Task 1: Python compilation flow.

PART 0: Introduction.

In this task, you will use the `ast` module to extract Abstract Syntax Trees from functions and modify them according to the requirement. See the lecture 3 for details.

Please install `pytest` module to run tests.

```
pip install pytest
# or pip3 if you use your global environment.
```

Now you can run tests by running:

```
# inside task1/ directory:
pytest <filename>
```

The more tests you pass, the better the grade. Happy hacking!

PART 1: If/Loop counter.

Implement a decorator `my_counter` which counts the number of `If` operators and `Loop` (`for`/`while`) operators. It must write this number as attribute of the function, e.g.

```
@my_counter
def my_function():
    if 1:
        for i in range(100):
            if i % 2 == 0:
                print(i)

print(my_function.num_loops)
# Output: 1
print(my_function.num_ifs)
# Output: 2
```

PART 2. Recursion detector.

Implement a function `has_recursion` which detects a usage of recursion inside the function. For instance:

```
def func1(i: int):
    if i > 0:
        func1(i)

def func2(i: int):
    return

def func3_1(i: int):
    func3_2(i)

def func3_2(i: int):
    if i > 0:
        func3_1(i)

has_recursion(func1)
> True
has_recursion(func2)
> False
has_recursion(func3_2)
> True
has_recursion(func3_1)
> True
```

It must detect coupled recursion of any depth.

Tests

1. Simple recursion
2. Coupled recursion
3. Large example
4. Aliases (*)

Constant expression evaluation.

Implement two decorators: `constexpr` and `eval_const_exprs`. The first one must mark the function as "pure", i.e. a function which does not change the global state and is not a closure. The second one must look for the calls of the marked functions with constant arguments, compute the result and replace this call with it.

For instance:

```
@constexpr
def f(a, b):
    return a + b
```

```
@eval_const_exprs
def my_function(a):
    return f(3, 6) + f(a, 3)

code(my_function)
> def my_function(a):
>     return 9 + f(a, 3)
```

- Mark functions decorated as `constexpr`. You can do that by simply adding a flag attribute to the function (`f.is_marked_constexpr = True`) and check it using `hasattr` call.
- Extra tests involve more advanced constant argument detection - like `f(3+3, 6)`.