

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФГАОУ высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Кафедра инфокоммуникаций

Основы кроссплатформенного программирования

**Отчет по лабораторной работе №5
Работа со списками в языке Python**

Выполнил студент группы

ИТС-б-з-22-1

Рябов З.А. « » _____ 2023г.

Подпись студента _____

Работа защищена « » _____ 2023г.

Проверил доцент, кандидат технических
наук, доцент кафедры инфокоммуникаций

Воронкин Роман Александрович

(подпись)

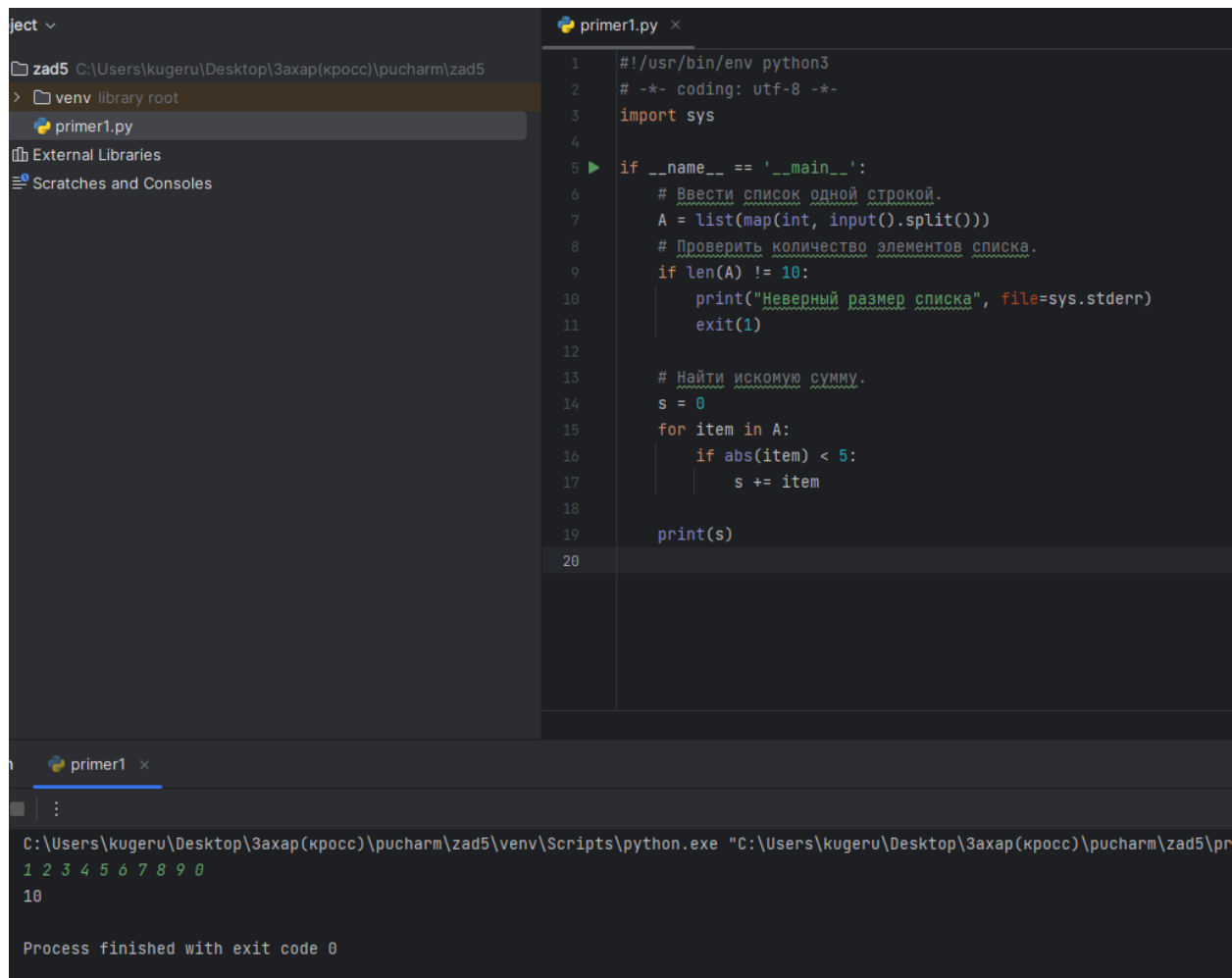
Ставрополь, 2023 г.

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

Ход работы:

Создал общедоступный репозиторий на GitHub (<https://github.com/DemoGood/zadanie5>).

Пример 1. Ввести список A из 10 элементов, найти сумму элементов, меньших по модулю 5, и вывести ее на экран.



The screenshot shows a Python IDE with a file explorer on the left and a code editor on the right. The file explorer shows a project named 'zad5' with a subdirectory 'venv' and a file 'primer1.py'. The code editor shows the following Python code:

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

Below the code editor, there is a terminal window showing the execution of the program. The command prompt shows the command to run the program, and the output shows the input list '1 2 3 4 5 6 7 8 9 0' and the resulting sum '10'.

```
C:\Users\kugeru\Desktop\Захар(кросс)\pucharm\zad5\venv\Scripts\python.exe "C:\Users\kugeru\Desktop\Захар(кросс)\pucharm\zad5\pr
1 2 3 4 5 6 7 8 9 0
10
Process finished with exit code 0
```

Рисунок 1 – Окно программы примера 1

Пример 2. Написать программу, которая для целочисленного списка определяет, сколько положительных элементов располагается между его максимальным и минимальным элементами.

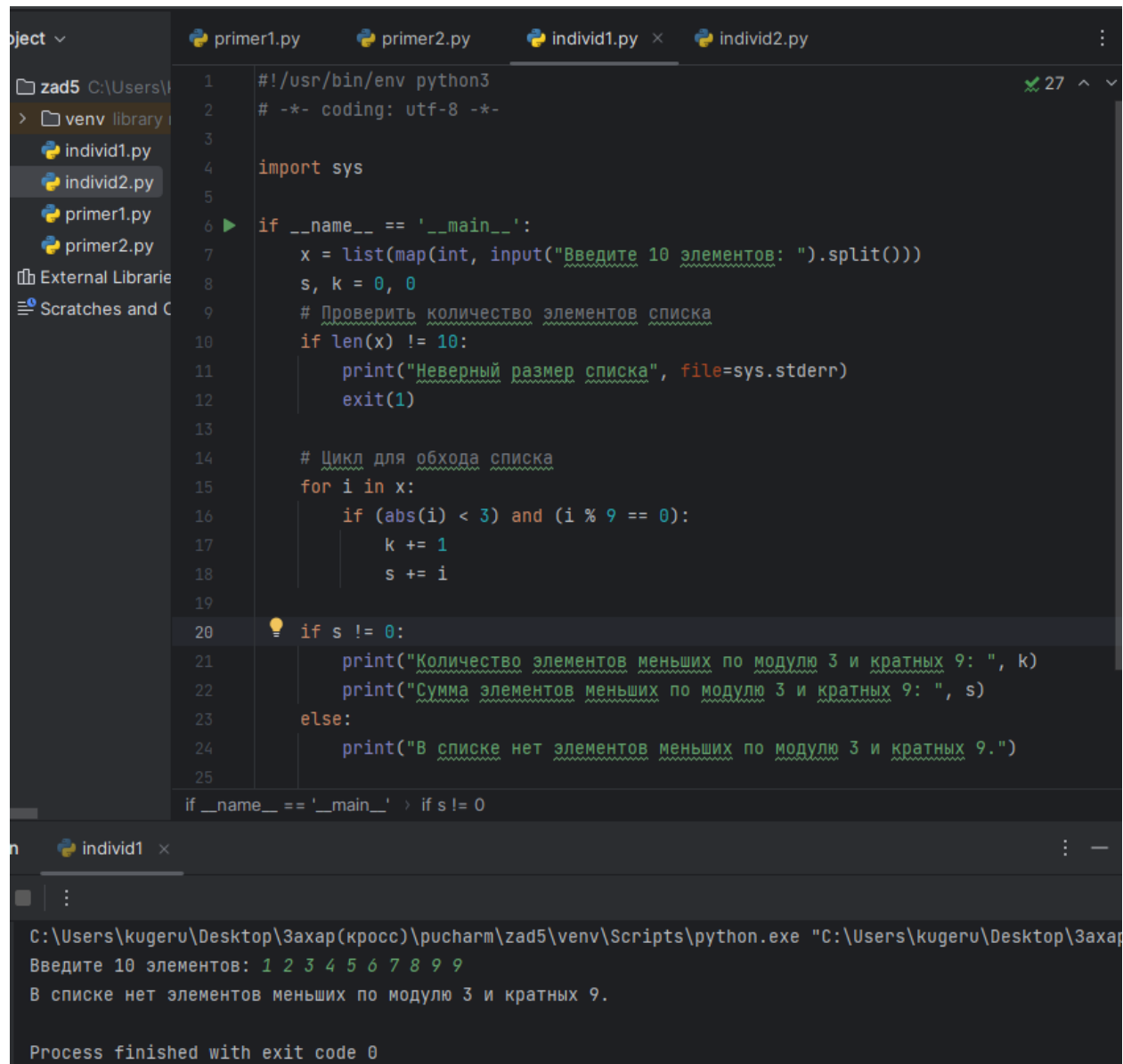
```
primer1.py x primer2.py x
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6  if __name__ == '__main__':
7      # Ввести список одной строкой.
8      a = list(map(int, input().split()))
9      # Если список пуст, завершить программу.
10     if not a:
11         print("Заданный список пуст", file=sys.stderr)
12         exit(1)
13
14     # Определить индексы минимального 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         if item >= a_max:
21             i_max, a_max = i, item
22
23     # Проверить индексы и обменять их местами.
24     if i_min > i_max:
25         i_min, i_max = i_max, i_min
26
27     # Посчитать количество положительных элементов.
28     count = 0
29     for item in a[i_min + 1:i_max]:
30         if item > 0:
31             count += 1
32
33     print(count)
34
```

```
C:\Users\kugeru\Desktop\Захар(кросс)\pucharm\zad5\venv\Scripts\python
1 2 3 4 5
3
Process finished with exit code 0
```

Рисунок 2 – Окно программы примера 2

Индивидуальное задание 1.

13. Ввести список *A* из 10 элементов, найти сумму элементов, меньших по модулю 3 и кратных 9, их количество и вывести результаты на экран.



```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6  if __name__ == '__main__':
7      x = list(map(int, input("Введите 10 элементов: ").split()))
8      s, k = 0, 0
9      # Проверить количество элементов списка
10     if len(x) != 10:
11         print("Неверный размер списка", file=sys.stderr)
12         exit(1)
13
14     # Цикл для обхода списка
15     for i in x:
16         if (abs(i) < 3) and (i % 9 == 0):
17             k += 1
18             s += i
19
20     if s != 0:
21         print("Количество элементов меньших по модулю 3 и кратных 9: ", k)
22         print("Сумма элементов меньших по модулю 3 и кратных 9: ", s)
23     else:
24         print("В списке нет элементов меньших по модулю 3 и кратных 9.")
25
if __name__ == '__main__': if s != 0
```

Process finished with exit code 0

Рисунок 3 – Окно программы для первой задачи и проверка кода на работоспособность.

Индивидуальное задание 2.

Составить программу с использованием одномерных массивов для решения задачи на переупорядочивание элементов массива. Для сортировки допускается использовать метод *sort* с заданным параметром *key* (<https://docs.python.org/3/howto/sorting.html>) и объединение нескольких списков. Номер варианта необходимо получить у преподавателя.

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

1. количество элементов списка, равных 0;
2. сумму элементов списка, расположенных после минимального элемента.

Упорядочить элементы списка по возрастанию модулей элементов.

```
primer1.py  primer2.py  individ1.py  individ2.py ×

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6  if __name__ == '__main__':
7      x = list(map(float, input("Введите элементы списка: ").split()))
8      if len(x) == 0:
9          print("Список нyc!", file=sys.stderr)
10         exit(1)
11         k = 0 # Счетчик количества нулей
12         s = 0 # Счетчик суммы элементов, расположенных после мин. элемента
13         m = x[0] # Присваиваем минимальному значению первый элемент списка
14         j = 1 # Индекс минимального элемента
15         t = 0 # Счетчик итераций первого цикла
16
17         # Цикл для нахождения минимального элемента и количества нулей
18         for i in x:
19             t += 1 # Счетчик итераций
20             if i == 0:
21                 k += 1
22             if i < m:
23                 m = i
24                 j = t # запоминаем индекс минимального значения
25
26         # Цикл для нахождения суммы элементов, расположенных после мин. элемента
27         for i in range(len(x)):
28             if i > j - 1:
29                 s += x[i]
30
31         # Упорядочить элементы списка по возрастанию модуля
32         x1 = x
33         for i in range(len(x)):
34             x1[i] = abs(x1[i])
35
36         x1.sort()
37         x = x1
38
39         print("Упорядоченный список: ", x)
40         print("Количество элементов списка равных 0: ", k)
41         print("Сумма элементов списка, расположенных после мин. элемента: ", s)
42
```

```
C:\Users\kugeru\Desktop\Захар(кросс)\pucharm\zad5\venv\Scripts\python
Введите элементы списка: 2 1 -1 2 2
Упорядоченный список: [1.0, 1.0, 2.0, 2.0, 2.0]
Количество элементов списка равных 0: 0
Сумма элементов списка, расположенных после мин. элемента: 4.0

Process finished with exit code 0
```

Рисунок 4 – Окно программы для второй задачи и проверка кода на работоспособность.

Ответы на контрольные вопросы:

1. Что такое списки в языке Python?

Список (list) – это структура данных для хранения объектов различных типов. В нем можно хранить объекты различных типов. Размер списка 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 на какие-либо другие объекты или структуры.

2. Как осуществляется создание списка в Python?

Для создания списка нужно заключить элементы в квадратные скобки.

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 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 `append` можно использовать для добавления элемента в список.

Метод `insert` можно использовать, чтобы вставить элемент в список.

9. Как выполнить сортировку списка?

Для сортировки списка нужно использовать метод `sort`. Для сортировки списка в порядке убывания необходимо вызвать метод `sort` с аргументом `reverse=True`.

10. Как удалить один или несколько элементов из списка?

Удалить элемент можно, написав его индекс в методе `pop`. Если не указывать индекс, то функция удалит последний элемент. Элемент можно удалить с помощью метода `remove`. Оператор `del` можно использовать для тех же целей.

Можно удалить несколько элементов с помощью оператора среза.

Можно удалить все элементы из списка с помощью метода `clear`.

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.

В языке Python есть две 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. Какие существуют функции агрегации для работы со списками?

Для работы со списками Python предоставляет следующие функции:

1. len(L) - получить число элементов в списке L
2. min(L) - получить минимальный элемент списка L
3. max(L) - получить максимальный элемент списка L
4. sum(L) - получить сумму элементов списка L, если список L

содержит только числовые значения.

13. Как создать копию списка? `copy.copy(x)`

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

Функция `sorted()` в Python возвращает отсортированный список из элементов в итерируемом объекте. `list.sort()` на 13% быстрее, чем `sorted()`.

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

Функция `sort()` очень похожа на `sorted()`, но в отличие от `sorted` она ничего не возвращает и не вносит изменений в исходную последовательность. Более того, `sort()` является методом класса `list` и может использоваться только со списками. Синтаксис: `List_name.sort(key, reverse=False)` Параметры: ключ: Функция, которая служит ключом для сравнения сортировки. реверс: Если `true`, то список сортируется в порядке убывания.

Вывод: приобрел навык по работе со списками при написании программ с помощью языка программирования Python.