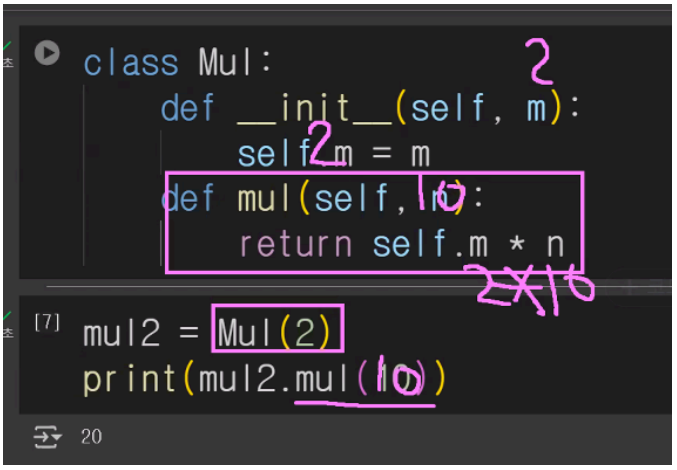
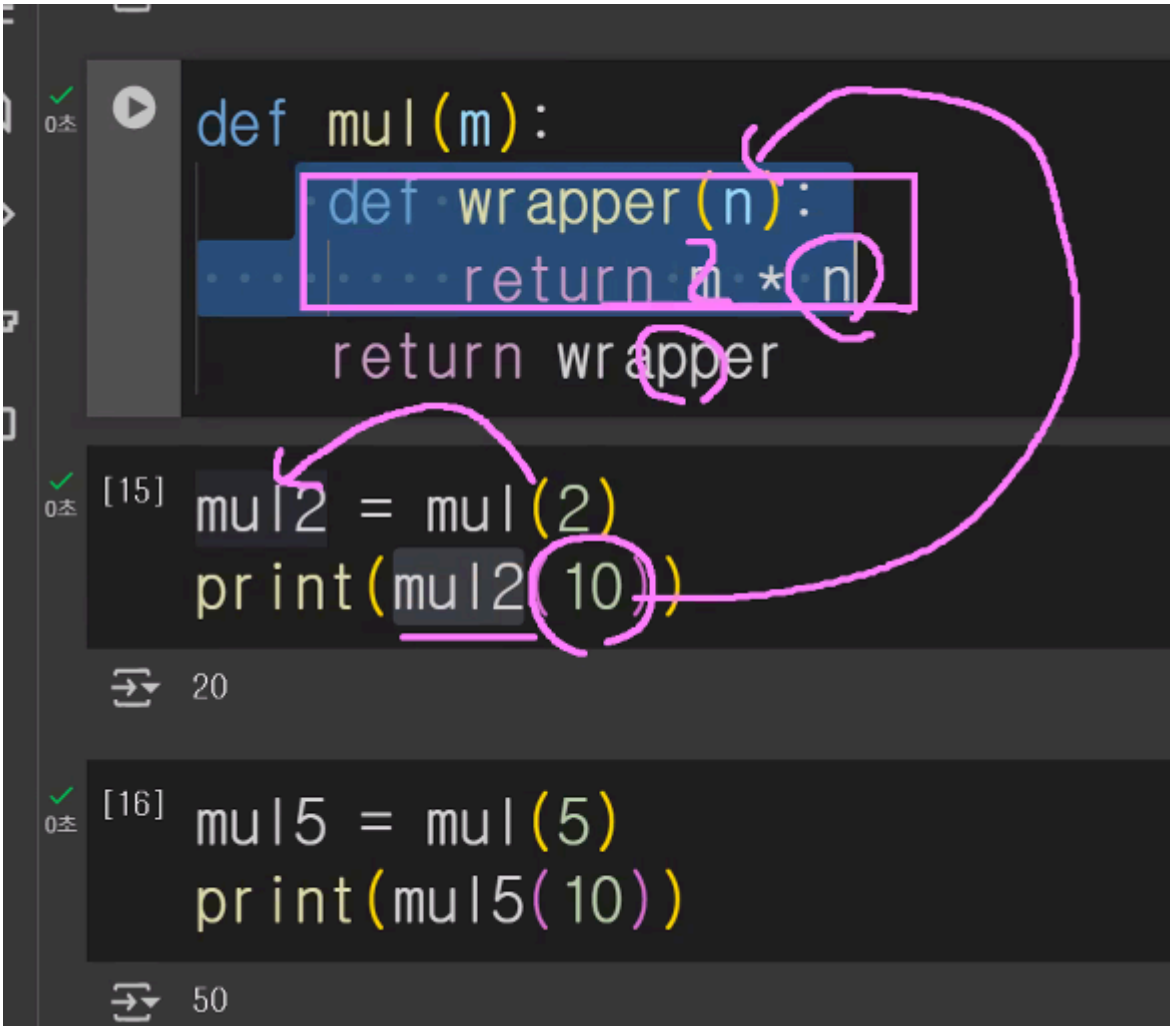
Day 2 - Closure and Decorator

Closure

- 외부 함수의 변수를 참조하는 내부 함수
- 외부 함수가 호출된 후에도 그 변수의 상태를 기억하고 사용할 수 있는 기능으로, 주로 데이터 은닉과 상태 유지에 활용됩니다.



- 클라스에서 활용법



Decorater

- **데코레이터**는 **기존 함수를 수정하지 않고 기능을 추가**하는 함수입니다. 다른 함수를 인자로 받아서 새로운 함수를 반환합니다. 주로 @ 기호로 사용됩니다.

ex)

```
def func1(a, b):
    result = a + b
    return result

def func2(a, b):
    result = a * b
    return result

# 함수를 보내면 감싸서 시간계산을 해주는 반환해주는 함수이다. (데코레이터)
def elapsed(func):
    def wrapper(a, b):
        start = time.time()
        print('함수가 시작되었습니다.')
        result = func(a, b)
        end = time.time()
        print(f'함수 수행시간: {end - start}')
        return result
    return wrapper
```

```
deco1 = elapsed(func1) # func1 함수가 elapsed 함수 안에 감싸져있다.
result = deco1(10, 3) # 10, 3을 매개변수로 넣으면 func1 -> wrapper 순으로 실행시킨다.
print(result)
```

```
함수가 시작되었습니다.
함수 수행시간: 5.984306335449219e-05
13
```