МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРОКАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра

инфокоммуникаций

Институт цифрового

развития

ОТЧЁТ

по лабораторной работе №2.11

Дисциплина: «Основы программной инженерии»

Тема: «Замыкание в языке Python»

Выполнил:

студент 2 курса

группы Пиж-б-о-21-1

Коныжев Максим

Викторович

Ставрополь 2022

Цель работы: приобретение навыков по работе с замыканиями при написании программ с помощью языка программирования Python версии 3.x.

1. Был создан репозиторий в Github в который были добавлены правила gitignore для работы IDE PyCharm, была выбрана лицензия MIT, сам репозиторий был клонирован на локальный сервер и был организован в соответствии с моделью ветвления git-flow.

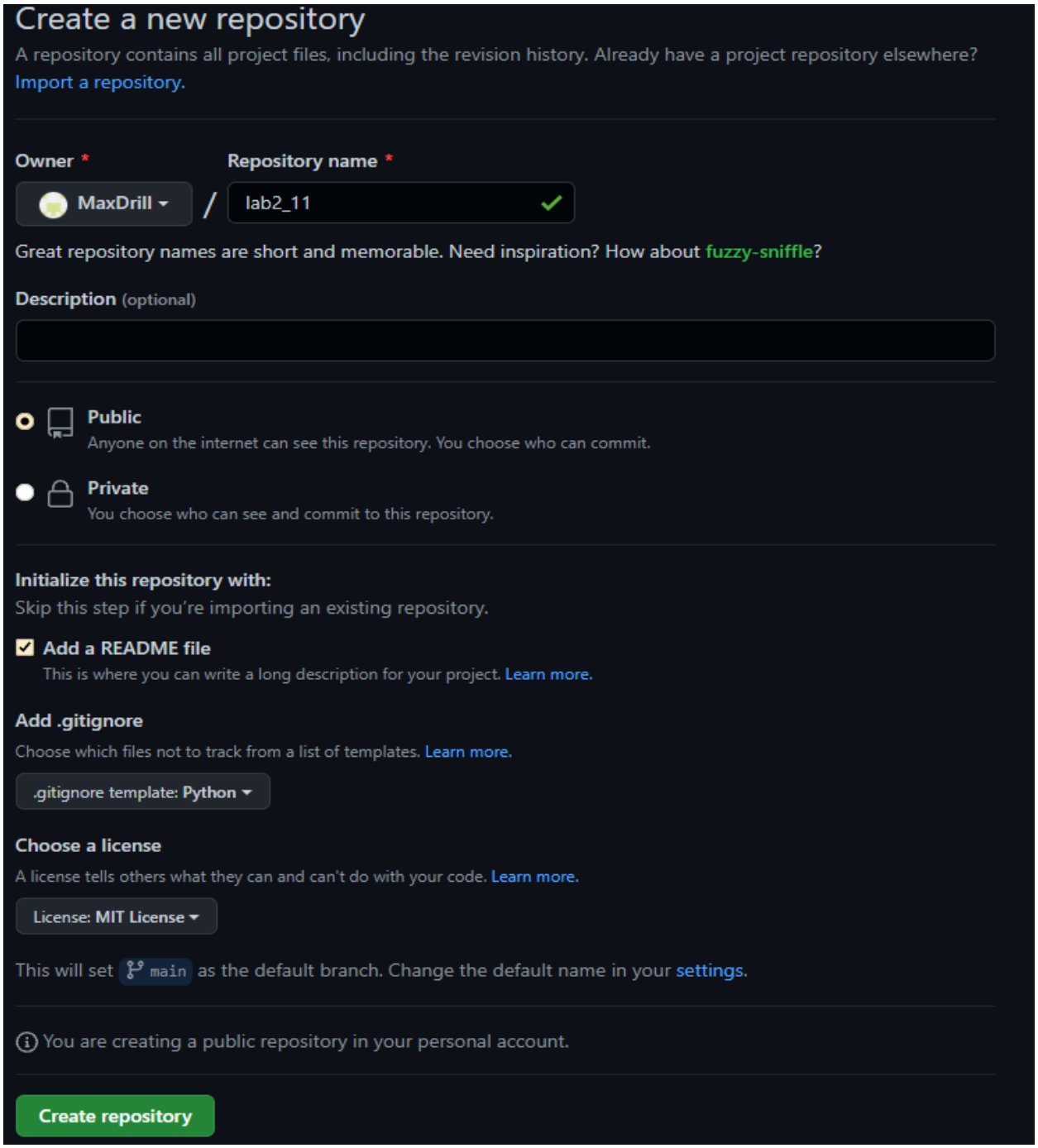

Рисунок 1 – Создание репозитория

```
C:\Users\UESR\gitproj>git clone https://github.com/MaxDrill/lab2_11.git
Cloning into 'lab2_11'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.
```

Рисунок 2 – Клонирование репозитория

```
C:\Users\UESR\gitproj\lab2_11>git flow init

Which branch should be used for bringing forth production releases?
   - main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [C:/Users/UESR/gitproj/lab2_11/.git/hooks]

C:\Users\UESR\gitproj\lab2_11>
```

Рисунок 3 – Организация репозитория с моделью ветвления git-flow

```
C:\Users\UESR\Desktop\2.1\proj\venv\Scri
example 1
19

example 2
(1, 2)
(3, (1, 2))
((1, 2), (3, (1, 2)))

Process finished with exit code 0
```

Рисунок 6 – Результат работы примера

3. Индивидуальное задание

Используя замыкания функций, объявите внутреннюю функцию, которая заключает строку s (s – строка, параметр внутренней функции) в произвольный тег, содержащийся в переменной tag – параметре внешней функции. Далее, на вход программы поступает две строки: первая с тегом, вторая с некоторым содержимым. Вторую строку нужно поместить в тег из первой строки с помощью реализованного замыкания. Результат выведите на экран.

