

**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 №2.9**

**Дисциплина: «Программирование на Python»**

**Тема: «Рекурсия в языке Python»**

Выполнил: студент 2 курса

группы ИВТ-б-о-21-1

Назаров Никита Юрьевич

## Выполнение работы.

Owner \*

NikitaNazarov179 ▾


/

Repository name \*


2.9 ✓

Great repository names are short and memorable. Need inspiration? How about **psychic-bassoon**?

Description (optional)

☒  **Public**

Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**

You choose who can see and commit to this repository.

**Initialize this repository with:**

Skip this step if you're importing an existing repository.

☒ **Add a README file**

This is where you can write a long description for your project. [Learn more.](#)

**Add .gitignore**


Choose which files not to track from a list of templates. [Learn more.](#)


.gitignore template: Python ▾

**Choose a license**

A license tells others what they can and can't do with your code. [Learn more.](#)

License: MIT License ▾

This will set  **main** as the default branch. Change the default name in your [settings](#).

 You are creating a public repository in your personal account.

Create repository

Рисунок 1 – создание репозитория.

```
1.py x
1 ▶ #!/usr/bin/env python3
2   # -*- coding: utf-8 -*-
3
4   # Пример обратного отсчета, написанного с использованием хвостовой рекурсии:
5
6
7   def countdown(n):
8       if n == 0:
9           print("Blastoff!")
10      else:
11          print(n)
12          countdown(n-1)
13 ▶ if __name__ == '__main__':
14     print ("Введите число: ")
15     n = int(input())
16     countdown(n)
```

if \_\_name\_\_ == '\_\_main\_\_'

Run: 1 x

Н:\Python\python.exe Н:/cross/git/nazarov/2.9/Примеры/1.py  
Введите число:  
5  
5  
4  
3  
2  
1  
Blastoff!

Process finished with exit code 0  
|

Рисунок 1 – проработал 1 пример

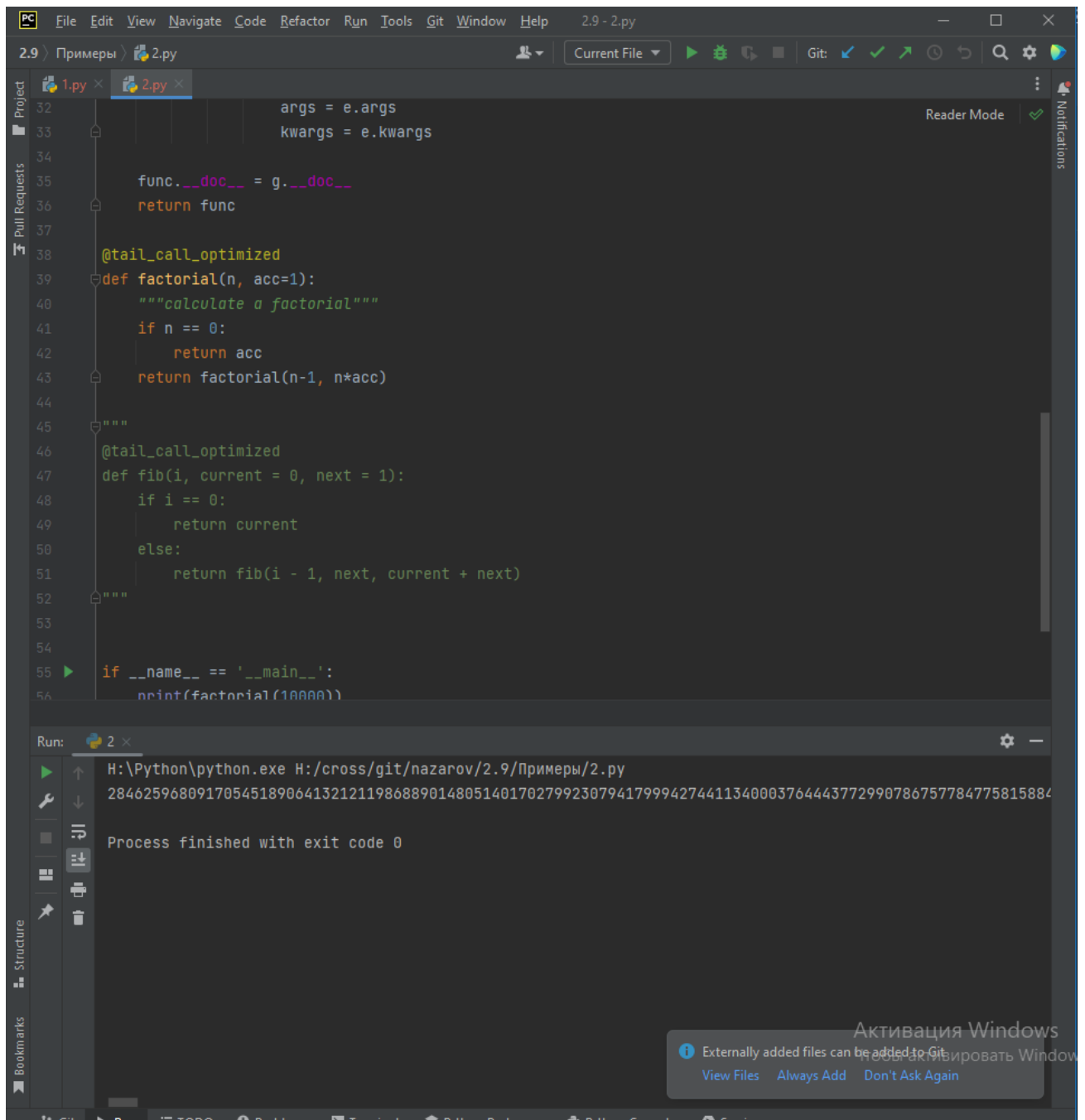


Рисунок 3 – проработал 2 пример.

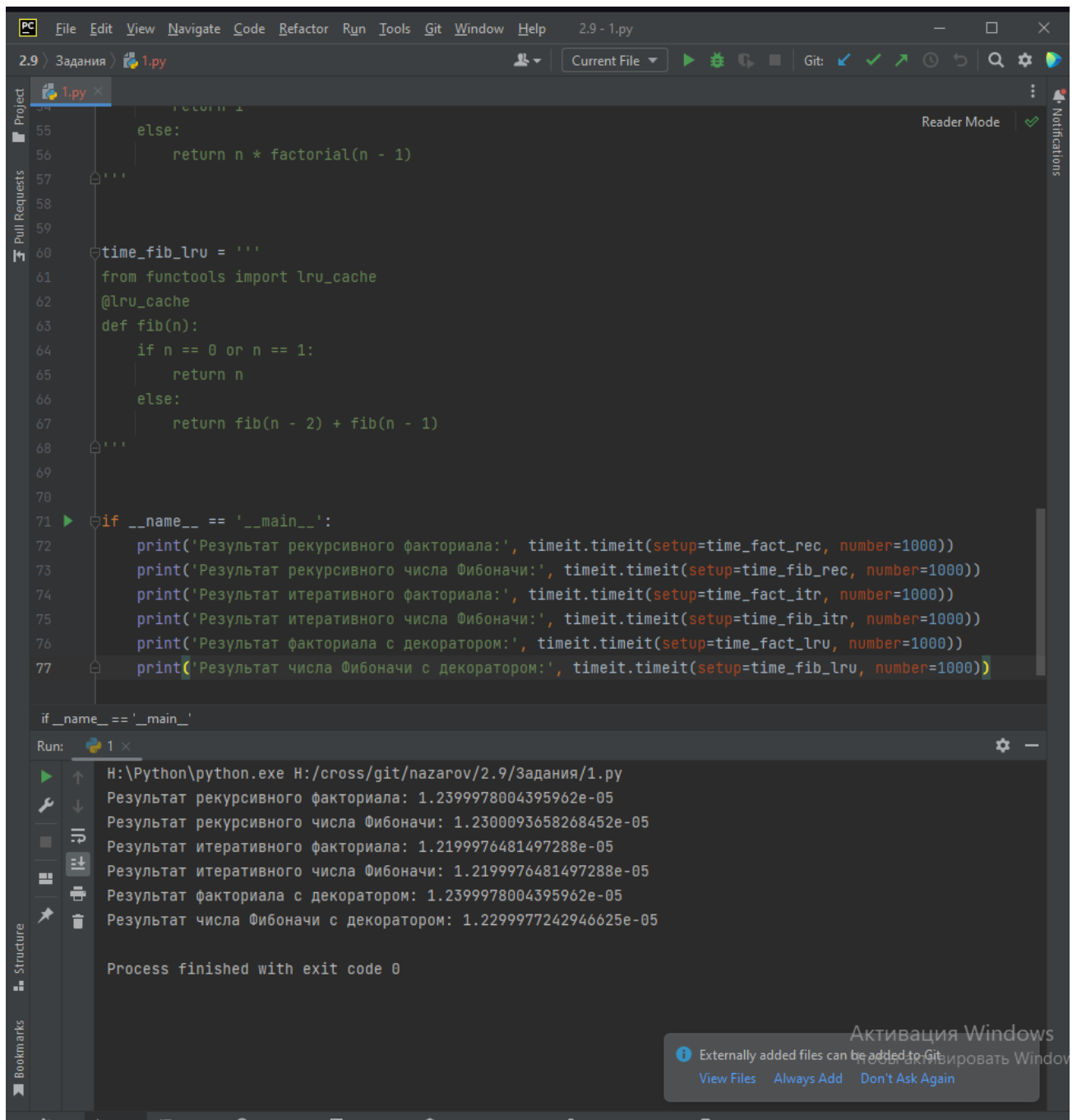


Рисунок 4 – 1 задание

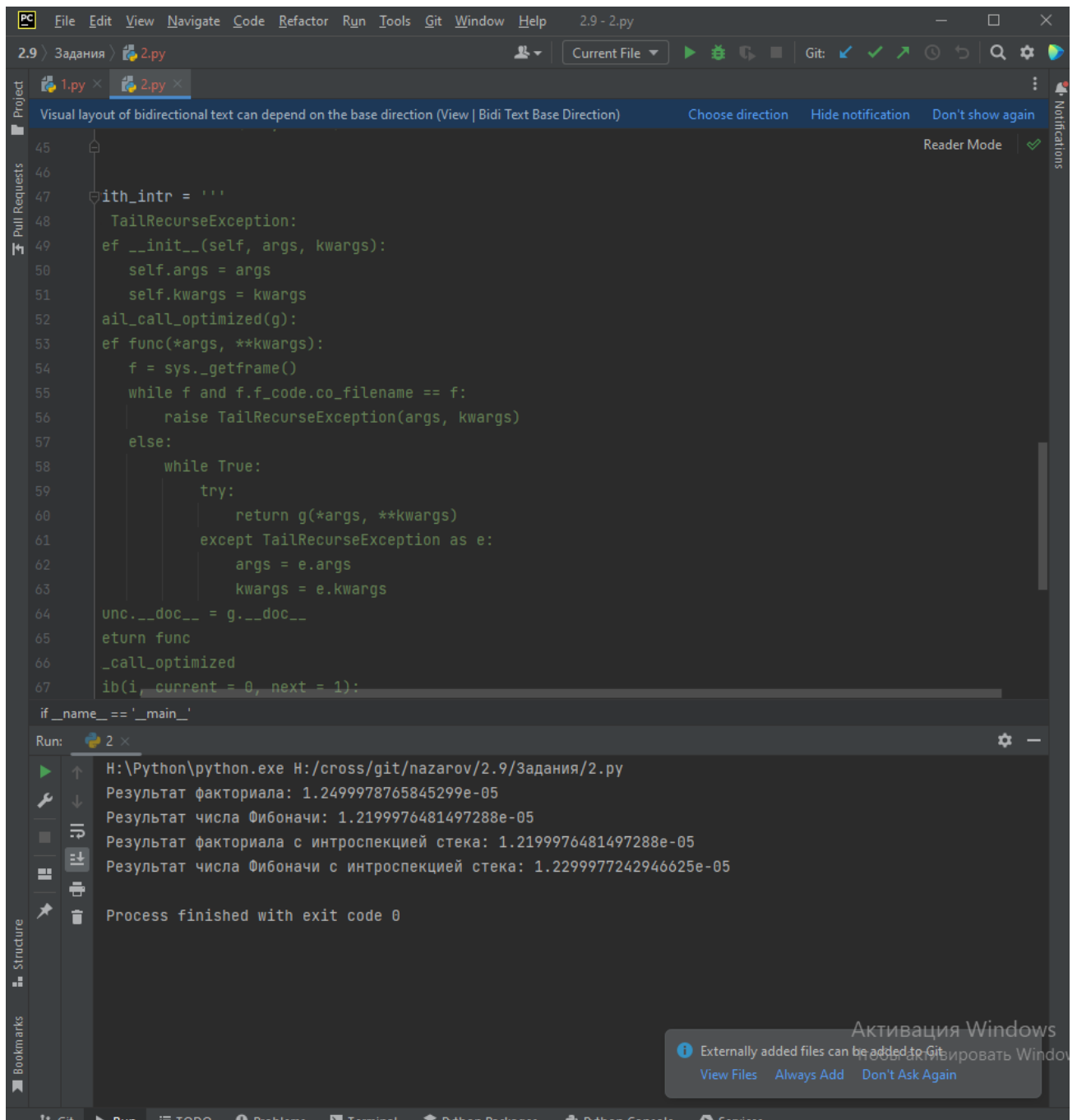


Рисунок 5 – 2 задание



самой, непосредственно (простая рекурсия) или через другие функции (сложная или косвенная рекурсия). Рекурсивная программа позволяет описать повторяющееся или даже потенциально бесконечное вычисление, причём без явных повторений частей программы и использования циклов.

## **2. Что называется базой рекурсии?**

База рекурсии – это такие аргументы функции, которые делают задачу настолько простой, что решение не требует дальнейших вложенных вызовов.

## **3. Самостоятельно изучите что является стеком программы. Как используется стек программы при вызове функций?**

Стек — это структура данных, в которой элементы хранятся в порядке поступления.

Стек хранит последовательность данных. Связаны данные так: каждый элемент указывает на тот, который нужно использовать следующим. Это линейная связь — данные идут друг за другом и нужно брать их по очереди. Из середины стека брать нельзя.

Главный принцип работы стека — данные, которые попали в стек недавно, используются первыми. Чем раньше попал — тем позже используется. После использования элемент стека исчезает, и верхним становится следующий элемент.

## **4. Как получить текущее значение максимальной глубины рекурсии в языке Python?**

Функция `sys.getrecursionlimit()` возвращает текущее значение предела рекурсии, максимальную глубину стека интерпретатора Python. Этот предел предотвращает бесконечную рекурсию от переполнения стека языка C и сбоя Python. Это значение может быть установлено с помощью `sys`.

## **5. Что произойдет если число рекурсивных вызовов превысит максимальную глубину рекурсии в языке Python?**

Существует предел глубины возможной рекурсии, который зависит от реализации Python. Когда предел достигнут, возникает исключение `RunTime`.

