МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙФЕДЕРАЦИИ

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»

Вариант 8

Выполнила: студентка 2 курса,группы ИВТ-б-о-21-1

Диченко Дина Алексеевна

Ставрополь 2022

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

Практическая часть:

1. Создала общедоступный репозиторий на GitHub.

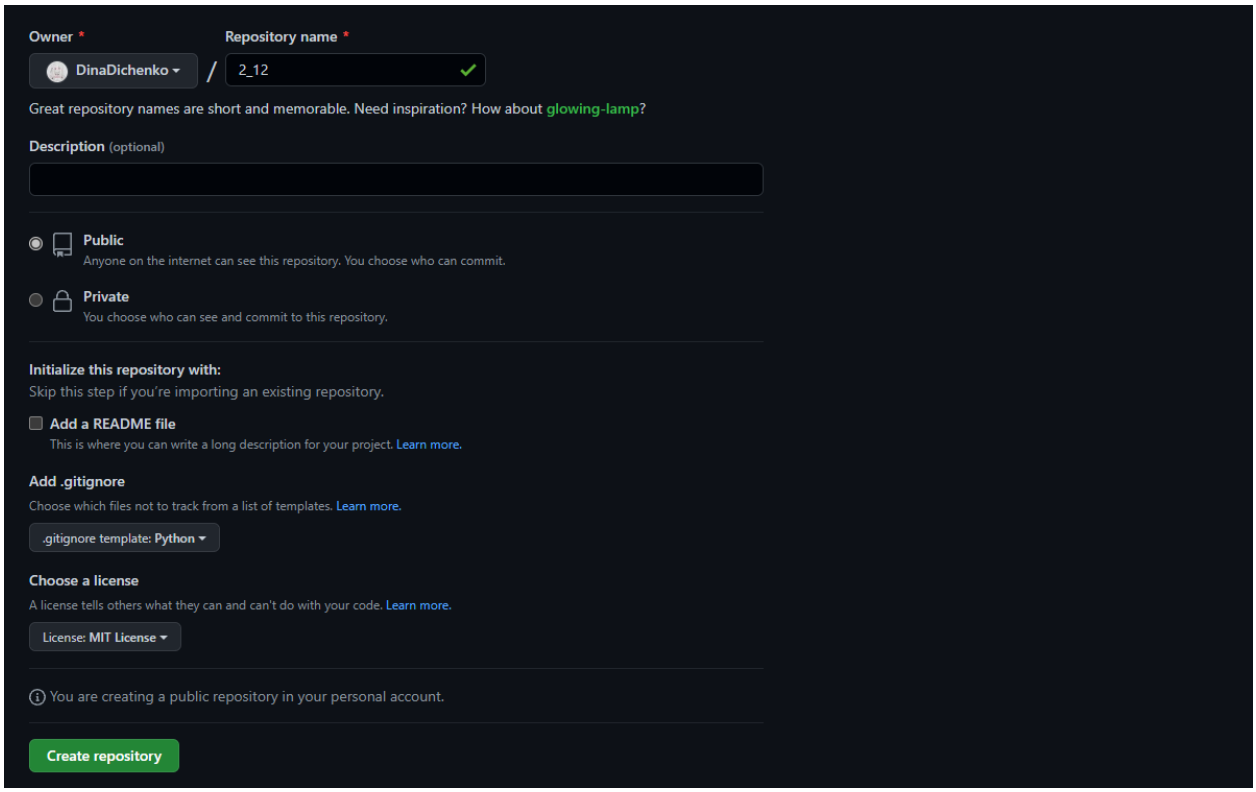

Рисунок 1. Создание репозитория

2. Выполнила клонирование созданного репозитория.

