### ****1. Метрики размера****

**Файл:** test2.py

| Метрика | Значение |
| --- | --- |
| Количество строк кода (SLOC) | 11 |
| Количество функций | 3 |

**Пояснение:**

* Учтены только строки с кодом (исключены пустые строки и комментарии, которых в файле нет).
* Функции: calculate\_sum, factorial, print\_numbers.

### ****2. Оценка сложности кода****

#### ****Цикломатическая сложность функций****

Формула: Количество ветвлений (if, for, while) + 1

| Функция | Ветвления | Сложность |
| --- | --- | --- |
| calculate\_sum | Нет | 1 |
| factorial | if n == 0 + рекурсия (else) | 2 |
| print\_numbers | for + if i % 2 == 0 + else | 3 |

**Пример расчёта для**print\_numbers**:**

* for (1) + if (1) + else (1) = 3.

### ****3. Выводы****

1. **Самая сложная функция:** print\_numbers (сложность = 3).
   * Причина: содержит вложенные условия (if-else) и цикл (for), что увеличивает риск ошибок (например, некорректная обработка граничных значений n).
2. **Потенциальные ошибки:**
   * В factorial: отсутствует проверка на отрицательные числа (может вызвать бесконечную рекурсию).
   * В print\_numbers: если n очень большое, возможна избыточная нагрузка на вывод.
3. **Рекомендации по улучшению:**
   * Добавить проверку if n < 0 в factorial.
   * В print\_numbers ограничить вывод при больших n (например, добавить if n > 1000: return).
   * Упростить print\_numbers, разделив логику проверки чётности и вывода.