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»

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

_____
(подпись)

Проверил:
Доцент департамента цифровых, робототехнических систем и электроники института перспективной инженерии Воронкин Р.А

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2025 г.

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

Ссылка на репозиторий: https://github.com/progwar/lab4

Ход работы:

1. Был создан общедоступный репозиторий с лицензией MIT и языком программирования Python.

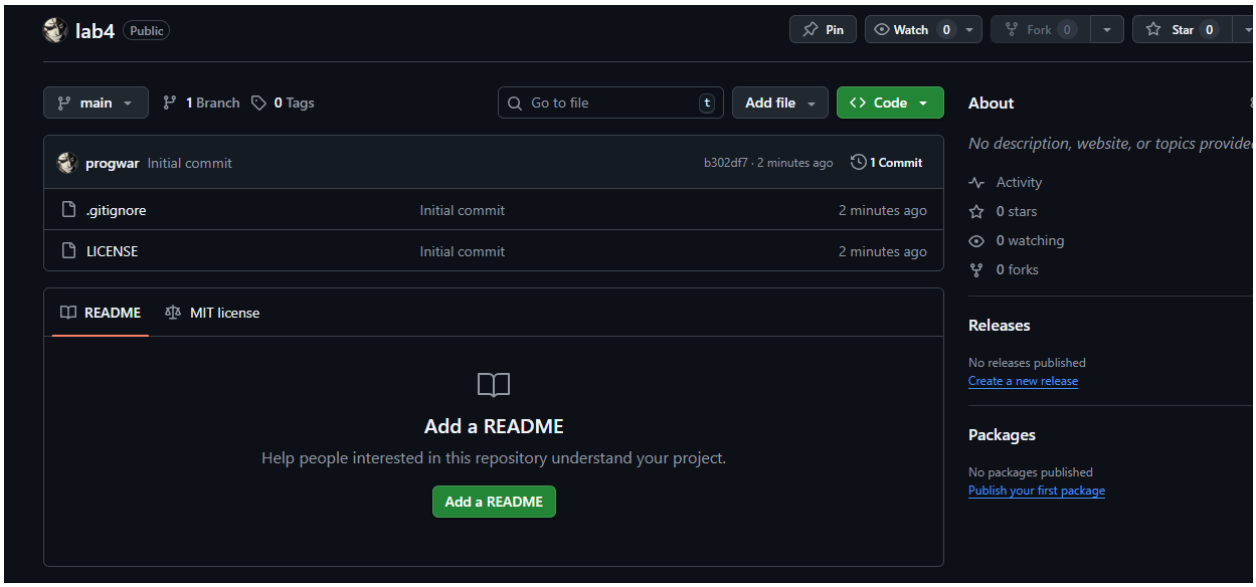


Рисунок 1. Созданный репозиторий

2. Было выполнено клонирование репозитория.

```
PS C:\WINDOWS\system32> cd "D:\github\lab4"
PS D:\github\lab4> git clone "https://github.com/progwar/lab4.git"
Cloning into 'lab4'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (4/4), done.
```

Рисунок 2. Клонирование репозитория

3. Файл .gitignore был дополнен необходимыми правилами.

```
# PyCharm
# User-specific stuff
.idea/**/workspace.xml
.idea/**/tasks.xml
.idea/**/usage.statistics.xml
.idea/**/dictionaries
.idea/**/shelf
#.idea/
```

Рисунок 3. Изменённый .gitignore

4. Репозиторий организован в соответствии с моделью ветвления git-flow.

```
PS D:\github\lab4> git flow init

Which branch should be used for bringing forth production releases?
   - main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [D:/github/lab4/.git/hooks]
```

Рисунок 4. Модель ветвления git-flow

5. Были проработаны примеры лабораторной работы.

