

Министерство науки и высшего образования Российской Федерации  
Федеральное государственное автономное образовательное учреждение  
высшего образования  
**«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Институт перспективной инженерии  
Департамент цифровых, робототехнических систем и электроники

**ОТЧЕТ  
ПО ЛАБОРАТОРНОЙ РАБОТЕ №4  
дисциплины  
«Программирование на Python»  
Вариант 19**

Выполнил:  
Поляков Никита Александрович  
2 курс, группа ИВТ-б-о-24-1,  
09.03.01 «Информатика и  
вычислительная техника»,  
направленность (профиль)  
«Программное обеспечение средств  
вычислительной техники и  
автоматизированных систем», очная  
форма обучения

---

(подпись)

Руководитель практики:  
Воронкин Р.А., доцент департамента  
цифровых, робототехнических систем и  
электроники института перспективной  
инженерии

---

(подпись)

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

Ставрополь, 2025 г.

**Тема:** Работа со списками и кортежами в языке Python

**Цель:** приобретение навыков по работе со списками и кортежами при написании программ с помощью языка программирования Python версии 3.x.

### Практическая часть:

Для начала был создан новый репозиторий на GitHub с установленными требуемыми параметрами:

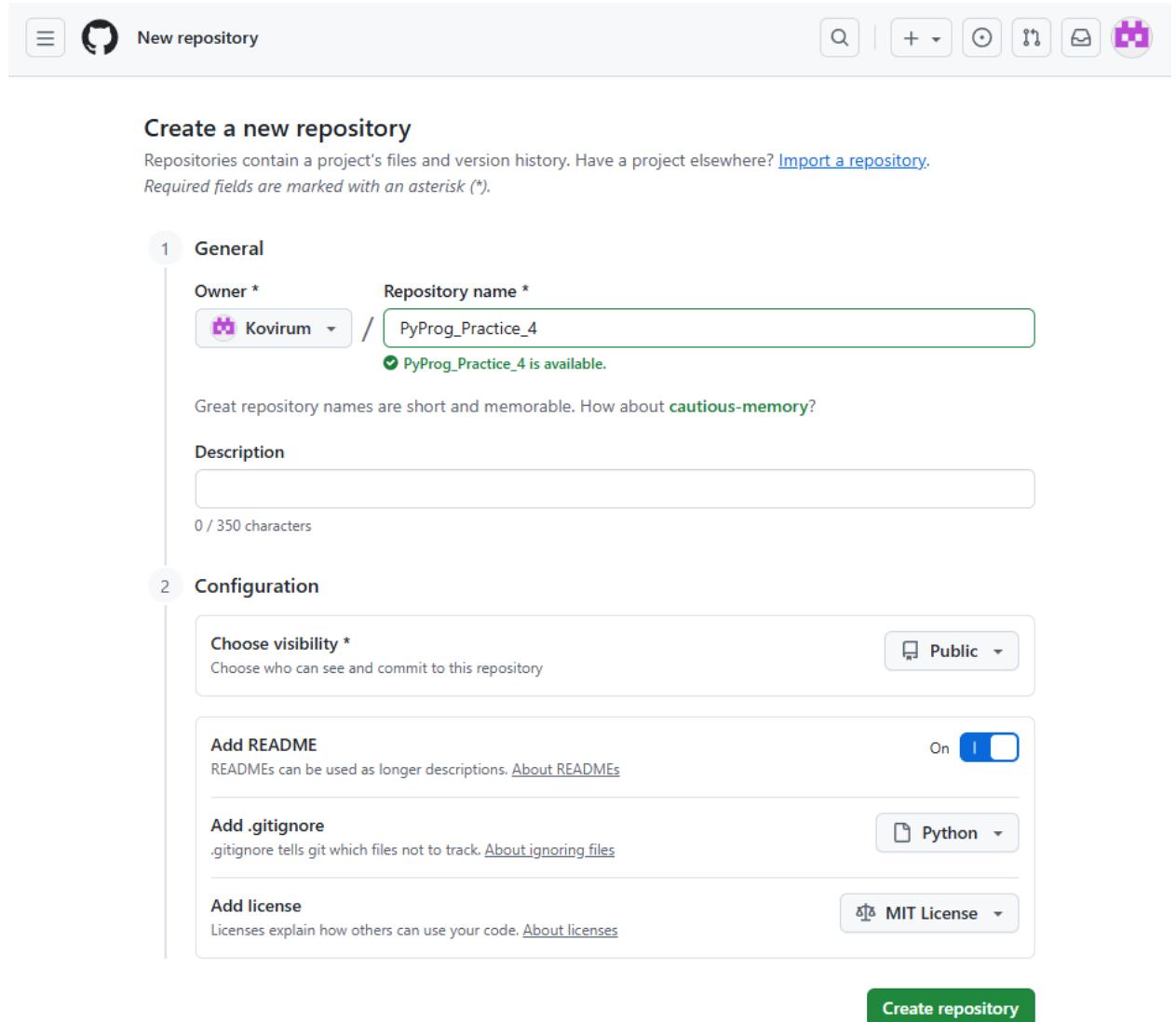


Рисунок 1. Окно создания репозитория

Ссылка на репозиторий: [https://github.com/Kovirum/PyProg\\_Practice\\_4](https://github.com/Kovirum/PyProg_Practice_4)

Далее репозиторий был клонирован на компьютер:

```
C:\Users\user\Desktop\Учёба\СКФУ\2 курс\1 семестр\Python\Лабораторная 4>git clone https://github.com/Kovirum/PyProg_Practice_4.git
Cloning into 'PyProg_Practice_4'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (5/5), done.

C:\Users\user\Desktop\Учёба\СКФУ\2 курс\1 семестр\Python\Лабораторная 4>cd PyProg_Practice_4

C:\Users\user\Desktop\Учёба\СКФУ\2 курс\1 семестр\Python\Лабораторная 4\PyProg_Practice_4>
```

Рисунок 2. Результат клонирования репозитория на компьютер

Далее в отдельных модулях языка Python были проработаны примеры лабораторной работы:

The screenshot shows the PyCharm IDE interface. The top navigation bar displays the project name "PyProg\_Practice\_4" and the current file "main". The left sidebar shows the project structure with a "examples" folder containing "example1" and "example2". The main editor window displays the content of "example1.py". The code is as follows:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import sys

