Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №15
дисциплины «Основы программной инженерии»

Выполнил:
Мелтонян Одиссей
2 курс, группа ПИЖ-б-о-22-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка
и сопровождение программного
обеспечения», очная форма обучения

_____
(подпись)

Проверил:
Кандидат технических наук, доцент
кафедры инфокоммуникаций
Воронкин Р. А.

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 г

Тема: Декораторы функций в языке Python

Цель работы: приобретение навыков по работе с декораторами функций при написании программ с помощью языка программирования Python

Ход работы:

1. Изучен теоретический материал работы

2. Создан общедоступный репозиторий на GitHub, в котором использована лицензия MIT и язык программирования Python

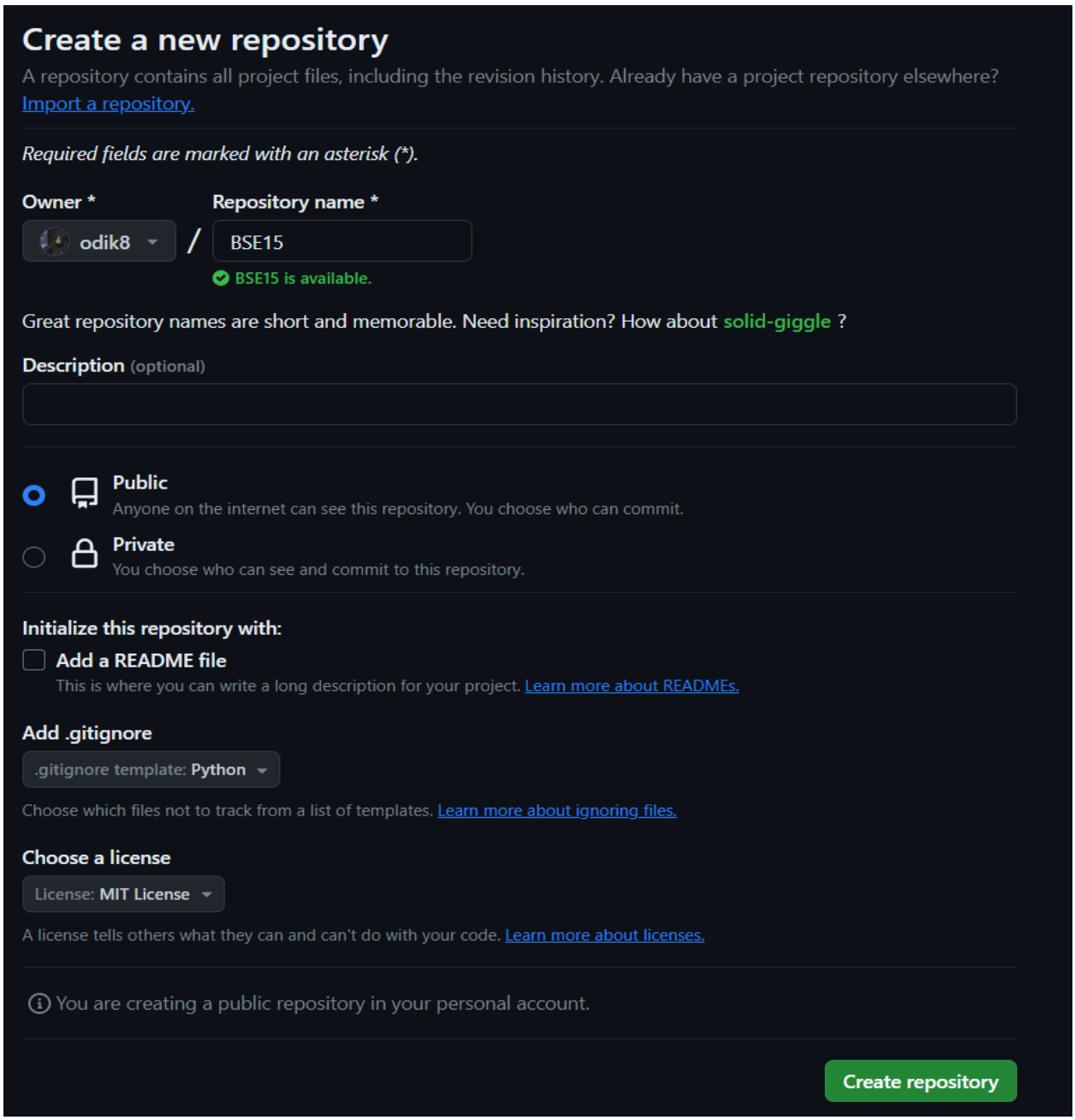

Рисунок 1 – Создание репозитория

3. Выполнено клонирование созданного репозитория

```
varfe@DESKTOP-E1O8KEH MINGW64 ~/Рабочий стол/Воронкин/ЛР15
$ git clone https://github.com/odik8/BSE15.git
Cloning into 'BSE15'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.
```

