



---

Python

# 패키지/예외처리

---

참고서적 : Do it! 점프 투 파이썬 | 박응용지음 | 이지스퍼블리싱

김선녕(sykim.lecture@gmail.com)

---



# 패키지 예외처리

- 도트(.)를 이용하여 파이썬 모듈을 계층적으로 관리할 수 있다
- 파이썬 패키지는 디렉터리와 파이썬 모듈로 이루어진다.
- 패키지 구조로 파이썬 프로그램을 만드는 것이 공동 작업이나 유지 보수 등 여러 면에서 유리하다
- 패키지 구조로 모듈을 만들면 다른 모듈과 이름이 겹치더라도 안전하게 사용할 수 있다

- 디렉토리와 파일 생성

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

- \_\_init\_\_.py 의 용도

- \_\_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)
- 하지만 하위 버전 호환을 위해 \_\_init\_\_.py 파일을 생성하는 것이 안전한 방법이다.

```
echo.py × render.py
game > sound > echo.py > ...
1  def echo_test():
2  |  |  print("echo")
```

```
echo.py × render.py ×
game > graphic > render.py > ...
1  def render_test():
2  |  |  print("render")
```

## echo 모듈을 import하여 실행하는 방법(3가지)

5

echo.py render.py package1.py X

package1.py > ...

```
1  # 첫번째
2  from game.sound.echo import echo_test
3  from game.sound import echo
4  import game.sound.echo
5  game.sound.echo.echo_test()
6
7  # 두번째
8  echo.echo_test()
9
10 # 세번째
11 echo_test()
12
```

터미널 문제 출력 디버그 콘솔

1: powershell



```
PS G:\workspace_script\study\python> python package1.py
echo
echo
echo
PS G:\workspace_script\study\python> █
```

# 불가능한 코드

6

```
echo.py  render.py  package1.py  package2.py X
package2.py
1  '''
2  특정 디렉토리의 모듈을 *를 이용하여 import할 때에는
3  해당 디렉토리의 __init__.py 파일에 __all__이라는 변수를 설정하고
4  import할 수 있는 모듈을 정의해 주어야 한다.
5  '''
6  from game.sound import *
7  echo.echo_test()
8
```

터미널 문제 1 출력 디버그 콘솔

1: powershell

```
PS G:\workspace_script\study\python> python package2.py
Traceback (most recent call last):
  File "package2.py", line 7, in <module>
    echo.echo_test()
NameError: name 'echo' is not defined
PS G:\workspace_script\study\python> 
```

```
game > sound > __init__.py > ...
1  '''
2  sound 디렉토리에서 *를 이용하여 import할 경우
3  이곳에 정의된 echo 모듈만 import된다는 의미
4  '''
5  __all__ = ['echo']
```

`package2.py`를 재 실행해 보면 다음과 같은 결과를 확인할 수 있다.

```
터미널  문제 1  출력  디버그 콘솔

PS G:\workspace_script\study\python> python package2.py
echo
PS G:\workspace_script\study\python> 
```

```
relative1.py X relative2.py
relative1.py > ...
1  from game.sound.echo import echo_test
2
3
4  def render_test():
5      print("render")
6      echo_test()
```

```
relative1.py relative2.py X
relative2.py > ...
1  import relative1
2  relative1.render_test()
3
```

터미널 문제 출력 디버그 콘솔

1: powershell



```
PS G:\workspace_script\study\python> python relative2.py
render
echo
PS G:\workspace_script\study\python> 
```





# 패키지 예외처리

```
1 f = open("noFile","r")
```

```
-----  
FileNotFoundError                                Traceback (most recent call last)  
<ipython-input-1-2867dc91f75f> in <module>()  
----> 1 f = open("noFile","r")
```

**FileNotFoundError:** [Errno 2] No such file or directory: 'noFile'

```
1 4/0
```

```
-----  
ZeroDivisionError                                Traceback (most recent call last)  
<ipython-input-2-221068dc2815> in <module>()  
----> 1 4/0
```

**ZeroDivisionError:** division by zero

```
1 a = [1,2,3]  
2 a[4]
```

```
-----  
IndexError                                        Traceback (most recent call last)  
<ipython-input-3-59fd82b1af02> in <module>()  
      1 a = [1,2,3]  
----> 2 a[4]
```

**IndexError:** list index out of range

```
try:  
    ...  
except:  
    ...
```

```
try:  
    ...  
except 발생 오류:  
    ...
```

```
try:  
    ...  
except 발생 오류 as 오류 메시지 변수:  
    ...
```

try 블록 수행 중 오류가 발생하면 except 블록이 수행된다.  
하지만 try블록에서 오류가 발생하지 않는다면 except 블록은 수행되지 않는다.

```
1 try:
2     4/0
3 except ZeroDivisionError as e:
4     print(e)
```

division by zero

```
1 # try ... else
2 # foo.txt라는 파일이 없다면 except절이 수행되고 foo.txt파일이 있다면 else절이 수행
3 try:
4     f = open('foo.txt', 'r')
5 except FileNotFoundError as e:
6     print(str(e))
7 else:
8     data = f.read()
9     f.close()
```

[Errno 2] No such file or directory: 'foo.txt'

```
1 # try .. finally
2 # try문 수행 도중 예외발생여부 상관없이 항상 수행된다.
3 f = open('foo.txt', 'w')
4 try:
5     # 로직 수행
6     pass
7 finally:
8     print("finally")
9     f.close()
```

finally

```
1 # 오류 회피하기
2 try:
3     f = open("nofile", 'r')
4 except FileNotFoundError: # 파일이 없더라도 오류를 발생시키지 않고 통과한다
5     pass
```

```
1 # 오류 일부러 발생시키기(raise 이용)
2 class Bird:
3     def fly(self):
4         raise NotImplementedError
```

```
1 class Eagle(Bird):
2     pass
3
4 eagle = Eagle()
5 eagle.fly() # Bird class에 raise문에 의해 NotImplementedError 발생 유발
```

---

```
NotImplementedError                                Traceback (most recent call last)
<ipython-input-19-01256a26320c> in <module>()
      3
      4 eagle = Eagle()
----> 5 eagle.fly()

<ipython-input-18-feb0189ed584> in fly(self)
      2 class Bird:
      3     def fly(self):
----> 4         raise NotImplementedError
```

NotImplementedError: