Общие замечания к тестовому заданию

Результат выполнения задание: архив с кодом или ссылка на репозиторий (пожалуйста, проверьте, что ссылка работает для анонимного пользователя).

Код не должен включать в себя содержание виртуального окружения, используйте PIP и requirements.txt или другой пакетный менеджер для сторонних библиотек.

Используемая версия Python: 3.8 или выше.

Будет плюсом написание тестов.

Будет плюсом оформление репозитория (.gitignore, README.md, каталог src для кода и т.д.).

Будет плюсом написать свои мысли по реализации (в файле README или INFO): почему применили тот или иной метод/подход, хотели, но не смогли/не посчитали нужным сделать то-то и то-то, нашли ошибку в требованиях к тестовому заданию или считаете его некорректным. Ну или любые другие комментарии.

1. Сервис-калькулятор

С помощью фреймворка FastAPI реализовать сервис, вычисляющий результат арифметического выражения и предоставляющий возможность просмотреть история запросов.

Общие требования

Сервис должен соответствовать Rest соглашениям: Get, Post запросы, HTTP коды ответов и т.д.

Аутентификация и авторизация не требуется.

Сервис должен быть "самодокументированным" - предоставлять описание форматов данных и запросов/ответов в формате OpenAPI с интерфейсом Swagger.

Описание конечных точек (endpoints)

/calc

Принимает в качестве входного значения арифметическое выражение и выдаёт либо ответ, либо ошибку.

*Обобщённый вид арифметического выражения: **[операция1](число)[(операция2)(число)]***, где:*

- операция1: плюс или минус: + -
- операция2: плюс, минус, умножение, деление: + - * /
- число: положительное вещественное число (decimal), имеющее ноль и более знаков после запятой
- круглые скобки: обязательный элемент
- квадратные скобки: необязательный элемент
- звёздочка (*): ноль и более повторений
- пробелы игнорируются

Примеры выражений и результат расчёта (информация по расчёту: см. ниже):

- +100.1 → 100.1
- -0 → 0
- -7 / 34.2 → -0.205
- - 6 * 2 → -11.98
- 2. / 1. → 2
- 5 + - 4 → ошибка
- *1 + 7 → ошибка

- 4 / 3 + → ошибка

Замечания по расчёту:

- арифметический приоритет операций игнорируется – все операции выполняются слева направо, то есть выражение 5 - 4 * 2 считается как (5 - 4) * 2, результат: 2
- количество значащих цифр после запятой: до 3 включительно, незначащих нули должны быть обрезаны
- округление математическое: 1.1234 → 1.123, 1.1235 → 1.124

Замечание по реализации:

- использование eval не допускается;
- будет плюсом написать реализацию, учитывающую приоритет операций и допускающей использование скобок (подсказка: польская нотация).

/history

Конечная точка возвращает последние 30 (по умолчанию) запросов к сервису в формате json (массив), где каждый запрос/ответ имеет вид:

- Успешно рассчитанный: {"request": "0.01 - 6 * 2", "response": "-11.980", "status": "success"}
- Запрос с ошибкой: {"request": "5 + - 4", "response": "", "status": "fail"}

Пример ответа для двух запросов: [{"request": "0.01 - 6 * 2", "response": "-11.980", "status": "success"}, {"request": "5 + - 4", "response": "", "status": "fail"}]

Дополнительные параметры запроса (могут использоваться совместно):

- limit (int) - ограничить количество выводимых запросов; при значениях меньше 1 и больше 30, возвращается ошибка;
- status (str) - отфильтровать успешные запросы или запросы с ошибкой; допустимые значения: success, fail. При других значениях возвращается ошибка.

Замечание по реализации: имеет смысл ограничить внутренне хранилище истории количеством значений по умолчанию (30), дабы постоянно запущенный сервис не съел память.

2. Игра "Крестики-нолики"

Реализовать программу, играющую в игру Крестики-нолики на поле 3*3

Общие требования

Один запуск программы - одна игра.

Компьютер играет сам с собой за двух игроков. В конце игры выводится история ходов обоих "игроков" и результат ("Ничья", "Выиграл игрок 1", "Выиграл игрок 2")

Будет плюсом

Возможность задать произвольный размер поля, например 4*20.

Возможность задать количество подряд идущих крестиков/ноликов для победы, например 5 для поля 20*20.

Использование осмысленного алгоритма или иного способа игры компьютера, отличного от случайного проставления крестиков/ноликов.