
Python

Class/Module/Package

클래스 모듈 패키지

클래스 변수/함수

클래스

```
In [1]: # 클래스 변수 : secret
class Service:
    secret = "영구는 배꼽이 두 개다."
```

```
In [2]: pey = Service()
pey.secret
```

Out [2]: '영구는 배꼽이 두 개다.'

```
In [6]: # 클래스 함수 : sum()
class Service:
    secret = "영구는 배꼽이 두 개다."
    def sum(self, a, b):
        result = a + b
        print("%s + %s = %s입니다." % (a, b, result))
```

```
In [9]: pey = Service()
pey.sum(1,1)

1 + 1 = 2입니다.
```

```
In [11]: # pey.sum(pey, 1, 1) : self는 호출시 이용했던 인스턴스(pey)로 바뀐다.
Service.sum(pey, 1,1)

1 + 1 = 2입니다.
```

```
In [11]: # pey.sum(pey, 1, 1) : self는 호출시 이용했던 인스턴스(pey)로 바뀐다.  
Service.sum(pey, 1, 1)  
  
1 + 1 = 2입니다.
```

```
In [13]: # pey라는 아이디와 홍길동이라는 이름을 연결해 주는 것이 바로 self이다.  
class Service:  
    secret = "영구는 배꼽이 두 개다."  
    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))
```

```
In [14]: pey = Service()  
pey.setname("홍길동")  
pey.sum(1, 1)  
  
홍길동님 1 + 1 = 2입니다.
```

```
In [15]: pey = Service()  
         pey.sum(1,1)
```

```
AttributeError                                Traceback (most recent call last)  
<ipython-input-15-83dafb30d622> in <module>()  
      1 pey = Service()  
----> 2 pey.sum(1,1)  
  
<ipython-input-13-a738c36ea69e> in sum(self, a, b)  
      5     def sum(self, a, b):  
      6         result = a + b  
----> 7         print("%s님 %s + %s = %s입니다." %(self.name,a, b, result))  
  
AttributeError: 'Service' object has no attribute 'name'
```

```
In [16]: class Service:  
         secret = "연구는 배급이 두 개다."  
         def __init__(self, name):  
             self.name = name  
         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))
```

```
In [17]: # __init__ : 인스턴스를 만들때 항상 실행된다.  
         pey = Service("홍길동")  
         pey.sum(1,1)
```

홍길동님 1 + 1 = 2입니다.

클래스의 구조

```
class 클래스이름[(상속 클래스명)]:  
    <클래스 변수 1>  
    <클래스 변수 2>  
    ...  
    <클래스 변수 N>  
  
    def 클래스함수1(self[, 인수1, 인수2,...]):  
        <수행할 문장 1>  
        <수행할 문장 2>  
        ...  
    def 클래스함수2(self[, 인수1, 인수2,...]):  
        <수행할 문장1>  
        <수행할 문장2>  
        ...  
    def 클래스함수N(self[, 인수1, 인수2,...]):  
        <수행할 문장1>  
        <수행할 문장2>  
        ...
```

사칙연산 클래스 만들기

클래스

```
In [18]: class FourCal:
         pass # 아무것도 수행하지 않는 문법. 임시로 코드를 작성할 때 주로 사용
```

```
In [19]: a = FourCal()
         type(a) # 객체의 타입을 출력
```

```
Out [19]: __main__.FourCal
```

```
In [20]: class FourCal:
         def setdata(self, first, second):
             self.first = first
             self.second = second
```

```
In [23]: a = FourCal()
         a.setdata(4,2)
         print(a.first)
```

4

```
In [24]: print(a.second)
```

2

```
In [25]: b = FourCal()
         b.setdata(3,7)
         print(b.first)
```

3

```
In [26]: print(b.second)
```

7

더하기 기능 만들기

클래스

```
In [27]: class FourCal:
        def setdata(self, first, second):
            self.first = first
            self.second = second
        def sum(self):
            result = self.first + self.second
            return result
```

```
In [28]: a = FourCal()
        a.setdata(4,2)
        print(a.sum())
```

