

## Лабораторная работа №7:

### Цель работы

1. Проверка с помощью создания контекста функций по считыванию настроек калькулятора и запись в файл результатов выполнения операций
2. Используя параметризацию тестов написать тесты функции `two_sum`.
3. Используя гипотезы с помощью `hypothesis` протестировать вычисление факториала.  
Используя параметрические тесты, проверить пограничные случаи, обработку некорректных данных при вычислении факториала.

### Комментарии по выполнению

#### Задача 1

Тестирование должно учитывать **развертывание и свертывание контекста**: подготовка временных файлов перед тестами и их корректное удаление после завершения. Тестирование можно проводить или с помощью `unittest` или с помощью `pytest`. Использовать `tempfile` (или другую для `unittest`) или `tmpdir` (для `pytest`) в тестах.

Проверьте (как минимум):

- Корректное считывание настроек из файла.
- Обработку ошибки `FileNotFoundError`.
- Обработку ошибки `ValueError` при некорректном JSON.

Для функции записи в файл проверьте:

- Корректную запись строки в файл.
- Создание файла, если он отсутствует.
- Добавление строк в существующий файл.

Модульные тесты должны быть вынесены в отдельный файл (например, `test_calculator_io.py`).

Код должен быть оформлен в соответствии со стандартом **PEP8**.  
Используйте комментарии и `docstrings` для описания функций.

#### Задача 2

Используя параметризацию тестов с использованием библиотеки `pytest`, написать тесты функции `two_sum`, которую вы реализовали ранее.

## Требования

### 1. Покрываемость тестами:

- Тесты должны быть реализованы с использованием **pytest**.
- Использовать параметризацию (`@pytest.mark.parametrize`) для проверки функции на разных входных данных.
- Обеспечить покрытие тестами следующих случаев:
  - **Базовый случай:** массив из нескольких элементов, включая положительные и отрицательные числа.
  - **Пограничные случаи:**
    - Минимальный размер массива (`len(nums) == 2`).
    - Числа в массиве с минимальными и максимальными значениями в пределах  $[-10^9, 10^9]$ .
  - **Особые случаи:**
    - Числа в массиве повторяются.
    - Все числа в массиве одинаковые.
- Тесты должны проверять как входные данные, так и корректность индексов (порядок не важен).

### 2. Тестируемая функция:

- Функция должна находиться в отдельном модуле (например, `solution.py`).
- Тесты должны быть реализованы в отдельном файле (например, `test_solution.py`).

## Задача 3

### Автоматическая генерация входных данных с Hypothesis:

- Использовать стратегию `@given` для генерации входных данных `n` в диапазоне  $0 \leq n \leq 100$ .
- Проверить, что функция корректно обрабатывает все допустимые входные значения.
- Проверить свойства факториала:
  - $(n+1)! = (n+1) \times n!$
  - Функция не возвращает отрицательных значений.

- Обработка некорректных данных: функция должна выбрасывать исключение `ValueError`, если `n` не является натуральным числом (например, отрицательное число или дробь).

## Задача 1:

### Тесты для работы с файлами

```
PS C:\Users\sambu\OneDrive\Рабочий стол\Учеба в Герцена\II курс\Программирование Python, 2 курс, 1 семестр\IP-7> pytest calculator/test_calculator.py -v
===== test session starts =====
platform win32 -- Python 3.13.1, pytest-8.3.4, pluggy-1.5.0 -- C:\Users\sambu\AppData\Local\Programs\Python\Python313\python.exe
cachedir: .pytest_cache
hypothesis profile 'default'
rootdir: C:\Users\sambu\OneDrive\Рабочий стол\Учеба в Герцена\II курс\Программирование Python, 2 курс, 1 семестр\IP-7
plugins: anyio-4.9.0, hypothesis-6.135.9
collected 5 items

calculator/test_calculator.py::test_read_settings_valid PASSED [ 20%]
calculator/test_calculator.py::test_read_settings_file_not_found PASSED [ 40%]
calculator/test_calculator.py::test_read_settings_invalid_json PASSED [ 60%]
calculator/test_calculator.py::test_write_to_file_new PASSED [ 80%]
calculator/test_calculator.py::test_write_to_file_append PASSED [100%]

===== 5 passed in 0.45s =====
```

## Задача 2:

### Тесты для функции `two_sum`

```
PS C:\Users\sambu\OneDrive\Рабочий стол\Учеба в Герцена\II курс\Программирование Python, 2 курс, 1 семестр\IP-7> pytest test_solution.py -v
===== test session starts =====
platform win32 -- Python 3.13.1, pytest-8.3.4, pluggy-1.5.0 -- C:\Users\sambu\AppData\Local\Programs\Python\Python313\python.exe
cachedir: .pytest_cache
hypothesis profile 'default'
rootdir: C:\Users\sambu\OneDrive\Рабочий стол\Учеба в Герцена\II курс\Программирование Python, 2 курс, 1 семестр\IP-7
plugins: anyio-4.9.0, hypothesis-6.135.9
collected 7 items

test_solution.py::test_two_sum_valid[nums0-9-expected_indices0] PASSED [ 14%]
test_solution.py::test_two_sum_valid[nums1-0-expected_indices1] PASSED [ 28%]
test_solution.py::test_two_sum_valid[nums2-6-expected_indices2] PASSED [ 42%]
test_solution.py::test_two_sum_valid[nums3-0-expected_indices3] PASSED [ 57%]
test_solution.py::test_two_sum_invalid[nums0-0] PASSED [ 71%]
test_solution.py::test_two_sum_invalid[nums1-1] PASSED [ 85%]
test_solution.py::test_two_sum_invalid[nums2-7] PASSED [100%]

===== 7 passed in 0.31s =====
```

## Задача 3:

### Тесты для факториала

```
PS C:\Users\sambu\OneDrive\Рабочий стол\Учеба в Герцена\II курс\Программирование Python, 2 курс, 1 семестр\IP-7> pytest test_fact.py -v
===== test session starts =====
platform win32 -- Python 3.13.1, pytest-8.3.4, pluggy-1.5.0 -- C:\Users\sambu\AppData\Local\Programs\Python\Python313\python.exe
cachedir: .pytest_cache
hypothesis profile 'default'
rootdir: C:\Users\sambu\OneDrive\Рабочий стол\Учеба в Герцена\II курс\Программирование Python, 2 курс, 1 семестр\IP-7
plugins: anyio-4.9.0, hypothesis-6.135.9
collected 5 items

test_fact.py::test_factorial_properties PASSED [ 20%]
test_fact.py::test_factorial_invalid_input[-1] PASSED [ 40%]
test_fact.py::test_factorial_invalid_input[-10] PASSED [ 60%]
test_fact.py::test_factorial_invalid_input[3.14] PASSED [ 80%]
test_fact.py::test_factorial_invalid_input[abc] PASSED [100%]

===== 5 passed in 0.40s =====
```