Рисунок 2 – Клонирование репозитория

4. Дополнен файл .gitignore необходимыми правилами для работы с IDE PyCharm

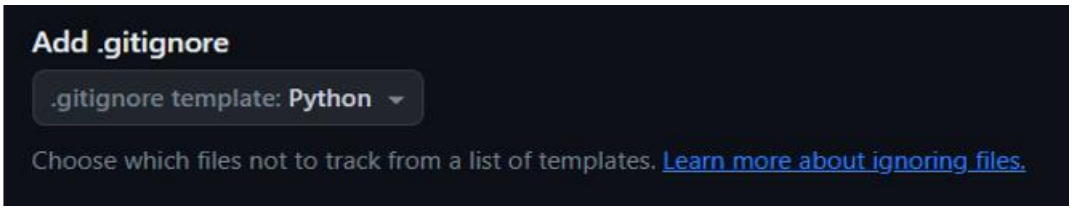

Рисунок 3 – Добавление файла .gitignore

5. Репозиторий организован в соответствие с моделью ветвления git-flow

```
varfe@DESKTOP-E1O8KEH MINGW64 ~/Рабочий стол/Воронкин/ЛР15
$ git flow init
Initialized empty Git repository in C:/Users/varfe/Рабочий стол/В
оронкин/ЛР15/.git/
No branches exist yet. Base branches must be created now.
Branch name for production releases: [master]
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/varfe/Рабочий стол/Воронки
н/ЛР15/.git/hooks]
```

Рисунок 4 – Инициализация git-flow

6. Создан проект PyCharm в папке репозитория

7. Индивидуальное задание.

Вариант 11. Объявите функцию с именем get_sq, которая вычисляет площадь прямоугольника по двум параметрам: width и height – ширина и высота прямоугольника и возвращает результат. Определите декоратор для этой функции с именем (внешней функции) func_show, который отображает результат на экране в виде строки (без кавычек): "Площадь прямоугольника: ". Вызовите декорированную функцию get_sq.

```python
#!/usr/bin/env python
1 usage  new *
def func_show(func):
    new *
    def wrapper(*args, **kwargs):
        result = func(*args, **kwargs)
        print(f"Площадь прямоугольника: {result}")
        return result

    return wrapper


1 usage  new *
@func_show
def get_sq(width, height):
    return width * height


if __name__ == "__main__":
    get_sq_result = get_sq( *args: 5, 8)

```

Рисунок 1 – Код решения задачи

```
C:\Users\varfe\AppData\Local\Programs\Py
Площадь прямоугольника: 40

Process finished with exit code 0
```

Рисунок 2 – Результат выполнения кода

8. Зафиксированы изменения в репозитории.

9. Добавлен отчет по лабораторной работе в формате PDF в папку doc репозитория. Зафиксируйте изменения.

10. Выполнено слияние ветки для разработки с веткой main.

11. Отправлены сделанные изменения на сервер GitHub.

12. Отправлен адрес репозитория GitHub на электронный адрес преподавателя.

Вывод: в ходе лабораторной работы были приобретены навыки по работе с декораторами функций при написании программ с помощью языка программирования Python

Вопросы для защиты работы:

1. Что такое декоратор? — это функция, которая позволяет обернуть другую функцию для расширения её функциональности без непосредственного изменения её кода.

2. Почему функции являются объектами первого класса? – Объектами первого класса в контексте конкретного языка программирования называются элементы, с которыми можно делать всё то же, что и с любым другим объектом: передавать как параметр, возвращать из функции и присваивать переменной. А в Python мы можем сохранять функции в переменные, передавать их в качестве аргументов и возвращать из других функций. Можно даже определить одну функцию внутри другой.

3. Каково назначение функций высших порядков? – Функции высших порядков в Python могут принимать другие функции в качестве аргументов

или возвращать их как результат. Они позволяют абстрагироваться от деталей реализации и писать более универсальный и гибкий код.

4. Как работают декораторы? – Декораторы работают, создавая обертку (wrapper) вокруг функции, которую они декорируют. Эта обертка может модифицировать поведение функции до или после её выполнения. Декоратор применяется к функции с использованием символа **@**, что делает код более читаемым и лаконичным.

5. Какова структура декоратора функций?

```
def decorator(func):
    def wrapper(*args, **kwargs):
        # Дополнительное поведение перед вызовом функции
        result = func(*args, **kwargs)
        # Дополнительное поведение после вызова функции
        return result
    return wrapper

@decorator
def function():
    # Код функции
```