# МИНИCTEPCTBO 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 |
| Богдашов Артём Владимирович |

Ставрополь 2022

# Выполнение работы:

1. Создал репозиторий в GitHub «rep Laba2.12» 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.

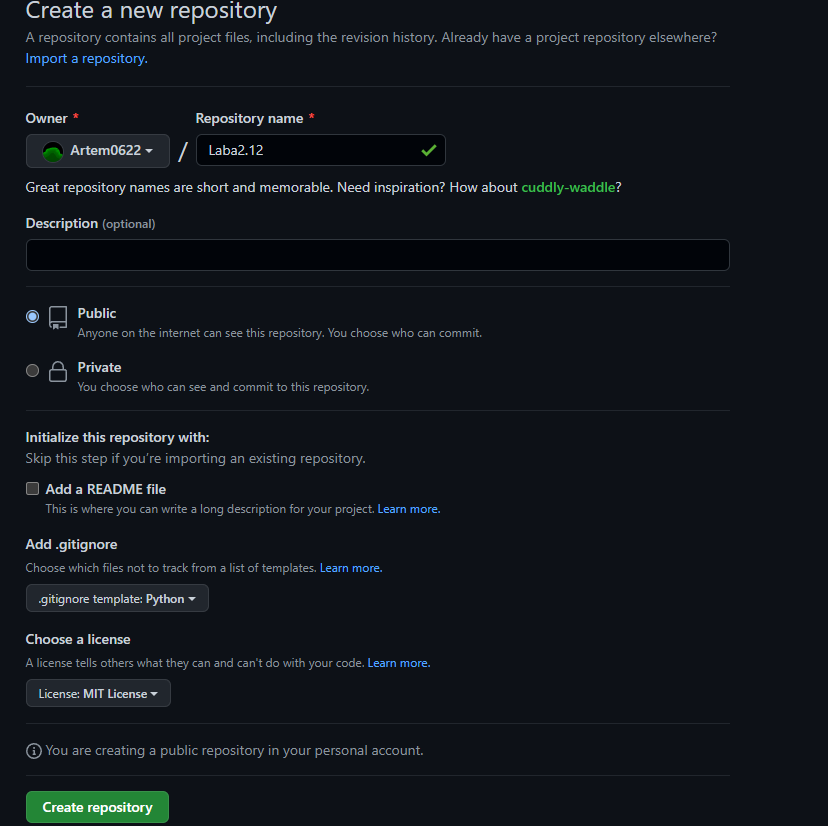


Рисунок 1.1 Создание репозитория

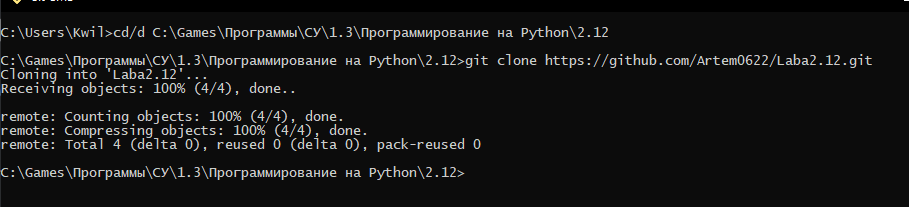


Рисунок 1.2 Клонирование репозитория

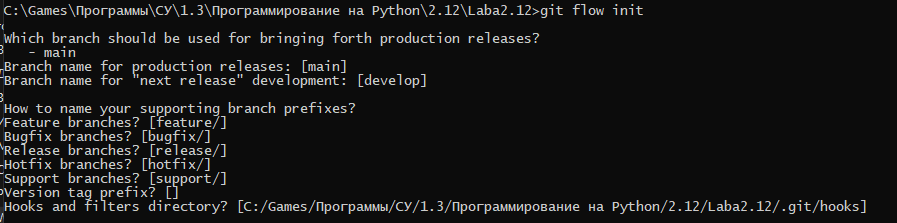


Рисунок 1.3 Организация репозитория в соответствии с моделью ветвления git-flow

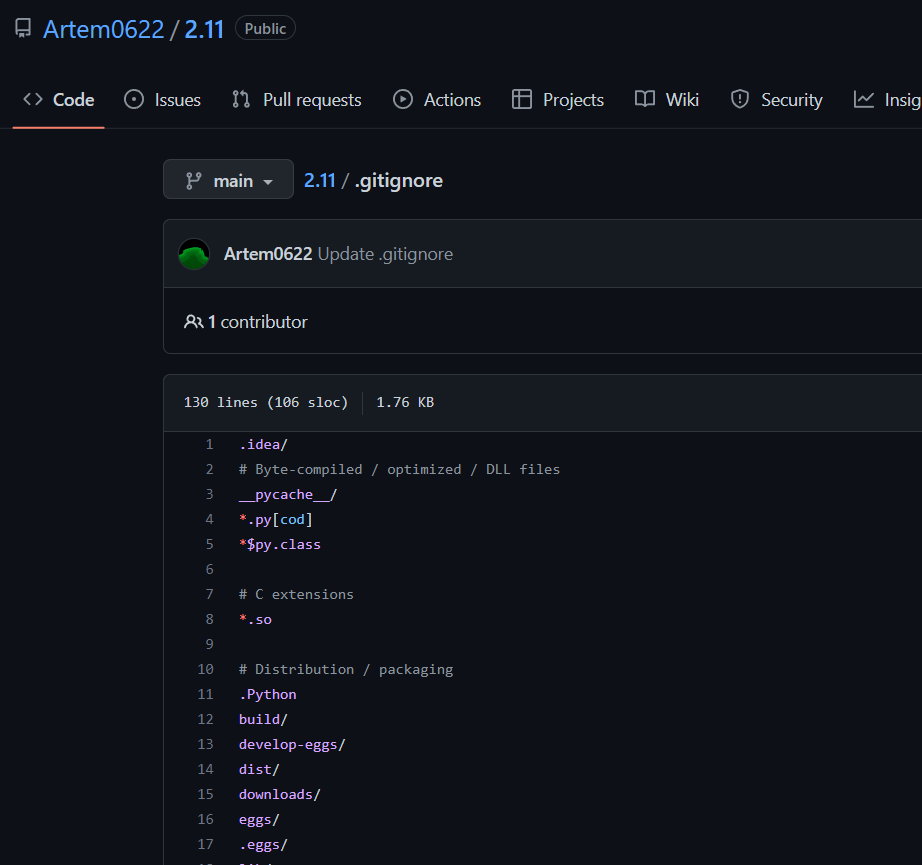


Рисунок 1.4 Изменение .gitignore

3.(3 вариант). Выполнил индивидуальное задание.

3. Вводятся два списка (каждый с новой строки) из слов, записанных через пробел. Имеется

функция, которая преобразовывает эти две строки в два списка слов и возвращает эти

списки. Определите декоратор для этой функции, который из этих двух списков

формирует словарь, в котором ключами являются слова из первого списка, а значениями –

соответствующие элементы из второго списка. Полученный словарь

должен возвращаться при вызове декоратора. Примените декоратор к первой функции и

вызовите ее. Результат (словарь) отобразите на экране.

