Пара №6

Задание №1: *Теоретическая часть*

1) Измерительные методы оценки программ — это системные подходы, использующие количественные и качественные показатели для измерения эффективности, результативности и воздействия программ. Эти методы включают сбор и анализ данных для оценки различных аспектов программы, таких как достижение целей, использование ресурсов, удовлетворенность участников и другие критерии.

Основные цели применения :

1. Оценка эффективности
2. Оптимизация ресурсов
3. Улучшение качества
4. Отчетность и прозрачность
5. Обучение и развитие
6. Сравнительный анализ

2)

1. Метрики размера

Строки кода (SLOC - Source Lines of Code):

Описание: Количество строк кода в программе

Использование: Оценка объема программного обеспечения, планирование ресурсов и сроков разработки, а также сравнение продуктивности

Количество функций

Описание: Количество функций или методов в программе.

Использование: Оценка сложности и модульности программы, выявление возможностей для рефакторинга и улучшения структуры кода.

2. Метрики сложности

Цикломатическая сложность

Описание: Метрика, измеряющая сложность программы путем подсчета количества независимых путей выполнения через граф потока управления.

Использование: Оценка сложности кода, выявление потенциально проблемных участков, которые могут быть трудными для понимания и сопровождения. Высокая цикломатическая сложность может указывать на необходимость рефакторинга.

3. Условия применения:

*Планирование и управление проектами* - метрики используются для оценки объема работ, планирования ресурсов и сроков выполнения задач. Пример: Оценка времени и ресурсов, необходимых для завершения проекта, на основе исторических данных и метрик предыдущих проектов.

*Оценка качества кода* - метрики качества, такие как покрытие тестами и коэффициент сопровождения, используются для оценки качества кода и выявления областей, требующих улучшения. Пример: Использование метрик для выявления участков кода с низким покрытием тестами и их последующего улучшения.

*Рефакторинг и оптимизация*- метрики сложности, такие как цикломатическая сложность, помогают выявить сложные и трудные для понимания участки кода, которые могут потребовать рефакторинга. Пример: Анализ цикломатической сложности для выявления функций, которые необходимо упростить и сделать более понятными.

Ограничения измерительных методов

Зависимость от контекста - метрики могут быть полезны только в определенном контексте и могут не подходить для всех типов проектов или ситуаций.

Субъективность и интерпретация - некоторые метрики могут быть субъективными и требовать интерпретации, что может привести к различным выводам в зависимости от интерпретатора.

Сложность сбора данных - сбор данных для метрик может быть сложным и трудоемким процессом, требующим значительных ресурсов и времени.

Ограниченная применимость - некоторые метрики могут быть применимы только к определенным аспектам программного обеспечения и не давать полной картины

Зависимость от инструментов - точность и полезность метрик могут зависеть от используемых инструментов и методов сбора данных, что может ограничивать их применимость и точность.

Задание №2: *Практическая часть*

Test1.py

1.Метрики размера

Количество строк кода - 80

Количество функций - 0

2. Сложность кода

1.while current\_date <= end\_date: — Ветвление 1

2.if response.status\_code == 200: — Ветвление 2

3.for valute in root.findall('Valute'): — Ветвление 3

4.if char\_code in currencies: — Ветвление 4

5.for currency in currencies: — Ветвление 5

6.if not df.empty: — Ветвление 6

7.for currency in currencies: — Ветвление 7

8.if df.empty: — Ветвление 8

Используя упрощенный метод (количество ветвлений + 1), цикломатическая сложность основного блока кода составляет 9 (8 ветвлений + 1).

Test2.py

1.*Метрики размера*

Количество строк кода -13

Количество функций - 3

2. Сложность кода

Ветвление внутри цикла добавляет один путь. Поэтому цикломатическая сложность этой функции равна 2 .

*Выводы:*

Самой сложной функцией является test1.py т.к

Обработка данных: Функция должна обрабатывать данные, проверять их на наличие, преобразовывать и сортировать даты. Это требует использования библиотек pandas и matplotlib, а также понимания работы с временными рядами и датами.

Визуализация данных: Создание графиков включает в себя множество параметров и настроек, таких как размер графика, стиль линии, размер и цвет маркеров, подписи осей, сетка и форматирование дат. Это требует знания библиотек для визуализации данных и понимания, как различные параметры влияют на конечный результат.

Сохранение и отображение графиков: Функция должна сохранять графики в файлы и отображать их на экране. Это требует понимания работы с файловой системой и управления графическими окнами.

Ошибки могут возникнуть в нескольких местах:

Обработка данных - если данные для какой-либо валюты отсутствуют, это может привести к ошибкам при попытке построения графика. Ошибки могут возникнуть при преобразовании строк в объекты даты, особенно если формат даты не соответствует ожидаемому.

Визуализация данных - неправильные параметры могут привести к некорректному отображению графиков, например, неправильные размеры, цвета или стили линий. Форматирование осей: Ошибки в форматировании осей могут привести к нечитаемым или некорректным подписям.

Сохранение и отображение графиков - Ошибки могут возникнуть при сохранении графиков в файлы, например, проблемы с правами доступа или неверные пути к файлам. Проблемы могут возникнуть при отображении графиков, особенно если графические окна не поддерживаются в текущей среде выполнения.