

PYTHON

클래스 / 모듈 / 패키지

8.7 우제혁

Index



001/ 클래스

002/ 모듈

003/ 패키지

1 클래스

클래스 개념

별모양 틀(클래스) 별모양 뽑기(인스턴스)

```
class service:  
    secret = "사실 집에 가고싶다"
```

```
pey = service()
```

```
pey.secret
```

```
class service:  
    secret = "사실 집에 가고싶다"
```

```
>>> pey = service()  
>>> pey.secret  
'사실 집에 가고싶다'
```

1 클래스

```
class service:
    def sum(self,a,b):
        result = a+b
        print("`%s+%s=%s입니다." %(a,b,result))
```

pey = service()

pey.sum(1,1)

```
class service:
    def sum(self,a,b):
        result = a+b
        print("`%s+%s=%s입니다." %(a,b,result))
```

```
>>> pey.sum(1,1)
Traceback (most recent call last):
  File "<pyshell#10>", line 1, in <module>
    pey.sum(1,1)
NameError: name 'pey' is not defined
```

```
>>> pey = service()
>>> pey.sum(1,1)
1+1=2입니다.
```

1 클래스

Self

```
def sum(self,a,b):
```

pey = service()

self에 pey가 맞음이 확인 되어 사용가능 로그인 개념

pey.sum(1,1)
service.sum(pey,1,1)

```
>>> pey.sum(1,1)  
1+1=2입니다.
```

```
>>> service.sum(pey,1,1)  
1+1=2입니다.
```

pey.sum(pey,1,1) -> 이렇게 쓰면 실행은 안됨

```
>>> pey.sum(pey,1,1)  
Traceback (most recent call last):  
  File "<pyshell#13>", line 1, in <module>  
    pey.sum(pey,1,1)  
TypeError: sum() takes 3 positional arguments but 4 were given
```

1 클래스

```
def setname(self,name):  
    self.name = name
```

```
def sum(self,a,b):  
    result = a+b  
    print("%님 %s+%s=%s입니다." %(self.name,a,b,result))
```

```
pey = service()  
pey.setname("홍길동")
```

pey값의 사용자 이름은 홍길동이라고 인식

```
class service:
```

```
def setname(self,name):  
    self.name = name  
  
def sum(self,a,b):  
    result = a+b  
    print("%s님 %s+%s=%s입니다." %(self.name,a,b,result))
```

```
>>> pey = service()  
>>> pey.setname("홍길동")  
>>> pey.sum(1,1)  
홍길동님 1+1=2입니다.
```

1 클래스

```
Def __init__(self,name):  
    self.name = name
```

```
def sum(self,a,b):  
    result = a+b  
    print("%님 %s+%s=%s입니다." %(self.name,a,b,result))
```

```
def __init__(self,name):  
    self.name = name  
  
def sum(self,a,b):  
    result = a+b  
    print("%s님 %s+%s=%s입니다." %(self.name,a,b,result))
```

__init__ -> 인스턴트 만들때 항상 실행

pey = service("홍길동 ") - 바로 name인자로 들어감

```
>>> pey = service("홍길동")  
>>> pey.sum(1,1)  
홍길동님 1+1=2입니다.
```

1 클래스

```
class fourcul:  
    def setdata(self,first,second):  
        self.first=first  
        self.second=second  
  
    def sum(self):  
        result=self.first+self.second  
        return result
```

a=fourcul() -> self 자리에 a가 들어감
a.setdata(4,5) -> self.first가 a.first로 바뀜
a.sum()->a가 첫번째 인수로 들어가서 원하는 값이 들어가진다

```
class fourcul:  
    def setdata(self,first,second):  
        self.first=first  
        self.second=second  
  
    def sum(self):  
        result=self.first+self.second  
        return result
```

```
>>> a = fourcul()  
>>> a.setdata(4,5)  
>>> a.sum()  
9
```


1 클래스

```
class housepark:
    lastname="박"

    def nameset(self,name):
        self.fullname=self.lastname+name

    def travel(self,where):
        print("%s,%s여행가다" %(self.fullname,where))

pey=housepark()
pey.nameset("명수")
pey.travel("부산")
```

```
class housepark:
    lastname="박"

    def nameset(self,name):
        self.fullname=self.lastname+name

    def travel(self,where):
        print("%s,%s여행가다" %(self.fullname,where))
```

```
>>> pey=housepark()
>>> pey.nameset("명수")
>>> pey.travel("부산")
박명수,부산여행가다
```

1 클래스

```
class housepark:  
    lastname="박"
```

```
    def __init__(self,name):
```

-> **__init__**으로 초기값 설정해주기

```
        self.fullname=self.lastname+name  
    def travel(self,where):  
        print("%s,%s여행가다" %(self.fullname,where))
```

```
pey=housepark("명수")  
pey.travel("부산")
```

```
class housepark:  
    lastname="박"  
  
    def __init__(self,name):  
        self.fullname=self.lastname+name  
  
    def travel(self,where):  
        print("%s,%s여행가다" %(self.fullname,where))
```

```
>>> pey=housepark("명수")  
>>> pey.travel("부산")  
박명수, 부산여행가다
```

