

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 titled 'PyProg_Practice_4' with a file explorer on the left and a code editor in the center. The file explorer shows a project structure with folders 'PyProg_Practice_4' and 'examples', and files 'example1.py', 'example2.py', 'tuple_example.py', 'README.md', 'LICENSE', and 'Scratches and Snippets'. The code editor displays the content of 'example1.py'.

```

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

```

Below the code editor is a 'Run' panel showing the execution of 'example1.py'. The command executed is '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.

Рисунок 3. Результат выполнения примера 1 лабораторной работы

The screenshot shows a Python IDE with a project named 'PyProg_Practice_4'. The file explorer on the left shows a directory structure with 'examples' containing 'example1', 'example2', 'tuple_example1', and 'tuple_example_2'. The main editor displays 'example2.py' with the following code:

```
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 Run console at the bottom shows the execution of the script. The command executed is: `C:\Users\user\AppData\Local\Programs\Python\Python313\python.exe "C:\Users\user\Desktop\Учёба\СКФУ\2 курс\PyProg_Practice_4\examples\example2.py"`. The output is: `1 4 2 3 7 5 6 9 8 10` followed by `8`. The status bar at the bottom indicates the file is 'example2.py' in the 'examples' directory, using Python 3.13 with CRLF line endings, UTF-8 encoding, and 4 spaces for indentation.

Рисунок 4. Результат выполнения примера 2 лабораторной работы

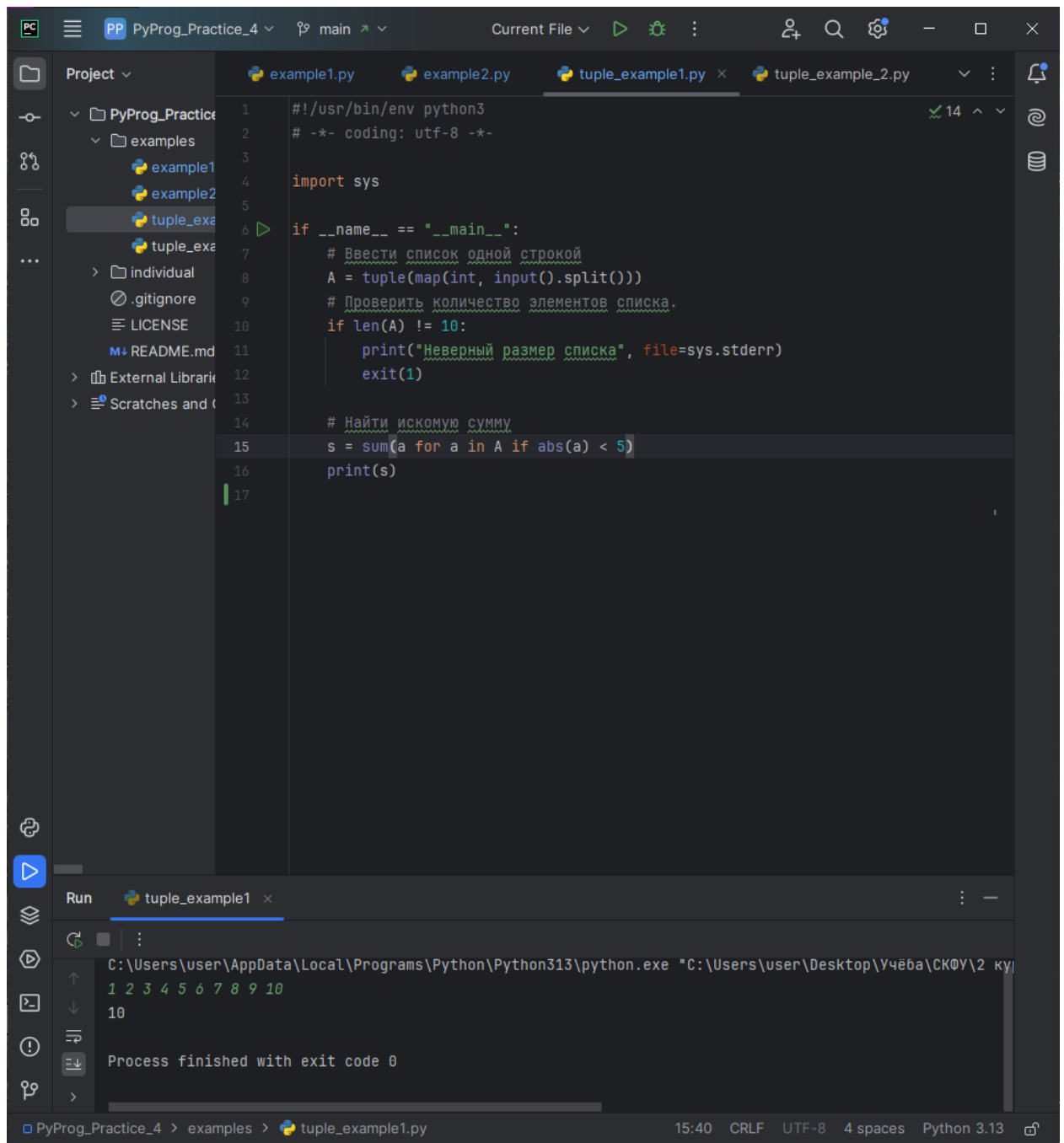


Рисунок 5. Результат выполнения примера 1 для кортежей для лабораторной работы

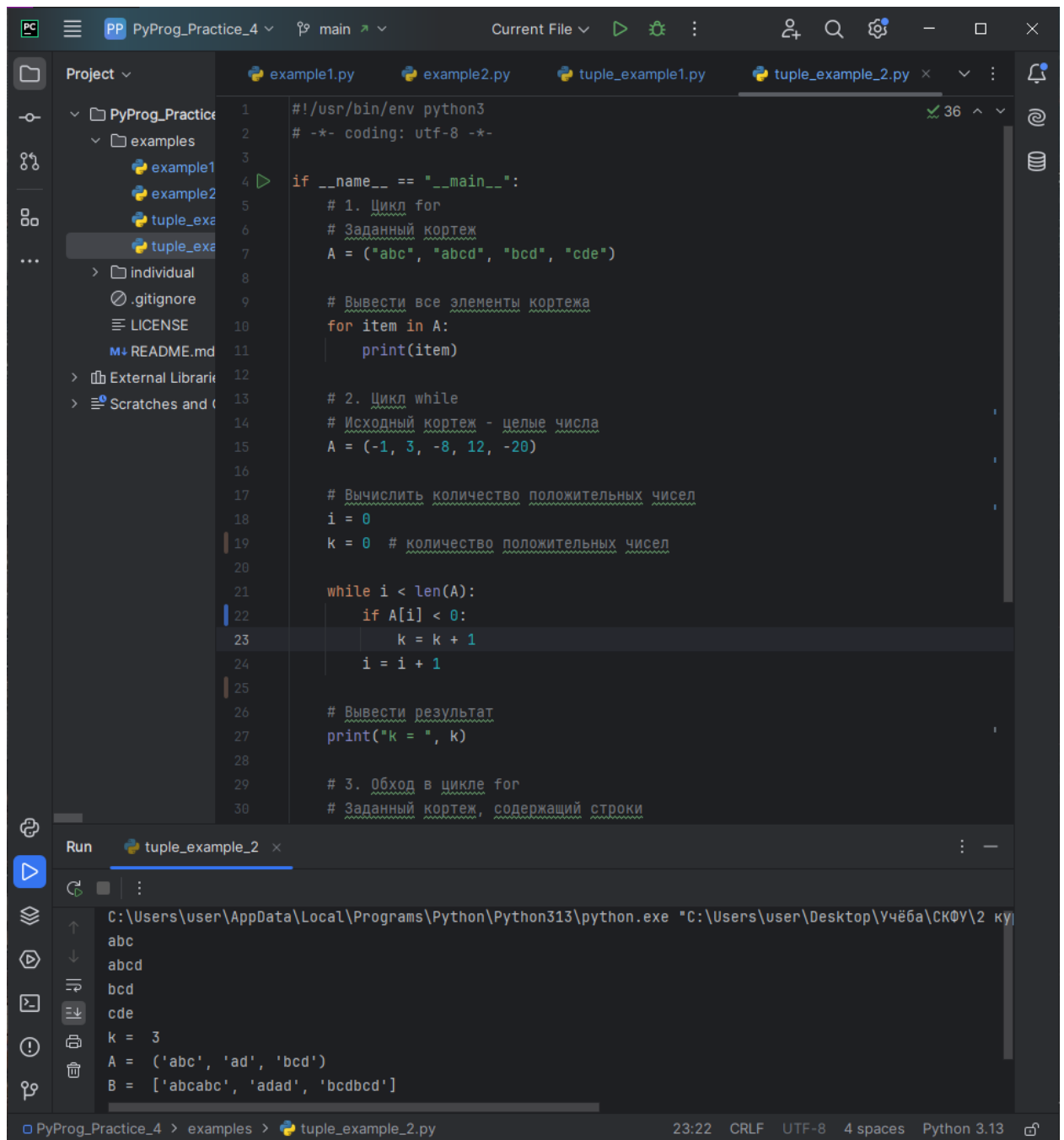


Рисунок 6. Результат выполнения примера 2 для кортежей для лабораторной работы

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

