Федеральное государственное автономное

образовательное учреждение

высшего образования

«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий

институт

Кафедра «Информатика»

кафедра

**ОТЧЕТ О ПРАКТИЧЕСКОЙ** **РАБОТЕ №4**

Слияние веток и модульные тесты

Тема

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Преподаватель |  |  |  |  |  | П.В.Пересунько |
|  |  |  |  | подпись, дата |  | инициалы, фамилия |
| Студент |  | КИ19-16/2б |  |  |  | С.Е.Шпаков |
|  |  | номер группы, зачетной книжки |  | подпись, дата |  | инициалы, фамилия |

Красноярск 2019

# СОДЕРЖАНИЕ

[1 Цель и задачи 3](#_Toc25284941)

[1.1 Цель 3](#_Toc25284942)

[1.2 Задачи 3](#_Toc25284943)

[2 Описание варианта задания 4](#_Toc25284944)

[2.1 Основное задание 4](#_Toc25284945)

[2.2 Дополнительное задание 4](#_Toc25284946)

[3 Ход выполнения 4](#_Toc25284947)

[4 Выводы 8](#_Toc25284948)

[СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ 10](#_Toc25284949)

# 1 Цель и задачи

## 1.1 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.

## 1.2 Задачи

Для выполнения практической работы необходимо выполнить следующие задачи:

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

# 2 Описание варианта задания

## 2.1 Основное задание

Вариант №23. Имеется алфавит *А*, состоящий только из четырех букв – «*О*», «*Е*», «*W*» и «*М*». Для каждого символа исходного алфавита *А* в соответствие поставлена комбинация символов из кодового алфавита *B*:

1. «*О*» - кодируется комбинацией «..»;
2. «*Е*» - кодируется комбинацией «.\_»;
3. «*W*» - кодируется комбинацией «\_.»;
4. «*М*» - кодируется комбинацией «\_\_».

Необходимо раскодировать строку, состоящую из букв кодового алфавита *B*.

## 2.2 Дополнительное задание

К алфавиту добавляется четыре символа. Для каждого символа алфавита *A* в соответствие поставлена обновленная комбинация символов из кодового алфавита *B*:

1. «*O*» - кодируется комбинацией «\_\_\_»;
2. «*E*» - кодируется комбинацией «\_\_.»;
3. «*W*» - кодируется комбинацией «\_.\_»;
4. «*M*» - кодируется комбинацией «\_..»;
5. «*C*» - кодируется комбинацией «.\_\_»;
6. «*A*» - кодируется комбинацией «.\_.»;
7. «*U*» - кодируется комбинацией «..\_»;
8. «*Q*» - кодируется комбинацией «…»;

# 3 Ход выполнения

Уже известный локальный Git нам нужно связать с удаленным сервером GitLab. Для этого мы получим SSH ключ командой “ssh-keygen” (рисунок 1). Для удобства будем использовать тип ключа Ed25519, потому что версия SSH позволяет, а ключ уже не будет километровым, обеспечивая равную защиту.

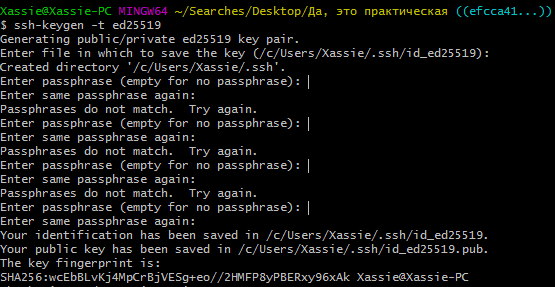


Рисунок 1 – Получение ключа SSH

После получения ключа мы можем непосредственно подключиться к серверу GitLab. Для этого необходима команда “ssh”, использованная как в примере (рисунок 2).

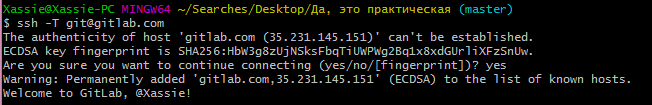


Рисунок 2 – Подключение к серверу GitLab

Далее переходим на другую ветку (git checkout –b caffeine), разделяем часть кода на функции, покрываем их юнит-тестами и своевременно коммитим. Чтобы не делать отчет ещё длиннее, чем он есть, показаны лишь сами коммиты (рисунок 3).

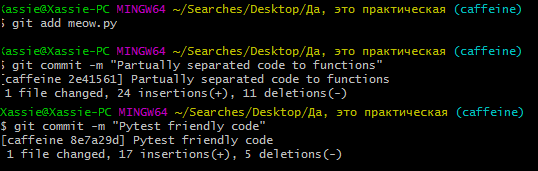


Рисунок 3 – Коммит изменений на новой ветке

Однако, коммитов получилось слишком много. Склеим их в один с помощью команды “git rebase” (рисунок 4).