if __name__ == "__main__":
    # Ввести список одной строкой
    A = list(map(int, input().split()))
    # Проверить количество элементов списка
    if len(A) != 10:
        print("Неверный размер списка", file=sys.stderr)
        exit(1)

    # Найти искомую сумму.
    s = 0
    for item in A:
        if abs(item) < 5:
            s += item

    print(s)
```

The bottom panel shows the "Run" tab selected, with "example1" listed. The run output window shows the command: "C:\Users\user\AppData\Local\Programs\Python\Python313\python.exe "C:\Users\user\Desktop\Учёба\СКФУ\2 курс\1 семестр\Python\Лабораторная 4\PyProg\_Practice\_4\examples\example1.py" and the resulting output: "1 2 3 4 5 6 7 8 9 10". Below the output, it says "Process finished with exit code 0". The status bar at the bottom indicates the file path "PyProg\_Practice\_4 > examples > example1.py", the time "17:26", and the Python version "Python 3.13".

Рисунок 3. Результат выполнения примера 1 лабораторной работы

The screenshot shows the PyCharm IDE interface. The top bar displays the project name "PyProg\_Practice\_4" and the current file "example2.py". The left sidebar shows the project structure with files like "example1.py", "example2.py", and "tuple\_example1.py", along with "examples", ".gitignore", "LICENSE", and "README.md". The main code editor window contains the following Python script:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import sys

if __name__ == "__main__":
    # Ввести список одной строкой
    a = list(map(int, input().split()))
    # Если список пуст, завершить программу
    if not a:
        print("Заданный список пуст", file=sys.stderr)
        exit(1)

    # Определить индексы инициального и максимального элементов
    a_min = a_max = a[0]
    i_min = i_max = 0
    for i, item in enumerate(a):
        if item < a_min:
            i_min, a_min = i, item
        if item >= a_max:
            i_max, a_max = i, item

    # Проверить индексы и обменять их местами
    if i_min > i_max:
        i_min, i_max = i_max, i_min

    # Посчитать количество положительных элементов
    count = 0
    for item in a[i_min+1:i_max]:
        if item > 0:
            count += 1

    print(count)
```

The "Run" tab in the bottom navigation bar is selected, showing the command "C:\Users\user\AppData\Local\Programs\Python\Python313\python.exe" and the output window below it. The output window shows the command and the resulting count of positive numbers:

```
C:\Users\user\AppData\Local\Programs\Python\Python313\python.exe "C:\Users\user\Desktop\Учёба\СКФУ\2 курс\PyProg_Practice_4\examples\example2.py"
1 4 2 3 7 5 6 9 8 10
8
```

The status bar at the bottom right indicates the time as 16:22, encoding as CRLF, character set as UTF-8, 4 spaces, and Python version as 3.13.

Рисунок 4. Результат выполнения примера 2 лабораторной работы

The screenshot shows the PyCharm IDE interface. In the top navigation bar, the project is named 'PyProg\_Practice\_4' and the current file is 'tuple\_example1.py'. The code editor displays the following Python script:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import sys

if __name__ == "__main__":
    # Ввести список одной строкой
    A = tuple(map(int, input().split()))
    # Проверить количество элементов списка.
    if len(A) != 10:
        print("Неверный размер списка", file=sys.stderr)
        exit(1)

    # Найти искомую сумму.
    s = sum(a for a in A if abs(a) < 5)
    print(s)
```

In the bottom right corner of the code editor, there is a status bar showing '14 ^ ^'.

The bottom half of the screen shows the terminal window. It has a 'Run' tab selected, showing the command 'C:\Users\user\AppData\Local\Programs\Python\Python313\python.exe "C:\Users\user\Desktop\Учёба\СКФУ\2 курс\PyProg\_Practice\_4\examples\tuple\_example1.py"'. The terminal output shows the user entering '1 2 3 4 5 6 7 8 9 10' and the program outputting '10'. Below the output, it says 'Process finished with exit code 0'. The status bar at the bottom of the terminal window also shows '15:40 CRLF UTF-8 4 spaces Python 3.13'.

Рисунок 5. Результат выполнения примера 1 для кортежей для лабораторной работы

The screenshot shows the PyCharm IDE interface. The project navigation bar at the top indicates the current file is `tuple_example_2.py`. The code editor displays Python code for tuple operations. The run output window below shows the execution results.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

if __name__ == "__main__":
    # 1. Цикл for
    # Заданный кортеж
    A = ("abc", "abcd", "bcd", "cde")

    # Вывести все элементы кортежа
    for item in A:
        print(item)

    # 2. Цикл while
    # Исходный кортеж - целые числа
    A = (-1, 3, -8, 12, -20)

    # Вычислить количество положительных чисел
    i = 0
    k = 0 # количество положительных чисел

    while i < len(A):
        if A[i] < 0:
            k = k + 1
        i = i + 1

    # Вывести результат
    print("k = ", k)

    # 3. Обход в цикле for
    # Заданный кортеж, содержащий строки
```

Run tuple\_example\_2

```
C:\Users\user\AppData\Local\Programs\Python\Python313\python.exe "C:/Users/user/Desktop/Учёба/СКФУ/2 курс/PyProg_Practice_4/examples/tuple_example_2.py"
abc
abcd
bcd
cde
k = 3
A = ('abc', 'ad', 'bcd')
B = ['abcabc', 'adad', 'bcdbcd']
```

23:22 CRLF UTF-8 4 spaces Python 3.13

Рисунок 6. Результат выполнения примера 2 для кортежей для лабораторной работы

Таким образом, в результате выполнения примеров данной лабораторной работы, каждый отдельный пример был реализован в виде модуля языка Python:

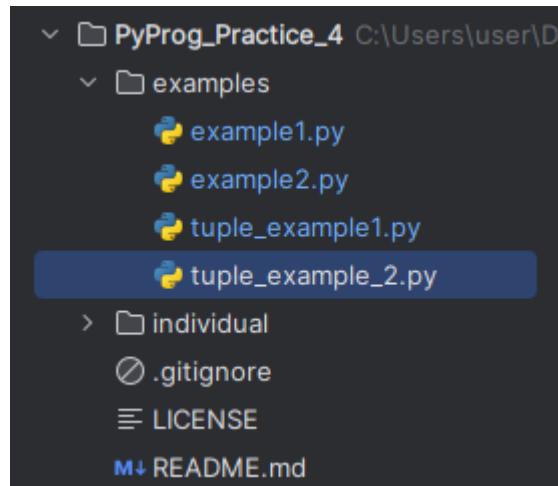


Рисунок 7. Структура проекта после выполнения примеров

Далее изменения были зафиксированы в репозитории:

```
C:\Users\user\Desktop\Учёба\СКФУ\2 курс\1 семestr\Python\Лабораторная 4\PyProg_Practice_4>git add .

C:\Users\user\Desktop\Учёба\СКФУ\2 курс\1 семestr\Python\Лабораторная 4\PyProg_Practice_4>git commit -m "feat: add lab work examples"
[main b2db37c] feat: add lab work examples
5 files changed, 111 insertions(+)
create mode 100644 examples/example1.py
create mode 100644 examples/example2.py
create mode 100644 examples/tuple_example1.py
create mode 100644 examples/tuple_example_2.py

C:\Users\user\Desktop\Учёба\СКФУ\2 курс\1 семestr\Python\Лабораторная 4\PyProg_Practice_4>
```

Рисунок 8. Результат фиксации изменений в репозитории после выполнения примеров лабораторной работы

Далее были выполнены индивидуальные задания:

Задание 1. Ввести список А из 10 элементов. Определить количество элементов, кратных 3 и индексы последнего такого элемента.

Выполнение задания:

Код:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import sys

if __name__ == "__main__":
    A = list(map(int, input().split()))

    if len(A) != 10:
        print("Введите ровно 10 элементов!",
file=sys.stderr)
```

```
exit(1)

cnt = 0
max_index = -1
for i, item in enumerate(A):
    if item % 3 == 0:
        cnt += 1
        max_index = i

print(f"Число элементов, кратных 3: {cnt}")
if max_index != -1:
    print(f"Наибольший индекс таких элементов: {max_index}")
```

Демонстрация работы программы:

The screenshot shows the PyCharm IDE interface. The top bar displays the project name 'PyProg\_Practice\_4' and the current file 'individual\_task\_1.py'. The code editor on the left shows a file structure for 'PyProg\_Practice' with subfolders 'examples' and 'individual', containing files like 'example1.py', 'example2.py', 'tuple\_example1.py', 'tuple\_example2.py', and 'individual\_task\_1.py'. The code in 'individual\_task\_1.py' is as follows:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import sys

if __name__ == "__main__":
    A = list(map(int, input().split()))

    if len(A) != 10:
        print("Введите ровно 10 элементов!", file=sys.stderr)
        exit(1)

    cnt = 0
    max_index = -1
    for i, item in enumerate(A):
        if item % 3 == 0:
            cnt += 1
            max_index = i

    print(f"Число элементов, кратных 3: {cnt}")
    if max_index != -1:
        print(f"Наибольший индекс таких элементов: {max_index}")


```

The bottom panel shows the terminal window with the command 'python individual\_task\_1.py' run. The output is:

```
C:\Users\user\AppData\Local\Programs\Python\Python313\python.exe "C:\Users\user\Desktop\Учёба\СКФУ\2 курс\individual_task_1.py"
1 2 3 4 5 6 7 8 9 10
Число элементов, кратных 3: 3
Наибольший индекс таких элементов: 8
Process finished with exit code 0
```

Рисунок 9. Результат работы программы индивидуального задания 1

Задание 2. В списке, состоящем из вещественных элементов, вычислить:

1. Произведение положительных элементов списка
2. Сумму элементов списка, расположенных до минимального значения

Упорядочить по возрастанию отдельно элементы, стоящие на четных местах, и элементы, стоящие на нечетных местах.

Выполнение задания:

Код:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import sys

if __name__ == "__main__":
    A = list(map(float, input().split()))

    pos_prod = 1
    min_el_ind = 0
    min_el_left_sum = 0

    for i, item in enumerate(A):
        if item > 0:
            pos_prod *= item
        if item < A[min_el_ind]:
            min_el_ind = i

    for item in A[:min_el_ind]:
        min_el_left_sum += item

    A_odd_el_sorted = sorted(A[::2])
    A_even_el_sorted = sorted(A[1::2])
    A_sorted = [item for pair in zip(A_odd_el_sorted,
A_even_el_sorted) for item in pair]

    print(f"Произведение положительных элементов:
{pos_prod}")
    print(f"Сумма элементов, расположенных до минимального:
{min_el_left_sum}")
    print(f"Упорядоченный список: {A_sorted}")
```

Демонстрация работы программы:

The screenshot shows the PyCharm IDE interface. The top bar displays the project name 'PyProg\_Practice\_4' and the current file 'individual\_task\_2.py'. The code editor on the left shows a Python script named 'individual\_task\_2.py' with code for calculating the product of positive elements, summing elements before the minimum, and sorting the array. The terminal window at the bottom shows the execution results: the product of positive elements is 15000.0, the sum of elements before the minimum is 20.0, and the sorted list is [0.0, 1.0, 2.0, 4.0, 3.0, 5.0, 5.0, 5.0]. The status bar at the bottom right indicates the time is 13:27, encoding is CRLF, character set is UTF-8, 4 spaces are used, and Python 3.13 is in use.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

if __name__ == "__main__":
    A = list(map(float, input().split()))

    pos_prod = 1
    min_el_ind = 0
    min_el_left_sum = 0

    for i, item in enumerate(A):
        if item > 0:
            pos_prod *= item
        if item < A[min_el_ind]:
            min_el_ind = i

    for item in A[:min_el_ind]:
        min_el_left_sum += item

    A_odd_el_sorted = sorted(A[::2])
    A_even_el_sorted = sorted(A[1::2])
    A_sorted = [item for pair in zip(A_odd_el_sorted, A_even_el_sorted) for item in pair]

print(f"Произведение положительных элементов: {pos_prod}")
print(f"Сумма элементов, расположенных до минимального: {min_el_left_sum}")
print(f"Упорядоченный список: {A_sorted}")
```

Рисунок 10. Результат работы программы индивидуального задания 2

**Задание 3.** Известны данные о численности населения (в миллионах жителей) и площади (в тысячах квадратных километров) 28 государств. Определить общую численность населения в «маленьких» государствах (чья площадь не превышает  $A$  тысяч квадратных километров).

**Выполнение задания:**

**Код:**

```
#!/usr/bin/env python3
```

```
# -*- coding: utf-8 -*-

import sys

if __name__ == "__main__":
    G_populations = tuple(map(int, input("Введите
численности населения (в млн. жителей) 28 государств:
").split()))
    G_squares = tuple(map(int, input("Введите площади 28
государств (в тыс. кв. км.): ").split()))
    A = int(input("Введите максимальную площадь
\"маленького\" государства (в тыс. кв. км.): "))

    if len(G_populations) != 28:
        print("Вы должны ввести ровно 28 чисел населений
государств!", file=sys.stderr)
        exit(1)
    if len(G_squares) != 28:
        print("Вы должны ввести ровно 28 площадей
государств!", file=sys.stderr)
        exit(1)

    small_g_population = 0
    for i, item in enumerate(G_populations):
        if G_squares[i] <= A:
            small_g_population += item

    print(f"Общая численность населения в \"маленьких\""
          "государствах: {small_g_population} млн. жителей")
```

Демонстрация работы программы:

The screenshot shows the PyCharm IDE interface. The top bar displays the project name 'PyProg\_Practice\_4' and the current file 'individual\_task\_3.py'. The code editor on the left shows the file structure of the project, including 'examples' and 'individual' subfolders containing various Python files like 'example1.py', 'example2.py', 'tuple\_exar1.py', 'tuple\_exar2.py', and several 'individual...' files. The main code area contains a script named 'individual\_task\_3.py' with the following content:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import sys

if __name__ == "__main__":
    G_populations = tuple(map(int, input("Введите численности населения (в млн. жителей) 28 государств: ").split()))
    G_squares = tuple(map(int, input("Введите площади 28 государств (в тыс. кв. км.): ").split()))
    A = int(input("Введите максимальную площадь \"маленького\" государства (в тыс. кв. км.): "))

    if len(G_populations) != 28:
        print("Вы должны ввести ровно 28 чисел населений государств!", file=sys.stderr)
        exit(1)
    if len(G_squares) != 28:
        print("Вы должны ввести ровно 28 площадей государств!", file=sys.stderr)
        exit(1)

    small_g_population = 0
    for i, item in enumerate(G_populations):
        if G_squares[i] <= A:
            small_g_population += item

    print(f"Общая численность населения в \"маленьких\" государствах: {small_g_population} млн. жителей")
```

The bottom panel shows the terminal window with the output of running the script. The user inputs the required data, and the script calculates and prints the total population of the countries with areas less than or equal to the specified maximum.

Рисунок 11. Результат работы программы индивидуального задания 4

Таким образом, после выполнения индивидуальных заданий структура проекта выглядит следующим образом:

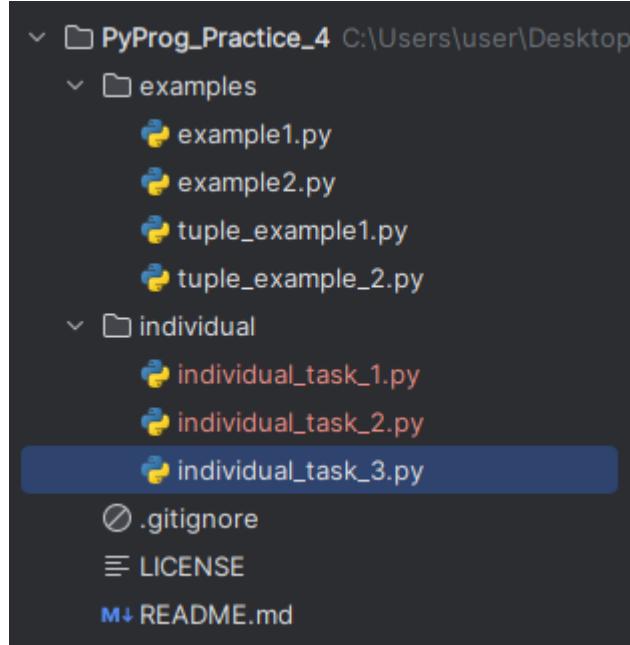


Рисунок 12. Структура проекта после выполнения индивидуальных заданий

Далее изменения были зафиксированы в репозитории:

```
C:\Users\user\Desktop\Учёба\СКФУ\2 курс\1 семестр\Python\Лабораторная 4\PyProg_Practice_4>git add .

C:\Users\user\Desktop\Учёба\СКФУ\2 курс\1 семестр\Python\Лабораторная 4\PyProg_Practice_4>git commit -m "feat: add lab work individual tasks"
[main 7826c42] feat: add lab work individual tasks
 3 files changed, 75 insertions(+)
 create mode 100644 individual/individual_task_1.py
 create mode 100644 individual/individual_task_2.py
 create mode 100644 individual/individual_task_3.py

C:\Users\user\Desktop\Учёба\СКФУ\2 курс\1 семестр\Python\Лабораторная 4\PyProg_Practice_4>
```

Рисунок 13. Результат фиксации изменений в репозитории после выполнения индивидуальных заданий

Далее был добавлен файл отчёта и отправлены изменения на сервер GitHub: