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 ЛАБОРАТОРНОЙ РАБОТЕ №12**

**дисциплины «Основы программной инженерии»**

|  |  |  |  |
| --- | --- | --- | --- |
|  | | Выполнил:  Звездин Алексей Сергеевич  2 курс, группа ПИЖ-б-о-22-1,  09.03.04 «Программная инженерия», направленность (профиль) «Разработка и сопровождение программного обеспечения», очная форма обучения  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | |
|  | | Руководитель практики:  Воронкин Р. А., доцент кафедры инфокоммуникаций  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | |
|  | |  | |

Отчет защищен с оценкой \_\_\_\_\_\_\_\_\_\_\_ Дата защиты\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ставрополь, 2023 г.

**Ход работы**

1. Я изучил теоретический материал работы

Изображение выглядит как текст, снимок экрана, программное обеспечение, Значок на компьютере

Автоматически созданное описание

Рисунок 1.1 – Изучение материала для лабораторной работы

1. Создал общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python

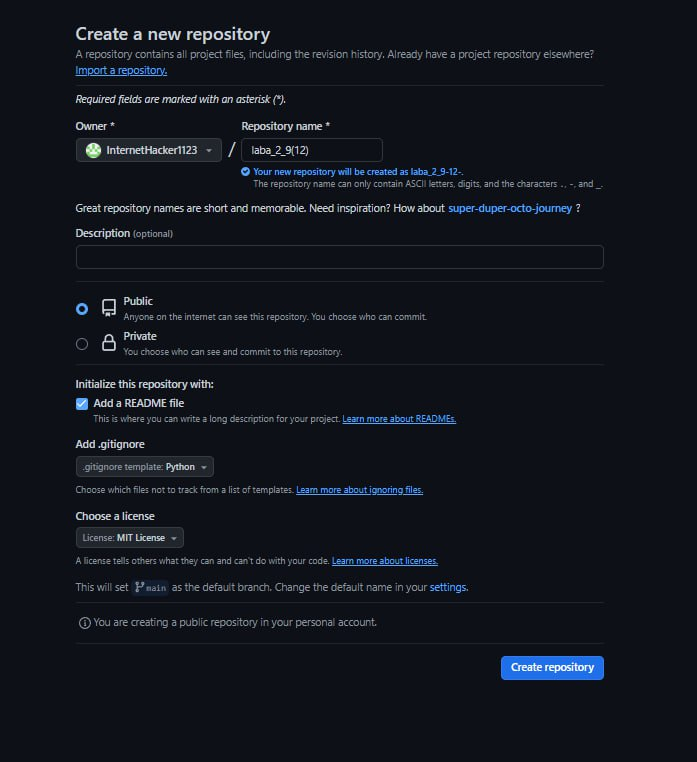


Рисунок 2.1 – Настройка репозитория

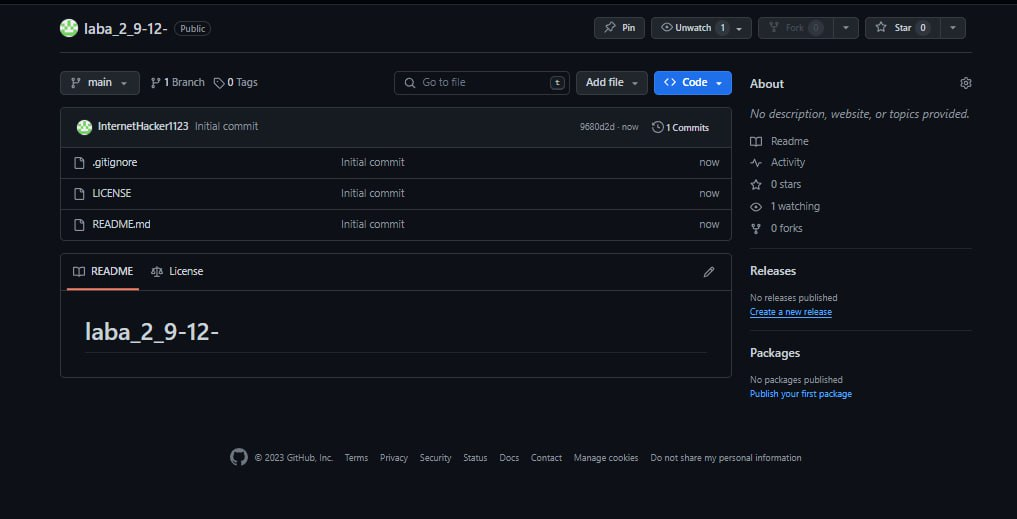


Рисунок 2.2 – Готовый репозиторий

1. Выполняю клонирование созданного репозитория



Рисунок 3.1 – Клонирование репозитория на локальный диск

1. Дополнил файл .gitignore необходимыми правилами для работы с VS Code

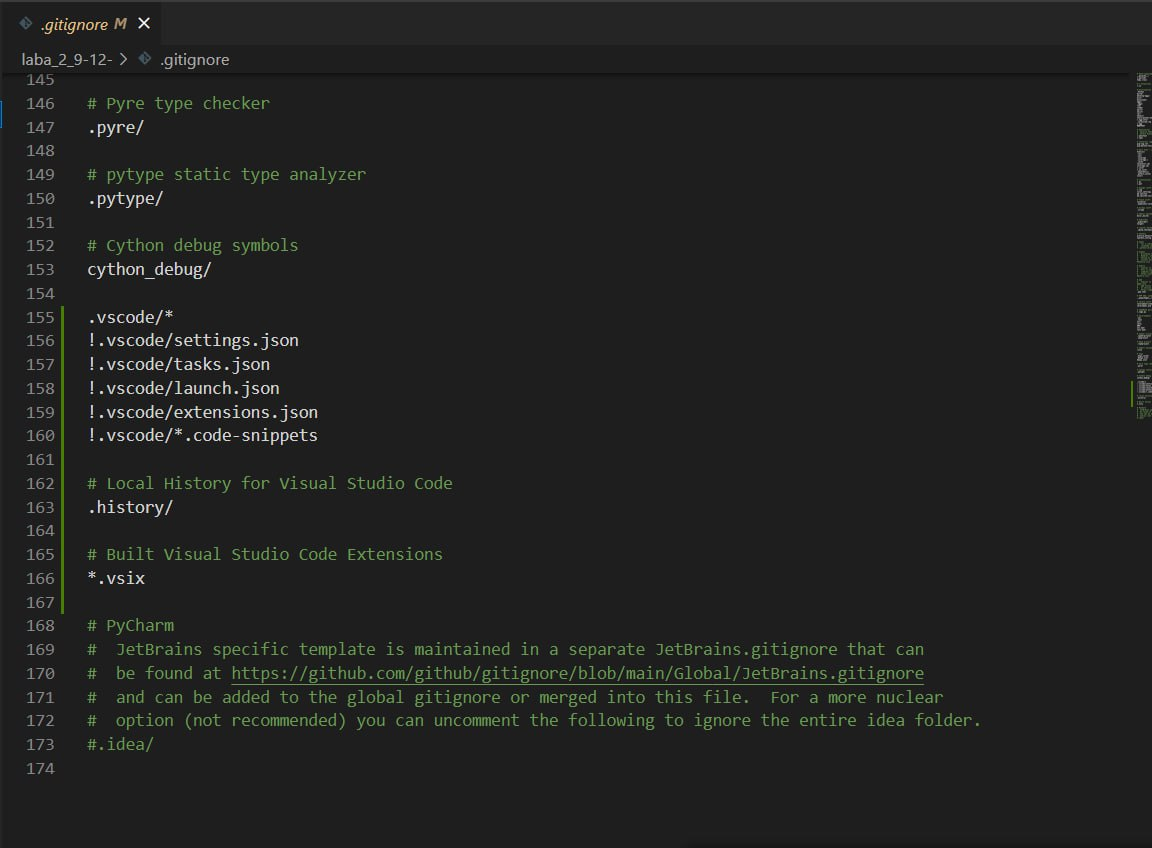


Рисунок 4.1 – .gitignore для VS Code

