# МИНИCTEPCTBO НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙ- СКОЙ ФЕДЕРАЦИИ

**Федеральное государственное автономное образовательное учреждение высшего образования**

# «СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

**Кафедра инфокоммуникаций**

# Основы кроссплатформенного программирования Отчет по лабораторной работе №2.12

Тема: «Декораторы функций в языке Python»

|  |
| --- |
| Выполнил студент группы  ИВТ-б-о-21-1 |
| Криворот В.Г. « » 20 г. |
| Подпись студента |
| Работа защищена « » 20 г. |
| Проверил доцент  Кафедры инфокоммуникаций, старший преподаватель  Воронкин Р.А.  (подпись) |

Ставрополь 2022

**Цель работы:** приобретение навыков по работе с декораторами функ- ций при написании программ с помощью языка программирования Python вер- сии 3.x..

# Ход работы:

1. **Создал репозиторий в GitHub,** дополнил правила в .gitignore для ра- боты с IDE PyCharm с ЯП Python, выбрал лицензию MIT, клонировал его на компьютер и организовал в соответствии с моделью ветвления git-flow.

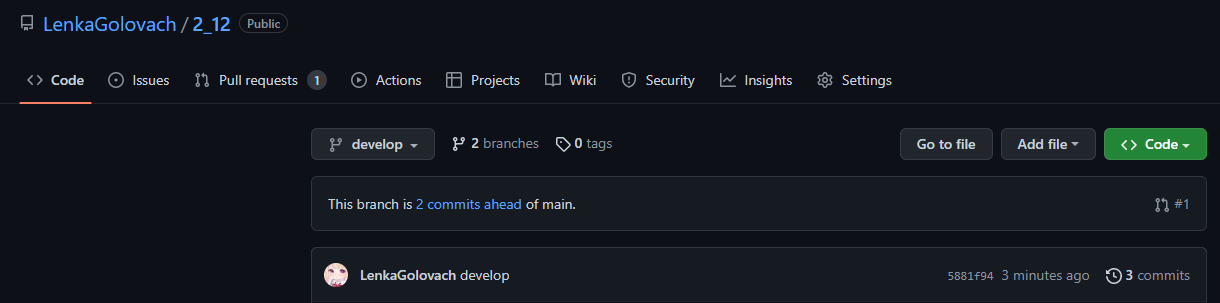


Рисунок 1.1 – Созданный репозиторий

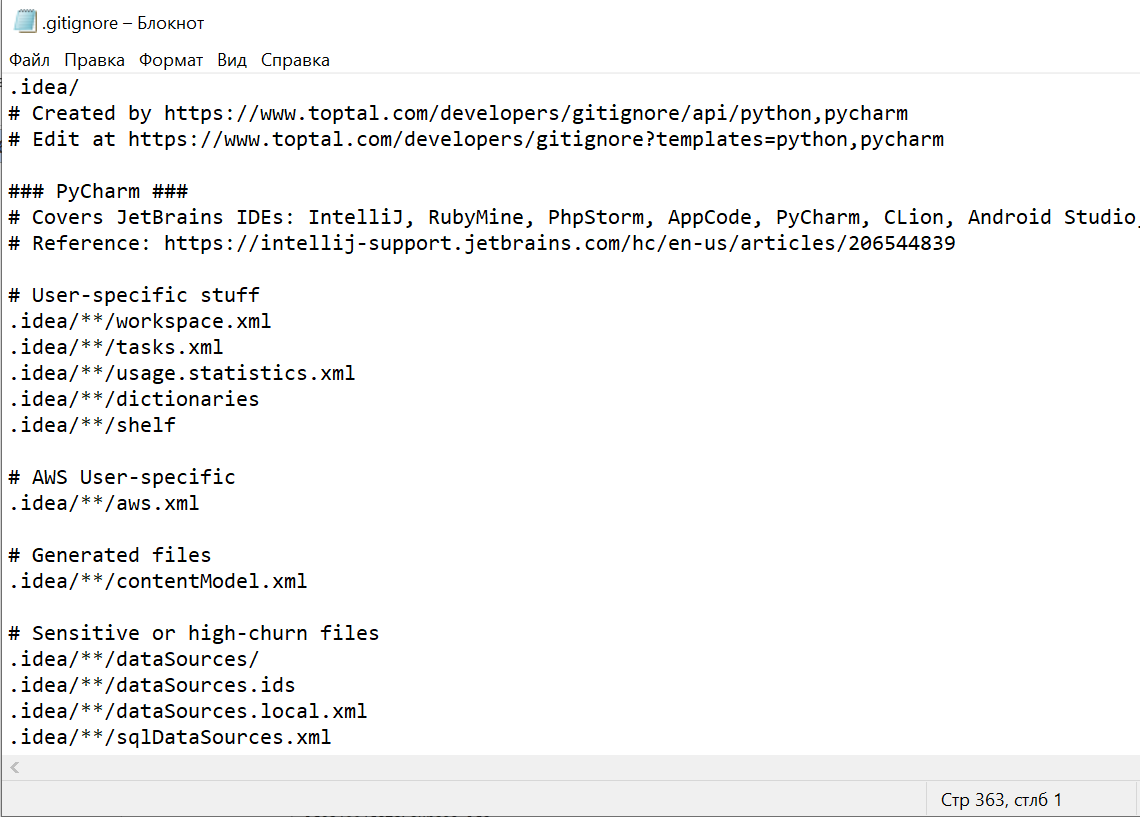


Рисунок 1.2 – Дополнил правила в .gitignore

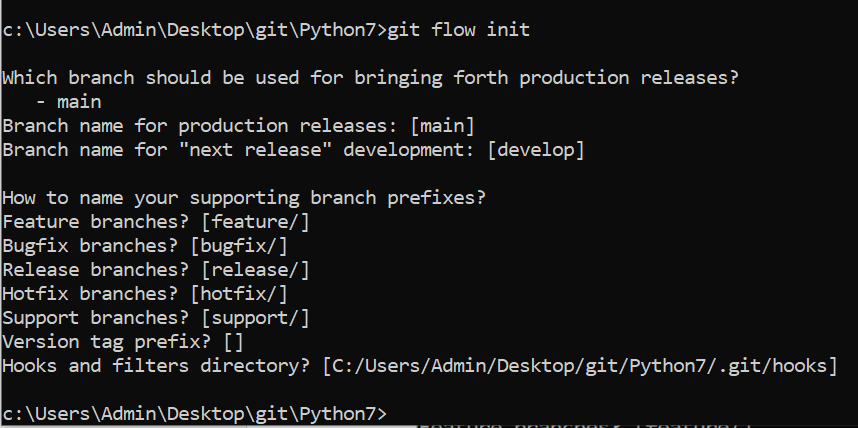


Рисунок 1.3 – Организация репозитория в соответствии с моделью ветвления git-flow

1. **Создал проект** Pycharm в папке репозитория, проработал примеры ЛР.

