

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

Институт цифрового
развития

Кафедра инфокоммуникаций

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ №5

**Дисциплины «Основы кроссплатформенного
программирования»**

Выполнил:
Гуляницкий Александр
Евгеньевич
1 курс, группа ИТС-б-о-21-1,
11.03.02 «Инфокоммуникационные
технологии и системы связи»,
направленность (профиль)
«Инфокоммуникационные системы и
сети», очная форма обучения

(подпись)

Руководитель практики: Воронкин Р.
А, канд. техн. наук, доцент кафедры
инфо коммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____


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

Ход работы:

Создал новый репозиторий <https://github.com/Alexander-its/laba-5> и начал отработку примеров

Пример 1:

P

 primer-1.py - C:/Users/GO_PB/Desktop/primer-1.py (3.10.4)

File Edit Format Run Options Window Help


```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
import sys

if __name__ == '__main__':
    # Ввести список одной строкой.
    A = list(map(int, input().split()))
    # Проверить количество элементов списка.
    if len(A) != 10:
        print("Неверный размер списка", file=sys.stderr)
        exit(1)
    # Найти искомую сумму.
    s = 0
    for item in A:
        if abs(item) < 5:
            s += item
    print(s)
```

```
|
|===== RESTART: C:/Users/GO_PB/Desktop/primer-1.py =====
| 1 2 3 4 5 6 7 8 9 0
| 10
>>>
```

исунок 1. Работа программы «Пример 1»

Пример 2:

 primer-2.py - C:/Users/GO_PB/AppData/Local/Programs/Python/Python310/primer-2.py (3.1.

File Edit Format Run Options Window Help

```
#!/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)
    # 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)
```

```
> == RESTART: C:/Users/GO_PB/AppData/Local/Programs/Python/Python310/primer-
123 123 456 346 3246457 235346 3
```

```
1
```

Рисунок 2. Работа программы «Пример 2»

Задание 1:

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

```
a = 0
s = 0
while True:
    a = int(input())

    if a == 0:
        break
    elif a < 0:
        s += 1
print(s)
```

```
===== RESTART: C
5
-20
-1
3
-7
0
3
|
```

Рисунок 3. Работа программы «Задание 1»

Задание 2:

В списке, состоящем из вещественных элементов, вычислить:

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

положительными элементами.

```
from random import randint

array = [randint(-5, 30) for i in range(30)]
pos_1 = 0
pos_2 = 0
for i in range(len(array)):
    if pos_1 and pos_2:
        break
    elif array[i] > 0 and not pos_1:
        pos_1 = i
    elif array[i] > 0 and not pos_2:
        pos_2 = i

modified_array = [i for i in array if i] + [0] * array.count(0)

print(f'Максимальный по модулю: {max(array, key=abs)}')
print(f'Сумма между: {sum(array[i] for i in range(pos_1, pos_2 + 1))}' if pos_1
print(f'Массив со смещением нулей: {modified_array}')
```

```
Максимальный по модулю: 30
Сумма между: 22
Массив со смещением нулей: [-1, 9, 13, -5, 30, 21, -5, 19, 12, 12, 17, 30, -5, 2
2, 16, 7, 10, -3, 11, 23, 18, 2, 10, 2, -4, 16, -5, 18, 27, 22]
```

Рисунок 4. Работа программы «Задание 2»

Вывод: Я приобрёл навыки по работе со списками с помощью языка программирования Python.

Контрольные вопросы

1. Что такое списки в языке Python?

Ответ:Список (*list*) – это структура данных для хранения объектов 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 ссылки на какие-либо другие объекты или структуры.

2. Как осуществляется создание списка в Python?

Ответ: Для создания списка нужно заключить элементы в квадратные скобки:

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

3. Как организовано хранение списков в оперативной памяти?

Ответ: 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?

Ответ: 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?

Ответ: Для объединения списков можно использовать оператор сложения (+).

Список можно повторить с помощью оператора умножения (*):

6. Как проверить есть ли элемент в списке?

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

7. Как определить число вхождений заданного элемента в списке?

Ответ: Метод `count` можно использовать для определения числа сколько раз данный элемент встречается в списке:

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

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

Метод `append` можно использовать для добавления элемента в список.

9. Как выполнить сортировку списка?

Ответ: Для сортировки списка нужно использовать метод `sort`.

Для сортировки списка в порядке убывания необходимо вызвать метод *sort* аргументом *reverse=True*.

10. Как удалить один или несколько элементов из списка?

Ответ: 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:

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

Ответ: *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. Как осуществляется доступ к элементам списков с помощью срезов?

Ответ: Слайсы (срезы) являются очень мощной составляющей *Python*, которая позволяет быстро и лаконично решать задачи выборки элементов из списка.

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?

Ответ: 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?

Ответ: Воспользоваться командой `copy.copy(x)`

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

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