1. Организовал свой репозиторий в соответствии с моделью ветвления git-flow

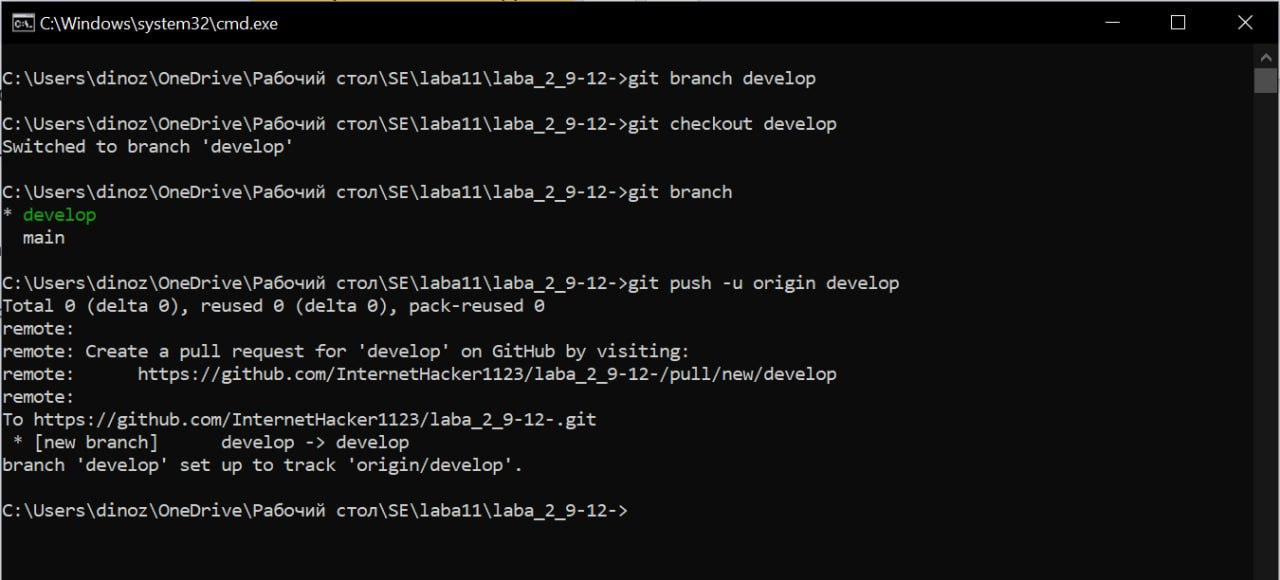


Рисунок 5.1 – Создание ветки develop от ветки main

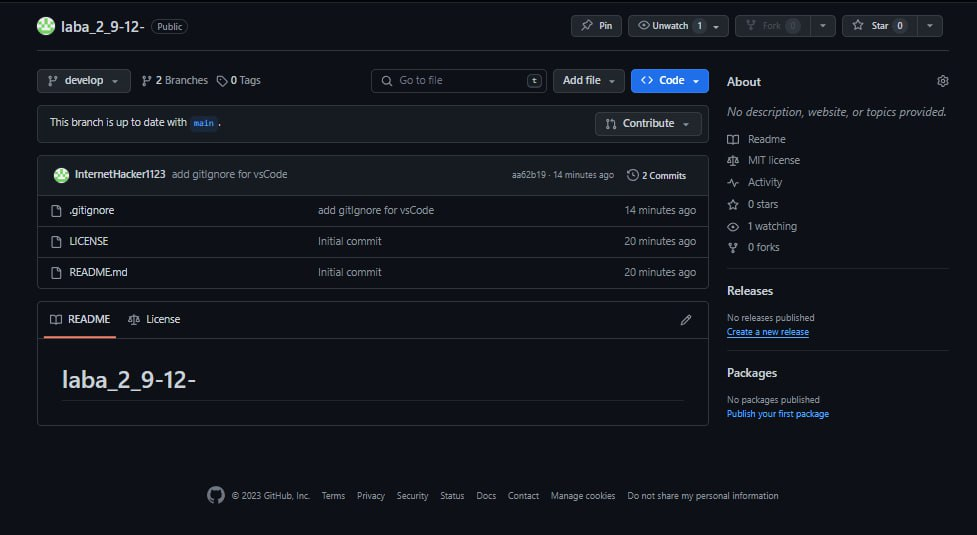


Рисунок 5.2 – Ветка develop на GitHub

1. Создал проект PyCharm в папке репозитория

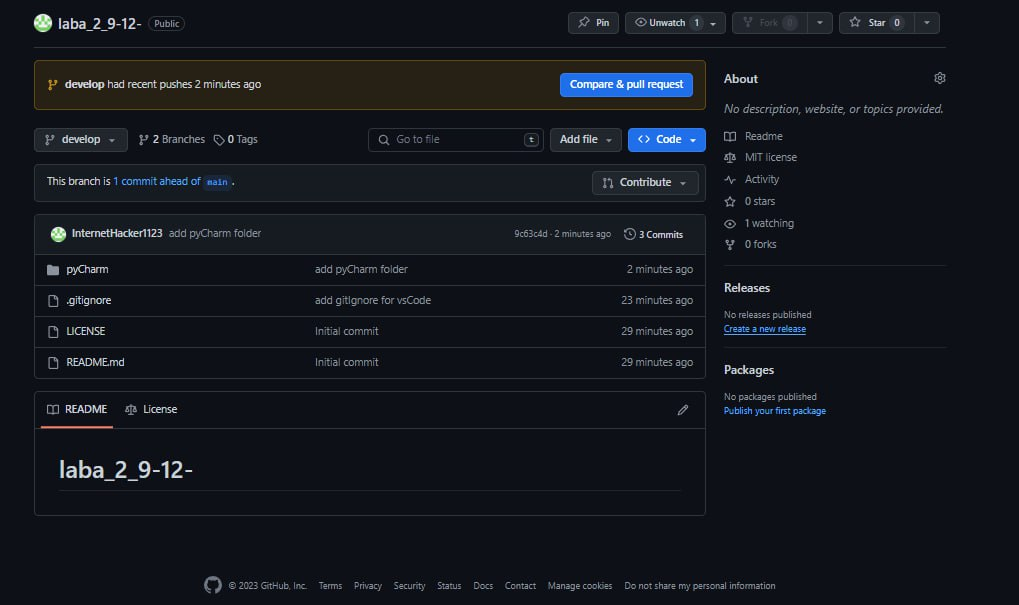
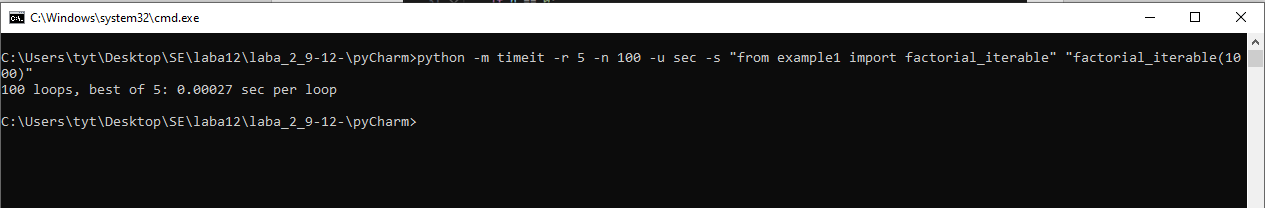


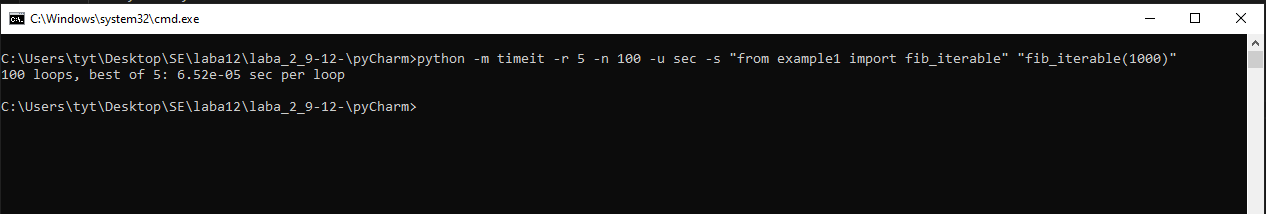
Рисунок 6.1 – Репозиторий с проектом PyCharm

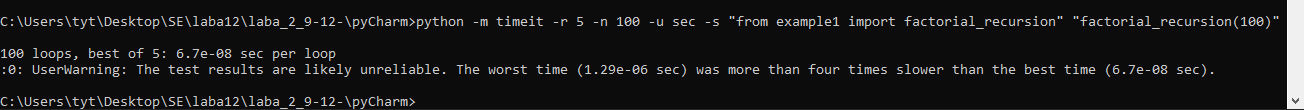
1. Самостоятельно изучил работу со стандартным пакетом Python timeit. Оценил с помощью этого модуля скорость работы итеративной и рекурсивной версий функций factorial и fib.

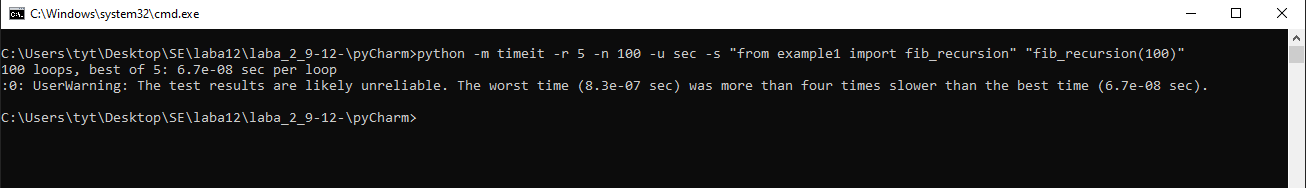


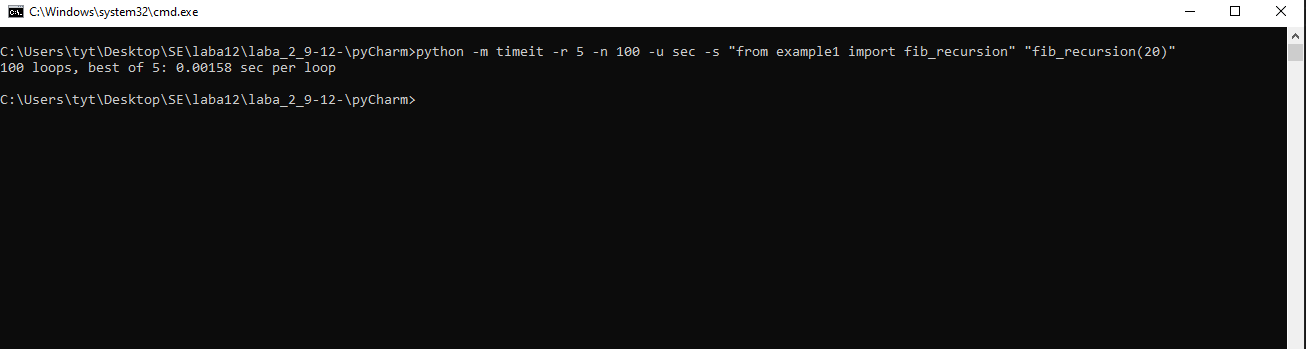
Рисунок 7.1 – Код программы example1.py

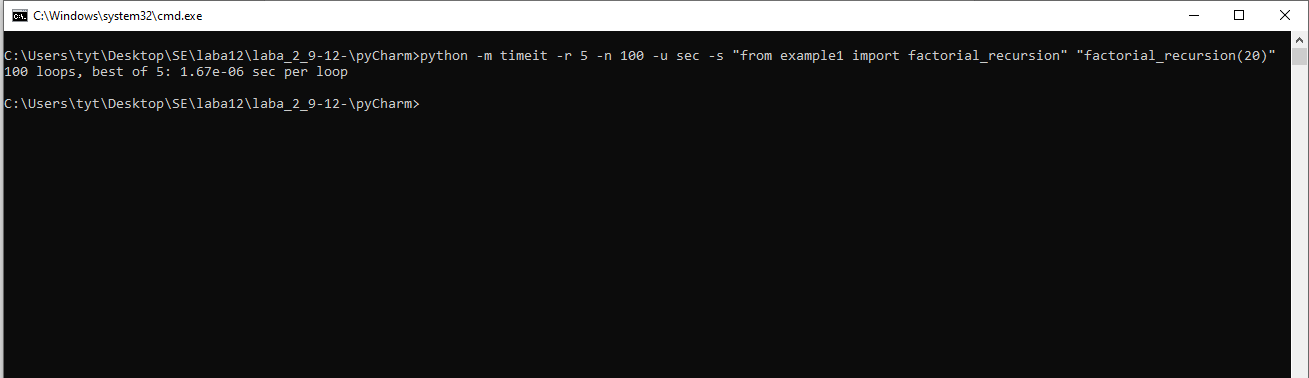
 Рисунок 7.2 – Результат замера времени выполнения

 Рисунок 7.3 – Результат замера времени выполнения

 Рисунок 7.4 – Результат замера времени выполнения

 Рисунок 7.5 – Результат замера времени выполнения

 Рисунок 7.6 – Результат замера времени выполнения

 Рисунок 7.7 – Результат замера времени выполнения

1. Самостоятельно проработал пример с оптимизацией хвостовых вызовов в Python. С помощью пакета timeit оценил скорость работы функций factorial и fib с использованием интроспекции стека и без использования интроспекции стека. Привел полученные результаты в отчет

