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 №2.12

Дисциплина: «Программирование на Python»

Тема: «Декораторы функций в языке Python»

Выполнил: студент 2 курса

группы ИВТ-б-о-21-1

Хашиев Адам Мухарбекович

Ставрополь 2022

Выполнение работы:

1. Создал репозиторий в GitHub «rep 2.6» в который добавил .gitignore, который дополнил правила для работы с IDE PyCharm с ЯП Python, выбрал лицензию MIT, клонировал его на лок. сервер и организовал в соответствии с моделью ветвления git-flow.

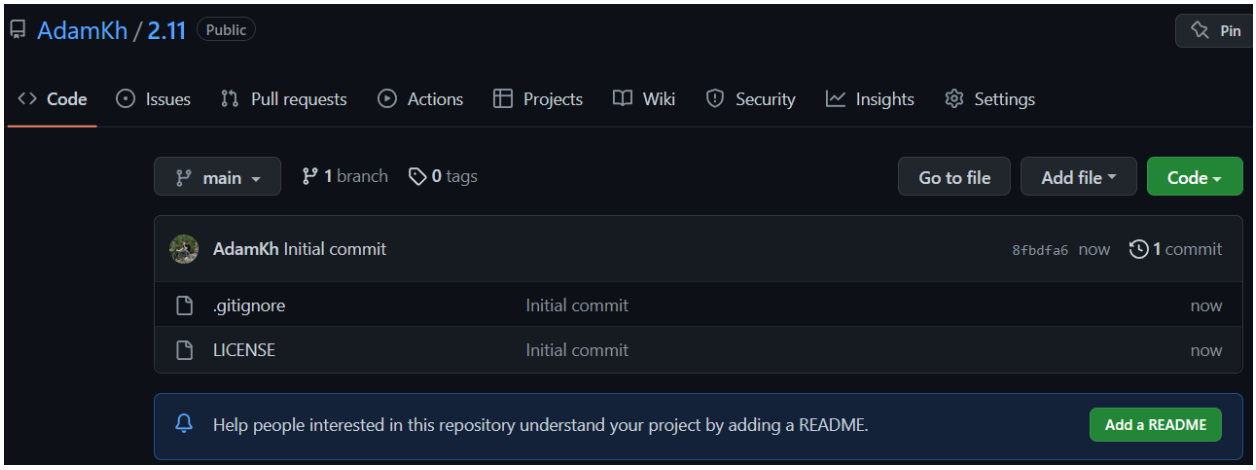

Рисунок 1.1 Создание репозитория

```
C:\Users\adamkh\Desktop\3 семестр\Программирование на Python\2.11>git clone https://github.com/AdamKh/2.11.git
Cloning into '2.11'...
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.

C:\Users\adamkh\Desktop\3 семестр\Программирование на Python\2.11>
```

Рисунок 1.2 Клонирование репозитория

```
C:\Users\adamkh\Desktop\3 семестр\Программирование на Python\2.11\2.11>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/adamkh/Desktop/3 семестр/Программирование на Python/2.11/2.11/.git/hooks]
```

Рисунок 1.3 Организация репозитория в соответствии с моделью ветвления git-flow

.gitignore – Блокнот

Файл Правка Формат Вид Справка

```
.idea/
# Created by https://www.toptal.com/developers/gitignore/api/python,pycharm
# Edit at https://www.toptal.com/developers/gitignore?templates=python,pycharm

### PyCharm ###
# Covers JetBrains IDEs: IntelliJ, RubyMine, PhpStorm, AppCode, PyCharm, CLion, Android Studio,
# Reference: https://intellij-support.jetbrains.com/hc/en-us/articles/206544839

# User-specific stuff
.idea/**/workspace.xml
.idea/**/tasks.xml
.idea/**/usage.statistics.xml
.idea/**/dictionaries
.idea/**/shelf

# AWS User-specific
.idea/**/aws.xml
```

Рисунок 1.4 Изменение .gitignore

2. Создал проект PyCharm в папке репозитория, проработал примеры ЛР.

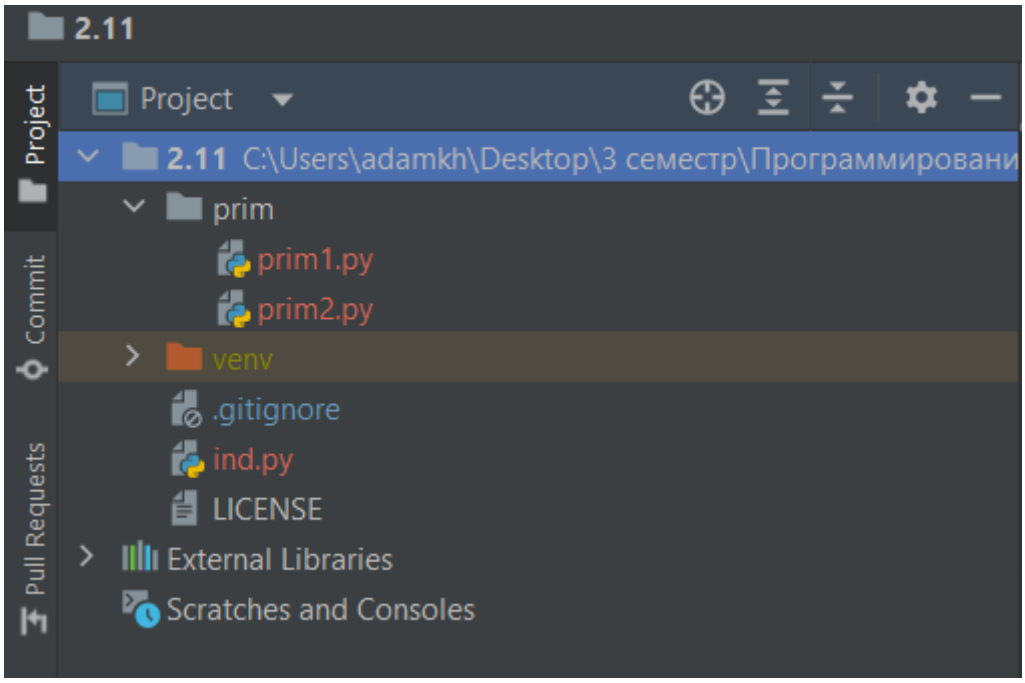

Рисунок 2.1 Создание проекта в PyCharm

```
Hello world!

Process finished with exit code 0
```

Рисунок 2.2 Рез-т выполнения программы 1-го примера

```
Функция-обёртка!
Оборачиваемая функция: <function hello_world at 0x00000162B7795990>
Выполняем обёрнутую функцию...
Hello world!
Выходим из обёртки

Process finished with exit code 0
```

Рисунок 2.3 Рез-т выполнения программы 2-го примера

3. (15 вариант). Выполнил индивидуальное задание.

```
Введите числа через пробел: 1 2 3 4 5
20

Process finished with exit code 0
```

Рисунок 3.1 Вывод программы индивидуального задания

4. Сделал коммит, выполнил слияние с веткой main, и запушил изменения в уд. репозиторий.

```
C:\Users\adamkh\Desktop\3 семестр\Программирование на Python\2.11\2.11>git add .

C:\Users\adamkh\Desktop\3 семестр\Программирование на Python\2.11\2.11>git commit -m "added progs"
[develop 4555710] added progs
 4 files changed, 199 insertions(+), 3 deletions(-)
 create mode 100644 ind.py
 create mode 100644 prim/prim1.py
 create mode 100644 prim/prim2.py
```

```
C:\Users\adamkh\Desktop\3 семестр\Программирование на Python\2.11\2.11>git push --set-upstream origin develop
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 8 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (7/7), 3.44 KiB | 1.72 MiB/s, done.
Total 7 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'develop' on GitHub by visiting:
remote:      https://github.com/AdamKh/2.11/pull/new/develop
remote:
To https://github.com/AdamKh/2.11.git
 * [new branch]      develop -> develop
branch 'develop' set up to track 'origin/develop'.
```

Рисунок 4.1 коммит и пуш изменений и переход на ветку main

```
C:\Users\adamkh\Desktop\3 семестр\Программирование на Python\2.11\2.11>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

C:\Users\adamkh\Desktop\3 семестр\Программирование на Python\2.11\2.11>git merge develop
Updating 8fbdfa6..4555710
Fast-forward
 .gitignore    | 157 ++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++--
 ind.py        |  17 +++++++
 prim/prim1.py |  12 +++++
 prim/prim2.py |  16 ++++++
 4 files changed, 199 insertions(+), 3 deletions(-)
 create mode 100644 ind.py
 create mode 100644 prim/prim1.py
 create mode 100644 prim/prim2.py

C:\Users\adamkh\Desktop\3 семестр\Программирование на Python\2.11\2.11>git push
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/AdamKh/2.11.git
   8fbdfa6..4555710  main -> main
```