6

곱하기, 빼기, 나누기 기능 만들기

클래스

```
In [29]: class FourCal:
    def setdata(self, first, second):
        self.first = first
        self.second = second
    def sum(self):
        result = self.first + self.second
        return result
    def mul(self):
        result = self.first * self.second
        return result
    def sub(self):
        result = self.first - self.second
        return result
    def div(self):
        result = self.first / self.second
        return result
```

```
In [30]: a = FourCal()
b = FourCal()
a.setdata(4,2)
b.setdata(3,7)
a.sum()
```

Out [30]: 6

```
In [31]: a.mul()
```

Out [31]: 8

곱하기, 빼기, 나누기 기능 만들기

클래스

```
In [32]: a.sub()
```

```
Out [32]: 2
```

```
In [33]: a.div()
```

```
Out [33]: 2.0
```

```
In [34]: b.sum()
```

```
Out [34]: 10
```

```
In [35]: b.mul()
```

```
Out [35]: 21
```

```
In [36]: b.sub()
```

```
Out [36]: -4
```

```
In [38]: b.div()
```

```
Out [38]: 0.42857142857142855
```

HousePark 클래스 만들기

클래스

```
In [39]: class HousePark:
         lastname = "박"
```

```
In [40]: pey = HousePark()
         pex = HousePark()
         print(pey.lastname)
```

박

```
In [41]: print(pex.lastname)
```

박

```
In [42]: class HousePark:
         lastname = "박"
         def setname(self, name):
             self.fullname = self.lastname + name
```

```
In [43]: pey = HousePark()
         pey.setname("응용")
         print(pey.fullname)
```

박응용

HousePark 클래스 만들기

클래스

```
In [44]: class HousePark:
          lastname = "박"
          def setname(self, name):
              self.fullname = self.lastname + name
          def travel(self, where):
              print("%s, %s여행을 가다." % (self.fullname, where))
```

```
In [45]: pey = HousePark()
          pey.setname("응용")
          pey.travel("부산")
```

박응용, 부산여행을 가다.

초기값 설정하기

클래스

```
In [46]: pey = HousePark()  
         pey.travel("부산")
```

```
AttributeError                                Traceback (most recent call last)  
<ipython-input-46-2574e615b80b> in <module>()  
      1 pey = HousePark()  
----> 2 pey.travel("부산")  
  
<ipython-input-44-04d8a13e6bff> in travel(self, where)  
      4     self.fullname = self.lastname + name  
      5     def travel(self, where):  
----> 6         print("%s, %s여행을 가다." %(self.fullname, where))  
  
AttributeError: 'HousePark' object has no attribute 'fullname'
```

초기값 설정하기

클래스

```
In [47]: class HousePark:
          lastname = "박"
          def __init__(self, name):
              self.fullname = self.lastname + name
          def travel(self, where):
              print("%s, %s여행을 가다." % (self.fullname, where))
```

```
In [48]: pey = HousePark()
```

```
-----
TypeError                                 Traceback (most recent call last)
<ipython-input-48-1fc05c75a068> in <module>()
----> 1 pey = HousePark()

TypeError: __init__() missing 1 required positional argument: 'name'
```

```
In [49]: pey = HousePark("응용")
          pey.travel("태국")
```

박응용, 태국여행을 가다.

클래스 상속/오버라이딩

클래스

```
In [50]: # 클래스의 상속
class HouseKim(HousePark):
    lastname = "김"
```

```
In [51]: juliet = HouseKim("줄리엣")
juliet.travel("독도")
```

김줄리엣, 독도여행을 가다.

```
In [52]: # 메서드 오버라이딩
class HouseKim(HousePark):
    lastname = "김"
    def travel(self, where, day):
        print("%s, %s여행을 %d일 가다." %(self.fullname, where, day))
```

```
In [53]: juliet = HouseKim("줄리엣")
juliet.travel("독도",3)
```

김줄리엣, 독도여행을 3일 가다.

연산자 오버로딩(Overloading)

클래스

```
In [54]: # 연산자 오버로딩 : 연산자(+,-,*,/..)를 객체끼리 사용할 수 있게 하는 기법
class HousePark:
    lastname = "박"
    def __init__(self, name):
        self.fullname = self.lastname + name
    def travel(self, where):
        print("%s, %s여행을 가다." % (self.fullname, where))
    def love(self, other):
        print("%s, %s 사랑에 빠졌네" % (self.fullname, other.fullname))
    def __add__(self, other):
        print("%s, %s 결혼했네" % (self.fullname, other.fullname))

class HouseKim(HousePark):
    lastname = "김"
    def travel(self, where, day):
        print("%s, %s여행을 %d일 가다." % (self.fullname, where, day))

pey = HousePark("응용")
juliet = HouseKim("줄리엣")
pey.love(juliet)
pey + juliet # __add__ 함수가 호출(객체끼리 더한다)
```

