

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:

Create a new repository

Repositories contain a project's files and version history. Have a project elsewhere? [Import a repository](#).
Required fields are marked with an asterisk (*).

1 General

Owner * / Repository name *

Kovirum / PyProg_Practice_4

✓ PyProg_Practice_4 is available.

Great repository names are short and memorable. How about **cautious-memory**?

Description

0 / 350 characters

2 Configuration

Choose visibility *
Choose who can see and commit to this repository

Public

Add README
READMEs can be used as longer descriptions. [About READMEs](#)

On

Add .gitignore
.gitignore tells git which files not to track. [About ignoring files](#)

Python

Add license
Licenses explain how others can use your code. [About licenses](#)

MIT License

Create repository

Рисунок 1. Окно создания репозитория

Ссылка на репозиторий: 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 an IDE window with a project named 'PyProg_Practice_4'. The file explorer on the left shows a folder 'examples' containing 'example1.py'. The editor displays the content of 'example1.py', which is a Python script that takes a list of numbers as input, checks if the length is 10, and calculates the sum of elements whose absolute value is less than 5. The script includes comments in Russian. Below the editor, the 'Run' panel shows the command used to execute the script: '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"'. The output shows the input '1 2 3 4 5 6 7 8 9 10' and the result '10'. The process finished with exit code 0.

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6  if __name__ == "__main__":
7      # Ввести список одной строкой
8      A = list(map(int, input().split()))
9      # Проверить количество элементов списка
10     if len(A) != 10:
11         print("Неверный размер списка", file=sys.stderr)
12         exit(1)
13
14     # Найти искомую сумму.
15     s = 0
16     for item in A:
17         if abs(item) < 5:
18             s += item
19
20     print(s)
21

```

```

Run example1
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"
1 2 3 4 5 6 7 8 9 10
10
Process finished with exit code 0

```

Рисунок 3. Результат выполнения примера 1 лабораторной работы

The screenshot shows the PyCharm IDE interface. The top toolbar includes icons for file operations, running, and debugging. The left sidebar displays the project structure for 'PyProg_Practice_4', with the 'examples' folder expanded to show 'example2.py'. The main editor window displays the code for 'example2.py', which is a Python script that takes a list of integers as input, finds the minimum and maximum values, and counts the number of positive elements. The code is as follows:

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6  if __name__ == "__main__":
7      # Ввести список одной строкой
8      a = list(map(int, input().split()))
9      # Если список пуст, завершить программу
10     if not a:
11         print("Заданный список пуст", file=sys.stderr)
12         exit(1)
13
14     # Определить индексы инициального и максимального элементов
15     a_min = a_max = a[0]
16     i_min = i_max = 0
17     for i, item in enumerate(a):
18         if item < a_min:
19             i_min, a_min = i, item
20
21         if item >= a_max:
22             i_max, a_max = i, item
23
24     # Проверить индексы и обменять их местами
25     if i_min > i_max:
26         i_min, i_max = i_max, i_min
27
28     # Посчитать количество положительных элементов
29     count = 0
30     for item in a[i_min+1:i_max]:
31         if item > 0:
32             count += 1
33
34     print(count)
```

The bottom panel shows the 'Run' output for 'example2.py'. The command executed is 'C:\Users\user\AppData\Local\Programs\Python\Python313\python.exe "C:\Users\user\Desktop\Учёба\СКФУ\2 курс\...'. The output shows the input '1 4 2 3 7 5 6 9 8 10' and the result '8'. The status bar at the bottom indicates the file is 'example2.py' in the 'examples' folder, using Python 3.13 with CRLF line endings and UTF-8 encoding.