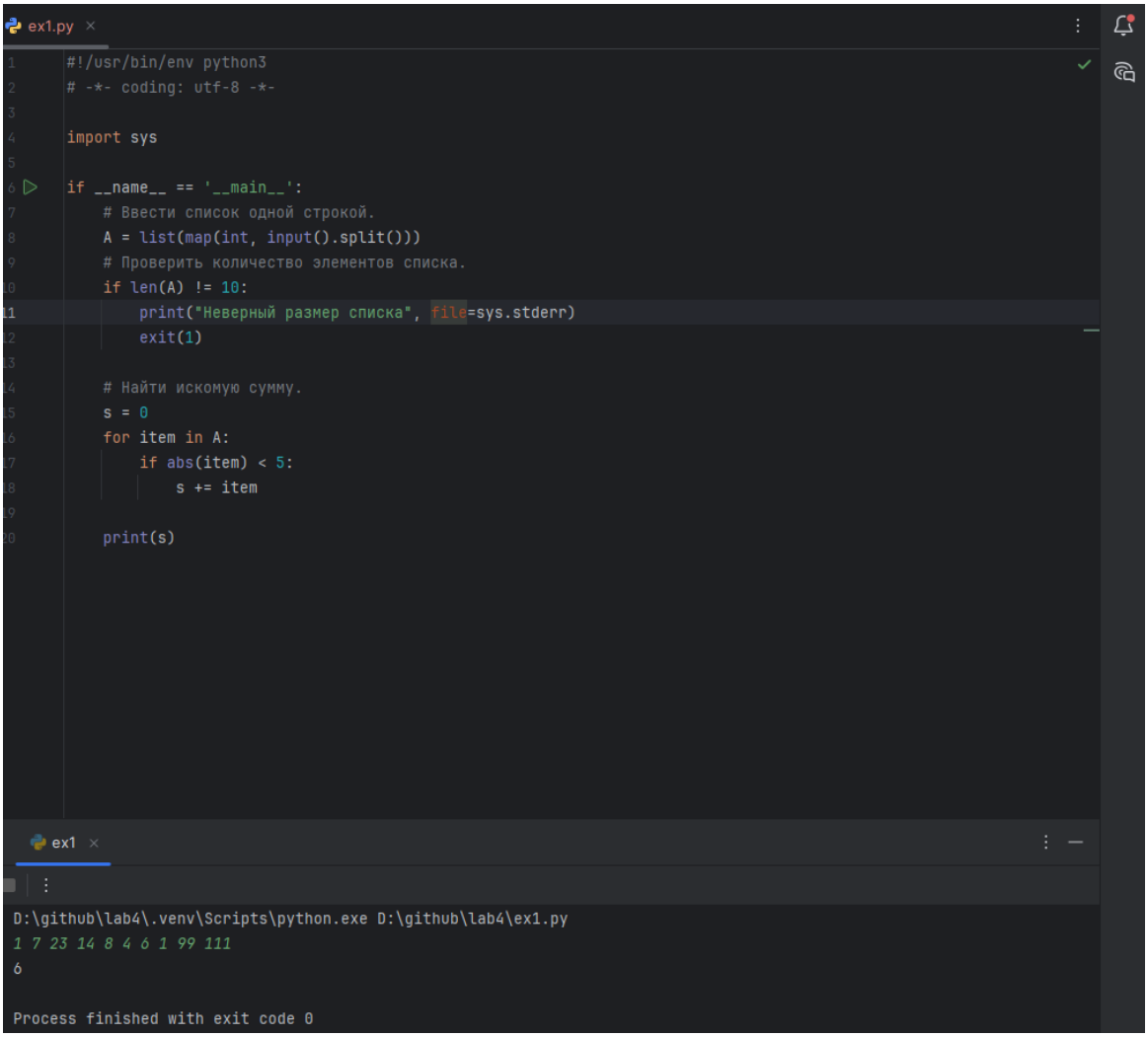


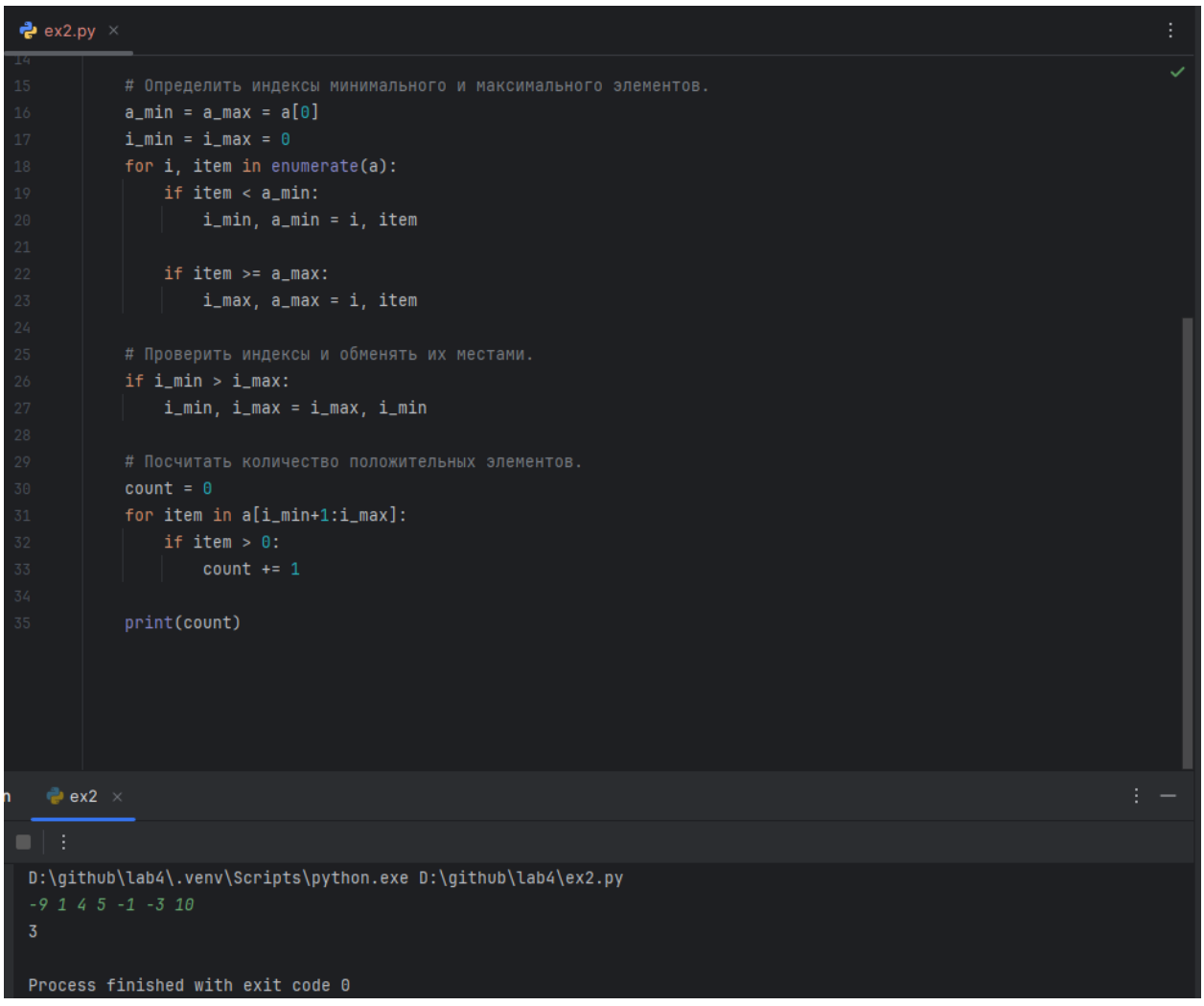
Рисунок 5. Пример 1

```python
#!/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)
```

Рисунок 6. Пример 2



```python
    # Определить индексы минимального 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)
```

```
D:\github\lab4\.venv\Scripts\python.exe D:\github\lab4\ex2.py
-9 1 4 5 -1 -3 10
3

Process finished with exit code 0
```

Рисунок 7. Пример 2 (продолжение)

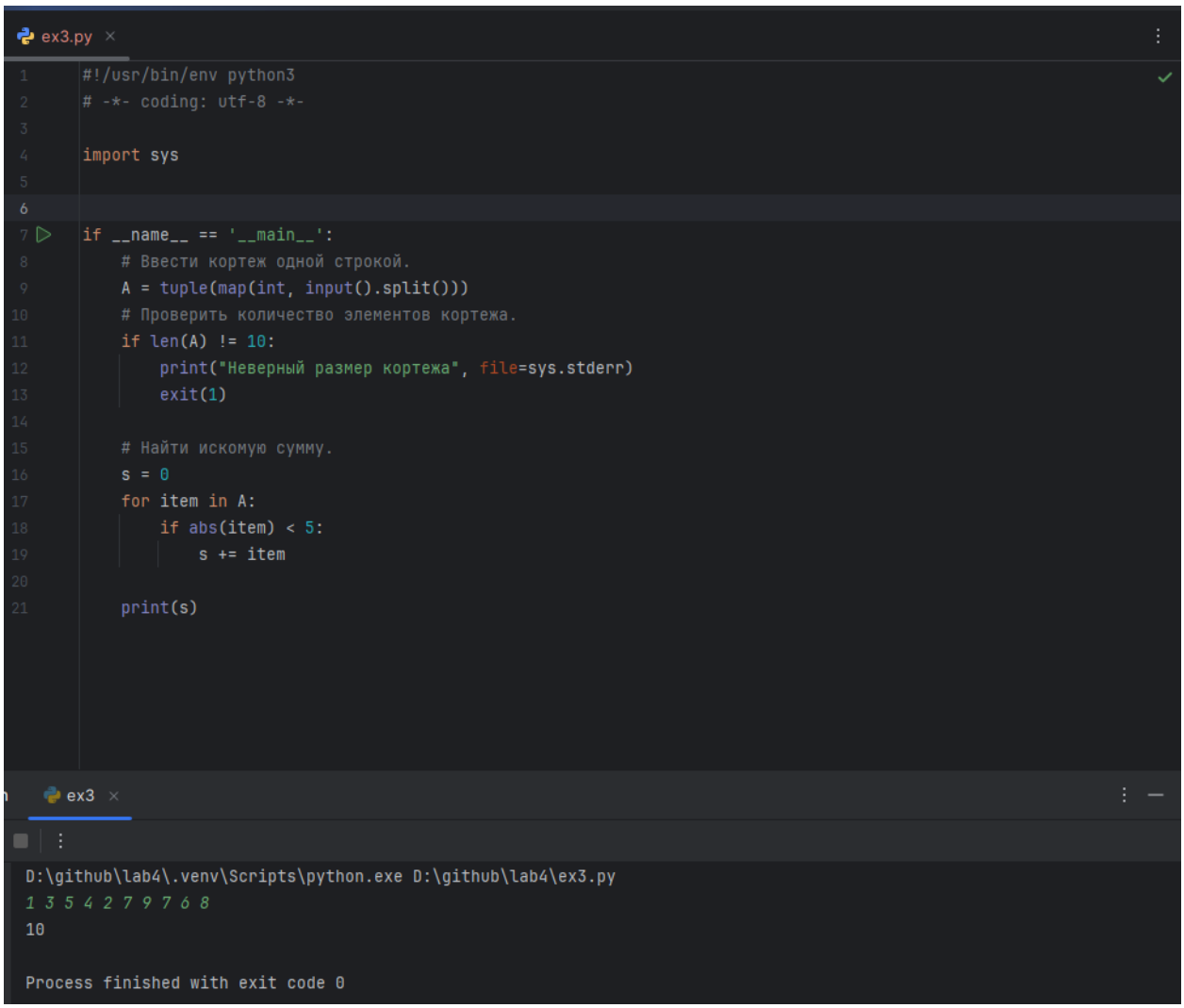


Рисунок 8. Пример 3

Индивидуальное задание 1:

Ввести список A из 10 элементов, найти разность положительных элементов, и вывести ее на экран.

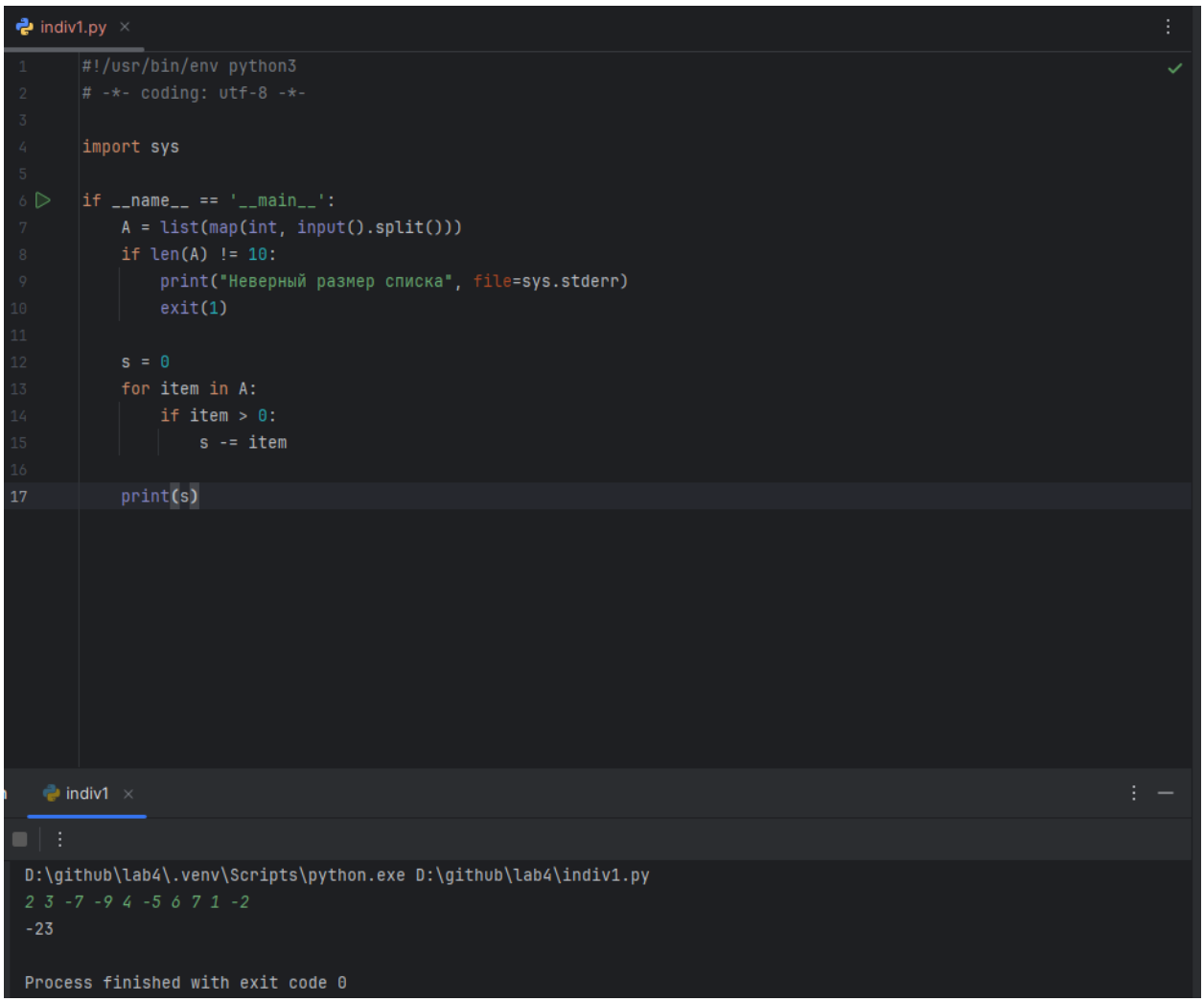


Рисунок 9. Индивидуальное задание 1

Индивидуальное задание 2:

В списке, состоящем из целых элементов, вычислить:

1) номер максимального элемента списка;

2) произведение элементов списка, расположенных между первым и вторым нулевыми элементами.

Преобразовать список таким образом, чтобы в первой его половине располагались элементы, стоявшие в нечетных позициях, а во второй половине элементы, стоявшие в четных позициях.

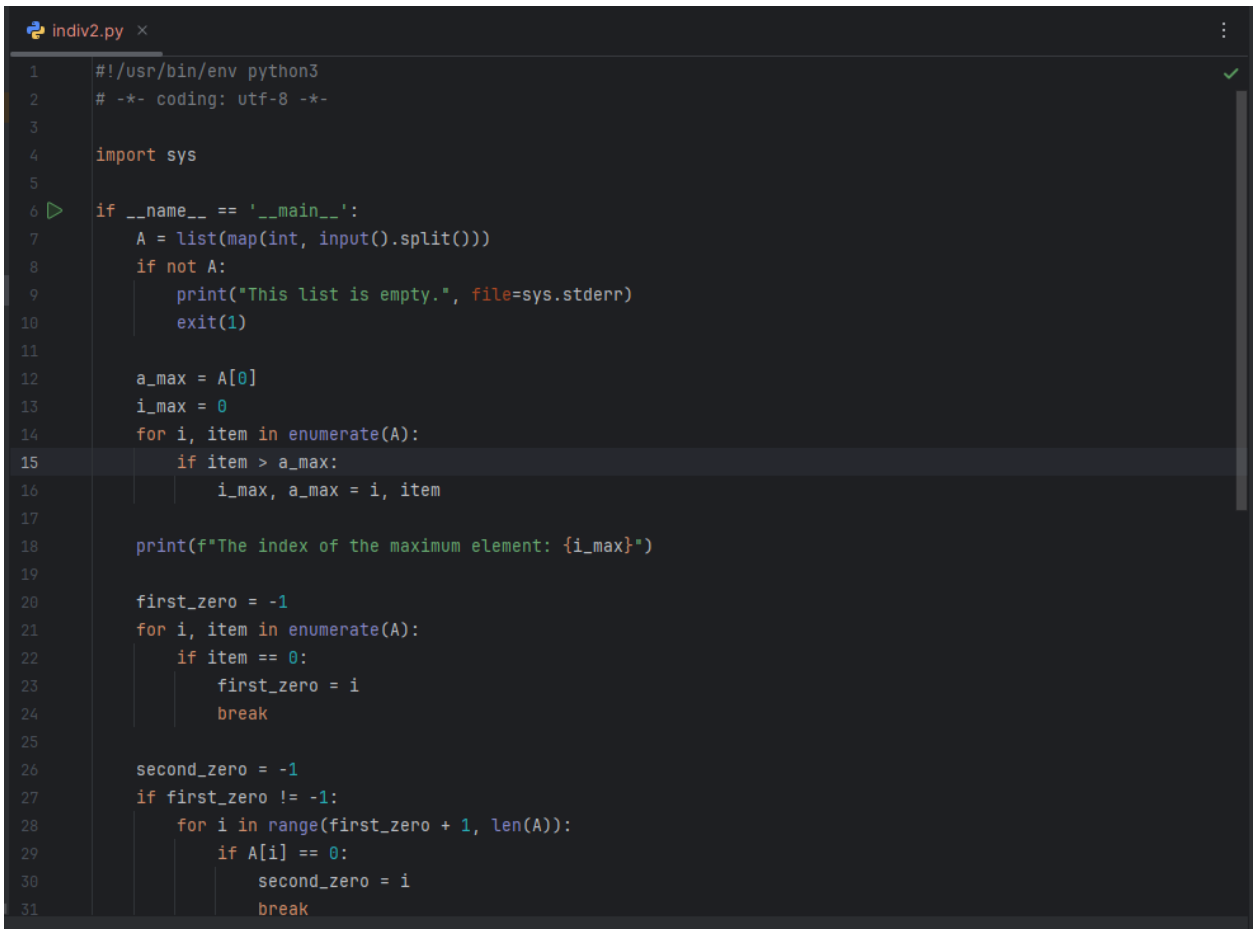


Рисунок 10. Индивидуальное задание 2

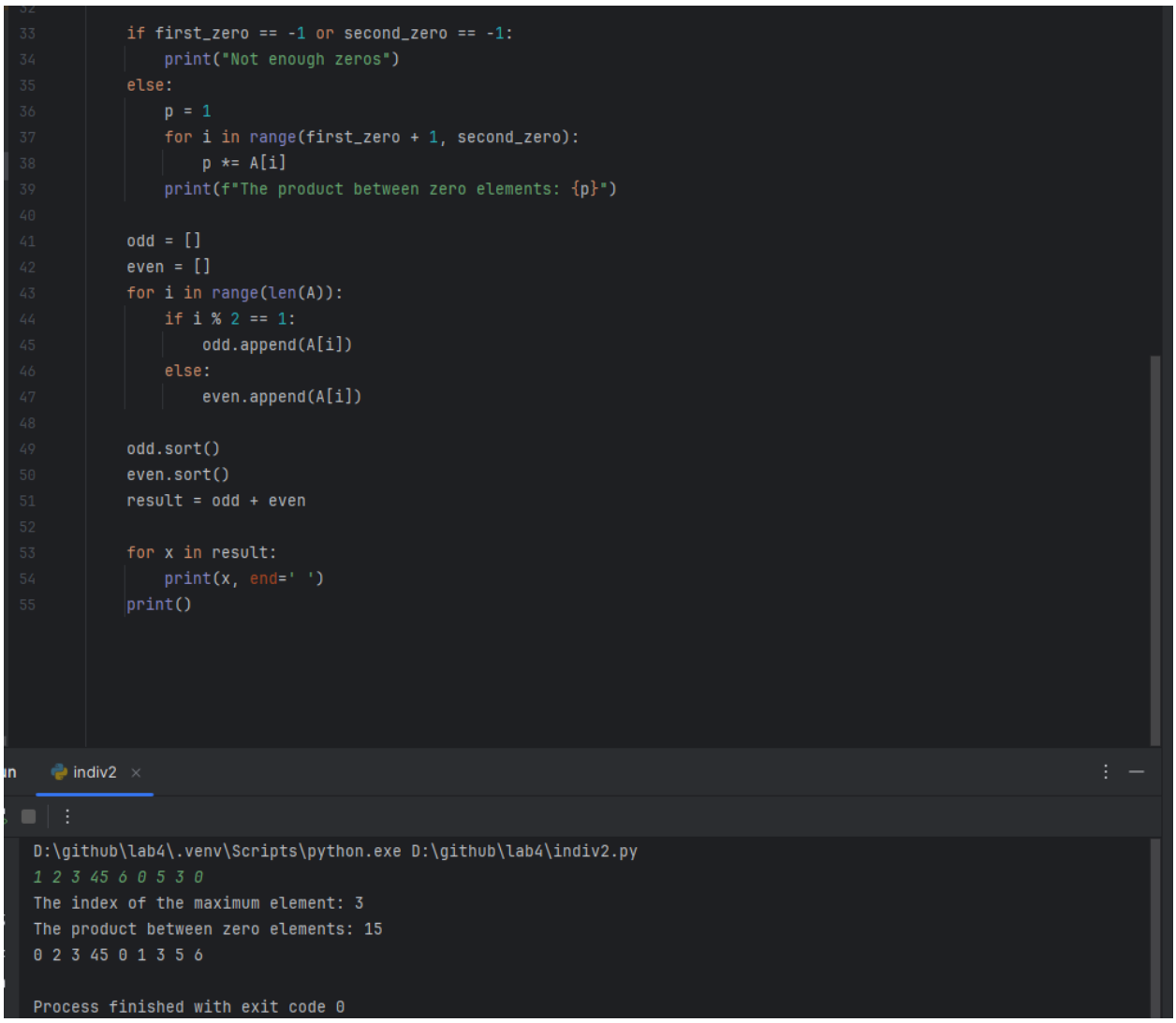


Рисунок 11. Индивидуальное задание 2 (продолжение)

Индивидуальное задание 3:

Дан кортеж целых чисел. Определить, есть ли в нем хотя бы одна пара соседних нечетных чисел. В случае положительного ответа определить номера элементов первой из таких пар.

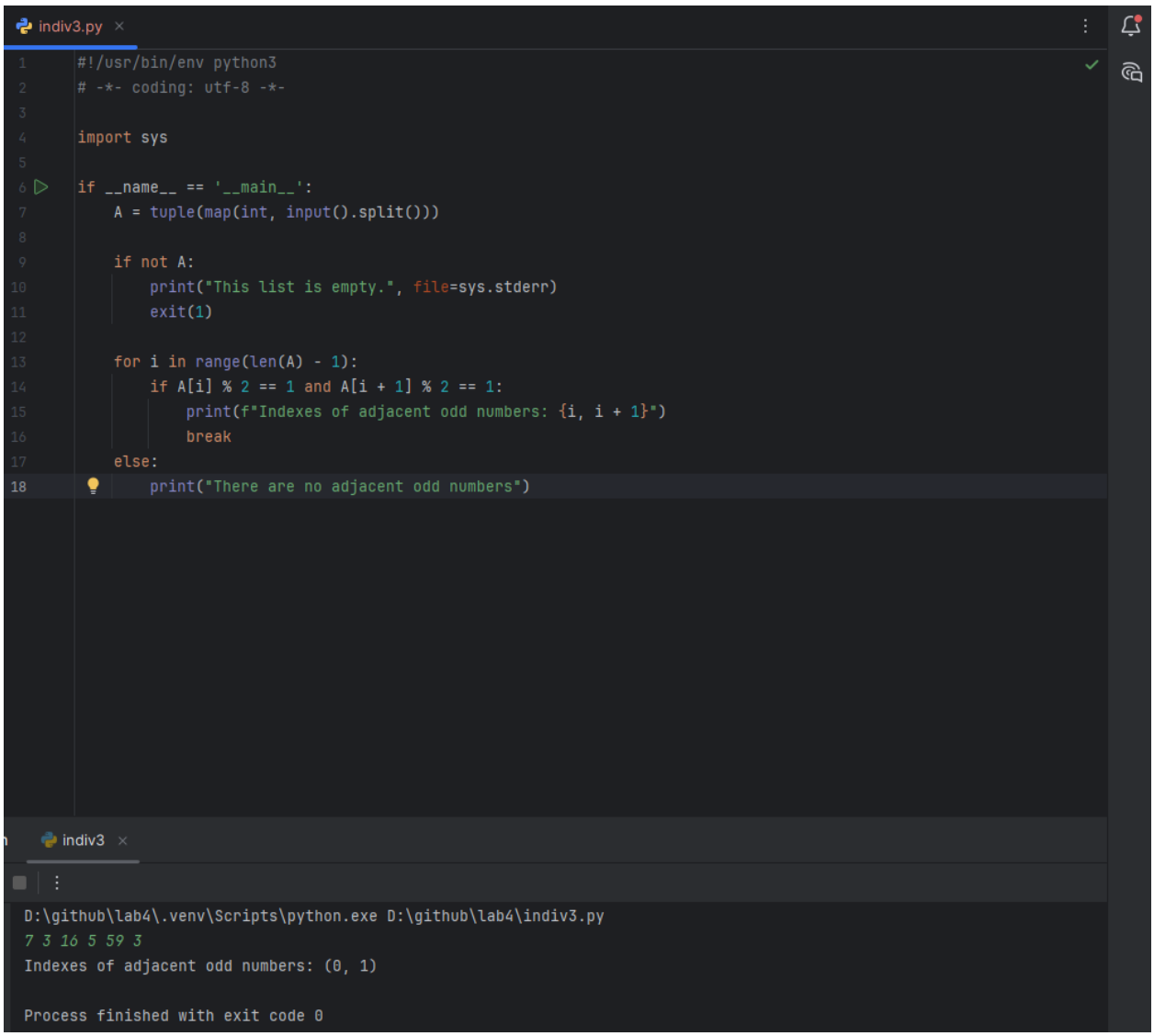


Рисунок 12. Индивидуальное задание 3

Контрольные вопросы:

1. Что такое списки в языке Python?

Список (list) – это структура данных для хранения объектов различных типов. Размер списков не статичен, его можно изменять. Список по своей природе является изменяемым типом данных. Переменная, определяемая как список, содержит ссылку на структуру в памяти, которая в свою очередь хранит ссылки на какие-либо другие объекты или структуры.

2. Как осуществляется создание списка в Python?

Для создания списка нужно элементы заключить в квадратные скобки: my_list = [1, 2, 3, 4, 5]

3. Как организовано хранение списков в оперативной памяти?

При создании списка в памяти резервируется область, которую можно условно назвать некоторым "контейнером", в котором хранятся ссылки на другие элементы данных в памяти. В отличии от таких типов данных как число или строка, содержимое "контейнера" списка можно менять.

4. Каким образом можно перебрать все элементы списка?

Перебрать все элементы списка можно с помощью следующего цикла:

```
my_list = ['Один', 'Два', 'Три', 'Четыре', 'Пять']
for elem in my_list:
    print(elem)
```

5. Какие существуют арифметические операции со списками?

Арифметические операции со списками: + (объединение списков), * (повтор списка).

6. Как проверить есть ли элемент в списке?

Для того, чтобы проверить, есть ли заданный элемент в списке Python необходимо использовать оператор in.

7. Как определить число вхождений заданного элемента в списке?

Метод count можно использовать для определения числа вхождений заданного элемента в списке.

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

Метод insert можно использовать для вставки элемента в список. Метод append можно использовать для добавления элемента в список.

9. Как выполнить сортировку списка?

Для сортировки списка нужно использовать метод sort.

10. Как удалить один или несколько элементов из списка?

Удалить элемент можно, написав его индекс в методе pop.

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

List Comprehensions (абстракция списков или списковое включение) является частью синтаксиса языка, которая предоставляет простой способ построения списков. В языке Python есть 2 очень мощные функции для работы

с коллекциями: map и filter. Они позволяют использовать функциональный стиль программирования, не прибегая к помощи циклов, для работы с такими типами как list, tuple, set, dict и т.п. Списковое включение позволяет обойтись без этих функций.