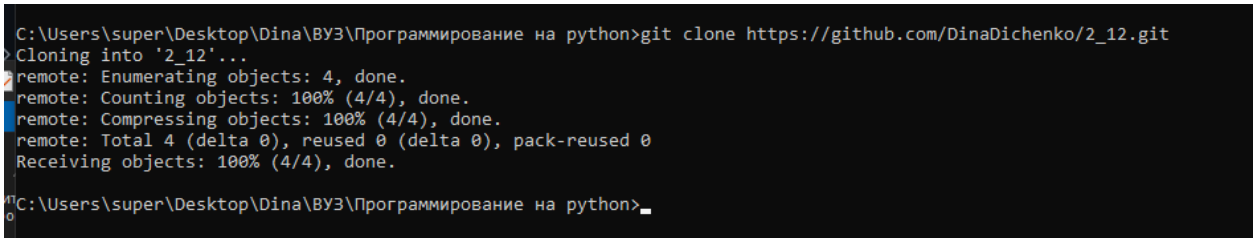

Рисунок 2. Клонирование репозитория

3. Дополнила файл .gitignore.

```
# 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, WebStorm and Rider
# Reference: https://intellij-support.jetbrains.com/hc/en-us/articles/206544839
.idea/
.idea

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

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

# Generated files
.idea/**/contentModel.xml

# Sensitive or high-churn files
.idea/**/dataSources/
.idea/**/dataSources.ids
.idea/**/dataSources.local.xml
.idea/**/sqlDataSources.xml
.idea/**/dynamic.xml
```

Рисунок 3. Изменение файла .gitignore

4. Организовала свой репозиторий в соответствие с моделью ветвления git-flow.

```
C:\Users\super\Desktop\Dina\ВУЗ\Программирование на python\2_12>git commit -m "gitignore"
[main 94cdf98] gitignore
 1 file changed, 155 insertions(+), 3 deletions(-)

C:\Users\super\Desktop\Dina\ВУЗ\Программирование на python\2_12>git branch develop

C:\Users\super\Desktop\Dina\ВУЗ\Программирование на python\2_12>git checkout develop
Switched to branch 'develop'

C:\Users\super\Desktop\Dina\ВУЗ\Программирование на python\2_12>git flow init

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

Which branch should be used for integration of the "next release"?
   - develop
Branch name for "next release" development: [develop] develop

How to name your supporting branch prefixes?
Feature branches? [feature/] fea
Bugfix branches? [bugfix/] bug
Release branches? [release/] branch
Hotfix branches? [hotfix/] hot
Support branches? [support/] supp
Version tag prefix? [] ver
Hooks and filters directory? [C:/Users/super/Desktop/Dina/ВУЗ/Программирование на python/2_12/.git/hooks] hooks

C:\Users\super\Desktop\Dina\ВУЗ\Программирование на python\2_12>
```

Рисунок 4. Организация репозитория в соответствии с git-flow

5. Проработала примеры лабораторной работы.

```
PS C:\Users\super> & C:/Users/super/AppData/Local/Programs/Python/Python310/py
Hello world!
PS C:\Users\super>
```