Код программы:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-


def get_tag(tag):
    x = tag


    def get_str(s):
        nonlocal x
        return ": ".join([x, s])
    return get_str


if __name__ == '__main__':
    test_fun = get_tag("test tag")
    print(test_fun("hello"))
```

```
C:\Users\UESR\Desktop\2.1\proj\venv\
test tag: hello

Process finished with exit code 0
```

Рисунок 7 – Результат работы программы

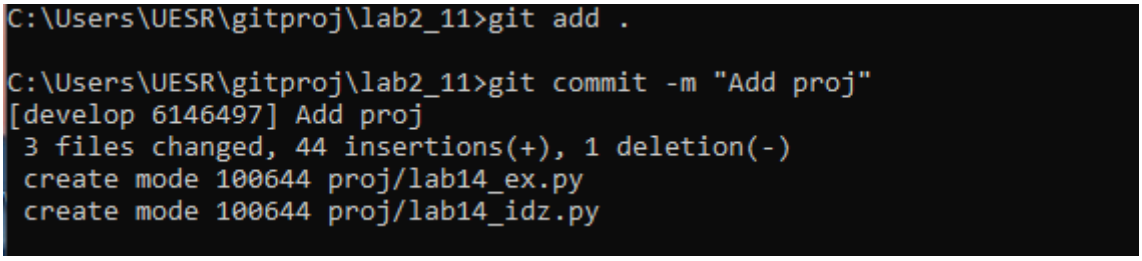

Рисунок 8 – Коммит изменений

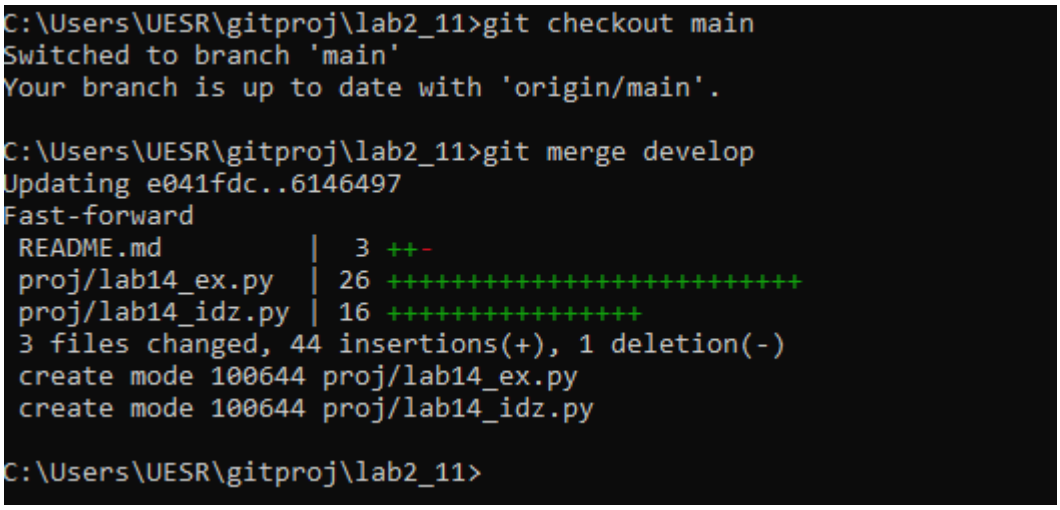

Рисунок 9 – Слияние веток

Вывод: в результате лабораторной работы были приобретены навыки по работе с замыканиями при написании программ с помощью языка программирования Python версии 3.x.

Контрольные вопросы

1. Что такое замыкание?

Замыкание – это функция, в теле которой присутствуют ссылки на переменные, объявленные вне тела этой функции.

2. Как реализованы замыкания в языке программирования Python?

Замыканием в языке Python называется функция, вложенная в другую функцию и использующая переменные внешней функции.

3. Что подразумевает под собой область видимости Local?

Переменный с областью видимости Local (локальные переменные) могут быть использованы только внутри того блока кода, где она была объявлена.

4. Что подразумевает под собой область видимости Enclosing?

Для вложенных функций переменные из функции более высокого уровня имеют данную область видимости.

5. Что подразумевает под собой область видимости Global?

Область видимости Global означает, что данная переменная может быть использована (видна) во всём модуле (файле с расширением .py).

6. Что подразумевает под собой область видимости Build-in?

Это переменный уровня интерпретатора. Для их использования не нужно импортировать модули.