### **Теоретическая часть**

1. ****Корректность программы****
   * **Определение:** Программа корректна, если её поведение соответствует заданной спецификации (предусловиям, постусловиям, инвариантам).
   * **Отличие от надёжности:** Надёжность — это способность программы работать без сбоев в различных условиях (включая некорректные входы).  
     **Пример:** Программа, которая всегда возвращает 0 при любом вводе, надёжна (не падает), но некорректна.
2. ****Эталоны корректности****
   * **Спецификация** — формальное описание ожидаемого поведения программы. Необходима для проверки корректности.
   * **Виды спецификаций:**
     + **Функциональные требования:** Что делает программа (например, "функция возвращает среднее положительных чисел").
     + **Предусловие:** Условия перед вызовом функции (например, numbers — непустой список).
     + **Постусловие:** Гарантии после выполнения (например, возврат average >= 0).
     + **Инвариант:** Условие, истинное на каждой итерации цикла (например, total >= 0 and count >= 0).

**Пример для** calculate\_positive\_average**:**

* + **Предусловие:** numbers — список чисел (может быть пустым).
  + **Постусловие:** Возвращает average >= 0; если положительных чисел нет, возвращает 0.

1. ****Методы проверки корректности****
   * **Тестирование:**
     + **Суть:** Проверка на конкретных входных данных.
     + **Плюсы:** Простота. **Минусы:** Не покрывает все случаи.
     + **Тест-кейс:** Входные данные + ожидаемый результат.
   * **Статический анализ:**
     + **Суть:** Анализ кода без выполнения (например, проверка типов).
     + **Пример:** Инструмент mypy обнаруживает несоответствие типов.
   * **Формальные методы:**
     + **Идея:** Математическое доказательство корректности (например, инварианты цикла).
     + **Инвариант цикла:** Условие, сохраняющееся на каждой итерации (например,total = sum положительных элементов до текущего шага).

**Сравнение:**

* + Для небольшой функции подходит тестирование.
  + Для большой системы — комбинация статического анализа и формальных методов.

### **Практическая часть**

#### ****1. Спецификация функции** calculate\_positive\_average**

* **Предусловие:**
  + numbers — список чисел (тип list[int] или list[float]).
* **Постусловие:**
  + Возвращает average >= 0; если положительных чисел нет, возвращает 0.
* **Инвариант цикла:**
  + total = сумма всех положительных элементов, обработанных до текущей итерации.
  + count = количество таких элементов.

#### ****2. Ручное тестирование****

| **Тест-кейс** | **Входные данные** | **Ожидаемый результат** | **Фактический результат** | **Прошел?** |
| --- | --- | --- | --- | --- |
| Обычный случай | [1, 2, 3, -4] | 2.0 | 2.0 | Да |
| Граничный случай (нет положительных) | [-1, -2] | 0 | 0 | Да |
| Нарушение предусловия (не список) | "123" | Ошибка (TypeError) | Ошибка (AttributeError) | Нет¹ |

¹ Функция не проверяет тип входных данных и упадёт при попытке итерации по строке.

#### ****3. Анализ корректности****

* **Строка A (**average = total / count**):**  
  Деление на ноль невозможно благодаря проверке if count > 0.
* **Пустой список (**numbers = []**):**  
  Функция вернёт 0, что соответствует постусловию.
* **Потенциальная ошибка:**  
  При нарушении предусловия (например, передача строки) возникнет AttributeError.

#### ****4. Выводы****

* **Корректность:** Функция корректна относительно очевидной спецификации, но не обрабатывает некорректные входы.
* **Метод проверки:** Для обнаружения ошибки в строке A (без count > 0) эффективнее всего **статический анализ** (инструмент вроде pylint обнаружил бы потенциальное деление на ноль).