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

Срезы задаются тройкой чисел, разделенных запятой: start, stop, step. Start – позиция, с которой нужно начать выборку, stop – конечная позиция, step – шаг. При этом необходимо помнить, что выборка не включает элемент определяемый stop.

13. Какие существуют функции агрегации для работы со списками?

Для работы со списками существуют следующие функции агрегации:

- len(L) – получить число элементов в списке
- min(L) – получить минимальный элемент списка
- max(L) – получить максимальный элемент списка
- sum(L) – получить сумму элементов списка, если список содержит только числовые значения.

14. Как создать копию списка?

Для создания копии списка необходимо использовать либо метод copy, либо оператор среза.

15. Самостоятельно изучите функцию sorted языка Python. В чем ее отличие от метода sort списков?

Функция sorted() в Python создаёт новый отсортированный список, не изменяя исходный. Это полезно, когда необходимо сохранить оригинальный порядок элементов. Отличие от метода sort() списков заключается в том, что метод sort() изменяет исходный список на месте и возвращает None. Этот подход эффективен по памяти, поскольку не создаёт новый список. Ещё одно отличие в том, что метод list.sort() определён только для списков, в то время как функция sorted() работает со всеми итерируемыми объектами (списки, кортежи, строки, множества, словари и т. д.).

16. Что такое кортежи в языке Python?

Кортеж (tuple) – это неизменяемая структура данных, которая по своему подобию очень похожа на список.

17. Каково назначение кортежей в языке Python?

Существует несколько причин, по которым стоит использовать кортежи вместо списков. Одна из них – это обезопасить данные от случайного изменения. Если мы получили откуда-то массив данных, и у нас есть желание поработать с ним, но при этом непосредственно менять данные мы не собираемся, тогда, это как раз тот случай, когда кортежи придутся как нельзя кстати. Используя их в данной задаче, мы дополнительно получаем сразу несколько бонусов - во-первых, это экономия места. Дело в том, что кортежи в памяти занимают меньший объем по сравнению со списками. Во-вторых – прирост производительности, который связан с тем, что кортежи работают быстрее, чем списки (т. е. на операции перебора элементов и т. п. будет тратиться меньше времени). Важно также отметить, что кортежи можно использовать в качестве ключа у словаря.

18. Как осуществляется создание кортежей?

Кортеж создается также как список, только вместо квадратных скобок используют круглые. При желании можно воспользоваться функцией tuple(). Для создания пустого кортежа, не нужно ничего писать в скобках.

19. Как осуществляется доступ к элементам кортежа?

Доступ к элементам кортежа осуществляется также как и к элементам списка – через указание индекса. Но изменять элементы кортежа нельзя.

20. Зачем нужна распадовая (деструктуризация) кортежа?

Обращение по индексу, это не самый удобный способ работы с кортежами. Дело в том, что кортежи часто содержат значения разных типов, и помнить, по какому индексу что лежит – очень непросто. Для этого существует деструктуризация.

21. Какую роль играют кортежи в множественном присваивании?

Благодаря тому, что кортежи легко собирать и разбирать, в Python удобно делать такие вещи, как множественное присваивание: (a, b, c) = (1, 2, 3)

22. Как выбрать элементы кортежа с помощью среза?

С помощью операции взятия среза можно получить другой кортеж: T2 = T1[i:j].

23. Как выполняется конкатенация и повторение кортежей?

Для кортежей можно выполнять операцию конкатенации, которая обозначается символом +. В простейшем случае общая форма операции следующая: T3 = T1 + T2. Кортеж может быть образован путем операции повторения, обозначаемой символом *. Общая форма: T2 = T1 * n (где n – количество повторений кортежа T1).

24. Как выполняется обход элементов кортежа?

Элементы кортежа можно последовательно просмотреть с помощью операторов цикла while или for, например:

```
A = ("abc", "abcd", "bcd", "cde")
for item in A:
    print(item)
```

25. Как проверить принадлежность элемента кортежу?

Для проверки вхождения элемента в кортеж используется операция in:

```
A = ("abc", "abcd", "bcd", "cde")
item = str(input("s = "))
if (item in A):
    print(item, " in ", A, " = True")
```

26. Какие методы работы с кортежами Вам известны?

Методы работы с кортежем:

- index() – поиск позиции элемента в кортеже
- count() – количество вхождения элемента в кортеж

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

Да, использование функций агрегации (таких как len(), sum(), max(), min() и др.) с кортежами в Python полностью допустимо и корректно. Кортежи

поддерживают те же операции, что и другие итерируемые объекты (например, списки), так как они являются последовательностями (sequence type).

28. Как создать кортеж с помощью спискового включения.

Создание кортежа с помощью спискового включения: a = tuple(map(int, input().split())).