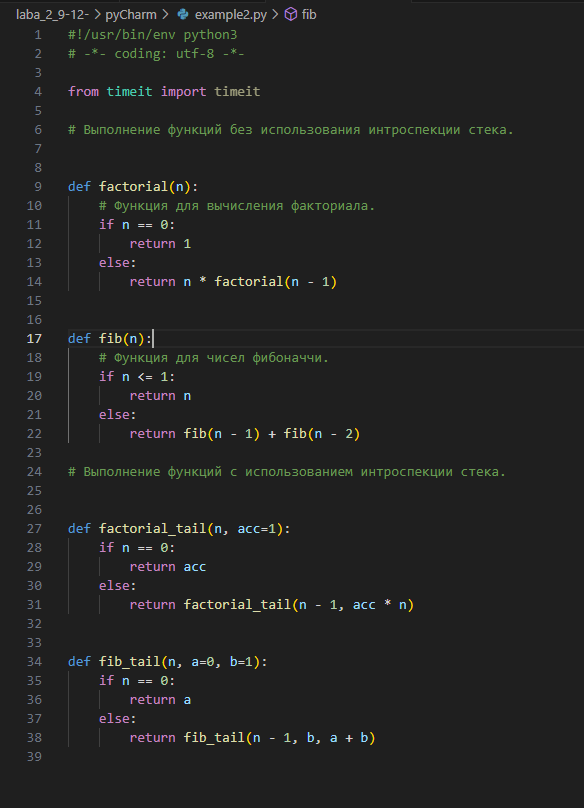


Рисунок 8.1 – Код программы example2.py в IDE PyCharm

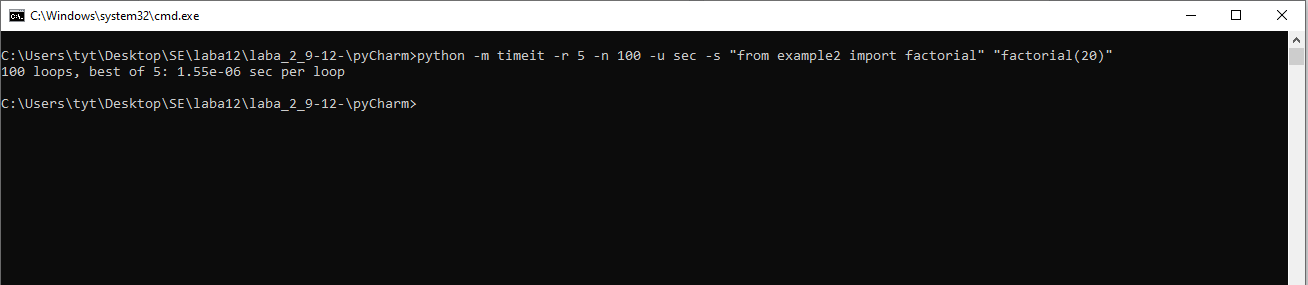
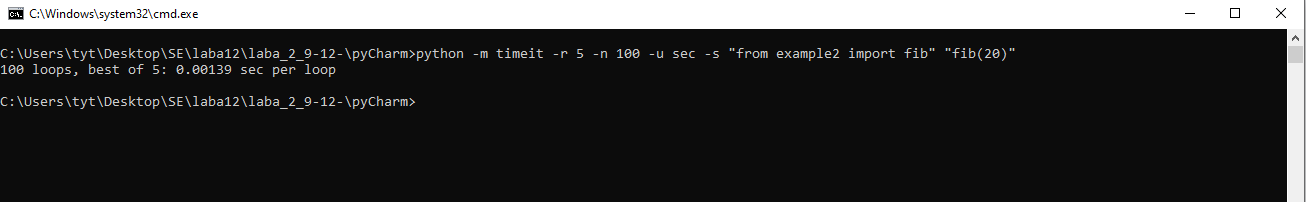


Рисунок 8.2 – Результат замера времени выполнения

 Рисунок 8.3 – Результат замера времени выполнения

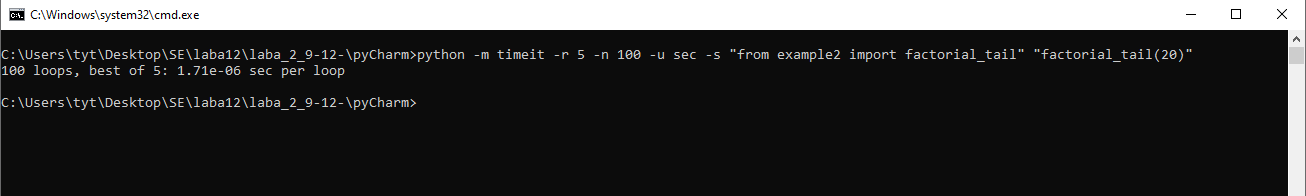
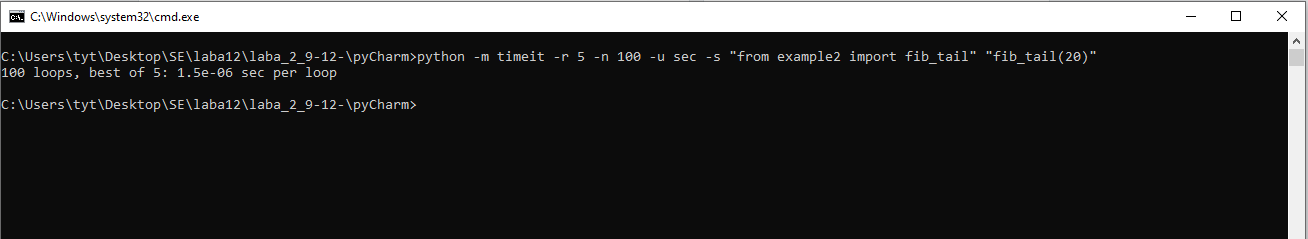


Рисунок 8.4 – Результат замера времени выполнения

 Рисунок 8.5 – Результат замера времени выполнения

1. Выполнил индивидуальные задания. Привел в отчете скриншоты работы программ решения индивидуального задания.