Рисунок 4. Результат выполнения примера 2 лабораторной работы

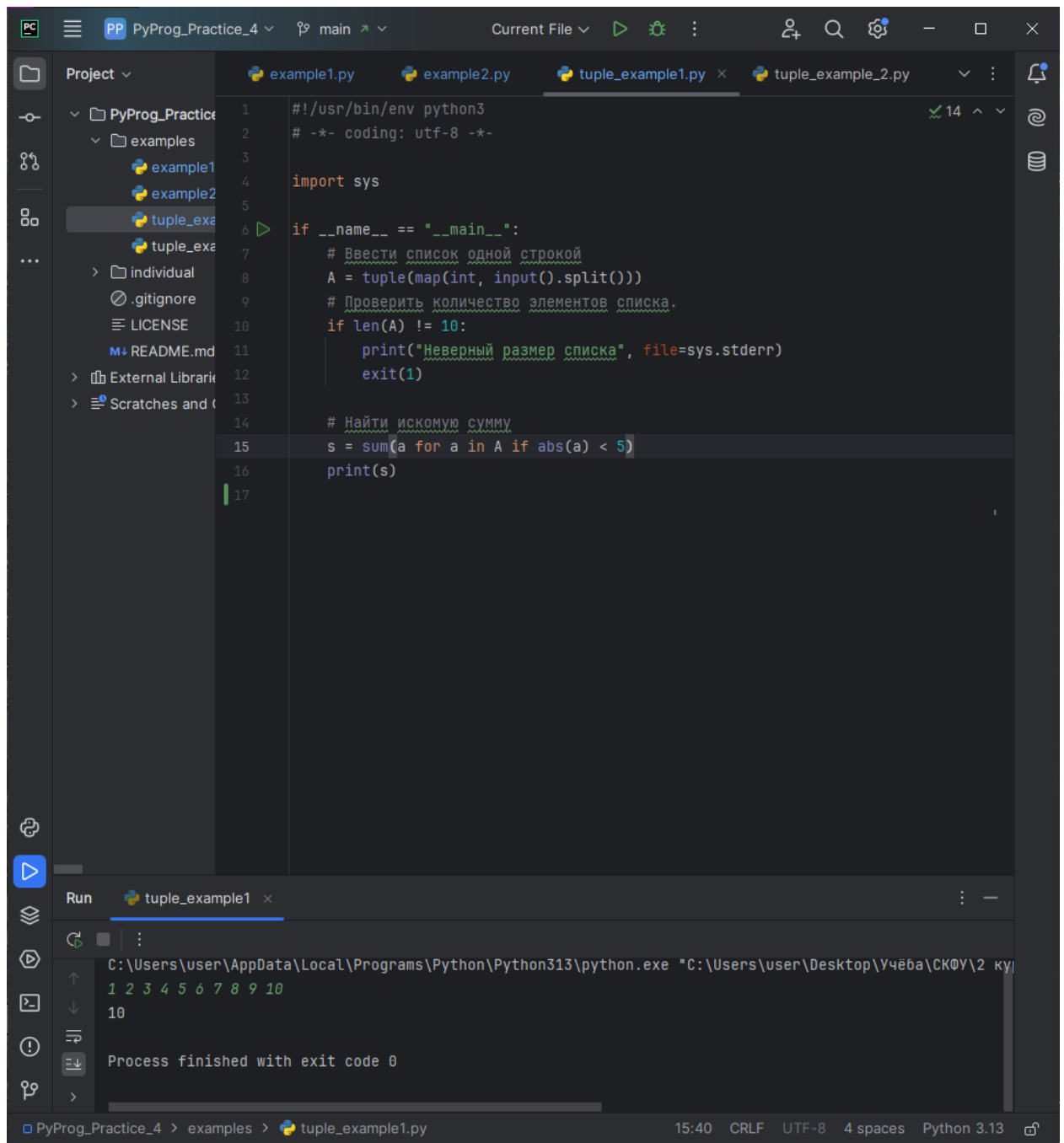


Рисунок 5. Результат выполнения примера 1 для кортежей для лабораторной работы

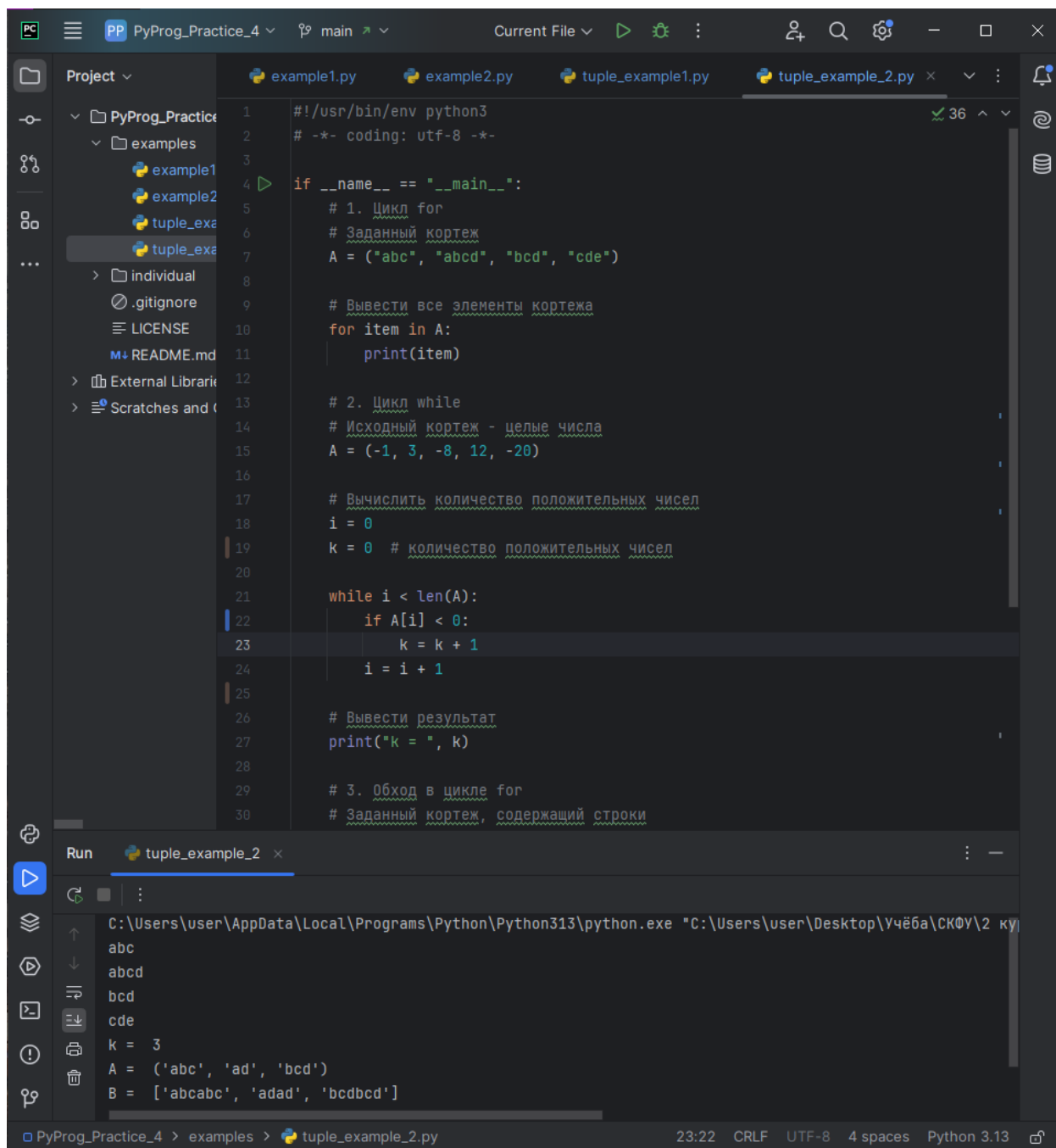


Рисунок 6. Результат выполнения примера 2 для кортежей для лабораторной работы

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

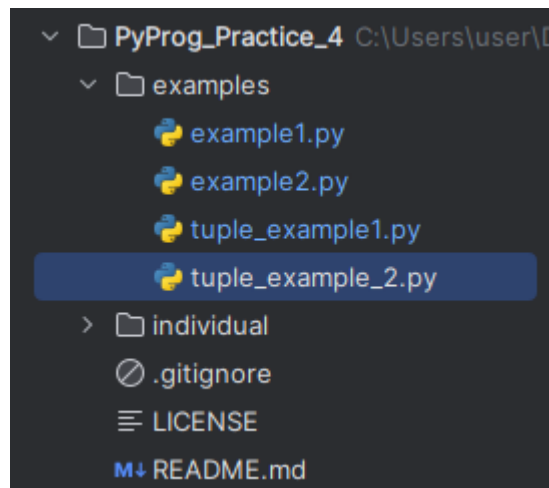


Рисунок 7. Структура проекта после выполнения примеров

Далее изменения были зафиксированы в репозитории:

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

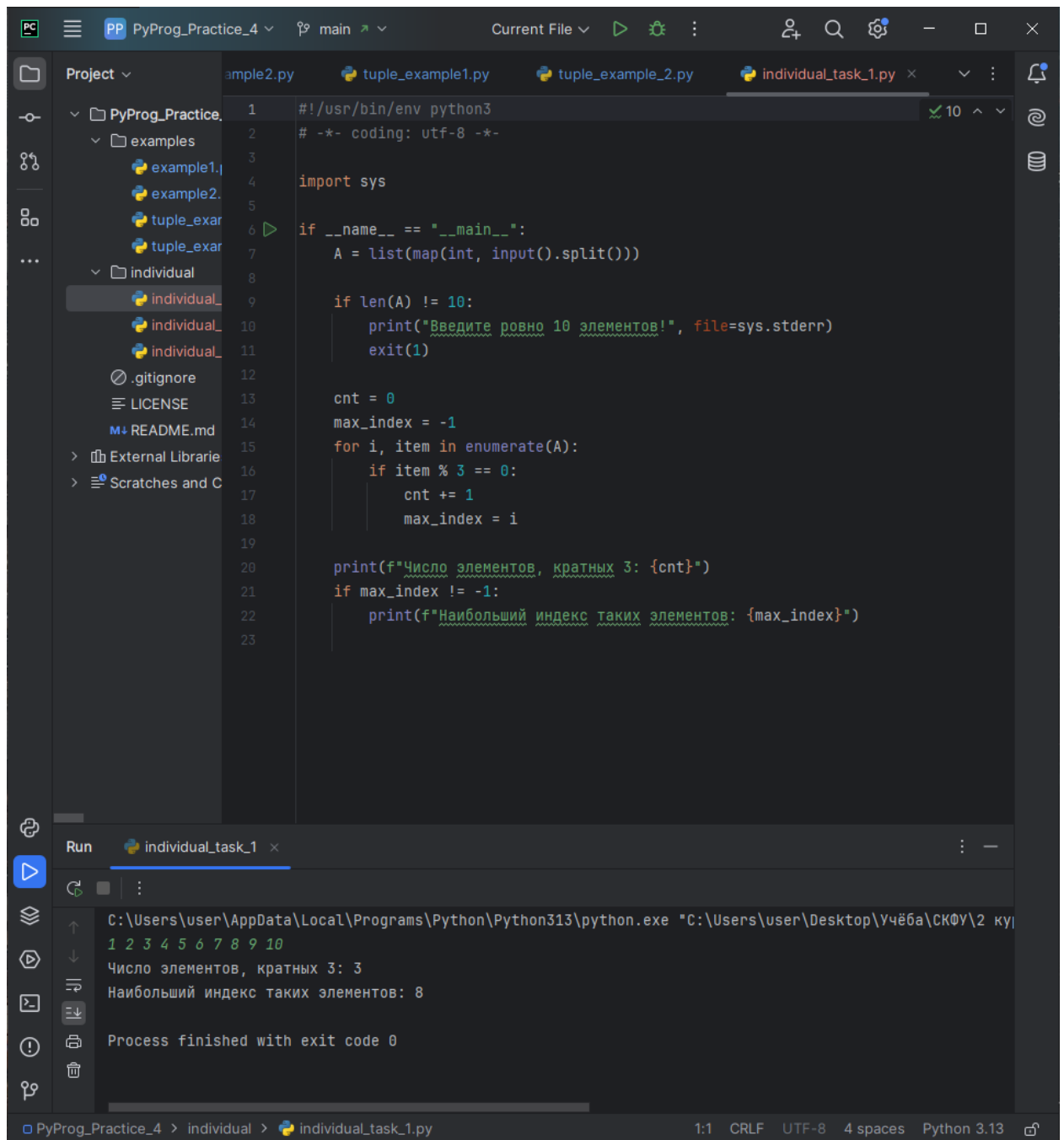


Рисунок 9. Результат работы программы индивидуального задания 1

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 toolbar includes icons for file operations, running, and debugging. The 'Project' sidebar on the left shows a directory structure with 'PyProg_Practice_4' containing 'examples' and 'individual' subdirectories. The main editor displays 'individual_task_2.py' with the following Python code:

```
#!/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}")
```