## **6. Как изменить максимальную глубину рекурсии в языке Python?**

С помощью `sys.setrecursionlimit(число)`.

## **7. Каково назначение декоратора `lru_cache`?**



Функция `lru_cache` предназначена для мемоизации (предотвращения повторных вычислений), т. е. кэширует результат в памяти. Полезный инструмент, который уменьшает количество лишних вычислений.

## **8. Что такое хвостовая рекурсия? Как проводится оптимизация хвостовых вызовов?**

Хвостовая рекурсия — частный случай рекурсии, при котором любой рекурсивный вызов является последней операцией перед возвратом из функции. Подобный вид рекурсии примечателен тем, что может быть легко заменён на итерацию путём формальной и гарантированно корректной перестройки кода функции. Оптимизация хвостовой рекурсии путём преобразования её в плоскую итерацию реализована во многих оптимизирующих компиляторах. В некоторых функциональных языках программирования спецификация гарантирует обязательную оптимизацию хвостовой рекурсии.

Типовой механизм реализации вызова функции основан на сохранении адреса возврата, параметров и локальных переменных функции в стеке и выглядит следующим образом:

1. В точке вызова в стек помещаются параметры, передаваемые функции, и адрес возврата.
2. Вызываемая функция в ходе работы размещает в стеке собственные локальные переменные.
3. По завершении вычислений функция очищает стек от своих локальных переменных, записывает результат (обычно — в один из регистров процессора).
4. Команда возврата из функции считывает из стека адрес возврата и выполняет переход по этому адресу. Либо непосредственно перед, либо сразу после возврата из функции стек очищается от параметров.