Рисунок 4.2 Слияние ветки main с develop

```
C:\Users\adamkh\Desktop\3 семестр\Программирование на Python\2.11\2.11>git push
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/AdamKh/2.11.git
   8fbdfa6..4555710  main -> main
```

Рисунок 4.3 Пуш изменений на удаленный сервер

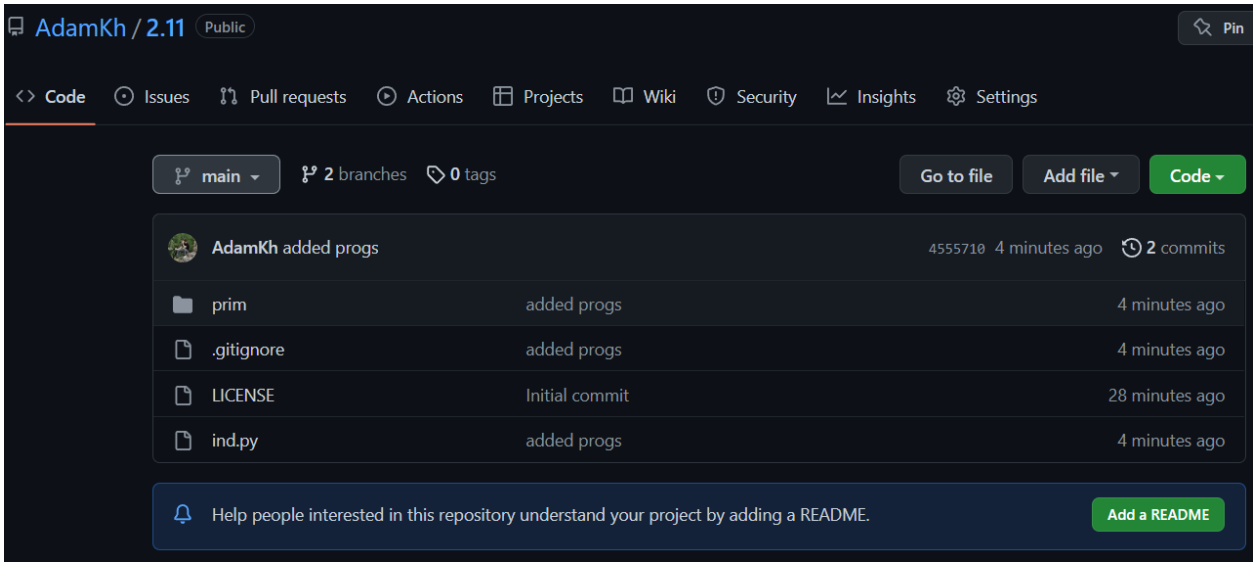

Рисунок 4.4 Изменения на удаленном сервере

Контр. вопросы и ответы на них:

1. Что такое декоратор?

Декоратор – это функция, которая позволяет обернуть другую функцию для расширения её функциональности без непосредственного изменения её кода.

2. Почему функции являются объектами первого класса?

Потому что с ними можно работать как с переменными, могут быть переданы как аргумент процедуры, могут быть возвращены как результат выполнения процедуры, могут быть включены в другие структуры данных.

3. Каково назначение функций высших порядков?

Основной задачей функций высших порядков является возможность принимать в качестве аргументов и возвращать другие функции.

4. Как работают декораторы?

Они берут декорируемую функцию в качестве аргумента и позволяет совершать с ней какие-либо действия до и после того, что сделает эта функция, не изменяя её.

5. Какова структура декоратора функций?

Функция decorator принимает в качестве аргумента функцию func, внутри функции decorator другая функций wrapper. В конце декоратора происходит возвращение функции wrapper.

6. Самостоятельно изучить как можно передать параметры декоратору, а не декорируемой функции?

Достаточно обернуть функцию декоратор в другую функцию, которая будет принимать аргументы. И сделать вывод функций wrapper и decorator.