
            print("child, before mother altered")
            super(child, self).altered()
            print("child, after mother altered")


>>> mom=mother()
>>> son=child()
>>> mom.altered()
mother altered
>>> son.altered()
child, before mother altered
mother altered
child, after mother altered
```

self / is-a / has-a

spy@securityfact.com

self

클래스의 함수 안에서 사용

접근한 인스턴스(객체)를 가리키는 변수

Java의 this와 비슷함
(클래스 내부의 object를 가리키는..)

```
class other(object):
    def overried(self):
        print("other override")

    def implicit(self):
        print("other implicit")

    def altered(self):
        print("other altered")
```

```
class new_example:
    def __init__(self, username):
        self.username=username

    def printName(self):
        print("%s, you can print your name₩n"%self.username)
```

is-a

연어 is-a 물고기

한 항목이 다른 항목 상속
(~는 …이다)

spy@securityfact.com

```
class other(object):
    def overried(self):
        print("other override")

    def implicit(self):
        print("other implicit")

    def altered(self):
        print("other altered")
```

class is-a object

spy@securityfact.com

has-a

연어 has-a 입

한 항목이 다른 항목 합성 or 특성을 가짐
(~는 …을 가졌다)

spy@securityfact.com

```
class child(object):
    def __init__(self):
        self.other=other( )
    def implicit(self):
        self.other.implicit( )
    def override(self):
        print("child   override")
    def altered(self):
        print("child, before other altered")
        self.other.altered( )
        print("child, after other altered")
```

child has-a other

See you later.

spy@securityfact.com