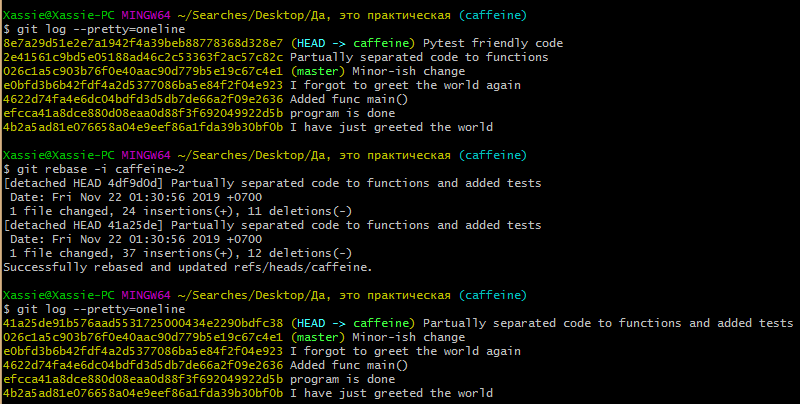


Рисунок 4 – Склеивание коммитов

Склеивать коммиты хорошо, но сливать ветки ещё лучше. Для того, чтобы соединить рабочую ветку с master, нужно в этот мастер перейти, а потом использовать команду “git merge” (рисунок 5).

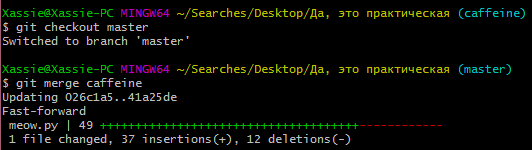


Рисунок 5 – Слияние веток

Всё это хорошо, но программу написать по варианту тоже надо. Но прежде небольшое отступление. Никита, спасибо, что мне просто надо было изменить словарь, чтобы реализовать дополнительный функционал. Конечно, нам надо покрыть это тоже модульными тестами, но у нас не появилось ничего особо нового лишь изменился словарь с алфавитами. Таким образом программа была полностью реализована (листинг 1).

Листинг 1 – Реализация программы по варианту задания

import argparse

ALPHABET = {

'\_\_\_': 'O',

'\_\_.': 'E',

'\_.\_': 'W',

'\_..': 'M',

'.\_\_': 'C',

'.\_.': 'A',

'..\_': 'U',

'...': 'Q',

}

def parse():

parser = argparse.ArgumentParser(description="Give a line to decode")

parser.add\_argument('n', type=str, help='Coded message')

R = parser.parse\_args()

return R.n

def decode(message):

dc = ''

res = ''

for i in message:

dc += i

if dc in ALPHABET:

res += ALPHABET[dc]

dc = ''

if not dc:

return res

def test\_decode():

assert decode('\_...\_..\_\_\_.\_\_\_\_..\_') == 'MACWOU'

assert decode('.\_\_\_\_\_\_..\_...\_.') == 'COMMA'

assert not decode('\_\_\_.\_...\_.\_.\_.')

assert not decode('I like trains')

def mcheck(message):

if set(message) == {'.', '\_'}:

return 1

def test\_message\_check():

assert mcheck('.\_\_\_.\_.\_\_\_.')

assert not mcheck('.---.-.---.')

assert not mcheck('I like trains')

def main():

print('Hello world!')

msg = parse()

if mcheck(msg):

result = decode(msg)

if result:

print(result)

else:

print('Impossible to decode')

else:

print('Please enter valid message')

if \_\_name\_\_ == '\_\_main\_\_':

main()

Сливаем рабочую ветку с мастером, удаляем рабочую ветку командой “git branch –d caffeine” и, наконец-то, заливаем всё это дело на GitLab, установив соединение и запушив в удаленный репозиторий (рисунок 6).

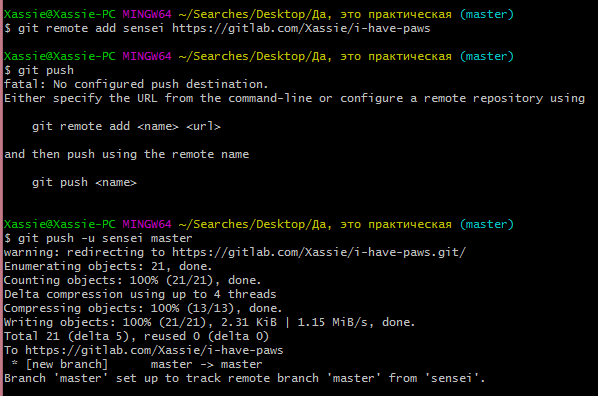


Рисунок 6 – Push ветки в удаленный репозиторий

# 4 Выводы

В результате работы были изучены принципы работы с ветками, модульными тестами и немного с удаленными репозиториями. Склеивали, сливали, меняли ветки, коммиты. Подружились с GitLab, по крайней мере до того момента, когда он поймет, что авторизировался я через GitHub.

# СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. СТО 4.2-07-2014 Система менеджмента качества Общие требования к построению, изложению и оформлению документов учебной деятельности. – Взамен СТО 4.2–07–2012 ; введ. 09.01.2014, - Красноярск : ИПК СФУ, 2014 – 60 с.