1

클래스-상속

```
class housepark:
    lastname="박"
```

```
def __init__(self,name):
    self.fullname=self.lastname+name
def travel(self,where):
    print("%s,%s여행가다" %(self.fullname,where))
```

```
class housekim(housepark):
    lastname ="김"    -> 상속
```

```
j=housekim("유진")
j.travel("뉴욕")->클래스에 travel이 없어도 상속을 해줬기때문에 가능
```

```
class housepark:
    lastname="박"

    def __init__(self,name):
        self.fullname=self.lastname+name

    def travel(self,where):
        print("%s,%s여행가다" %(self.fullname,where))

class housekim(housepark):
    lastname ="김"
```

```
>>> j=housekim("유진")
>>> j.travel("뉴욕")
김유진,뉴욕여행가다
```

1 클래스-오버로딩

```
class housepark:  
    lastname="박"
```

```
    def __init__(self,name):  
        self.fullname=self.lastname+name  
    def travel(self,where):  
        print("%s,%s여행가다" %(self.fullname,where))
```

```
class housekim(housepark):  
    lastname ="김"
```

```
    def travel(self,where,day):  
        print("%s,%s로%s만큼 여행가다" %(self.fullname,where,day))
```

```
j=housekim("유진")  
j.travel("뉴욕",3)   같은 메소드로 새롭게 만들어 주면 됨
```

```
class housepark:  
    lastname="박"  
  
    def __init__(self,name):  
        self.fullname=self.lastname+name  
  
    def travel(self,where):  
        print("%s,%s여행가다" %(self.fullname,where))  
  
class housekim(housepark):  
    lastname ="김"  
  
    def travel(self,where,day):  
        print("%s,%s로%s만큼 여행가다" %(self.fullname,where,day))
```

```
>>> j=housekim("유진")  
>>> j.travel("뉴욕",3)  
김유진,뉴욕로3만큼 여행가다
```

1

클래스-연산자 오버로딩

```
class housepark:  
    lastname="박"
```

```
def __init__(self,name):  
    self.fullname=self.lastname+name
```

```
def __add__(self,other):  
    print("%s,%s 함께여행" %(self.fullname,other.fullname))
```

```
class housekim(housepark):  
    lastname ="김"
```

```
j=housekim("유진")  
pey=housepark("명수")  
pey + j
```

```
class housepark:  
    lastname="박"  
  
    def __init__(self,name):  
        self.fullname=self.lastname+name  
  
    def __add__(self,other):  
        print("%s,%s 함께여행" %(self.fullname,other.fullname))  
  
class housekim(housepark):  
    lastname ="김"
```

```
>>> j=housekim("유진")  
>>> pey=housepark("명수")  
>>> pey+j  
박명수,김유진 함께여행
```

모듈 만들고 불러오기

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

```
Microsoft Windows [Version 10.0.17763.615]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Users\User>cd C:\w제혁

C:\w제혁>dir
8-23-2019 10:04 AM <DIR>
블룸 일련 번호: 06F0-02A7

C:\w제혁 디렉터리

2019-08-05 오후 06:35 <DIR> .
2019-08-05 오후 06:35 <DIR> ..
2019-05-01 오후 08:50 897,094 1 우제혁_ftz_level9 풀이.pdf
2019-07-18 오후 07:26 71,611 c언어 공부.pptx
2019-04-05 오전 08:49 <DIR> c언어 콘서트_Sources
2019-07-06 오후 10:28 <DIR> john-1.9.0-jumbo-1-win64
2019-08-05 오후 06:22 31 mod1.py
2019-07-30 오후 11:52 ppt 애플링
2019-08-05 오후 02:19 108,064 python공부.pptm
2019-08-01 오후 07:37 <DIR> SOP 발표 자료
2019-08-05 오후 04:39 460 test.py
2019-08-05 오후 06:35 __pycache__
2019-06-14 오전 10:52 <DIR> 과제
2019-07-08 오후 08:00 42,869 방화 계획.pptx
2019-04-11 오후 11:49 <DIR> 방화 문제
2019-05-31 오전 11:35 <DIR> 인간관계와 소통
2019-06-13 오후 11:58 <DIR> 정보보호 수업자료
2019-03-30 오후 05:16 <DIR> 조사한 이미지
2019-04-10 오후 02:10 <DIR> 컴활
6개 파일 1,120,069 바이트
13개 디렉터리 90,718,728,192 바이트 남음

C:\w제혁>python
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul 8 2019, 19:29:22) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more
>>>
```

```
>>> import mod1
>>> mod1.sum(2,3)
5
```

```
def sum(a,b):  
    return a+b  
def safe_sum(a,b):  
    if type(a) != type(b):  
        print("못더함")  
        return  
    else:  
        result = sum(a,b)  
        return result  
  
print(safe_sum('a',1))  
print(safe_sum(1,14))  
print(safe_sum(10,1.2))
```

```
C:\뽕제혁>python mod1.py  
못더함  
None  
15  
못더함  
None  
C:\뽕제혁>
```

