Теоретическая часть

1. Определение:

Измерительные методы оценки программ — это набор метрик и инструментов, используемых для количественной оценки характеристик программного обеспечения, таких как размер, сложность, качество и производительность.

Основные цели их применения:

Оценка качества кода.

Выявление потенциальных проблем (например, высокой сложности).

Улучшение сопровождаемости и читаемости кода.

Планирование ресурсов для разработки и тестирования.

2. Примеры метрик:

Метрики размера:

Строки кода: Показывает объём кода. Чем больше, тем сложнее поддержка.

Количество функций: Отражает модульность кода. Большое количество функций может указывать на избыточность.

Метрики сложности:

Цикломатическая сложность: Измеряет количество независимых путей в коде. Высокая сложность увеличивает риск ошибок.

Метрики качества:

Коэффициент сопровождения: Оценивает легкость поддержки кода.

Покрытие тестами: Показывает процент кода, проверенный тестами.

3. Условия применения:

Метрики применяются для анализа кода, оценки трудозатрат.

Ограничения:

Метрики не всегда отражают реальное качество кода.

Некоторые метрики могут быть misleading (например, из-за комментариев).

Требуют интерпретации в контексте проекта.

Практическая часть

Код из файла test2.py:

1. Метрики размера:

Количество строк кода (без пустых строк и комментариев): 13 строк.

Количество функций: 3.

2. Оценка сложности кода:

Цикломатическая сложность:

calculate\_sum: 1 (нет ветвлений).

factorial: 2 (одно условие if n == 0).

print\_numbers: 3 (цикл for + условие if i % 2 == 0).

3. Выводы:

Самая сложная функция: print\_numbers, так как имеет наибольшую цикломатическую сложность (3).

Потенциальные ошибки:

В factorial возможна бесконечная рекурсия при отрицательном n.

В print\_numbers ошибки маловероятны, но высокая сложность усложняет тестирование.

Анализ кода из файла test1.py:

1. Метрики размера:

Количество строк кода: 65.

Количество функций: 0

2. Оценка сложности кода:

Цикломатическая сложность: Высокая из-за вложенных условий и циклов.

Рекомендации:

Разбить код на меньшие функции.

Добавить обработку ошибок для HTTP-запросов.

Предложения по улучшению кода:

Использовать try-except для обработки ошибок запросов.

Вынести повторяющийся код (например, создание графиков) в отдельные функции.

3. Выводы:

Нужно разбить код на функции.