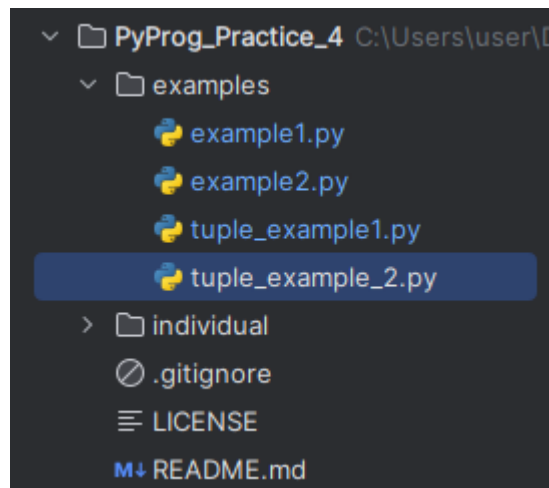


Рисунок 7. Структура проекта после выполнения примеров

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:

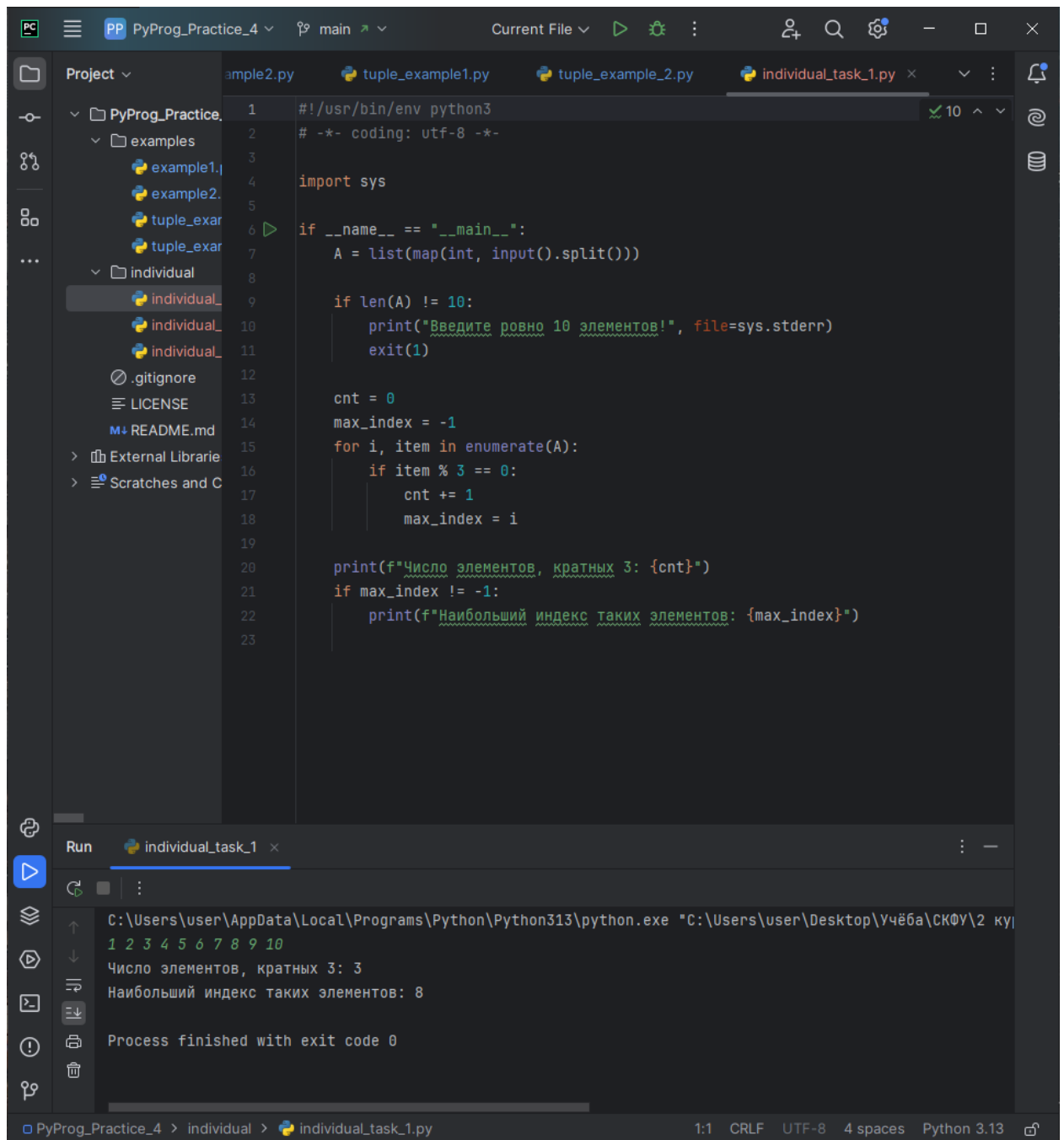


Рисунок 9. Результат работы программы индивидуального задания 1

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:

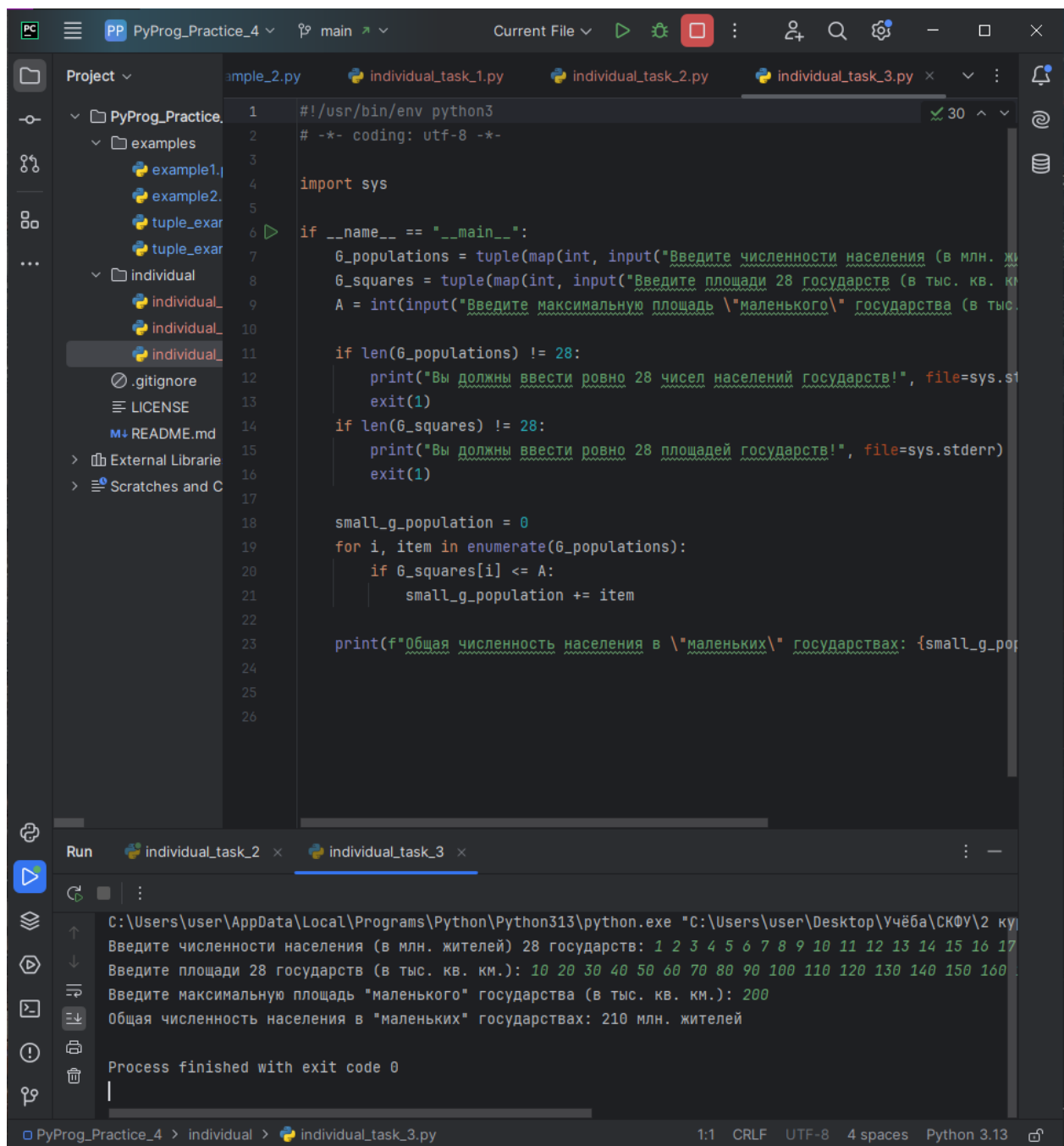


Рисунок 11. Результат работы программы индивидуального задания 4

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

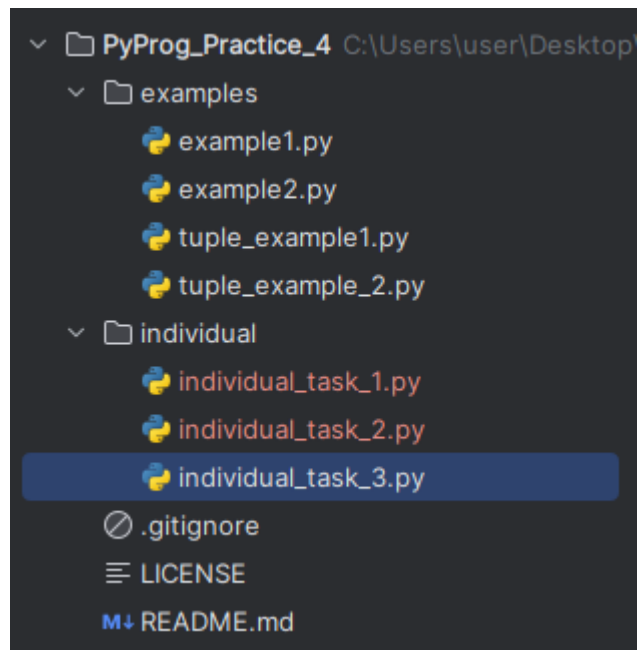


Рисунок 12. Структура проекта после выполнения индивидуальных заданий

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:

