Практическая работа №4. Слияние веток и модульные тесты

1 Цель

Изучить принципы работы с ветками в Git, с модульными тестами в Python и с удаленными репозиториями. Научиться создавать ветки, сливать ветки, заливать изменения в репозиторий и делать модульные тесты.

2 Задачи

Для выполнения практической работы необходимо выполнить следующие задачи:

- установить соединение с удаленным сервером в Gitlab по SSH;
- перейти на новую ветку, в которой вы будете реализовывать вашу программу;
- сделать проверку ввода отдельной функцией, сделать коммит(-ы);
- покрыть модульными тестами (юнит-тестами) весь код функции проверки ввода, сделать коммит(-ы);
- сплющить коммиты текущей ветки в один;
- слить текущую рабочую ветку в ветку master;
- перейти обратно на рабочую ветку;
- реализовать задание согласно варианту, сделать коммит(-ы);
- покрыть функцию, выполняющее задание, модульными тестами, сделать коммит(-ы);
- сплющить коммиты текущей ветки в один (все те коммиты, которые были созданы после сплющивания);
- слить текущую рабочую ветку в мастер;
- удалить рабочую ветку;
- залить ветку master в удаленный репозиторий;
- оформить отчет по практической работе;
- ответить на вопросы и выполнить дополнительные задания.

3 Пояснения к заданию

Практическая, выполненная не по своему варианту, не может быть засчитана. Программа должна быть выполнена на языке Python.

Обычно модульные тесты не используют ввод с клавиатуры (хотя это и можно сделать с переопределением потоков ввода), поэтому для проверки ввода нужно создать отдельную функцию, которая на вход будет принимать введенную пользователем строку, а возвращать уже обработанное значение. Например, если вам нужно принимать только целое положительное число, то функция проверки должна принимать на вход введенную пользователем строку, и возвращать число в том случае, если пользователь правильно его ввел. Иначе функция должна генерировать исключение ValueError, так как значение некорректное. Тоже самое, например, если пользователь ввел строку с цифрами, а по заданию допускаются только буквы алфавита.

Покрытие всего кода означает, что в процессе запуска тестов каждая строка кода будет запущена. Кроме того, надо выделить классы эквивалентности для параметров функции, и проверить все возможные тестовые случи, когда функция может сработать некорректно.

Тоже самое касается функции, которая будет реализовывать основной функционал программы. Эта функция будет принимать на вход параметры, которые задал пользователь, и возвращать результат. Эта функция не должна возвращать то, что будет напечатано на экран. И тем более печатать на экран результат. Например, если у вас в задании надо проверить, является ли слово палиндромом, функция должна будет принимать строку, и возвращать True, если это палиндром, и False в другом случае. Основная причина делать так в том, что интерфейс ПО может быстро меняться. Например, вы захотели изменить формат вывода с «WOW is palindrome» на «Word “WOW” is palindrome». Если у вас функция возвращает булево значение, то нужно будет поменять только вывод на экран в функции main. Иначе, нужно будет поменять формирование строки в функции проверки слова на «полиндромность» и

модульные тесты. А если эта функция будет выводить результат сразу на экран, то проверить ее модульными тестами будет совсем неудобно.

Итого, у вас в программе должно быть как минимум 3 функции: функция проверки ввода, функция, которая решает задачу, и функция main, которая содержит в себе ввод данных с клавиатуры, вызов функции, решающей ваш вариант и вывод результата на экран.

Дополнительным заданием будет разрешение конфликта файла при слиянии веток.