한번에 실행

```
C:\#제혁>python
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul 8 2019, 19:29:22) [MSC v.1916 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> import mod1
>>>
```

```
C:\#제혁>python mod1.py
못더함
None
15
못더함
None

C:\#제혁>
```



```
if __name__=="__main__":  
    print(safe_sum('a',1))  
    print(safe_sum(1,14))  
    print(safe_sum(10,1.2))
```

```
C:\₩제혁>python  
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul 8 2019, 19:29:22) [MSC v.1916 32 bit (Intel)] on win32  
Type "help", "copyright", "credits" or "license" for more information.  
>>> import mod1  
>>>
```

->실행할때 `__name__` 이라는 변수를 `__main__` 으로 세팅을 한다.

3 패키지

패키지는 도트(.)를 이용해 모듈을 계층적으로 관리할 수 있게해주는것

A.B는 A가패키지명이 되고 B는 A 패키지의 B 모듈이 된다

3 패키지

이름	수정된 날짜	유형	크기
Download	2019-03-30 오후...	파일 폴더	
Intel	2018-12-19 오후...	파일 폴더	
Nexon	2019-04-12 오후...	파일 폴더	
PerfLogs	2018-09-15 오후...	파일 폴더	
Program Files	2019-06-20 오후...	파일 폴더	
Program Files (x86)	2019-06-20 오후...	파일 폴더	
Python	2019-08-07 오후...	파일 폴더	
Riot Games	2019-06-26 오후...	파일 폴더	
Temp	2019-08-01 오후...	파일 폴더	
User_manual	2018-12-19 오후...	파일 폴더	
Windows	2019-08-05 오후...	파일 폴더	
사용자	2019-03-29 오후...	파일 폴더	
재혁	2019-09-06 오후...	파일 폴더	
rhdssetup.log	2018-12-19 오후...	텍스트 문서	3KB
Setup.log	2018-12-19 오후...	텍스트 문서	1KB

내 PC > 로컬 디스크 (C:) > Python	
이름	수
game	20

내 PC > 로컬 디스크 (C:) > Python > game	
이름	수정된 날짜
graphic	2019-08-07 오후...
sound	2019-08-07 오후...
__init__.py	2019-08-07 오후...

내 PC > 로컬 디스크 (C:) > Python > game > graphic		
이름	수정된 날짜	유형
__init__.py	2019-08-07 오후...	Pythor
render.py	2019-08-07 오후...	Pythor

내 PC > 로컬 디스크 (C:) > Python > game > sound		
이름	수정된 날짜	유형
__init__.py	2019-08-07 오후...	Pythor
echo.py	2019-08-07 오후...	Pythor

```
def echo_test():  
    print("echo")
```

```
def render_test():  
    print("render")
```

3 패키지

```
C:\>set PYTHONPATH=C:/Python
```

```
C:\>python
```

```
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul 8 2019, 19:29:22) [MSC v.1916 32 bit (Intel)] on win32  
Type "help", "copyright", "credits" or "license" for more information.
```

game 패키지를 참조할 수 있도록
set 명령으로 PYTHONPATH 환경변수에 C:/Python 디렉터리를 추가

3 패키지

패키지 안의 함수 실행하기

```
>>> import game.sound.echo
>>> game.sound.echo.echo_test()
echo
```

```
>>> from game.sound import echo
>>> echo.echo_test()
echo
```

```
>>> from game.sound.echo import echo_test
>>> echo_test()
echo
```

```
>>> import game
>>> game.sound.echo.echo_test
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: module 'game' has no attribute 'sound'
```

Import game 하면 game안의 파일만 사용가능하다

```
>>> import game.sound.echo.echo_test
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ModuleNotFoundError: No module named 'game.sound.echo.echo_test'
```

Import 의 마지막 항목은
반드시 패키지나 모듈이어야 한다

3 패키지

relative 패키지

 render.py - C:\Python\game\graphic\render.py

File Edit Format Run Options Window Help

```
from game.sound.echo import echo_test
```

```
def render_test():  
    print("render")  
    echo_test()
```

```
>>> from game.graphic.render import render_test  
>>> render_test()  
render  
echo
```

다른 디렉터리의 모듈이 또다른 디렉터리의 모듈을 사용하고 싶을때
relative를 사용하면 된다

The slide features a white background with a green border. In the top-left corner, there is a green triangle pointing downwards. In the bottom-right corner, there is a green triangle pointing upwards.

Q&A