

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 ПРАКТИЧЕСКОЙ РАБОТЕ №15
дисциплины «Программирование на Python»
Вариант 23

Выполнил:
Мотовилов Вадим Борисович
2 курс, группа ИВТ-б-о-22-1,
09.03.01 «Информатика и
вычислительная техника»,
направленность (профиль)
«Информатика и вычислительная
техника», очная форма обучения

(подпись)

Руководитель практики:
Воронкин Роман Александрович

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2023 г.

Порядок выполнения работы:

1. Создал репозиторий и скопировал его

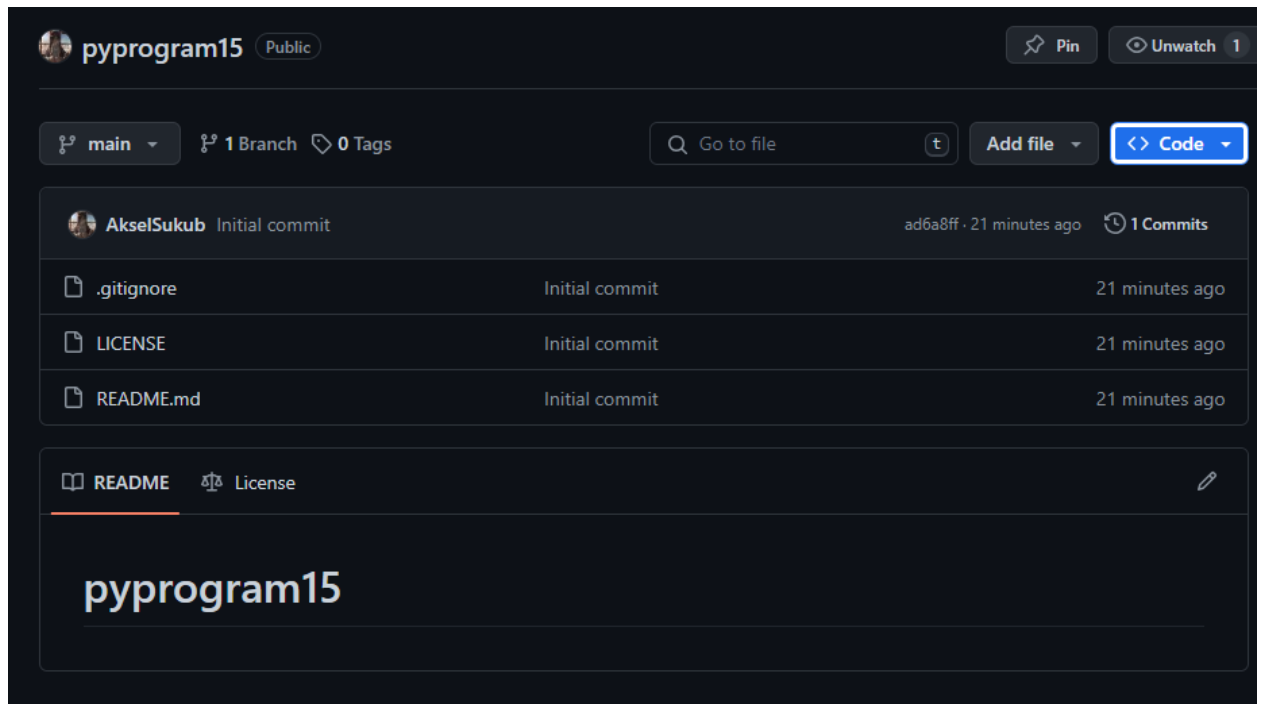


Рисунок 1. Созданный репозиторий

2. Изменил файл .gitignore и README.rm и добавил git flow

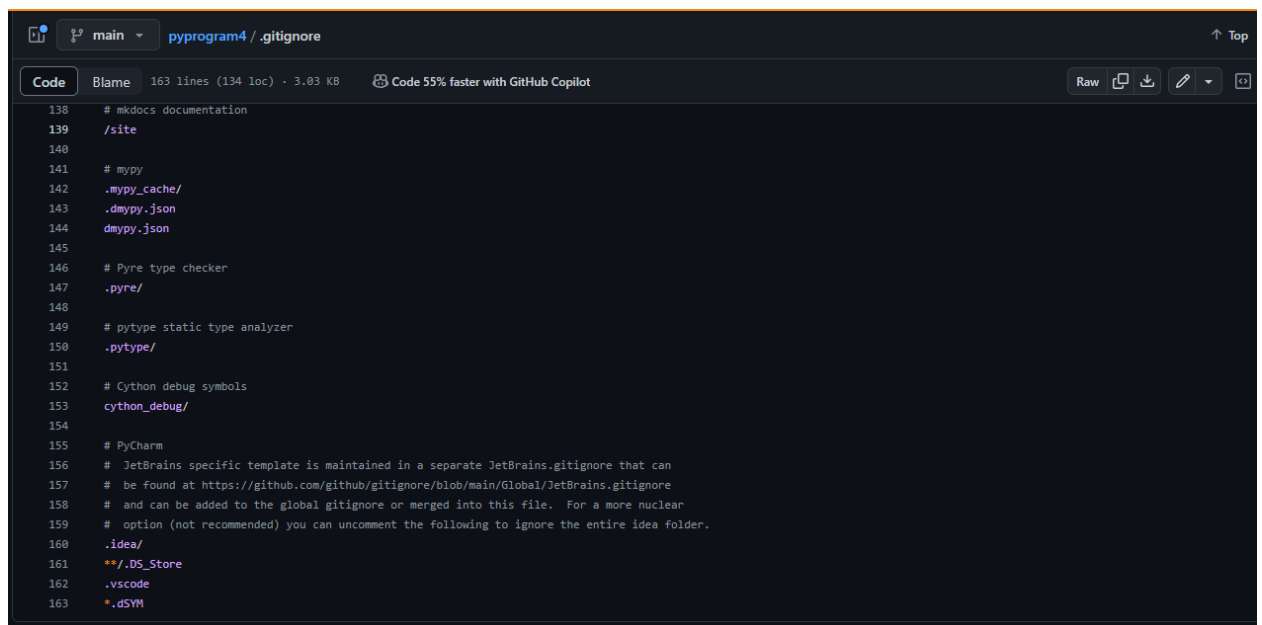
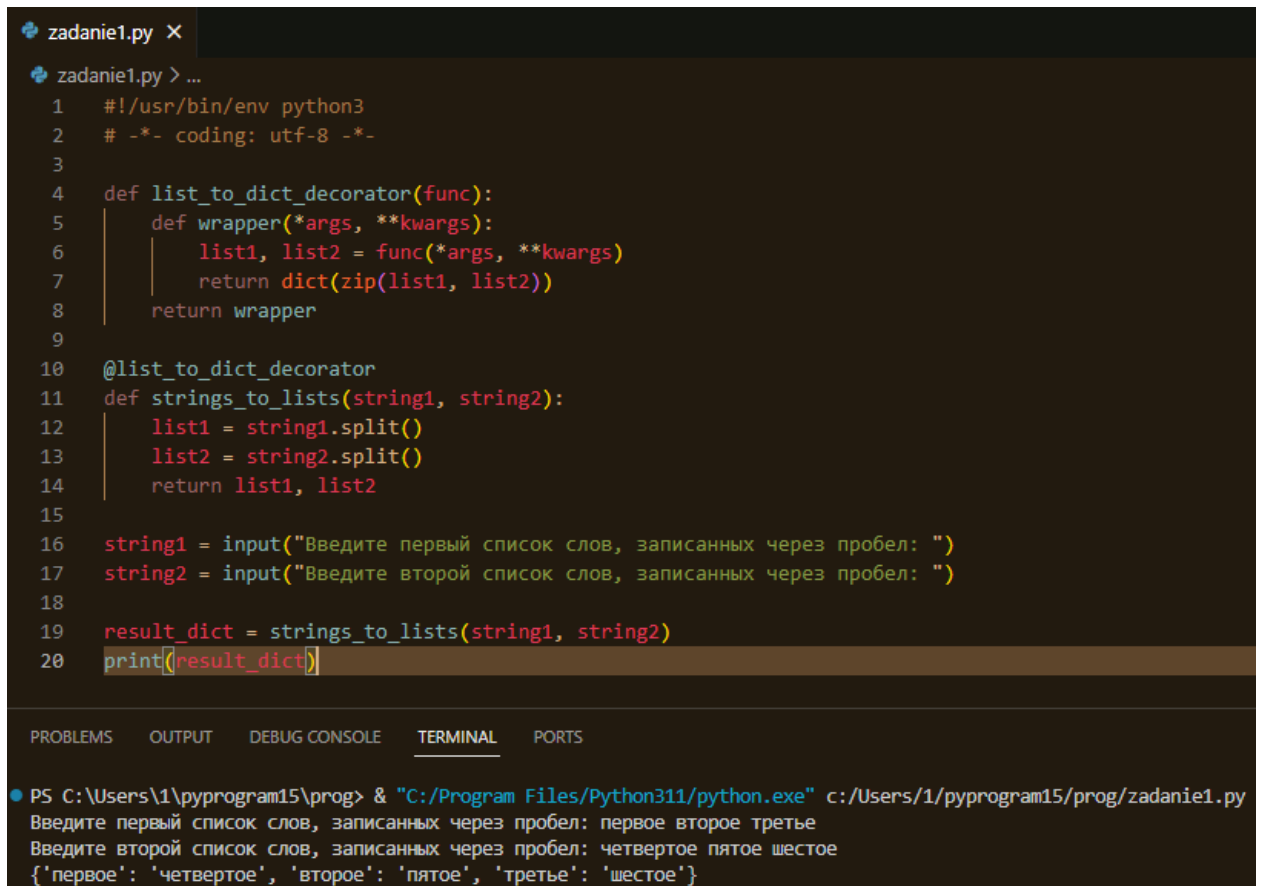