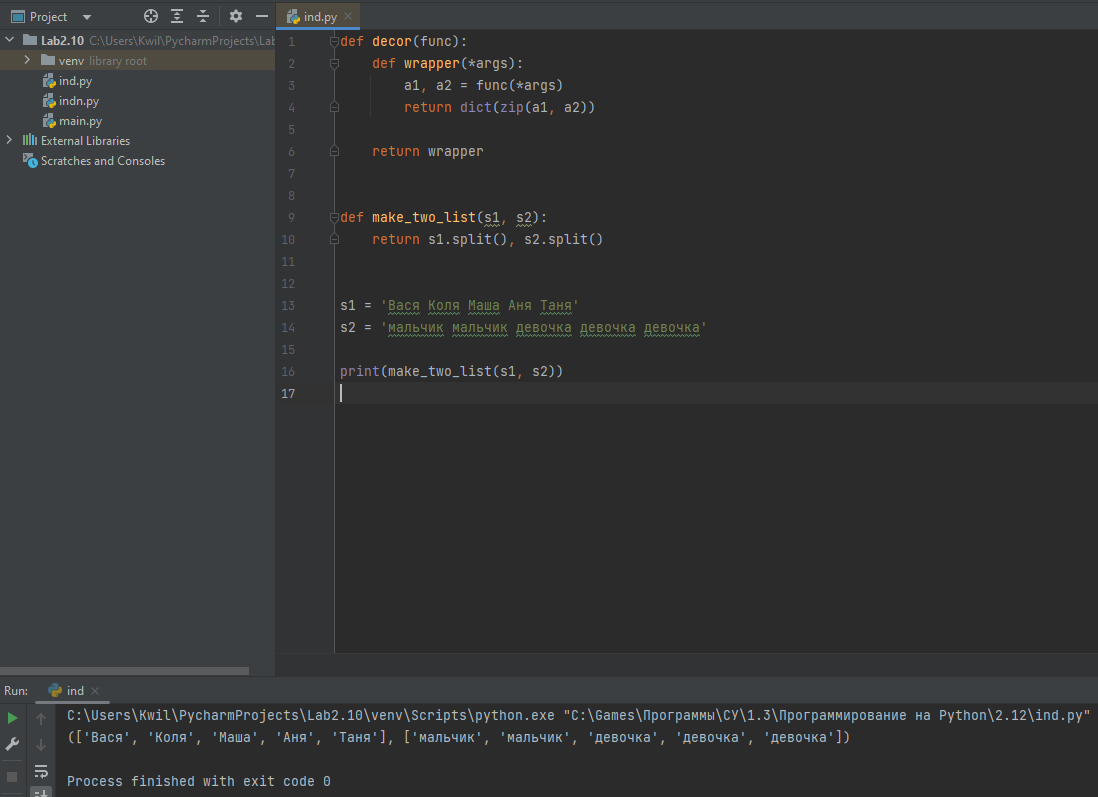


Рисунок 2.1 Вывод программы индивидуального задания

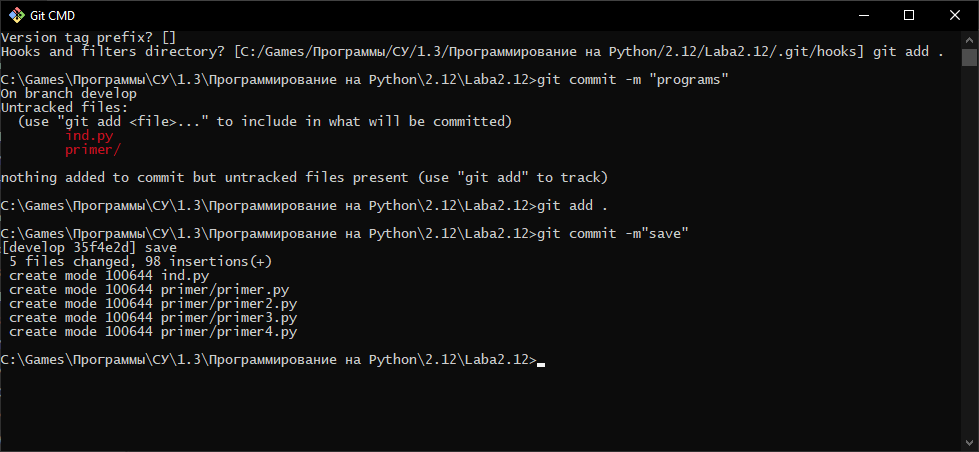


Рисунок 3.1 Коммит изменений

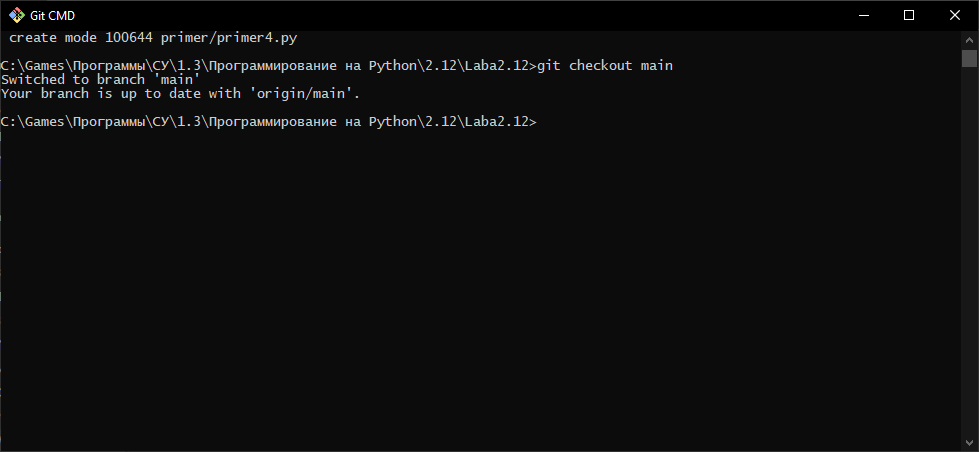


Рисунок 3.2 Переход на ветку main

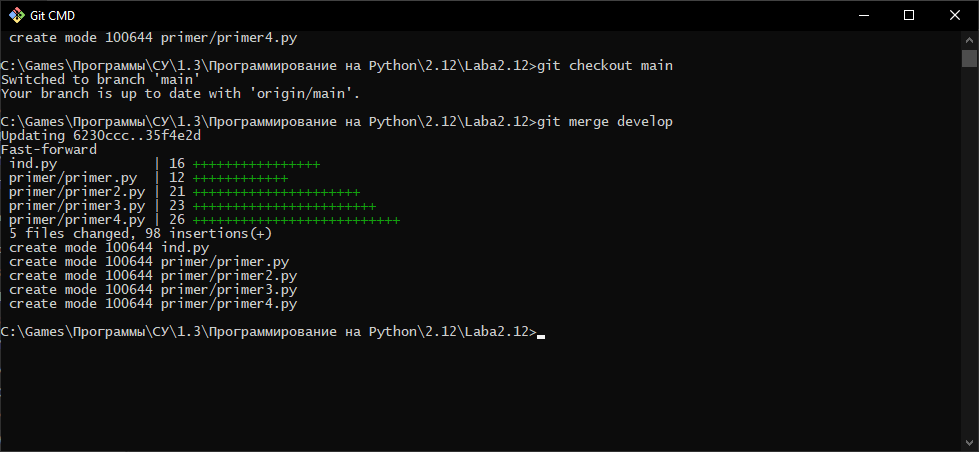