박응용, 김줄리엣 사랑에 빠졌네
박응용, 김줄리엣 결혼했네

HousePark 클래스 완성하기

클래스

```
In [57]: class HousePark:
    lastname = "박"
    def __init__(self, name):
        self.fullName = self.lastname + name
    def travel(self, where):
        print("%s, %s여행을 가다." % (self.fullName, where))
    def love(self, other):
        print("%s, %s 사랑에 빠졌네" % (self.fullName, other.fullName))
    def fight(self, other):
        print("%s, %s 싸우네" % (self.fullName, other.fullName))
    def __add__(self, other):
        print("%s, %s 결혼했네" % (self.fullName, other.fullName))
    def __sub__(self, other):
        print("%s, %s 이혼했네" % (self.fullName, other.fullName))

    pey = HousePark("응용")
    juliet = HouseKim("줄리엣")
    pey.travel("부산")
    juliet.travel("부산",3)
    pey.love(juliet)
    pey + juliet
    pey.fight(juliet)
    pey - juliet
```

박응용, 부산여행을 가다.
김줄리엣, 부산여행을 3일 가다.
박응용, 김줄리엣 사랑에 빠졌네
박응용, 김줄리엣 결혼했네
박응용, 김줄리엣 싸우네
박응용, 김줄리엣 이혼했네

- 다음과 같이 동작하는 클래스 Calculator를 작성해 보자.

```
In [4]: cal1 = Calculator([1,2,3,4,5])
        print(cal1.sum())
        print(cal1.avg())

        cal2 = Calculator([6,7,8,9,10])
        print(cal2.sum())
        print(cal2.avg())
```