```
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 "docs: add lab work report"
[main c4cb9c9] docs: add lab work report
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 "doc/\320\237\320\276\320\273\321\217\320\272\320\276\320\262_\320\233\320\260\320\261_4.pdf"
C:\Users\user\Desktop\Учёба\СКФУ\2 курс\1 семестр\Python\Лабораторная 4\PyProg_Practice_4>
```

Рисунок 14. Фиксация добавления отчёта в локальный репозиторий

```
C:\Users\user\Desktop\Учёба\СКФУ\2 курс\1 семестр\Python\Лабораторная 4\PyProg_Practice_4>git push -u origin main
Enumerating objects: 27, done.
Counting objects: 100% (27/27), done.
Delta compression using up to 12 threads
Compressing objects: 100% (25/25), done.
Writing objects: 100% (25/25), 1.03 MiB | 673.00 KiB/s, done.
Total 25 (delta 10), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (10/10), completed with 1 local object.
To https://github.com/Kovirum/PyProg_Practice_4.git
5bfb0aa..c4cb9c9 main -> main
branch 'main' set up to track 'origin/main'.
C:\Users\user\Desktop\Учёба\СКФУ\2 курс\1 семестр\Python\Лабораторная 4\PyProg_Practice_4>
```

Рисунок 15. Результат отправки изменений на сервер GitHub

В результате изменения были успешно отправлены на сервер и на главной странице репозитория теперь отражены все изменения:

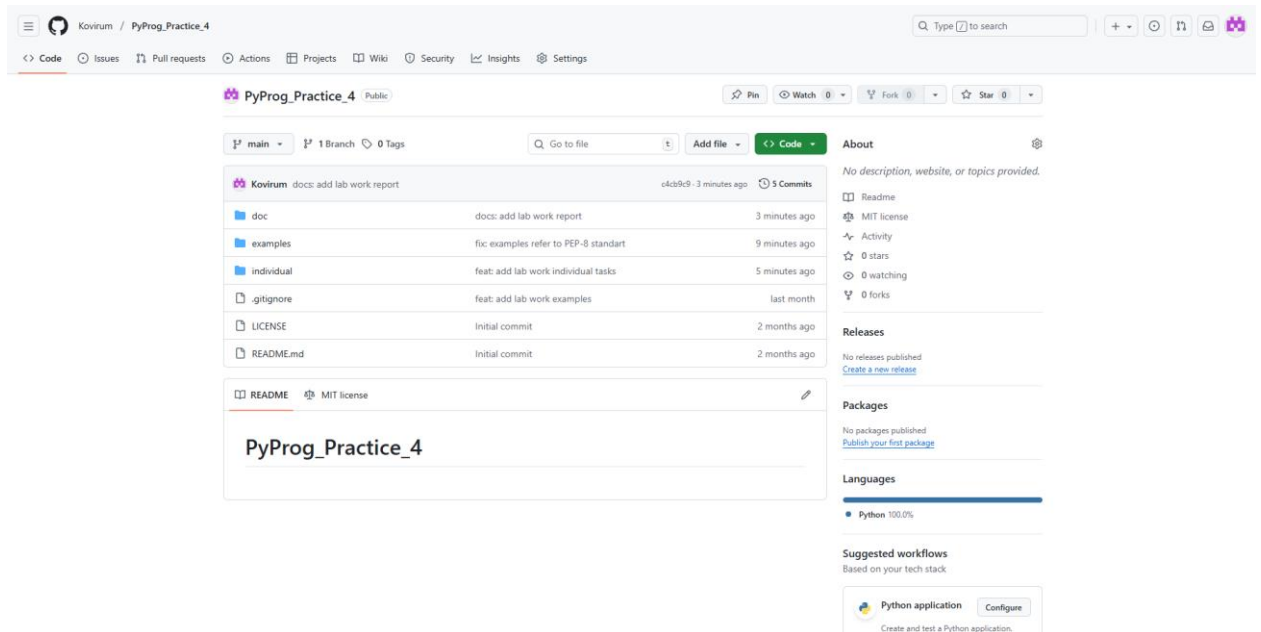


Рисунок 16. Страница проекта на GitHub после отправки изменений

Контрольные вопросы:

1. Что такое списки в языке Python?

Список (list) — это изменяемая (mutable) упорядоченная коллекция элементов произвольных типов.

2. Как осуществляется создание списка в Python?

Создать список можно с помощью квадратных скобок [] или функции list():

my_list = [1, 2, 3] или my_list = list("abc").

3. Как организовано хранение списков в оперативной памяти?

Список хранится как динамический массив ссылок на объекты, что позволяет быстро обращаться к элементам по индексу и изменять размер.

4. Каким образом можно перебрать все элементы списка?

С помощью цикла for:

for item in my_list:

print(item)

Или через индексы с range().

5. Какие существуют арифметические операции со списками?

Конкатенация (+) и повторение (*):

$[1, 2] + [3, 4] \rightarrow [1, 2, 3, 4]$,

$[1, 2] * 3 \rightarrow [1, 2, 1, 2, 1, 2]$.

6. Как проверить есть ли элемент в списке?

С помощью оператора in:

if element in my_list:

print("Найден")

7. Как определить число вхождений заданного элемента в списке?

