# 1. Теоретическая часть Корректность: Программа корректна, если для всех входных данных, удовлетворяющих спецификации (предусловиям), она завершает выполнение и возвращает результат, соответствующий спецификации (постусловиям). Корректность vs Надежность: Надежность — способность работать без сбоев в реальных условиях, даже при некорректных данных. Пример надёжной, но некорректной программы: калькулятор, который не падает при вводе, но возвращает неправильный результат. Спецификация: Это формальное описание того, что должна делать программа. Без спецификации нельзя точно проверить корректность. Виды спецификаций: - Функциональные требования — что делает программа; - Предусловия — что должно быть истинно до вызова; - Постусловия — что гарантируется после; - Инварианты — утверждения, сохраняющиеся в цикле. Для функции calculate\_positive\_average(numbers): Предусловие: numbers — список чисел. Постусловие: возвращает среднее положительных чисел; если их нет — 0. Методы проверки корректности: - Тестирование: запуск с примерами. Плюсы — просто. Минусы — не покрывает всё. - Статический анализ: анализ кода без выполнения. Ищет ошибки типа типов, мёртвого кода. - Формальные методы: математическое доказательство. Инвариант — утверждение, сохраняющееся при каждой итерации цикла. Сравнение методов: Для простой функции — лучше ручное тестирование. Для большой системы — комбинация всех методов.

# 2. Практическая часть

## 2.1 Спецификация Предусловие: numbers — список чисел (не None, не строка). Постусловие: если положительные числа есть — вернуть их среднее; иначе вернуть 0. Инвариант: на каждой итерации total — сумма положительных чисел, просмотренных до текущего, count — их количество.

## Таблица тест-кейсов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Входные данные | Ожидаемый результат | Фактический результат | Прошел? |
| 1 | [1, -2, 3, 4] | 2.666... | 2.666... | ✅ |
| 2 | [-5, -1, 0] | 0 | 0 | ✅ |
| 3 | abc | TypeError | Ошибка исполнения | ✅ |

## 2.3 Анализ корректности и ошибок Почему не делится на 0: Перед делением выполняется if count > 0 — защита от деления на ноль. Пустой список: [] → count == 0 → возвращается 0. Соответствует постусловию. Ошибка при нарушении предусловия: Если передать строку или None, произойдет TypeError. Можно улучшить: - Проверять тип входных данных (isinstance(numbers, list)). - Проверять типы внутри списка.

## 3. Выводы Функция корректна относительно своей спецификации. Она возвращает правильный результат при корректном входе и защищена от деления на 0. Однако не надёжна — не обрабатывает некорректные типы данных. Если бы не было проверки count > 0, ошибку помогло бы найти тестирование или формальная верификация.