Factorial Tasks

1. Create the Factorial Program

Write a Python file named factorial.py.

The program should compute the factorial of a given non-negative integer.

It must read a single integer from standard input and write the calculated factorial to standard output.

The logic for calculating the factorial should be encapsulated in a function called compute_factorial(number).

The code to read the input, call the function, and print the output should be wrapped in a main() function.

Hint: The factorial of 0 is 1. The function should raise a ValueError for negative numbers.

2. Test the Core Function

Create a new file named test_factorial.py.

Inside, create a unittest.TestCase class to test the compute_factorial function.

Write at least three different test methods:

test_valid_input: Tests that the factorial of a standard positive integer (e.g., 5) is calculated correctly.

test_invalid_input: Tests that the function correctly raises a ValueError when given a negative number.

test_boundary: Tests the boundary condition, ensuring the factorial of 0 is correctly handled.

3. Run Tests from the Command Line

Demonstrate how to use the unittest Python package from the command line to:

Run all the tests you've written.

Run only the test_boundary test individually.

Run any test whose name contains the substring "valid".

4. Test the Main Function's I/O (Classic Method)

Create a new test case class in test_factorial.py to test the main() function.

This test should verify both the program's calculation and its interaction with the console.

Emulate the console I/O by using the setUp and tearDown instance methods to substitute sys.stdin and sys.stdout with io.StringIO objects.

5. Test the Main Function's I/O (Mocking Method)

Create a final test case class that also tests the main() function.

Instead of using setUp/tearDown, use the unittest.mock.patch context manager to manage the mocking of standard input and standard output.