Рисунок 2. Измененный файл .gitignore

3. Выполнил задания

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.



```
zadanie1.py X
zadanie1.py > ...
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  def list_to_dict_decorator(func):
5      def wrapper(*args, **kwargs):
6          list1, list2 = func(*args, **kwargs)
7          return dict(zip(list1, list2))
8      return wrapper
9
10 @list_to_dict_decorator
11 def strings_to_lists(string1, string2):
12     list1 = string1.split()
13     list2 = string2.split()
14     return list1, list2
15
16 string1 = input("Введите первый список слов, записанных через пробел: ")
17 string2 = input("Введите второй список слов, записанных через пробел: ")
18
19 result_dict = strings_to_lists(string1, string2)
20 print(result_dict)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\1\pyprogram15\prog> & "C:/Program Files/Python311/python.exe" c:/Users/1/pyprogram15/prog/zadanie1.py
Введите первый список слов, записанных через пробел: первое второе третье
Введите второй список слов, записанных через пробел: четвертое пятое шестое
{'первое': 'четвертое', 'второе': 'пятое', 'третье': 'шестое'}
```

Рисунок 3 . Выполнение 1 задания

Ответы на контрольные вопросы:

1. **Декоратор** — это функция в Python, которая принимает другую функцию в качестве аргумента и расширяет или изменяет её поведение без изменения её исходного кода. Декораторы обычно используются для добавления функциональности к функциям или методам, например, для логирования, кеширования, проверки типов и т.д.

2. **Функции являются объектами первого класса**, потому что в Python функции можно передавать как аргументы другим функциям, возвращать их как значения из других функций, присваивать их переменным и хранить их в структурах данных. Это позволяет использовать функции в

качестве абстракций поведения и создавать высокоуровневые структуры данных и алгоритмы.

3. **Функции высших порядков** — это функции, которые принимают другие функции в качестве аргументов и/или возвращают функции как результат. Они используются для создания абстракций поведения и структурирования кода. Примеры функций высших порядков в Python — это встроенные функции `map`, `filter` и `reduce`.

4. **Декораторы работают**, принимая функцию в качестве аргумента и возвращая новую функцию, которая расширяет или изменяет поведение исходной функции. Когда декорированная функция вызывается, вызывается вместо неё функция, возвращённая декоратором.

5. **Структура декоратора функций** обычно включает в себя определение декоратора, который принимает функцию в качестве аргумента, определение внутренней функции, которая расширяет или изменяет поведение исходной функции, и возвращение этой внутренней функции из декоратора.

6. **Передача параметров декоратору** обычно требует добавления ещё одного уровня вложенности в декоратор. Внешний уровень принимает параметры декоратора, средний уровень принимает функцию, а внутренний уровень принимает аргументы функции. Вот пример декоратора, который принимает параметры:

```
def decorator_with_args(decorator_arg1, decorator_arg2):  
    def decorator(func):  
        def wrapper(*args, **kwargs):  
            print(f"Decorator arg1 = {decorator_arg1}, arg2 = {decorator_arg2}")  
            return func(*args, **kwargs)  
        return wrapper  
    return decorator  
  
@decorator_with_args("hello", "world")
```

```
def my_func(x, y):  
    return x + y
```

```
print(my_func(1, 2))
```

В этом коде декоратор `decorator_with_args` принимает два аргумента, а затем возвращает декоратор, который принимает функцию и возвращает обёртку вокруг этой функции.