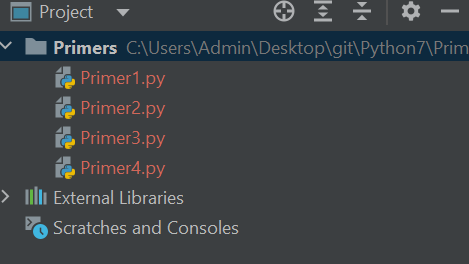
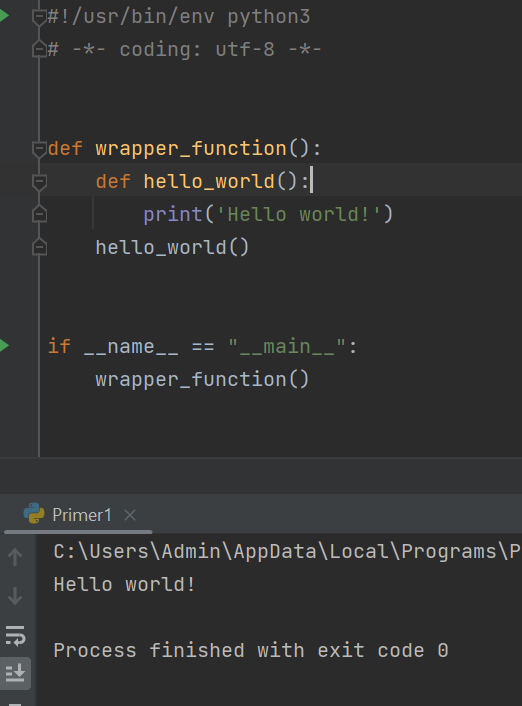


Рисунок 2.1 – Созданные проекты



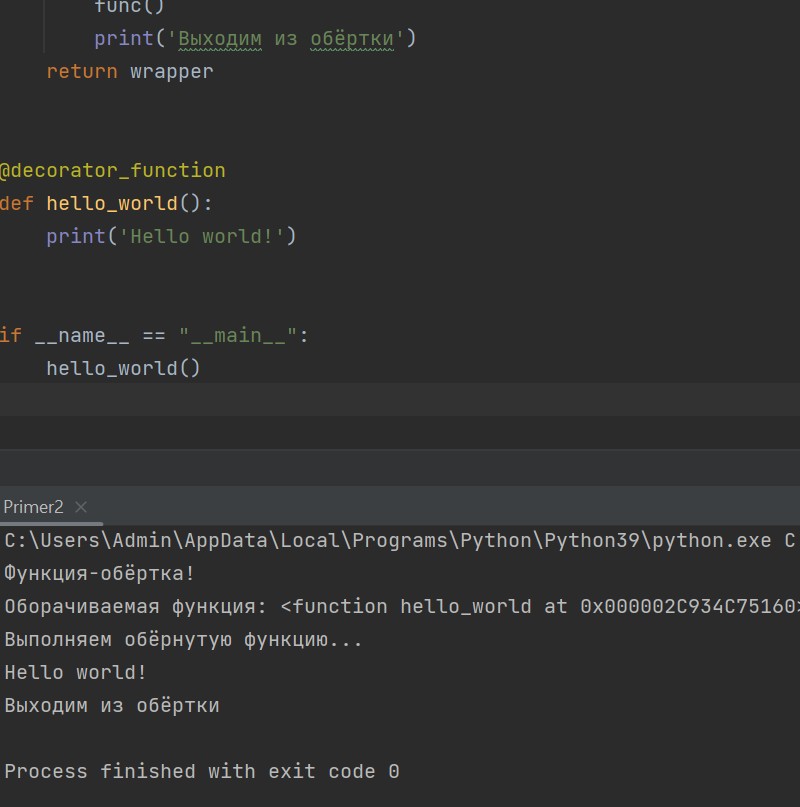
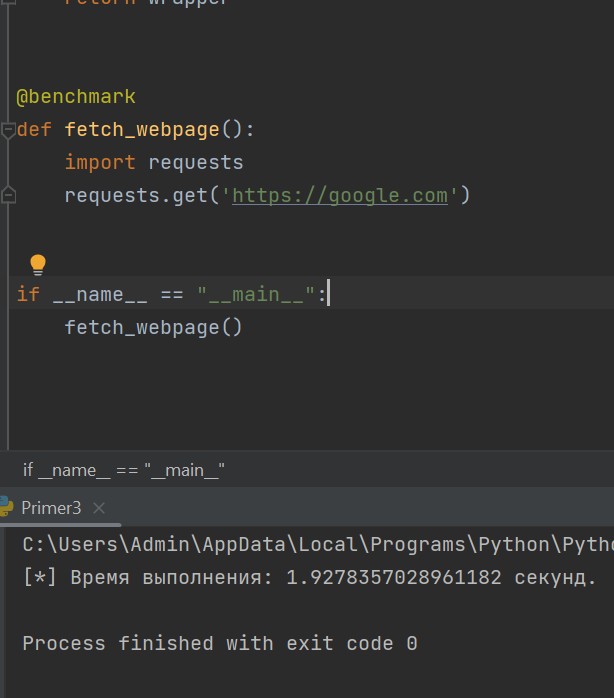
Рисунок 2.2 – Результат выполнения примера №1

