Тестовое задание на позицию QA Automation Engineer.

Общие требования:

- Выполнять на python 3.9.8.
- Результат присылать в виде ссылки на github репозиторий.
- Можно использовать любые подходящие для выполнения задания python-пакеты, если не указано обратное.

Задания:

1. pytest:
    - На сайте https://en.wikipedia.org/wiki/Programming_languages_used_in_most_popular_websites
      есть таблица «**Programming languages used in most popular websites**»
    - Необходимо реализовать параметризованный тест, проверяющий, что в этой таблице нет строк, у которых значение в столбце «Popularity(unique visitors per month)» меньше передаваемого в качестве параметра в тест значения.
    - Если такие строки в таблице есть, тест выводит сообщение об ошибке, перечисляя строки с ошибками в виде, пример: *“Yahoo (Frontend:JavaScript|Backend:PHP) has X unique visitors per month. (Expected more than Y)”*
    - Тест должен запускаться для значений: [10^7, 1.5 * 10^7, 5 * 10^7, 10^8, 5 * 10^8, 10^9, 1.5 * 10^9]
    - При реализации теста необходимо учитывать, что данные из этой таблицы могут понадобиться и в других тестах. Будет плюсом реализовать хранение данных из таблицы в виде датаклассов.
    - Подсказка для задания: есть несколько способов получения содержимого таблицы и использование selenium не самый оптимальный из них.
2. DevOps
    - Создать простое Rest API приложение на Flask. У api должно быть два эндпоинта:
        1. «/» get метод, возвращающий список записей из базы данных (колонку «text»)

2. «/new» post метод с данными {text:<текст записи>} – метод по добавлению новой записи в базу данных

- Приложение работает с базой данных MongoDB (поднята как докер контейнер). Нам нужна только 1 коллекция с полем «text». Данные из базы не должны теряться при рестарте или передеплое контейнера приложения.
- Написать 2 модуля с юниттестами на приложение (тесты могут быть просто заглушками типа assert True т.к. они нужны для проверки пайплайна деплоя). В одном модуле все тесты всегда проходят, во втором всегда падают.
- Написать Dockerfile и docker-compose.yml для получившегося приложения
- Создать джобу в дженкинсе с пайплайном редеплоя уже поднятого приложения с наименьшим возможным даунтаймом. (Пример сценария: поднятие дополнительного инстанса приложения -> запуск юниттестов -> при успехе тестов новый инстанс становится основным, при падении тестов получаем список ошибок, новый инстанс тушится)
  В джобе должен быть параметр, отвечающий за выбор модуля тестов для запуска. Это нужно для проверки обоих кейсов. Для проверки поместить файл джобы (groovy или xml) в репозиторий приложения.