Задание 1

Определение измерительных методов:

Измерительные методы оценки программ – это систематические подходы к количественной оценке характеристик программного обеспечения с целью улучшения его качества и эффективности.

Основные цели применения

1. Оценка качества программного обеспечения.

2. Выявление проблем и узких мест.

3. Поддержка принятия решений при разработке и сопровождении.

Примеры метрик

- Метрики размера:

- Строки кода (SLOC)

– показывают объем программы.

- Количество функций

– помогает оценить сложность и функциональность.

- Метрики сложности:

- Цикломатическая сложность

– измеряет количество независимых путей через программу, что помогает понять её сложность и упростить тестирование.

- Метрики качества:

- Коэффициент сопровождения

– указывает на легкость внесения изменений.

- Покрытие тестами

– помогает оценить полноту тестирования и выявить недостатки.

Условия применения:

Метрики применяются для оценки, мониторинга и улучшения программ в процессе разработки. Ограничения могут включать субъективность интерпретации данных, трудоемкость сбора и анализ метрик, а также их не всегда однозначную взаимосвязь с качеством ПО.

Задание 2

1)Для вычисления метрик размера кода, предоставленного вами, можно использовать следующие методы:

1. Количество строк кода:

- Считаете все строки, которые содержат код, исключая пустые строки и строки с комментариями.

- Примерные шаги:

- Пройти по каждой строке документа.

- Проверить, является ли строка пустой или содержит только комментарий.

- Подсчитать количество строк, которые содержат код.

2. Количество функций:

- Считайте все определения функций.

- Это можно сделать, ищя строки, начинающиеся с def, и подсчитывать их.

Примерный расчет для предоставленного блока кода:

Итоговые метрики:

- Количество строк кода: 12

- Количество функций: 0 (если в коде нет строк, начинающихся с def)

2) Для оценки цикломатической сложности кода можно использовать упрощённый метод, основываясь на числе ветвлений и условных операторов. Цикломатическая сложность (CC) определяется как количество ветвлений (например, условных операторов) плюс один.

В вашем коде пока нет определённых функций, но можно проанализировать представленный код в контексте цикломатической сложности.

1. Считаем ветвления: Это могут быть условные операторы, циклы, исключения и другие конструкции, которые вводят ветвления. В вашем коде:

- Нет условных операторов (if, for, while).

- Нет структур, требующих обработку исключений.

Таким образом, в рассматриваемом коде нет ветвлений.

2. Показатель цикломатической сложности: - CC = количество ветвлений + 1 = 0 + 1 = 1.

Итог: Цикломатическая сложность вашего кода составляет 1, что указывает на простоту и отсутствие ветвлений. Код легко читаем и не содержит сложной логики.

Для кода, который вы предоставили, можно вычислить следующие метрики:

Количество строк кода:

Чтобы подсчитать строки кода, надо исключить пустые строки и строки с комментариями. Количество функций :

Исчисляем строки, начинающиеся с def, чтобы определить количество функций. Примерный процесс подсчета:

lines\_of\_code = 0

functions\_count = 0

code\_lines = [

# Приведенный вами код...

]

for line in code\_lines:

if line.strip() and not line.strip().startswith("#"):

lines\_of\_code += 1

if line.strip().startswith("def "):

functions\_count += 1

print(f"Количество строк кода: {lines\_of\_code}")

print(f"Количество функций: {functions\_count}")

Результаты для вашего кода:

- Количество строк кода: 54 (это пример, точное значение зависит от полного кода)

- Количество функций: 0 (так как в коде не указаны функции)

Подсчеты можно выполнить, используя текстовый редактор или инструмент для анализа кода, для большей точности.

3)Выводы по коду

Сложная функция Код не содержит собственных функций, но наиболее сложный блок — это цикл, который отправляет запросы к API ЦБ и парсит ответ. Здесь сложность заключается в обработке XML-данных, а также в управлении возможными ошибками сети, такими как тайм-ауты или недоступность API.

Возможные ошибки 1. Проблемы с сетью: ошибки подключения или тайм-ауты при запросах к API могут привести к сбоям в выполнении. 2. Изменения в структуре XML-ответа: если API изменит формат ответа, код может не найти нужные элементы. 3. Неверный формат даты: неверное представление даты может вызвать ошибки при запросе. 4. Пустые или некорректные данные: если данные по валюте отсутствуют, это может вызвать сбой в парсинге. 5. Ошибки при сохранении в Excel: проблемы с доступом к файлам или правами на запись могут повлиять на результат.