Рисунок 2.3 – Результат выполнения примера №2



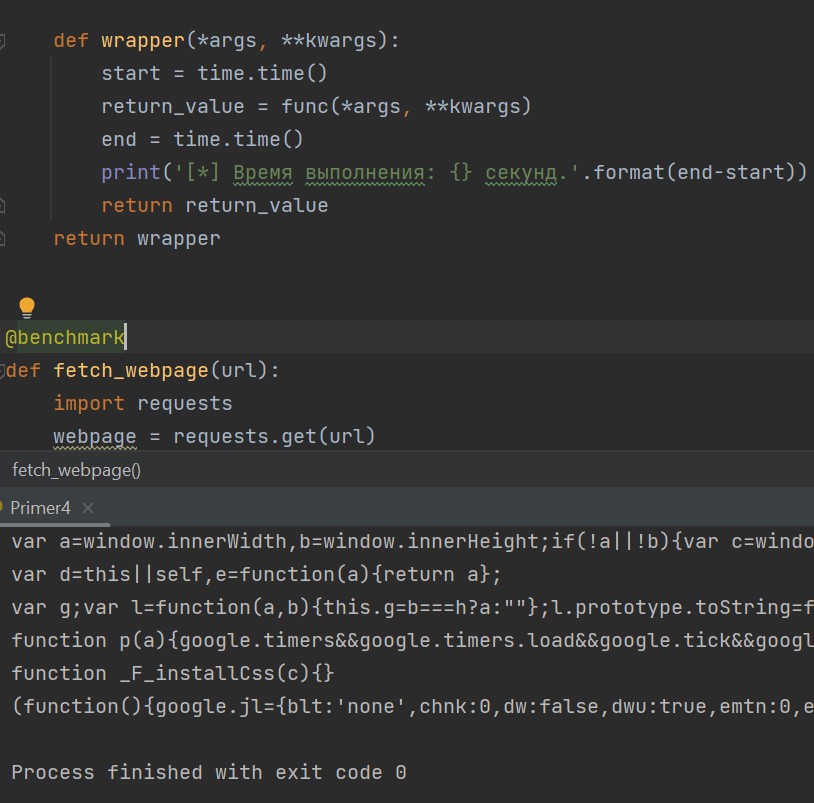
Рисунок 2.4 – Результат выполнения примера №3

Рисунок 2.5 – Результат выполнения примера №4

**Индивидуальное задание. В – 17.** Объявите функцию, которая вычисляет периметр многоугольника и возвращает  
вычисленное значение. Длины сторон многоугольника передаются в виде коллекции (списка  
или кортежа). Определите декоратор для этой функции, который выводит на экран  
сообщение: «Периметр фигуры равен = <число>». Примените декоратор к функции и  
вызовите декорированную функцию.

