

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

Институт цифрового развития
Кафедра инфокоммуникаций

«Работа со списками в языке Python»

ОТЧЕТ
по лабораторной работе №7
дисциплины
«Основы программной инженерии»

Выполнила:

Кувшин Ирина Анатольевна

2 курс, группа ПИЖ-б-о-21-1,

09.03.04 «Программная инженерия»,

направленность (профиль) «Разработка

и сопровождение программного

обеспечения», очная форма обучения

(подпись)

Проверил:

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

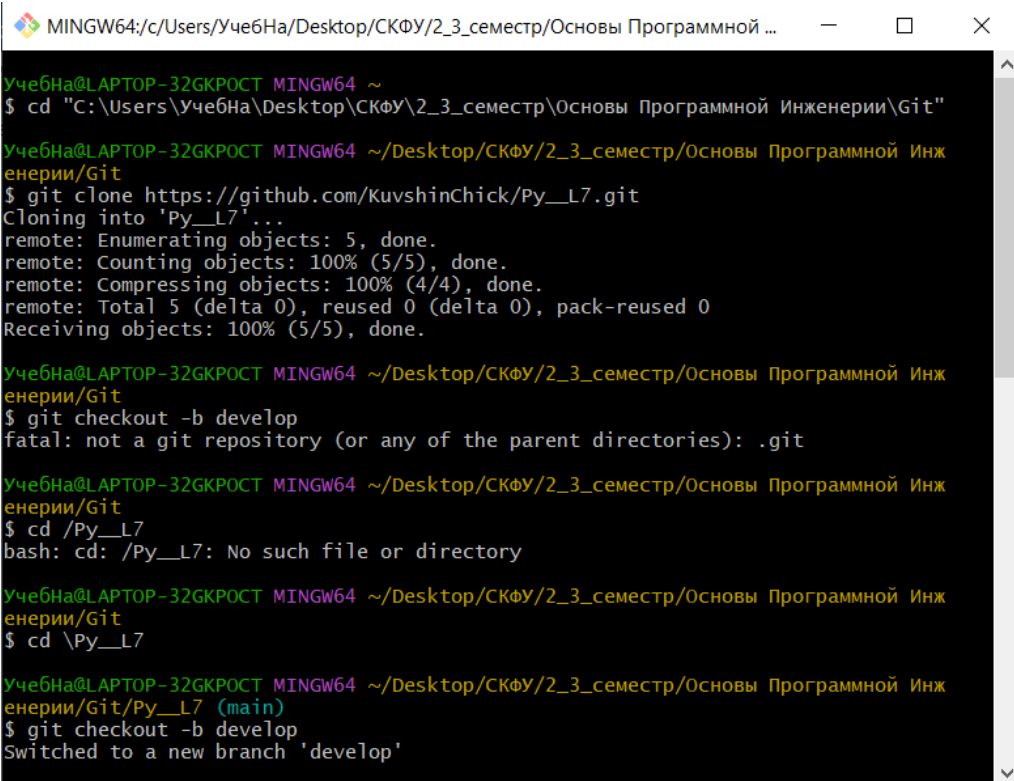
Ставрополь, 2022 г.

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

Ссылка на репозиторий: https://github.com/KuvshinChick/Py__L7.git

Ход работы:

1. Изучить теоретический материал работы.
2. Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT и язык программирования Python.
3. Выполните клонирование созданного репозитория.
4. Дополните файл .gitignore необходимыми правилами для работы с IDE PyCharm.
5. Организуйте свой репозиторий в соответствии с моделью ветвления git-flow.



```
MINGW64/c:/Users/Учебна/Desktop/СКФУ/2_3_семестр/Основы Программной ...
Учебна@LAPTOP-32GKP0CT MINGW64 ~
$ cd "C:\Users\Учебна\Desktop\СКФУ\2_3_семестр\Основы Программной Инженерии\Git"

Учебна@LAPTOP-32GKP0CT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Git
$ git clone https://github.com/KuvshinChick/Py__L7.git
Cloning into 'Py__L7'...
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
Receiving objects: 100% (5/5), done.

Учебна@LAPTOP-32GKP0CT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Git
$ git checkout -b develop
fatal: not a git repository (or any of the parent directories): .git

Учебна@LAPTOP-32GKP0CT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Git
$ cd /Py__L7
bash: cd: /Py__L7: No such file or directory

Учебна@LAPTOP-32GKP0CT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Git
$ cd \Py__L7

Учебна@LAPTOP-32GKP0CT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инженерии/Git/Py__L7 (main)
$ git checkout -b develop
Switched to a new branch 'develop'
```

Рисунок 6.1 – Клонирование репозитория и создание ветки develop

```

Учебна@LAPTOP-32GKP0CT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инж
нерии/Git/Py__L7 (develop)
$ git status
On branch develop
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   .gitignore
        modified:   README.md

no changes added to commit (use "git add" and/or "git commit -a")

Учебна@LAPTOP-32GKP0CT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инж
нерии/Git/Py__L7 (develop)
$ git add .

Учебна@LAPTOP-32GKP0CT MINGW64 ~/Desktop/СКФУ/2_3_семестр/Основы Программной Инж
нерии/Git/Py__L7 (develop)
$ git commit -m "modified .gitignore & readme"
[develop a13a71e] modified .gitignore & readme
2 files changed, 5 insertions(+), 3 deletions(-)

```

Рисунок 6.2 – Обновление .gitignore и readme

6. Создайте проект PyCharm в папке репозитория.
7. Проработайте примеры лабораторной работы. Создайте для каждого примера отдельный модуль языка Python. Зафиксируйте изменения в репозитории.
8. Приведите в отчете скриншоты результатов выполнения каждой из программ примеров при различных исходных данных вводимых с клавиатуры.

Пример 1.

Ввести список A из 10 элементов, найти сумму элементов, меньших по модулю 5, и вывести ее на экран.

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6
7  if __name__ == '__main__':
8      # Ввести список одной строкой.
9      A = list(map(int, input().split()))
10     # Проверить количество элементов списка.
11     if len(A) != 10:
12         print("Неверный размер списка", file=sys.stderr)
13         exit(1)
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

```

Рисунок 6.3 – Код программы-примера

```

"C:\Program Files\Python310\python.exe"
"C:\Users\УчебНа\Desktop\СКФУ\2_3_семестр\Основы
Инженерии\Git\Py__L7\PyCharm\ex_1.py"
1 5 6 12 7 10 23 4 7 0
5
Process finished with exit code 0

```

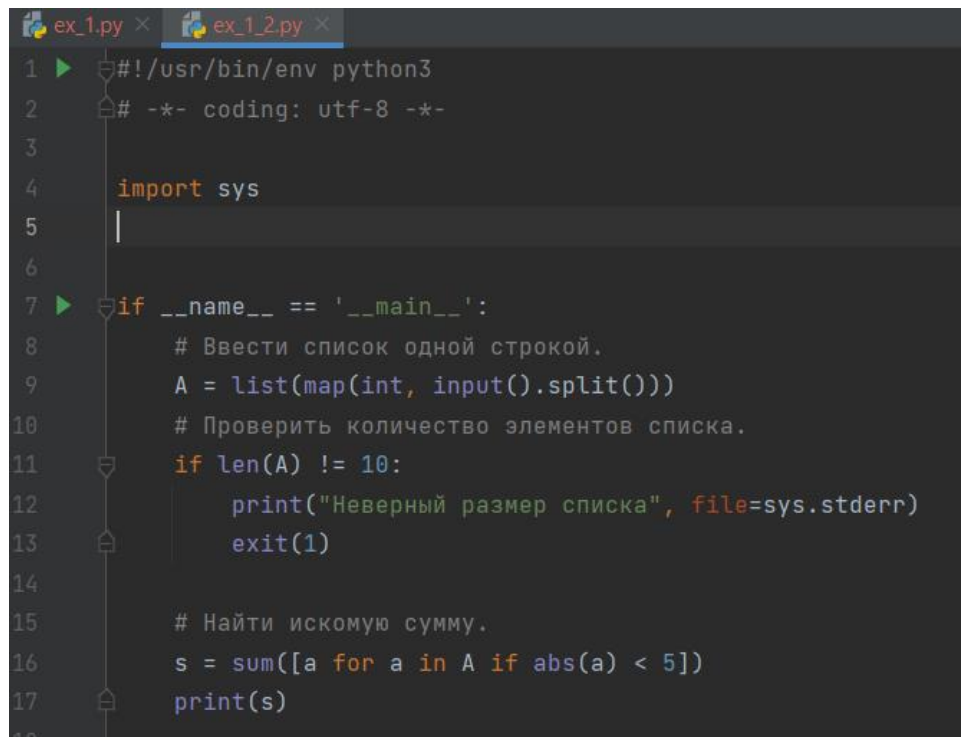
Рисунок 6.4 – Результат программы

```

"C:\Program Files\Python310\python.exe"
"C:\Users\УчебНа\Desktop\СКФУ\2_3_семестр\Осн
Инженерии\Git\Py__L7\PyCharm\ex_1.py"
1 2 5
Неверный размер списка
Process finished with exit code 1

```

Рисунок 6.5 – Результат программы (ошибка)

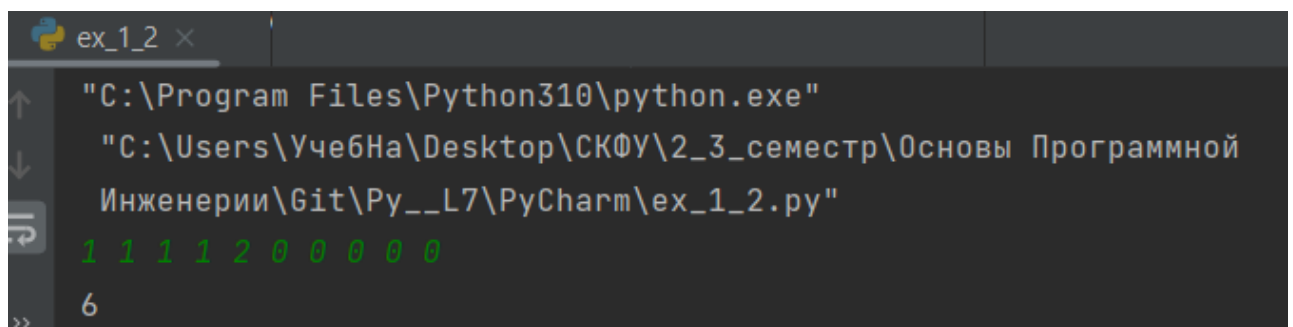


```

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

```

Рисунок 6.6 – Код программы-примера (с помощью списковых включений)



```

"C:\Program Files\Python310\python.exe"
"C:\Users\УчебHa\Desktop\СКФУ\2_3_семестр\Основы Программной
Инженерии\Git\Py__L7\PyCharm\ex_1_2.py"
1 1 1 1 2 0 0 0 0 0
6

```

Рисунок 6.7 – Результат программы

Пример 2.

Написать программу, которая для целочисленного списка определяет, сколько положительных элементов располагается между его максимальным и минимальным элементами.

```

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

```

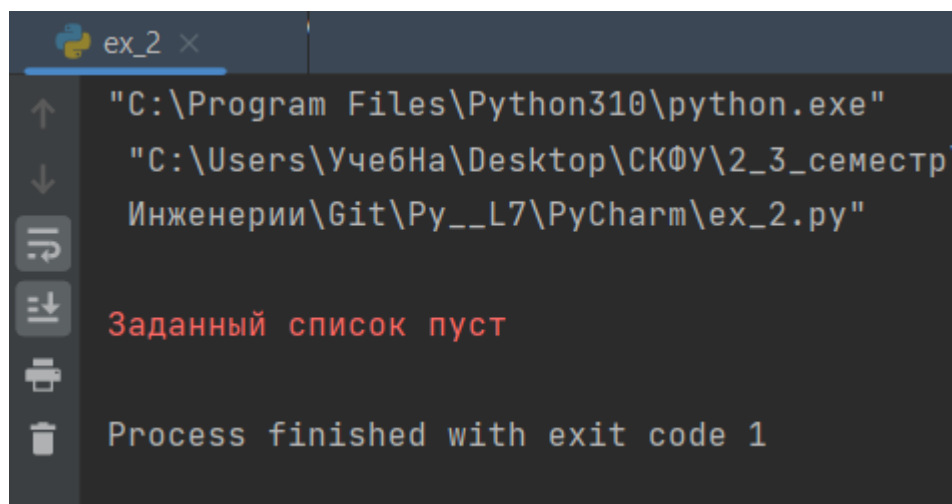
Рисунок 6.8 – Код программы-примера

```

ex_2 x
"C:\Program Files\Python310\python.exe"
"C:\Users\УчебНа\Desktop\СКФУ\2_3_семестр\Основы Программн
Инженерии\Git\Py_L7\PyCharm\ex_2.py"
0 5 1 2 8 1 3 4
3
Process finished with exit code 0

```

Рисунок 6.9 – Результат программы



```
ex_2 x
"C:\Program Files\Python310\python.exe"
"C:\Users\УчебHa\Desktop\СКФУ\2_3_семестр
Инженерии\Git\Py__L7\PyCharm\ex_2.py"
Заданный список пуст
Process finished with exit code 1
```

Рисунок 6.10 – Результат программы (ошибка)

9. Приведите в отчете скриншоты работы программ решения индивидуальных заданий.
10. Зафиксируйте сделанные изменения в репозитории.
11. Добавьте отчет по лабораторной работе в формате PDF в папку doc репозитория. Зафиксируйте изменения.
12. Выполните слияние ветки для разработки с веткой main / master.
13. Отправьте сделанные изменения на сервер GitHub.
14. Отправьте адрес репозитория GitHub на электронный адрес преподавателя.

Индивидуальные задания: (Вариант 15)

Задание 1.

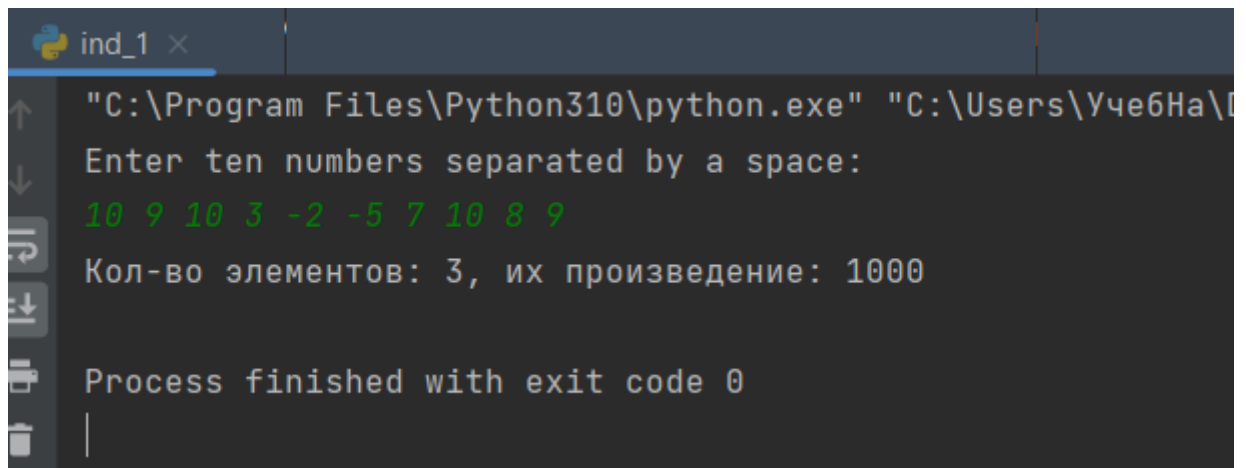
15. Ввести список A из 10 элементов, найти произведение элементов, больших 8 и меньших 18 и кратных 10, их количество и вывести результаты на экран.

```
ex_1.py x ex_1_2.py x ex_2.py x ind_1.py x
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import sys
5
6 if __name__ == '__main__':
7     # Ввести список одной строкой.
8     print("Enter ten numbers separated by a space: ")
9     num_list = list(map(int, input().split()))
10    # Проверить количество элементов списка.
11    if len(num_list) != 10:
12        print("Неверный размер списка", file=sys.stderr)
13        exit(1)
14
15    # Найти искомое произведение.
16    # (Условия '>8' '<18' '%10 == 0', дают нам, что элемент равен 10)
17    sum_of_nums = 0
18    for item in num_list:
19        if item == 10:
20            sum_of_nums += 1
21
22    print(f"Кол-во элементов: {sum_of_nums}, их произведение: {10**sum_of_nums}")
23
```

Рисунок 6.11 – Код программы

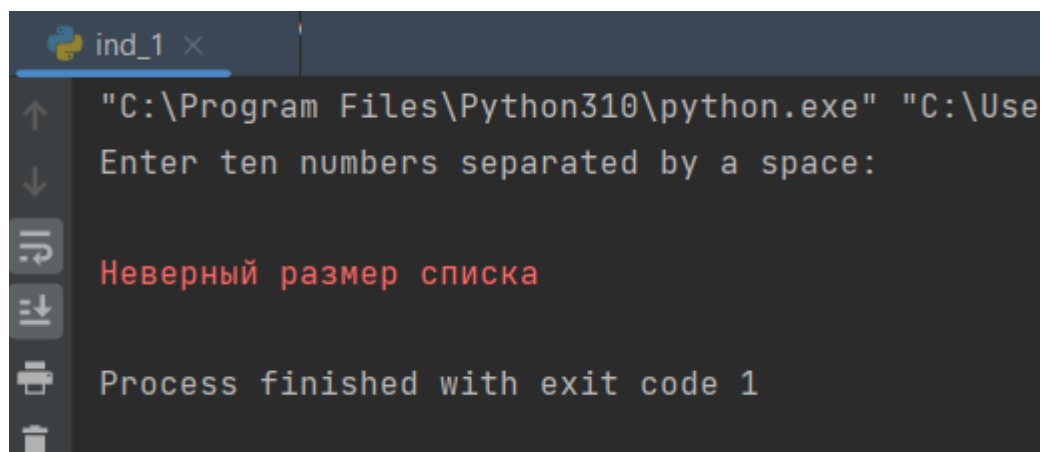
```
ex_2.py x ind_1.py x ind_1_2.py x ind_2.py x
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import sys
5
6 if __name__ == '__main__':
7     # Ввести список одной строкой.
8     print("Enter ten numbers separated by a space: ")
9     num_list = list(map(int, input().split()))
10    # Проверить количество элементов списка.
11    if len(num_list) != 10:
12        print("Неверный размер списка", file=sys.stderr)
13        exit(1)
14
15    # Найти искомое произведение.
16    # new_list = [y for y in num_list if y > 8 and y < 18 and y % 10 == 0]
17    # (Условия '>8' '<18' '%10 == 0', дают нам, что элемент равен 10)
18    new_list = [y for y in num_list if y == 10]
19
20    print(f"Кол-во элементов: {len(new_list)}, их произведение: {10 ** len(new_list)}")
21
```

Рисунок 6.11_2 – Код программы через List Comprehensions



```
ind_1 x
"C:\Program Files\Python310\python.exe" "C:\Users\УчебНа\I
Enter ten numbers separated by a space:
10 9 10 3 -2 -5 7 10 8 9
Кол-во элементов: 3, их произведение: 1000
Process finished with exit code 0
```

Рисунок 6.12 – Результат программы



```
ind_1 x
"C:\Program Files\Python310\python.exe" "C:\Use
Enter ten numbers separated by a space:
Неверный размер списка
Process finished with exit code 1
```

Рисунок 6.13 – Результат программы (ошибка)

Задание 2.

15. В списке, состоящем из вещественных элементов, вычислить:

1. количество отрицательных элементов списка;
2. сумму модулей элементов списка, расположенных после

минимального по модулю элемента.

Заменить все отрицательные элементы списка их квадратами и упорядочить элементы списка по возрастанию.

```

ex_2.py x ind_1.py x ind_1_2.py x ind_2.py x
1  ▶ #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6
7  ▶ if __name__ == '__main__':
8      # Ввести список одной строкой.
9      num_list = list(map(float, input().split()))
10     # Если список пуст, завершить программу.
11     if not num_list:
12         print("Заданный список пуст", file=sys.stderr)
13         exit(1)
14
15     # Определить индекс минимального элемента.
16     # Найти кол-во отрицательных
17     a_min = num_list[0]
18     i_min = 0
19     neg_num_sum = 0
20     for i, item in enumerate(num_list):
21         if item < a_min:
22             i_min, a_min = i, item
23         if item < 0:
24             neg_num_sum += 1
25
26 if __name__ == '__main__':

```

```

ex_2.py x ind_1.py x ind_1_2.py x ind_2.py x
25
26     # Нахождение суммы модулей элементов
27     abs_sum = 0
28     for item in num_list[i_min+1::]:
29         abs_sum += abs(item)
30
31     # Замена элементов списка
32     for i, item in enumerate(num_list):
33         if item < 0:
34             num_list[i] = num_list[i]**2
35
36     # Упорядочивание списка
37     num_list.sort()
38     # Приведение списка к формату двух чисел после запятой
39     # num_list = ['%.2f' % elem for elem in num_list]
40
41     print(f"Кол-во отрицательных элементов: {neg_num_sum}")
42     print(f"Сумма модулей элементов, расположенных после", end=' ')
43     print(f"минимального по модулю элемента {abs_sum}")
44     print(f"Модифицированный массив: {num_list}")
45

```

Рисунок 6.14 – Код программы

```

ind_2 x
"C:\Users\kuvsh\Desktop\CK0Y\2_3_семестр\Основы Программной Инженерии\Git\Py_L7\PyCharm\venv\Script
1 0 -2 5 -7 1 2
Кол-во отрицательных элементов: 2
Сумма модулей элементов, расположенных после минимального по модулю элемента 3.0
Модифицированный массив: [0.0, 1.0, 1.0, 2.0, 4.0, 5.0, 49.0]

Process finished with exit code 0

```

Рисунок 6.15 – Результат программы

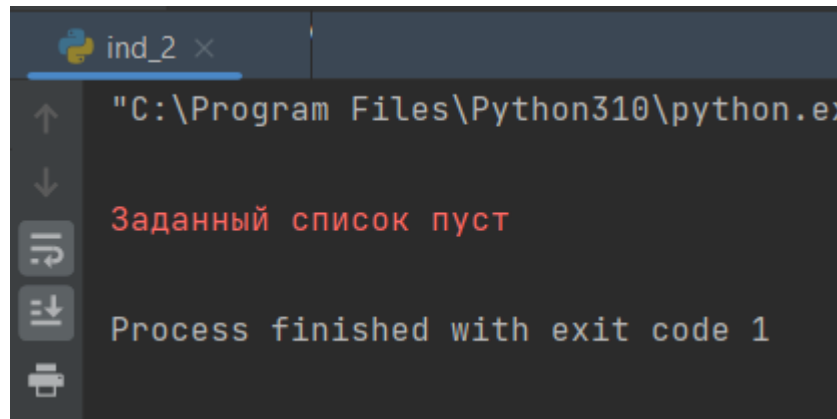


Рисунок 6.16 – Результат программы (ошибка)

Вопросы для защиты работы

1. Что такое списки в языке Python?

Список (list) – это структура данных для хранения объектов различных типов.

2. Как осуществляется создание списка в Python?

Для создания списка нужно заключить элементы в квадратные скобки:

```
my_list = [1, 2, 3, 4, 5]
```

3. Как организовано хранение списков в оперативной памяти?

Как уже было сказано выше, список является изменяемым типом данных. При его создании в памяти резервируется область, которую можно условно назвать некоторым “контейнером”, в котором хранятся ссылки на другие элементы данных в памяти. В отличие от таких типов данных как число или строка, содержимое “контейнера” списка можно менять.

4. Каким образом можно перебрать все элементы списка?

Читать 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?

Для объединения списков можно использовать оператор сложения (+):

Список можно повторить с помощью оператора умножения (*):

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 сколько раз данный элемент встречается в списке

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

Метод insert можно использовать, чтобы вставить элемент в список

```
my_list = [1, 2, 3, 4, 5]
my_list.insert(1, 'Привет')
print(my_list)
```

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

```
my_list = ['один', 'два', 'три', 'четыре', 'пять']
removed = my_list.pop(2)
print(my_list)
print(removed)
```

Результат:

```
['один', 'два', 'четыре', 'пять']
три
```

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

List Comprehensions чаще всего на русский язык переводят как абстракция списков или списковое включение, является частью синтаксиса языка, которая предоставляет простой способ построения списков. Проще всего работу list comprehensions показать на примере. Допустим вам необходимо создать список целых чисел от 0 до n, где n предварительно задается. Классический способ решения данной задачи выглядел бы так:

```
>>> n = int(input())
7
>>> a=[]
>>> for i in range(n):
    a.append(i)

>>> print(a)
[0, 1, 2, 3, 4, 5, 6]
```

Использование *list comprehensions* позволяет сделать это значительно проще:

```
>>> n = int(input())
7
>>> a = [i for i in range(n)]
>>> print(a)
[0, 1, 2, 3, 4, 5, 6]
```

или вообще вот так, в случае если вам не нужно больше использовать *n*:

```
>>> a = [i for i in range(int(input()))]
7
>>> print(a)
[0, 1, 2, 3, 4, 5, 6]
```

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

Слайс задается 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 Python предоставляет следующие функции:

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

Для функций `min` и `max` элементы списка должны быть сравнимы между собой.

14. Как создать копию списка?

Поэтому для создания копии списка необходимо использовать либо метод `copy`, либо использовать оператор среза.

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

Функция `sorted()` в Python, выполняет сортировку.

Выполняет сортировку последовательности по возрастанию/убыванию.

Метод `sort()` работает только со списками и сортирует уже имеющийся список. Данный метод ничего не возвращает. А метод `sorted()` работает с любыми итерируемыми объектами и возвращает новый отсортированный список. В качестве итерируемых объектов могут выступать списки, строки, кортежи и другие.