```
def f(ham: str, eggs: str = 'eggs') -> str:
    print("Annotations:", f.__annotations__)
    print("Arguments:", ham, eggs)
    return ham + ' and ' + eggs

f('spam')
# 여기서 annotation이 그냥 설명을 위한 추가적인 기능인지?
```

```
a = [1,2,3,4]
a.extend([1,2,3,4])
# append와 extend의 차이점은?
```

```
# 그러면 import 시에 submodule까지 dot notation으로 불러오는 이유는 쓰기
편하게 하기 위해서?
# 아니면 submodule 중에서 겹치는 경우가 존재하기 때문에?
# 이 설명이 맞는지?
def scope_test():
    def do_local():
        spam = "local spam"
        # 밖의 spam에 접근 못하므로 그대로 test spam
    def do_nonlocal():
        nonlocal spam
        spam = "nonlocal spam"
        # nonlocal로 bind하면 가장 가까운 scope로 re-bound 된다
        # 이때는 바깥의 spam이랑 bound됨
    def do_global():
        global spam
        spam = "global spam"
        # 이미 nonlocal에 bind되어서
        # global로 bind가 된다

    spam = "test spam"
    do_local()
    print("After local assignment:", spam)
    do_nonlocal()
    print("After nonlocal assignment:", spam)
    do_global()
    print("After global assignment:", spam)
    # 여기서는 global에 접근이 되기 전에
    # 함수 내의 spam에 접근해서 출력한다

scope_test()
print("In global scope:", spam)

# 또한 이해가 잘 안되는 말
# global 대입은 모듈 수준의 연결을 바꿉니다
```

박준형