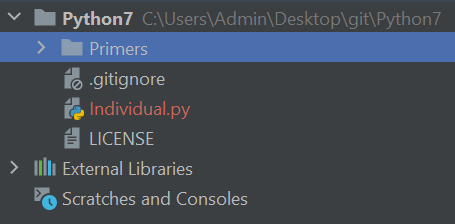


Рисунок 3.1 – Созданный проект

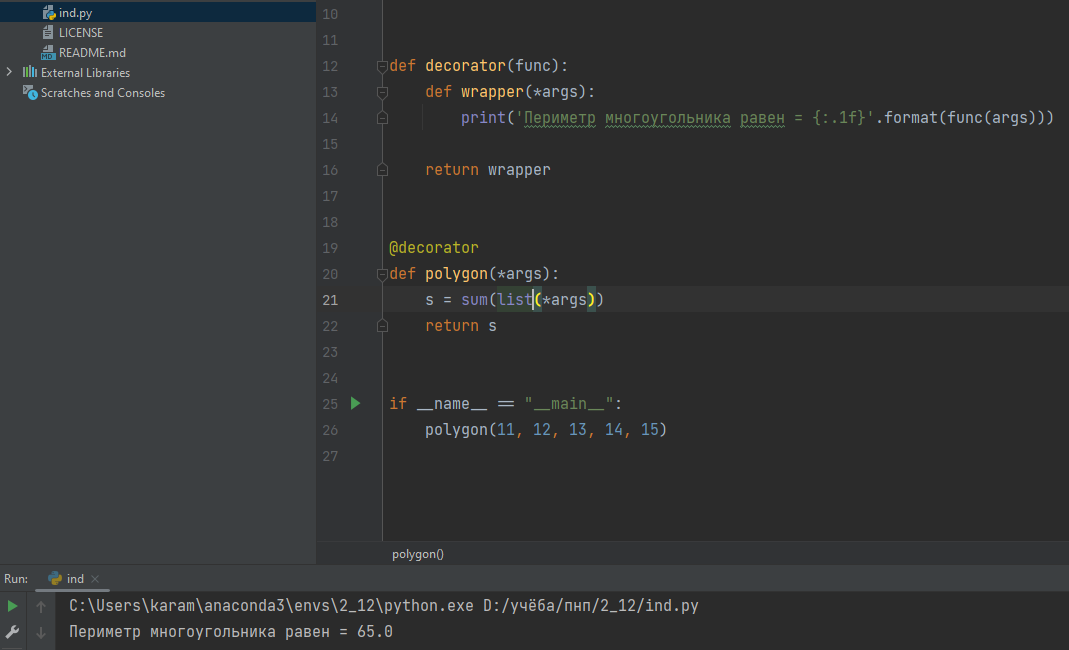


Рисунок 3.2 – Результат выполнения программы

**Вывод:** в результате выполнения лабораторной работы были приобре- тены практические навыки и теоретические сведения по работе с декорато- рами функций при написании программ с помощью языка программирования Python версии 3.x..

# Ответы на 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 прак- тику метапрограммирования, когда программы могут работать с другими про- граммами как со своими данными.

# Почему функции являются объектами первого класса?

В Python всё является объектом, а не только объекты, которые вы созда- ёте из классов. В этом смысле он (Python) полностью соответствует идеям объ- ектно-ориентированного программирования. Это значит, что в Python всё это

* объекты:
  + числа;
  + строки;
  + классы (да, даже классы!);
  + функции (то, что нас интересует).

Тот факт, что всё является объектами, открывает перед нами множество возможностей. **Мы можем сохранять функции в переменные, передавать их в качестве аргументов и возвращать из других функций. Можно даже определить одну функцию внутри другой. Иными словами, функции — это объекты первого класса.**

# Каково назначение функций высших порядков?

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

# 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, ко- торая обёртывает функцию-аргумент и затем изменяет её поведение. Декора- тор возвращает эту обёртку.

# Какова структура декоратора функций?

В 4 вопросе пример. **Здесь decorator\_function() является функцией- декоратором.** Как вы могли заметить, она является функцией высшего по- рядка, 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.

# Самостоятельно изучить как можно передать параметры декора- тору, а не декорируемой функции?

В декоратор можно передать и сам параметр. В этом случае нужно доба- вить ещё один слой абстракции, то есть – ещё одну функцию-обёртку. Это обя- зательно, поскольку аргумент передаётся декоратору. Затем, функция, которая вернулась, используется для декорации нужной.