

## 1. 모듈

파이썬 모듈은 전역변수, 함수 등 을 모아둔 파일 입니다.

## 2. 패키지 (= 라이브러리)

패키지는 모듈을 디렉토리형식으로 구조화한 것입니다.

### 2.1 패키지 만들기

모듈들은 넣어둔 디렉토리명이 패키지명이 됩니다.

#### 모듈 만들기

모듈에 대해 자세히 살펴보기 전에 간단한 모듈을 한번 만들어 보자.

```
# mod1.py
def add(a, b):
    return a + b

def sub(a, b):
    return a - b
```

위와 같이 add와 sub 함수만 있는 파일 mod1.py를 만들고 C:\doit 디렉터리에 저장하자. 이 mod1.py 파일이 바로 모듈이다. 지금까지 에디터로 만들어 온 파일과 다르지 않다.

※ 파이썬 확장자 .py로 만든 파이썬 파일은 모두 모듈이다.

## 모듈 불러오기

우리가 만든 mod1.py 파일, 즉 모듈을 파이썬에서 불러와 사용하려면 어떻게 해야 할까?

먼저 다음과 같이 명령 프롬프트 창을 열고 mod1.py를 저장한 디렉터리(이 책에서는 C:\doit)로 이동한 다음 대화형 인터프리터를 실행한다.

※ 대화형 인터프리터를 실행할 때 나타나는 버전 정보 등의 메시지는 생략했다.

```
C:\Users\pahkey>cd C:\doit
C:\doit>dir
...
2014-09-23 오후 01:53 49 mod1.py
...
C:\doit>python
>>>
```

★ 반드시 mod1.py를 저장한 C:\doit 디렉터리로 이동한 다음 예제를 진행해야 한다. 그래야만 대화형 인터프리터에서 mod1.py를 읽을 수 있다.

이제 다음과 같이 따라 해 보자.

```
>>> import mod1
>>> print(mod1.add(3, 4))
7
>>> print(mod1.sub(4, 2))
2
```

mod1.py를 불러오기 위해 import mod1 이라고 입력하였다. 실수로 import mod1.py 로 입력하지 않도록 주의하자. import는 이미 만들어 놓은 파이썬 모듈을 사용할 수 있게 해주는 명령어이다. mod1.py 파일에 있는 add 함수를 사용하기 위해서는 위 예와 같이 mod1.add 처럼 모듈 이름 뒤에 "."(도트 연산자)를 붙이고 함수 이름을 쓰면 된다.

★ ※ import는 현재 디렉터리에 있는 파일이나 파이썬 라이브러리가 저장된 디렉터리에 있는 모듈만 불러올 수 있다. 파이썬 라이브러리는 파이썬을 설치할 때 자동으로 설치되는 파이썬 모듈을 말한다.

import의 사용 방법은 다음과 같다.

```
import 모듈이름
```

여기에서 모듈 이름은 mod1.py에서 .py 확장자를 제거한 mod1만을 가리킨다.

때로는 `mod1.add`, `mod1.sub` 처럼 쓰지 않고 `add`, `sub` 처럼 모듈 이름 없이 함수 이름만 쓰고 싶은 경우도 있을 것이다. 그럴 때는 "from 모듈 이름 import 모듈 함수"를 사용하면 된다.

```
from 모듈이름 import 모듈함수
```

위 형식을 사용하면 위와 같이 모듈 이름을 붙이지 않고 바로 해당 모듈의 함수를 쓸 수 있다.

다음과 같이 따라 해 보자.

```
>>> from mod1 import add
>>> add(3, 4)
7
```

그런데 위와 같이 하면 mod1.py 파일의 add 함수만 사용할 수 있다. add 함수와 sub 함수를 둘 다 사용하고 싶다면 어떻게 해야 할까?

2가지 방법이 있다.

```
from mod1 import add, sub
```

첫 번째 방법은 위와 같이 from 모듈 이름 import 모듈 함수1, 모듈 함수2처럼 사용하는 것이다. 콤마로 구분하여 필요한 함수를 불러올 수 있다.

```
from mod1 import *
```

두 번째 방법은 위와 같이 \* 문자를 사용하는 방법이다. 07장에서 배울 정규 표현식에서 \* 문자는 "모든 것"이라는 뜻인데 파이썬에서도 마찬가지로 의미로 사용한다. 따라서 `from mod1 import *` 는 mod1.py의 모든 함수를 불러서 사용하겠다는 뜻이다.