Рисунок 3.3 Слияние ветки main с develop

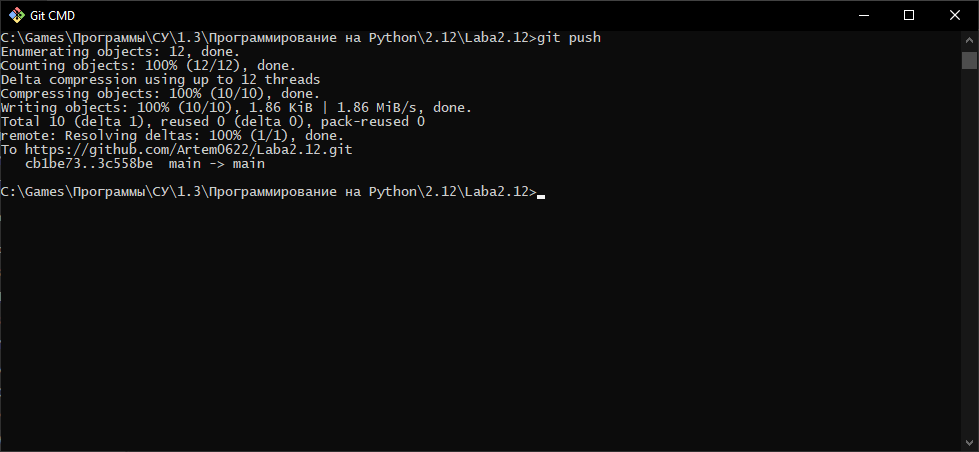


Рисунок 3.4 Пуш изменений

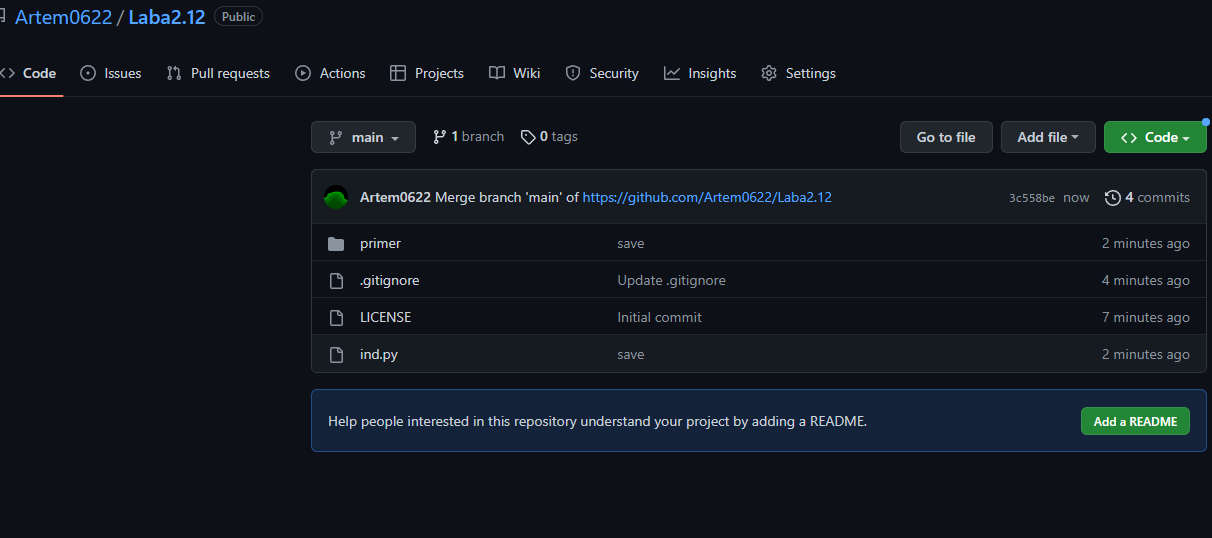


Рисунок 3.5 Изменение на уд сервере

# 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.