

## 02.파이썬 기본 문법-2

---

06. 함수

-----

**07. 모듈**

-----

08. 패키지

-----

## 7.1 모듈이란?

- 어떤 대상의 부분 혹은 조각  
예) 레고 블록, 벽돌, 자동차 부품들, LG-G5

## 7.2 파이썬의 모듈이란?

- 파이썬의 Module 은 [파일명].py 을 의미한다.
- 파이썬 파일로 변수, 함수, 클래스 등을 모아 놓은 파일이다.
- 다른 모듈에 의해서 호출되고 사용된다.
- 표준모듈, 사용자 생성 모듈, 서드 파티 모듈 등으로 나눌 수 있다.

**[실습]**

mymath.py 파일을 만들고 아래와 같이 메소드를 정의한 후 정의한 메소드를 사용해 봅니다.

[mymath.py]

```
pi = 3.14
```

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

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

```
def multi(a, b):  
    return a*b
```

```
def div(a, b):  
    return a/b
```

```
def area_circle(r):  
    return pi*r**2
```

```
print(plus(30,10))
```

```
print(minus(30,10))
```

```
print(multi(30,10))
```

```
print(div(30,10))
```

```
print(area_circle(5))
```

### 7.3 import 방법

#### import 모듈명(파일명)

import는 현재 디렉토리에 있는 파일이나 파이썬 라이브러리가 저장된 디렉토리에 있는 모듈만 불러올 수 있다.

#### [실습]

mymath 모듈의 내용을 다른 모듈에서 사용해 봅니다.

[main.py]

```
import mymath
```

```
print(mymath.plus(30,10))  
print(mymath.minus(30,10))  
print(mymath.multi(30,10))  
print(mymath.div(30,10))  
print(mymath.area_circle(5))
```

```
mymath.py
```

## 7.4 Namespace

- 모듈 내부의 **이름(변수, 함수, 클래스)**를 구분하는 역할을 한다.
- 필요한 내용만 골라서 호출 할 수 있다.
- from 과 import 키워드를 사용한다. (as 별명사용)
- import 는 모듈명 까지만 사용할 수 있다.

### 1) 모듈 전체를 import 하는 방법

#### import 모듈명

```
import mymath

print(mymath.plus(30,10))
print(mymath.minus(30,10))
print(mymath.multi(30,10))
print(mymath.div(30,10))
print(mymath.area_circle(5))
```

#### import 모듈명 as 별명

```
import mymath as mm

print(mm.plus(30,10))
print(mm.minus(30,10))
print(mm.multi(30,10))
print(mm.div(30,10))
print(mm.area_circle(5))
```

## 2) 모듈의 이름만 가져와서 사용 - 특정이름

```
from 모듈명 import 이름
```

```
from mymath import plus, minus
```

```
print(plus(30,10))
```

```
print(minus(30,10))
```

```
print(multi(30,10))
```

```
print(div(30,10))
```

```
print(area_circle(5))
```

이름을 여러 개 가져올 경우  
coma(,) 로 구분

이름을 가져오지 않아  
사용할 수 없음

## 3) 모듈의 이름만 가져와서 사용 - 모든이름

```
from 모듈명 import *
```

```
from mymath import *
```

모든 이름을 가져옴

```
print(plus(30,10))
```

```
print(minus(30,10))
```

```
print(multi(30,10))
```

```
print(div(30,10))
```

```
print(area_circle(5))
```

모든 이름을 가져와서  
이름만으로 사용가능

## 4) 별명사용

```
from 모듈명 import 이름 as 별명
```

```
from mymath import
```

모든 이름을 가져옴

```
print(plus(30,10))
```

```
print(minus(30,10))
```

```
print(multi(30,10))
```

```
print(div(30,10))
```

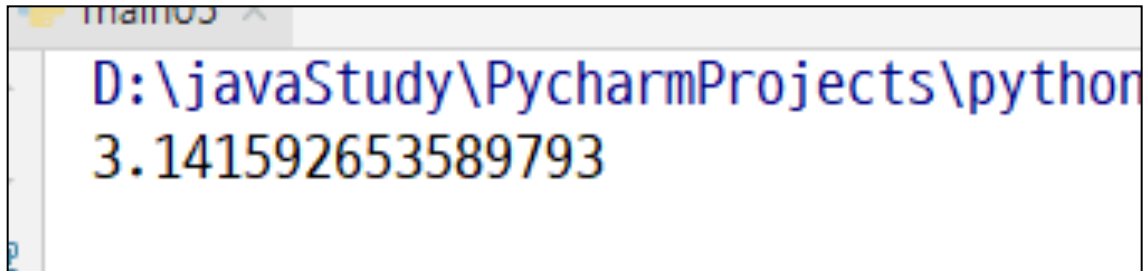
```
print(area_circle(5))
```

모든 이름을 가져와서  
이름만으로 사용가능



5) 같은 이름이 있는 경우 나중에 정의된 이름이 사용된다.

```
from mymath import pi      3.14
from math import pi      3.141592653589793
print(pi)
```



The screenshot shows a Python interpreter window with the following output:

```
D:\javaStudy\PycharmProjects\python
3.141592653589793
```

## 7.5 내장변수 `__name__` 과 모듈이름 `"__main__"`

- 모든 모듈은 모듈이름을 저장하고 있는 내장 변수 `__name__` 을 가지고 있다.
- 최상위모듈(인터프리터에 의해 실행된 모듈)의 `__name__` 변수의 값은 `"__main__"` 이다.

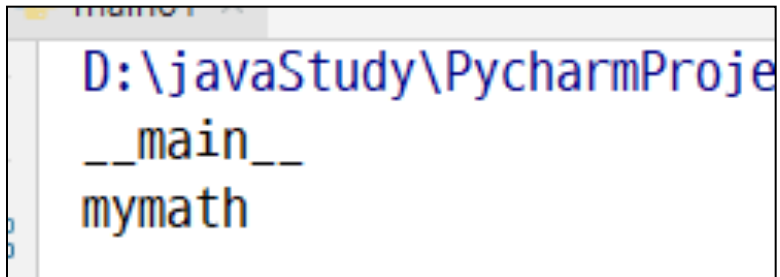
[main.py]

```
import mymath
```

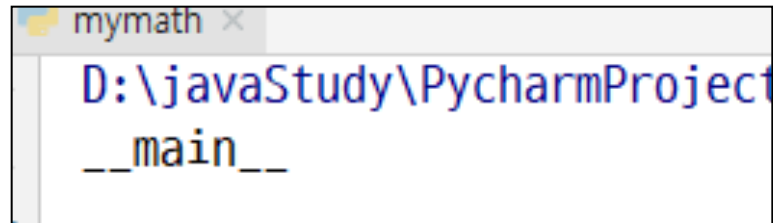
```
print(__name__)  
print(mymath.__name__)
```

[mymath.py]

```
print(__name__)
```



```
D:\javaStudy\PycharmProject  
__main__  
mymath
```



```
mymath x  
D:\javaStudy\PycharmProject  
__main__
```

[mymath.py]

[함수정의 부분 생략]

...

**if \_\_name\_\_ == "\_\_main\_\_" :**

```
print(area_circle(5))  
print(plus(30,10))  
print(minus(30,10))  
print(multi(30,10))  
print(div(30,10))  
print(area_circle(5))  
print(__name__)
```

**if 구문은**  
최상위모듈  
(인터프리터에 의해 실행된 모듈)  
일때만 실행된다.