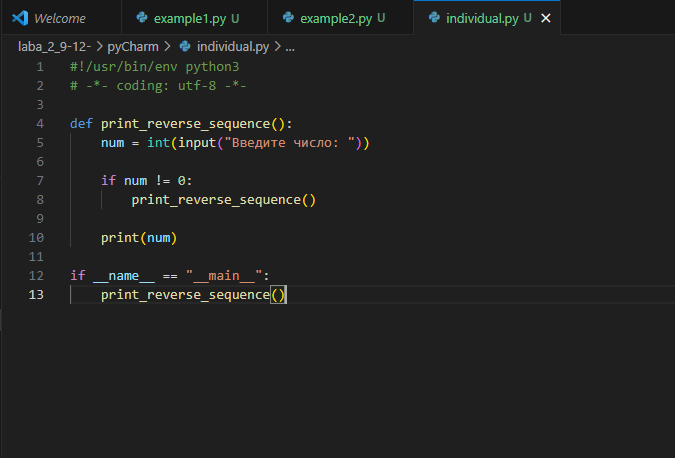


Рисунок 9.1 – Код программы individual.py в IDE PyCharm

1. Зафиксировал сделанные изменения в репозитории.

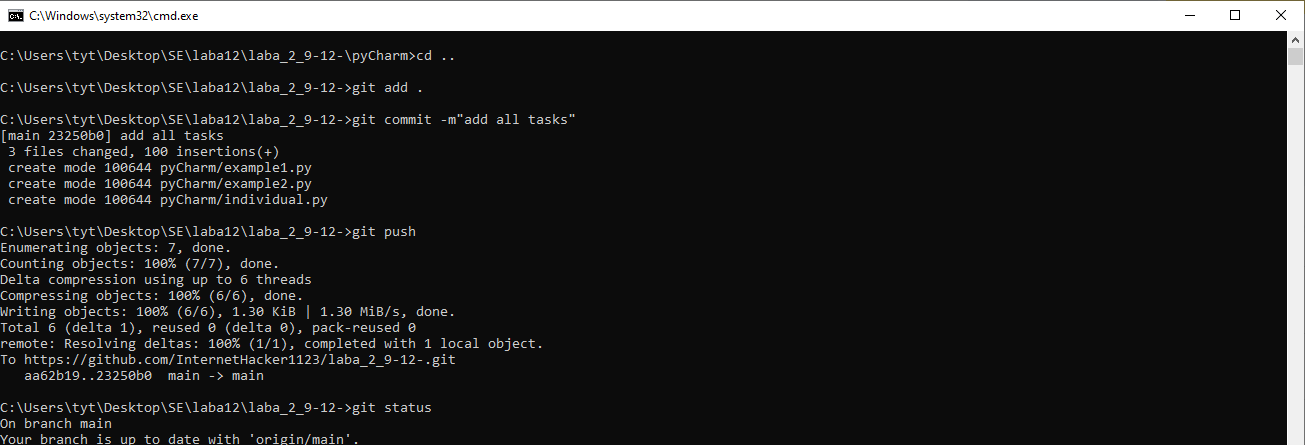


Рисунок 10.1 – Коммит файлов в репозитории git

**Контрольные вопросы**

1. Для чего нужна рекурсия?

Рекурсия функции нужна, когда требуется выполнить последовательность из одинаковых действий.

1. Что называется базой рекурсии?

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

1. Самостоятельно изучите что является стеком программы. Как используется стек программы при вызове функций?

Стек — это особая область памяти, которая используется для временного хранения данных во время выполнения программы. Стеки работает по принципу LIFO (last in, first out). Стек вызовов работает так: при вызове вложенной функции, основная функция, откуда был вызов останавливается и создается блок памяти по новый вызов. В ячейку памяти записываются значения переменных и адрес возврата.

1. Как получить текущее значение максимальной глубины рекурсии в языке Python?

Чтобы проверить текущие параметры лимита, нужно запустить: sys.getrecursionlimit()

1. 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 RuntimeError

1. Как изменить максимальную глубину рекурсии в языке Python?

Можно изменить предел глубины рекурсии с помощью вызова: sys.setrecursionlimit(limit)

1. Каково назначение декоратора lru\_cache?

Декоратор @lru\_cache() модуля functools оборачивает функцию с переданными в нее аргументами и запоминает возвращаемый результат соответствующий этим аргументам. Такое поведение может сэкономить время и ресурсы, когда дорогая или связанная с вводом/выводом функция периодически вызывается с одинаковыми аргументами.

1. 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. Чтобы оптимизировать рекурсивные функции, мы можем использовать декоратор @tail\_call\_optimized для вызова нашей функции