## if \_\_name\_\_ == "\_\_main\_\_": 의 의미

이번에는 mod1.py 파일을 다음과 같이 변경해 보자.

```
# mod1.py
def add(a, b):
    return a+b

def sub(a, b):
    return a-b

print(add(1, 4))
print(sub(4, 2))
```

add(1, 4) 와 sub(4, 2) 의 결과를 출력하는 다음 문장을 추가하였다.

```
print(add(1, 4))
print(sub(4, 2))
```

위에서 작성한 mod1.py 파일은 다음과 같이 실행할 수 있다.

```
C:\doit>python mod1.py
5
2
```

그런데 이 mod1.py 파일의 add와 sub 함수를 사용하기 위해 mod1 모듈을 import할 때는 좀 이상한 문제가 생긴다. 명령 프롬프트 창에서 다음을 따라 해 보자.

```
C:\Users\pahkey> cd C:\doit
C:\doit> python
Type "help", "copyright", "credits" or "license" for more information.
>>> import mod1
5
2
```

영동하게도 import mod1을 수행하는 순간 mod1.py가 실행이 되어 결과값을 출력한다. 우리는 단지 mod1.py 파일의 add와 sub 함수만 사용하려고 했는데 말이다.


↑ 원래 import하는 순간, 지정된 모듈이 실행된다.

이러한 문제를 방지하려면 `mod1.py` 파일을 다음처럼 변경해야 한다.

```
# mod1.py
def add(a, b):
    return a+b

def sub(a, b):
    return a-b

if __name__ == "__main__":
    print(add(1, 4))
    print(sub(4, 2))
```

 `if __name__ == "__main__"` 을 사용하면 `C:\doit>python mod1.py` 처럼 직접 이 파일을 실행했을 때는 `__name__ == "__main__"` 이 참이 되어 if문 다음 문장이 수행된다. 반대로 대화형 인터프리터나 다른 파일에서 이 모듈을 불러서 사용할 때는 `__name__ == "__main__"` 이 거짓이 되어 if문 다음 문장이 수행되지 않는다.

위와 같이 수정한 후 다시 대화형 인터프리터를 열고 실행해 보자.

```
>>> import mod1
>>>
```

아무 결과값도 출력되지 않는 것을 확인할 수 있다.

### [모듈을 불러오는 또 다른 방법]

우리는 지금껏 명령 프롬프트 창을 열고 모듈이 있는 디렉터리로 이동한 다음에 모듈을 사용할 수 있었다. 이번에는 모듈을 저장한 디렉터리로 이동하지 않고 모듈을 불러와서 사용하는 방법에 대해 알아보자.

먼저 다음과 같이 이전에 만든 mod2.py 파일을 `C:\doit\mymod` 로 이동시킨다.

```
C:\Users\pahkey>cd C:\doit
C:\doit>mkdir mymod
C:\doit>move mod2.py mymod
1개 파일을 이동했습니다.
```

그리고 다음 예를 따라 해 보자.

#### 1. `sys.path.append`(모듈을 저장한 디렉터리) 사용하기

먼저 `sys` 모듈을 불러온다.

```
C:\doit>python
>>> import sys
```

`sys` 모듈은 파이썬을 설치할 때 함께 설치되는 라이브러리 모듈이다. `sys`에 대해서는 뒤에서 자세하게 다룰 것이다. 이 `sys` 모듈을 사용하면 파이썬 라이브러리가 설치되어 있는 디렉터리를 확인할 수 있다.

다음과 같이 입력해 보자.

```
>>> sys.path
['', 'C:\\Windows\\SYSTEM32\\python37.zip', 'c:\\Python37\\DLLs',
 'c:\\Python37\\lib', 'c:\\Python37', 'c:\\Python37\\lib\\site-packages']
```

`sys.path` 는 파이썬 라이브러리가 설치되어 있는 디렉터리를 보여 준다. 만약 파이썬 모듈이 위 디렉터리에 들어 있다면 모듈이 저장된 디렉터리로 이동할 필요 없이 바로 불러서 사용할 수 있다. 그렇다면 `sys.path` 에 `C:\doit\mymod` 디렉터를 추가하면 아무 곳에서나 불러 사용할 수 있지 않을까?

※ 명령 프롬프트 창에서는 `/`, `\` 든 상관 없지만, 소스 코드 안에서는 반드시 `/` 또는 `\\` 기호를 사용해야 한다.

당연하다. `sys.path`의 결괏값이 리스트이므로 우리는 다음과 같이 할 수 있다.

```
>>> sys.path.append("C:/doit/mymod")
>>> sys.path
['', 'C:\\Windows\\SYSTEM32\\python37.zip', 'c:\\Python37\\DLLs',
'c:\\Python37\\lib', 'c:\\Python37', 'c:\\Python37\\lib\\site-packages',
'C:/doit/mymod']
>>>
```

`sys.path.append`를 사용해서 `C:/doit/mymod` 라는 디렉터리를 `sys.path`에 추가한 후 다시 `sys.path`를 보면 가장 마지막 요소에 `C:/doit/mymod` 라고 추가된 것을 확인할 수 있다.

자, 실제로 모듈을 불러와서 사용할 수 있는지 확인해 보자.

```
>>> import mod2
>>> print(mod2.add(3,4))
7
```

이상 없이 불러와서 사용할 수 있다.