Методом count():

my_list.count(5) вернёт количество элементов 5.

8. Как осуществляется добавление (вставка) элемента в списке?

- append(x) — добавить в конец.
- insert(i, x) — вставить на позицию i.
- extend(iterable) — добавить несколько элементов.

9. Как выполнить сортировку списка?

Метод sort() сортирует список на месте:

my_list.sort()

Или sorted() для получения нового отсортированного списка.

10. Как удалить один или несколько элементов из списка?

- remove(x) — удалить первый элемент равный x.
- pop(i) — удалить элемент по индексу i.
- del my_list[i] или del my_list[i:j].
- clear() — очистить весь список.

11. Что такое списковое включение и как с его помощью осуществлять обработку списков?

Списковое включение (list comprehension) — компактный способ создания списка:

$[x**2 \text{ for } x \text{ in range}(10) \text{ if } x \% 2 == 0]$

Эквивалентно циклу с условием.

12. Как осуществляется доступ к элементам списков с помощью срезов?

Синтаксис `list[start:stop:step]`:

`my_list[1:5]` — элементы с индексами 1 до 4,

`my_list[::-1]` — разворот списка.

13. Какие существуют функции агрегации для работы со списками?

`len()`, `sum()`, `min()`, `max()`, а также `all()`, `any()`.

14. Как создать копию списка?

– Поверхностная копия: `list.copy()` или `my_list[:]`.

– Глубокая копия: `import copy; copy.deepcopy(my_list)`.

15. Функция `sorted()` vs метод `sort()`

`sorted()` возвращает новый отсортированный список, не изменяя исходный.

`list.sort()` сортирует список на месте и возвращает `None`.

`sorted()` работает с любыми итерируемыми объектами.

16. Что такое кортежи в языке Python?

Кортеж (tuple) — это неизменяемая (immutable) упорядоченная коллекция элементов произвольных типов.

17. Каково назначение кортежей в языке Python?

Используются для хранения фиксированных данных, в качестве ключей словарей, для возврата нескольких значений из функции.

18. Как осуществляется создание кортежей?

С помощью круглых скобок `()` или функции `tuple()`:

`my_tuple = (1, 2, 3)` или `my_tuple = tuple([1, 2, 3])`.

Кортеж из одного элемента требует запятой: `(5,)`.

19. Как осуществляется доступ к элементам кортежа?

Так же, как к элементам списка — по индексу:

`my_tuple[0]` или через срезы `my_tuple[1:3]`.

20. Зачем нужна распаковка (деструктуризация) кортежа?

Распаковка позволяет присвоить элементы кортежа отдельным переменным в одной строке:

```
a, b, c = (1, 2, 3) # a=1, b=2, c=3
```

Удобно для возврата нескольких значений из функций.

21. Какую роль играют кортежи в множественном присваивании?

Кортежи позволяют менять значения переменных без временной переменной:

```
a, b = b, a # обмен значениями
```

Используется неявное создание кортежей.

22. Как выбрать элементы кортежа с помощью среза?

Синтаксис как у списков: `tuple[start:stop:step]`

Пример: `my_tuple[1:4]` — элементы с индексами 1, 2, 3.

23. Как выполняется конкатенация и повторение кортежей?

– Конкатенация: $(1, 2) + (3, 4) \rightarrow (1, 2, 3, 4)$

– Повторение: $(1, 2) * 3 \rightarrow (1, 2, 1, 2, 1, 2)$

24. Как выполняется обход элементов кортежа?

Как и для списка — через цикл `for`:

```
for item in my_tuple:
```

```
    print(item)
```

25. Как проверить принадлежность элемента кортежу?

С помощью оператора `in`:

```
if 5 in my_tuple:
```

```
    print("Найден")
```

26. Какие методы работы с кортежами Вам известны?

Только два метода:

– `count(x)` — количество вхождений элемента `x`

– `index(x)` — индекс первого вхождения элемента `x`

27. Допустимо ли использование функций агрегации таких как `len()`, `sum()` и т. д. при работе с кортежами?

Да, все стандартные функции агрегации работают с кортежами:

```
len(my_tuple),    sum(my_tuple),    min(my_tuple),    max(my_tuple),  
all(my_tuple), any(my_tuple).
```

28. Как создать кортеж с помощью спискового включения.

Используется генератор кортежа (tuple comprehension), но фактически это генераторное выражение:

```
my_tuple = tuple(x**2 for x in range(5)) # (0, 1, 4, 9, 16)
```

Или с условием:

```
my_tuple = tuple(x for x in range(10) if x % 2 == 0) # (0, 2, 4, 6, 8)
```

Вывод:

В результате выполнения данной лабораторной работы были приобретены навыки по работе со списками и кортежами при написании программ с помощью языка программирования Python версии 3.x.