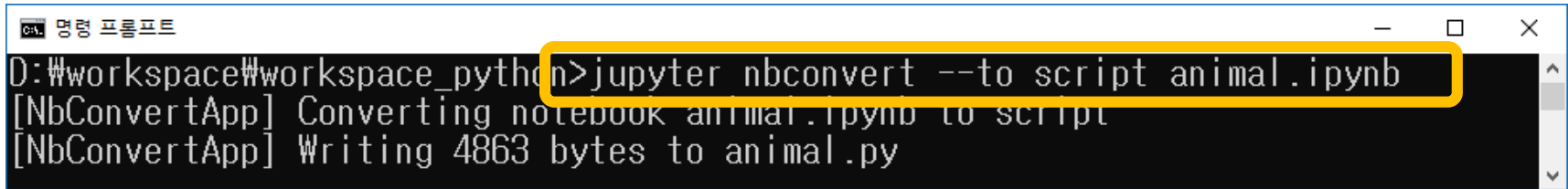
```
15
3.0
40
8.0
```

클래스 모듈 패키지

주피터 모듈 import준비

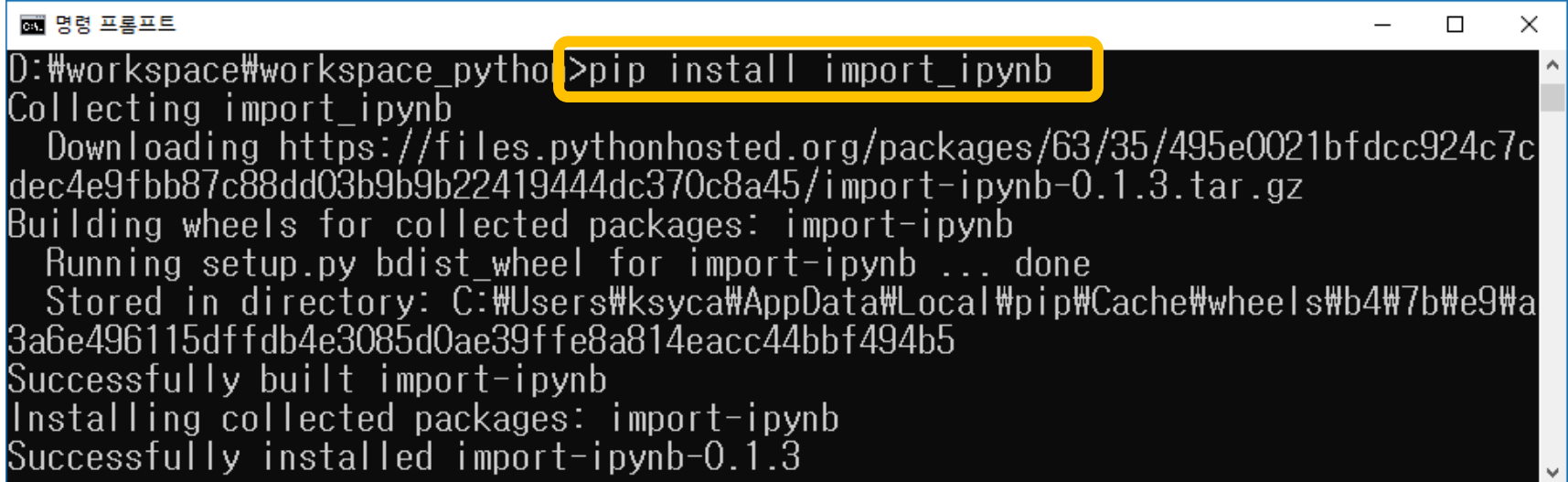
모듈

- *.ipynb를 *.py로 변경
 - jupyter nbconvert --to script file_name.ipynb



```
명령 프롬프트
D:\workspace\workspace_python>jupyter nbconvert --to script animal.ipynb
[NbConvertApp] Converting notebook animal.ipynb to script
[NbConvertApp] Writing 4863 bytes to animal.py
```

- import_ipynb 설치
 - pip install import_ipynb



```
명령 프롬프트
D:\workspace\workspace_python>pip install import_ipynb
Collecting import_ipynb
  Downloading https://files.pythonhosted.org/packages/63/35/495e0021bfdcc924c7cdec4e9fbb87c88dd03b9b9b22419444dc370c8a45/import-ipynb-0.1.3.tar.gz
Building wheels for collected packages: import-ipynb
  Running setup.py bdist_wheel for import-ipynb ... done
  Stored in directory: C:\Users\ksyca\AppData\Local\pip\Cache\wheels\b4\7b\9a3a6e496115dffdb4e3085d0ae39ffe8a814eacc44bbf494b5
Successfully built import-ipynb
Installing collected packages: import-ipynb
Successfully installed import-ipynb-0.1.3
```

- python_lecture52.ipynb

```
In [3]: # python_lecture52.ipynb
def sum(a, b):
    return a + b
```

- python_lecture53.ipynb

```
In [6]: # python_lecture53.ipynb
import import_ipynb
import python_lecture52
print(python_lecture52.sum(3,4))
```

7

- python_lecture52.ipynb

```
In [3]: # python_lecture52.ipynb
def sum(a, b):
    return a + b
```

```
In [4]: def safe_sum(a, b):
        if type(a) != type(b):
            print("더할수 있는 것이 아닙니다.")
            return
        else:
            result = sum(a, b)
            return result
```

- python_lecture53.ipynb

```
In [1]: # python_lecture53.ipynb
import import_ipynb
import python_lecture52
print(python_lecture52.sum(3,4))
```

importing Jupyter notebook from python_lecture52.ipynb
7

```
In [2]: print(python_lecture52.safe_sum(3, 4))
```

7

```
In [3]: print(python_lecture52.sum(20,30))
```

50

sum, safe_sum 함수처럼 쓰고 싶은 경우

모듈

```
In [8]: from python_lecture52 import sum  
sum(3,4)
```

Out [8]: 7

```
In [9]: from python_lecture52 import sum, safe_sum  
safe_sum(3,4)
```

Out [9]: 7

```
In [10]: from python_lecture52 import *
```

import 수행시 모듈이 실행?

모듈

- python_lecture52.ipynb

```
In [1]: def sum(a, b):  
        return a+b  
  
        def safe_sum(a, b):  
            if type(a) != type(b):  
                print("더할수 있는 것이 아닙니다.")  
                return  
            else:  
                result = sum(a, b)  
                return result  
  
        print(safe_sum('a', 1))  
        print(safe_sum(1, 4))  
        print(sum(10, 10.4))  
        print(__name__ )
```

```
더할수 있는 것이 아닙니다.  
None  
5  
20.4  
__main__
```

- python_lecture54.ipynb

```
In [1]: # python_lecture54.ipynb  
import import_ipynb  
import python_lecture52
```

```
importing Jupyter notebook from python_lecture52.ipynb  
더할수 있는 것이 아닙니다.  
None  
5  
20.4  
python_lecture52
```

C:\python_workspace>python python_lecture52.py
더할 수 있는 것이 아닙니다.

None

5

20.4

__main__

C:\python_workspace>python
>>> import python_lecture52
더할 수 있는 것이 아닙니다.

None

5

20.4

python_lecture52

- 파이썬은 자동으로 실행되는 메인함수가 없다.
- 대신 들여쓰기 하지 않은 모든 코드(level 0 코드)를 실행한다.
- `__name__`은 현재 모듈의 이름을 담고 있는 내장변수
 - import로 모듈을 가져왔을 때 모듈의 이름으로 설정된다.
 - 모듈이 직접 실행하면 “`__name__`”은 “`__main__`”으로 설정된다.

```
In [1]: # python_lecture54.ipynb
import import_ipynb
import python_lecture52
print(__name__)
```

```
importing Jupyter notebook from python_lecture52.ipynb
더할수 있는 것이 아닙니다.
```

```
None
```

```
5
```

```
20.4
```

```
python_lecture52
```

```
__main__
```

if __name__ == "__main__":

모듈

```
In [1]: # python_lecture54.ipynb
def sum(a, b):
    return a+b

def safe_sum(a, b):
    if type(a) != type(b):
        print("더할수 있는 것이 아닙니다.")
        return
    else:
        result = sum(a, b)
        return result

if __name__ == "__main__":
    print(safe_sum('a', 1))
    print(safe_sum(1, 4))
    print(sum(10, 10.4))
    print(__name__ )
```

더할수 있는 것이 아닙니다.

None

5

20.4

__main__

if __name__ == "__main__":

모듈

```
In [1]: # python_lecture54.ipynb
import import_ipynb
import python_lecture52
print(__name__ )
```

```
importing Jupyter notebook from python_lecture55.ipynb
__main__
```

```
In [1]: # python_lecture56
PI = 3.141592

class Math:
    def solv(self, r):
        return PI * (r ** 2)

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

if __name__ == "__main__":
    print(PI)
    a = Math()
    print(a.solv(2))
    print(sum(PI , 4.4))
```

```
3.141592
12.566368
7.5415920000000005
```

```
In [3]: import import_ipynb
import python_lecture56
print(python_lecture56.PI)
```

3.141592

```
In [4]: a = python_lecture56.Math()
print(a.solv(2))
```

12.566368

```
In [5]: print(python_lecture56.sum(python_lecture56.PI, 4.4))
```

7.5415920000000005

```
In [6]: result = python_lecture56.sum(3, 4)
print(result)
```

7

모듈 불러오는 다른 방법

모듈

\$jupyter_home\Testmodule 디렉터리를 생성한후 python_lecture56.py 모듈을 저장한 후 다음을 따라 해보자.

```
In [8]: import sys
        sys.path
```

```
Out[8]: ['',
         'c:\\\\users\\\\user\\\\appdata\\\\local\\\\programs\\\\python\\\\python36\\\\python36.zip',
         'c:\\\\users\\\\user\\\\appdata\\\\local\\\\programs\\\\python\\\\python36\\\\DLLs',
         .....,
         'C:\\\\Users\\\\user\\\\.ipython']
```

```
In [13]: sys.path.append("G:\\python_workspace\\jupyter\\Testmodule")
        sys.path
```

```
Out[13]: ['',
         'c:\\\\users\\\\user\\\\appdata\\\\local\\\\programs\\\\python\\\\python36\\\\python36.zip',
         'c:\\\\users\\\\user\\\\appdata\\\\local\\\\programs\\\\python\\\\python36\\\\DLLs',
         .....,
         'C:\\\\Users\\\\user\\\\.ipython',
         'G:\\python_workspace\\jupyter\\Testmodule',
         'G:\\python_workspace\\jupyter\\Testmodule']
```

```
In [12]: import import_ipynb
        import python_lecture56
        print(python_lecture56.sum(3,4))
```

클래스 모듈 패키지

패키지란?

- 도트(.)를 이용하여 파이썬 모듈을 계층적(디렉터리 구조)으로 관리할 수 있게 해준다.
- 파이썬 패키지는 디렉터리와 파이썬 모듈로 이루어지며 구조는 오른쪽과 같다.
- 패키지 구조로 파이썬 프로그램을 만드는 것이 공동 작업이나 유지 보수 등 여러 면에서 유리하다.
- 또한 패키지 구조로 모듈을 만들면 다른 모듈과 이름이 겹치더라도 더 안전하게 사용할 수 있다.

```
game/  
  __init__.py  
  sound/  
    __init__.py  
    echo.py  
    wav.py  
  graphic/  
    __init__.py  
    screen.py  
    render.py  
  play/  
    __init__.py  
    run.py  
    test.py
```


- 디렉토리와 파일 생성

```
/game/__init__.ipynb  
/game/sound/__init__.ipynb  
/game/sound/echo.ipynb  
/game/graphic/__init__.ipynb  
/game/graphic/render.ipynb
```

- echo.ipynb

```
In [2]: def echo_test():  
        print ("echo")
```

- render.ipynb

```
In [1]: def render_test():  
        print ("render")
```

- __init__.ipynb

```
In [ ]:
```

패키지 안의 함수 실행하기

패키지

```
In [7]: import import_ipynb
import game.sound.echo
game.sound.echo.echo_test()
```

echo

```
In [8]: from game.sound import echo
echo.echo_test()
```

echo

```
In [9]: from game.sound.echo import echo_test
echo_test()
```

echo

불가능한 코드(신규 파일 생성후 실행)

패키지

```
In [1]: # 불가능한 코드
import import_ipynb
import game
game.sound.echo.echo_test()
```

```
-----
AttributeError                                Traceback (most recent call last)
<ipython-input-1-3c9da701caf1> in <module>()
      2 import import_ipynb
      3 import game
----> 4 game.sound.echo.echo_test()
```

AttributeError: module 'game' has no attribute 'sound'

```
In [1]: import import_ipynb
import game.sound.echo.echo_test
```

importing Jupyter notebook from G:\python_workspace\jupyter\game\sound\echo.ipynb

```
-----
ModuleNotFoundError                          Traceback (most recent call last)
<ipython-input-1-0875a565ec1b> in <module>()
      1 import import_ipynb
----> 2 import game.sound.echo.echo_test
```

ModuleNotFoundError: No module named 'game.sound.echo.echo_test'; 'game.sound.echo' is not a package

- `__init__.py` 파일은 해당 디렉터리가 패키지의 일부임을 알려주는 역할을 한다.
- 만약 `game`, `sound`, `graphic` 등 패키지에 포함된 디렉터리에 `__init__.py` 파일이 없다면 패키지로 인식되지 않는다.
- python3.3 버전부터는 `__init__.py` 파일 없이도 패키지로 인식이 된다(PEP 420).
 - <https://www.python.org/dev/peps/pep-0420/>
- 하지만 하위 버전 호환을 위해 `__init__.py` 파일을 생성하는 것이 안전한 방법이다.

```
In [1]: '''
특정 디렉터리의 모듈을 *를 이용하여 import할 때에는
해당 디렉터리의 __init__.py 파일에 __all__이라는 변수를 설정하고
import할 수 있는 모듈을 정의해 주어야 한다.
'''

import import_ipynb
from game.sound import *
echo.echo_test()
```

NameError Traceback (most recent call last)

<ipython-input-1-5bbefd8786f0> in <module>()

1 import import_ipynb

2 from game.sound import *

—> 3 echo.echo_test()

NameError: name 'echo' is not defined

```
In [2]: ''' /game/sound/__init__.ipynb(py)
sound 디렉터리에서 * 기호를 이용하여 import할 경우
이곳에 정의된 echo 모듈만 import된다는 의미
'''

__all__ = ['echo']
```

```
In [1]: import import_ipynb
        from game.sound.echo import echo_test

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



```
In [1]: import import_ipynb
        from ..sound.echo import echo_test

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

```
In [1]: # 신규파일에서 실행
        import import_ipynb
        from game.graphic.render import render_test
        render_test()
```

```
importing Jupyter notebook from G:\python_workspace\jupyter\game\graphic\render.ipynb
importing Jupyter notebook from G:\python_workspace\jupyter\game\sound\echo.ipynb
render
```