고급통계프로그래밍 Solution 2

컴퓨터과학부 2017920054 이호준

과제 index는 번역본 기준입니다.

Ex 3.4

```
def do_twice(f):
    f()
    f()
```

Solution 1

```
def print_spam():
    print("spam")

do_twice(print_spam)

# stdout
spam
spam
```

- `do_twice()`를 호출하면 함수의 인자로 함수 객체가 전달되어, 이 함수 내부에서 `print_spam()` 을 두 번 호출해서 결과적으로 **"spam"**이 두번 출력된다.

Solution 2

```
def do_twice(func, arg):
    func(arg)
    func(arg)
```

- `do_twice()`의 인자를 두 개로 변경하고, 기존의 f(→func) 뿐만 아니라 여기에 전달할 매개변수 arg를 선언한다.
- 함수 내부는 기존 구조와 유사하나, 기존에 인자 없이 호출하던 것과 달리 매개변수로 받아온 arg를 함께 전달해준다.

Solution 3

```
def print_twice(arg):
    print(arg)
    print(arg)

# test
print_twice("Sol 3")

# stdout
Sol 3
Sol 3
```

- 기존의 `print_spam()` 에서 출력의 인자를 함수의 인자로 받아서 할당해주는 함수 `print_twice()` 를 생성한다.

Solution 4

```
do_twice(print_twice, "spam")

# stdout
spam
spam
spam
spam
```

- 기본적으로 `do_twice()` 는 1번째 인자로 들어온 함수 객체를 2번 실행한다. 그런데 이에 해당하는 함수 `print_twice()` 는 내부에서 `print()` 를 호출한다. 따라서 2번 호출이 2회 일어나므로 2*2 = 4회 `print()` 실행이 이루어진다.

Solution 5

```
def do_four(func, arg):
    do_twice(func, arg)
    do_twice(func, arg)

do_four(print, "Sol 5")
```

- Sol. 4와 유사한 로직으로, `do_twice()` 를 2번 호출하는 함수 `do_four()` 를 만들어 주면 2줄의 코드로도 함수를 4번 실행할 수 있다.

Ex 5.3

Solution 1

```
def check_fermat(a, b, c, n):
    if (n > 2) and ((a ** n) + (b ** n) == (c ** n)):
        print("페르마가 틀렸다!")
    else:
        print("아냐, 그건 아니지.")

check_fermat(3, 4, 5, 2)

# stdout
아냐, 그건 아니지.
```

- 4개의 인자를 입력받아서 주어진 조건 - $n > 2, a^n + b^n = c^n$ - 을 만족하면 **"페르마가 틀렸다!"**를, 아니면 **"아냐, 그건 아니지."**를 출력하는 코드

Solution 2

```
def check_fermat_with_input():
    a = int(input("a를 입력하세요: "))
    b = int(input("b를 입력하세요: "))
    c = int(input("c를 입력하세요: "))
    n = int(input("n를 입력하세요: "))
    check_fermat(a, b, c, n)

check_fermat_with_input()
```

- **"...를 입력하세요: "**의 프롬프트와 함께 입력을 4번 받고, 이전에 만들어 둔 `check_fermat()`를 통해 주어진 입력 인수들이 페르마 정리의 반례에 해당하는지 확인하는 코드