Рисунок 5. Результат работы примера 1

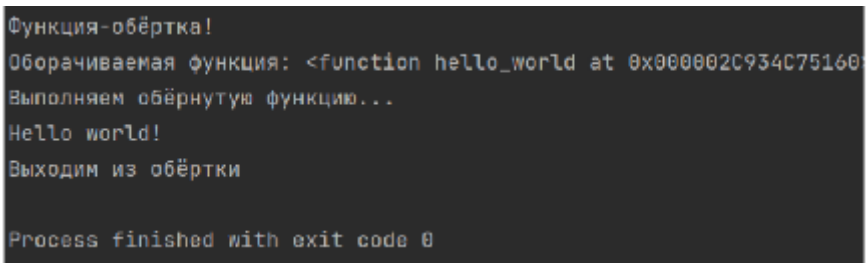

Рисунок 6. Результат работы примера 2

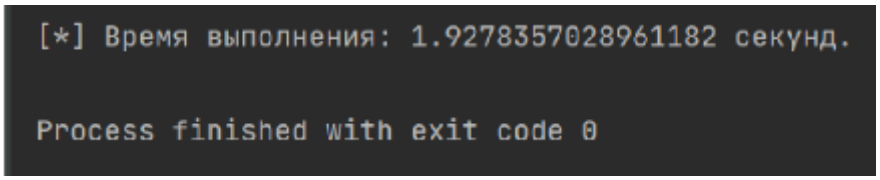

Рисунок 7. Результат работы примера 3

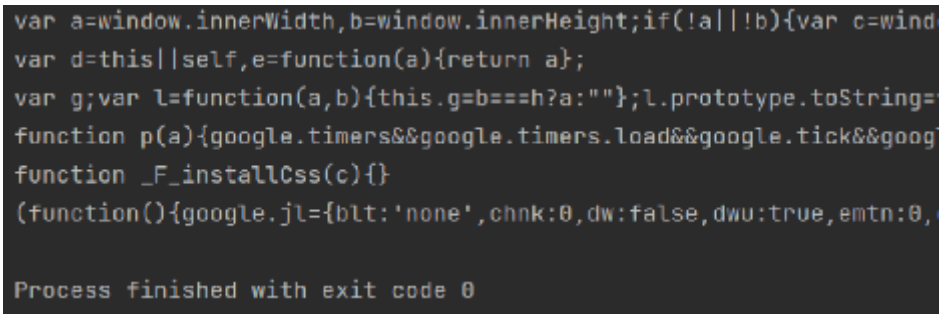

Рисунок 8. Результат работы примера 4

6. Выполнила индивидуальное задание.

Объявите функцию, которая вычисляет площадь круга и возвращает вычисленное значение. В качестве аргумента ей передается значение радиуса. Определите декоратор для этой функции, который выводит на экран сообщение: «Площадь круга равна = <число>». В строке выведите числовое значение с точностью до сотых. Примените декоратор к функции и вызовите декорированную функцию.

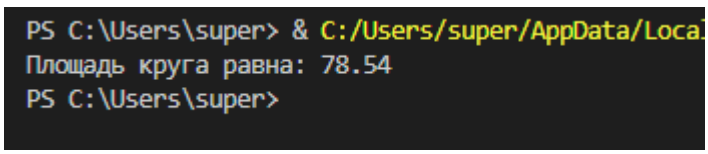

Рисунок 9. Результат работы индивидуального задания

Контрольные вопросы:

1. Что такое декоратор?

Декоратор — это функция, которая позволяет обернуть другую функцию для расширения её функциональности без непосредственного изменения её кода. Вот почему декораторы можно рассматривать как практику метапрограммирования, когда программы могут работать с другими программами как со своими данными.

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.Каково назначение функций высших порядков?

3. Каково назначение функций высших порядков?

Функции высших порядков — это такие функции, которые могут принимать в качестве аргументов и возвращать другие функции.

4. Как работают декораторы?

```
def decorator_function(func):
    def wrapper():
        print('Функция-обёртка!')
        print('Оборачиваемая функция: {}'.format(func))
        print('Выполняем обёрнутую функцию...')
        func()
        print('Выходим из обёртки')
    return wrapper
```

Здесь decorator_function() является функцией-декоратором. Как вы могли заметить, она является функцией высшего порядка, так как принимает функцию в качестве аргумента, а также возвращает функцию. Внутри decorator_function() мы определили другую функцию, обёртку, так сказать, которая обёртывает функцию-аргумент и затем изменяет её поведение. Декоратор возвращает эту обёртку. модифицировали её поведение. Однако выражение с @ является всего лишь синтаксическим сахаром для hello_world = decorator_function(hello_world). Иными словами, выражение @decorator_function вызывает decorator_function() с hello_world в качестве аргумента и присваивает имени hello_world возвращаемую функцию.

5. Какова структура декоратора функций?

Пример в 4 вопросе.

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

В декораторе можно передать и сам параметр. В этом случае нужно добавить еще один слой, то есть – еще одну функцию-обертку. Это обязательно, поскольку аргумент передается декоратору. Затем, функция, которая вернулась, используется для декорации нужной.

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