The 'Run' console at the bottom shows the execution output for 'individual_task_2.py':

```
C:\Users\user\AppData\Local\Programs\Python\Python313\python.exe "C:\Users\user\Desktop\Учѐба\СКФУ\2 курс\PyProg_Practice_4\individual\individual_task_2.py"
5 5 5 5 0 1 3 4 2
Произведение положительных элементов: 15000.0
Сумма элементов, расположенных до минимального: 20.0
Упорядоченный список: [0.0, 1.0, 2.0, 4.0, 3.0, 5.0, 5.0, 5.0]
Process finished with exit code 0
```

The status bar at the bottom indicates the file path 'PyProg_Practice_4 > individual > individual_task_2.py', the time '13:27', and settings 'CRLF UTF-8 4 spaces Python 3.13'.

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 А 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 a Python IDE with a project named 'PyProg_Practice_4'. The project structure includes folders 'examples' and 'individual'. The 'individual' folder contains files 'individual_1.py', 'individual_2.py', and 'individual_3.py'. The 'individual_2.py' file is open, showing Python code that prompts the user for population and area data for 28 countries, calculates the total population of small countries, and prints the result.

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import sys
5
6 if __name__ == "__main__":
7     G_populations = tuple(map(int, input("Введите численности населения (в млн. жителей) 28 государств (в тыс. кв. км.): ").split()))
8     G_squares = tuple(map(int, input("Введите площади 28 государств (в тыс. кв. км.): ").split()))
9     A = int(input("Введите максимальную площадь \"маленького\" государства (в тыс. кв. км.): ").split()[0])
10
11     if len(G_populations) != 28:
12         print("Вы должны ввести ровно 28 чисел населений государств!", file=sys.stderr)
13         exit(1)
14     if len(G_squares) != 28:
15         print("Вы должны ввести ровно 28 площадей государств!", file=sys.stderr)
16         exit(1)
17
18     small_g_population = 0
19     for i, item in enumerate(G_populations):
20         if G_squares[i] <= A:
21             small_g_population += item
22
23     print(f"Общая численность населения в \"маленьких\" государствах: {small_g_population} млн. жителей")
24
25
26
```

The terminal window shows the execution of the program. The user input is: "1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28" for population and "10 20 30 40 50 60 70 80 90 100 110 120 130 140 150 160 170 180 190 200 210 220 230 240 250 260 270 280" for area. The output is: "Общая численность населения в \"маленьких\" государствах: 210 млн. жителей".

Рисунок 11. Результат работы программы индивидуального задания 4

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

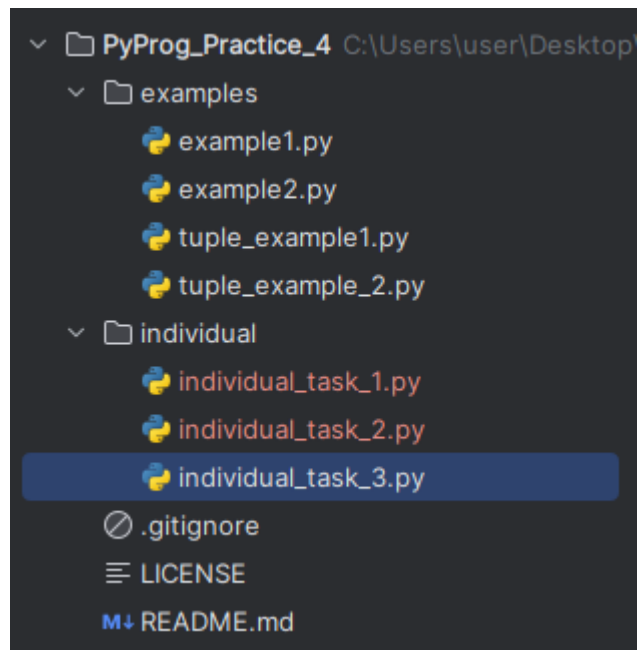


Рисунок 12. Структура проекта после выполнения индивидуальных заданий

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: