

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ  
Федеральное государственное автономное  
образовательное учреждение высшего образования  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Кафедра инфокоммуникаций  
«Работа со списками в языке Python»**

**Отчет по лабораторной работе № 2.4  
по дисциплине «Основы кроссплатформенного  
программирования»**

Выполнил студент группы ИВТ-б-о-21-1

Шайдеров Дмитрий Викторович.

«20 » мая 2022г.

Подпись студента \_\_\_\_\_

Работа защищена « » \_\_\_\_\_ 20\_\_ г.

Проверил Воронкин Р.А. \_\_\_\_\_  
(подпись)

Ставрополь 2022

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

**Порядок выполнения работы:**

1. Создал общедоступный репозиторий на GitHub, в котором использована лицензия MIT.

Owner \* Repository name \*

dshayderov / lw7 ✓

Great repository names are short and memorable. Need inspiration? How about [ideal-c](#)

Description (optional)

☒ Public  
Anyone on the internet can see this repository. You choose who can commit.

☐ Private  
You choose who can see and commit to this repository.

Initialize this repository with:  
Skip this step if you're importing an existing repository.

☒ Add a README file  
This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore  
Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: Python ▾

Choose a license  
A license tells others what they can and can't do with your code. [Learn more.](#)

License: MIT License ▾

2. Выполните клонирование созданного репозитория.

```
Asus@LAPTOP-09A994CG MINGW64 /c/My projects/2
$ git clone https://github.com/dshayderov/lw7.git
Cloning into 'lw7'...
remote: Enumerating objects: 11, done.
remote: Counting objects: 100% (11/11), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 11 (delta 2), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (11/11), done.
Resolving deltas: 100% (2/2), done.
```

Рисунок 1 - Клонирование репозитория

3. Организуйте свой репозиторий в соответствии с моделью ветвления git-flow.

```

Asus@LAPTOP-09A994CG MINGW64 /c/My projects/2
$ cd "C:\My projects\2\lw7"

Asus@LAPTOP-09A994CG MINGW64 /c/My projects/2/lw7 (main)
$ git checkout -b develop
Switched to a new branch 'develop'

```

Рисунок 2 - Ветвление по модели git-flow

4. Создайте проект PyCharm в папке репозитория.

Project	29.05.2022 18:40	Папка с файлами	
.gitignore	29.05.2022 18:37	Файл "GITIGNORE"	2 КБ
LICENSE	29.05.2022 18:37	Файл	2 КБ
README.md	29.05.2022 18:37	Файл "MD"	1 КБ

Рисунок 3 - Проект PyCharm

5. Проработайте примеры лабораторной работы. Создайте для каждого примера отдельный модуль языка Python. Зафиксируйте изменения в репозитории.

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

```

example_1 x
"C:\My projects\2\lw7\Project\venv\
0 1 2 3 4 5 6 7 8 9
10

```

Рисунок 4 – Результат выполнения программы

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

```

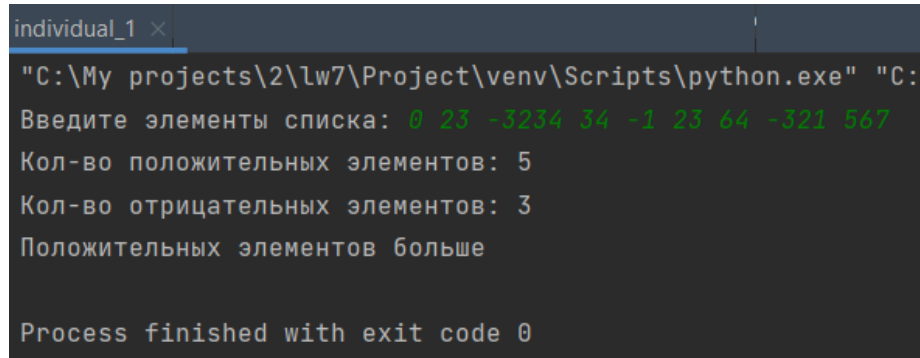
exmple_2 x
"C:\My projects\2\lw7\Project\venv\
1 4 5 2 12 43 78 23 45 78 09 98
10

```

Рисунок 5 - Результат выполнения программы

6. Выполните индивидуальные задания (Вариант 27).

Задание 1. Для заданного списка определить, каких элементов больше: положительных или отрицательных. Вывести на экран их количество.



```
individual_1 x
"C:\My projects\2\lw7\Project\venv\Scripts\python.exe" "C:
Введите элементы списка: 0 23 -3234 34 -1 23 64 -321 567
Кол-во положительных элементов: 5
Кол-во отрицательных элементов: 3
Положительных элементов больше

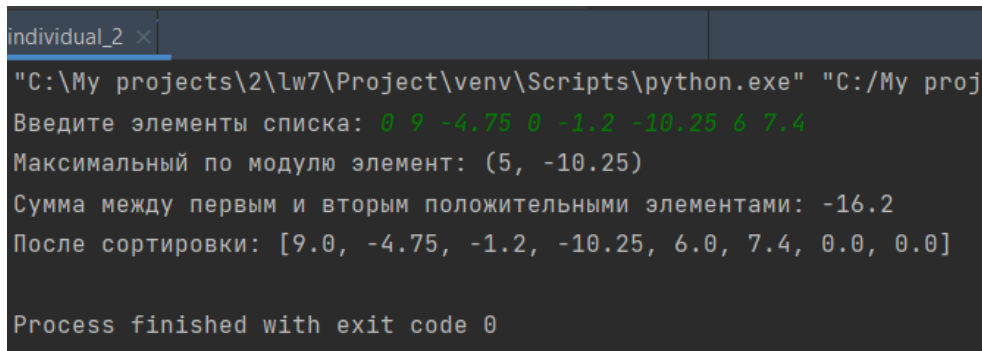
Process finished with exit code 0
```

Рисунок 6 - Результат выполнения программы

Задание 2 (Вариант 8). В списке, состоящем из вещественных элементов, вычислить:

1. максимальный по модулю элемент списка;
2. сумму элементов списка, расположенных между первым и вторым положительными элементами.

Преобразовать список таким образом, чтобы элементы, равные нулю, располагались после всех остальных.



```
individual_2 x
"C:\My projects\2\lw7\Project\venv\Scripts\python.exe" "C:/My proj
Введите элементы списка: 0 9 -4.75 0 -1.2 -10.25 6 7.4
Максимальный по модулю элемент: (5, -10.25)
Сумма между первым и вторым положительными элементами: -16.2
После сортировки: [9.0, -4.75, -1.2, -10.25, 6.0, 7.4, 0.0, 0.0]

Process finished with exit code 0
```

Рисунок 7 - Результат выполнения программы

**Контрольные вопросы:**

**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]`

Список может выглядеть так: `my_list = ['один', 'два', 'три', 'четыре', 'пять']`.

Можно смешивать типы содержимого: `my_list = ['один', 10, 2.25, [5, 15], 'пять']`.

### **3. Как организовано хранение списков в оперативной памяти?**

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?**

Перебор элементов списка состоит в том, что мы в цикле просматриваем все элементы этого 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()`, в порядке возрастания будет(`list1.sort()`). Для сортировки списка в порядке убывания необходимо вызвать метод `sort` с аргументом `reverse=True`(`list1.reverse()`).

### **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 чаще всего на русский язык переводят как абстракция списков или списковое включение, является частью синтаксиса языка, которая предоставляет простой способ построения списков.

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

`list[::]`

### **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

### **14. Как создать копию списка?**

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

Отличие заключается в том, что метод `list.sort()` определён только для списков, в то время как `sorted()` работает со всеми итерируемыми объектами.

**Вывод:** приобрел навыки по работе со списками при написании программ с помощью языка программирования Python версии 3.x.