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 №7**

**дисциплины «Программирование на Python»**

**Вариант 23**

|  |  |  |
| --- | --- | --- |
|  | Выполнил:  Омонкулов Исомиддин Валижон угли  2 курс, группа ИВТ-б-о-22-1,  09.03.01 «Информатика и вычислительная техника», направленность (профиль) «Программное обеспечение средств вычислительной техники и автоматизированных систем», очная форма обучения  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | |
|  | Руководитель практики:  Воронкин Р. А.  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | |
|  | |  | |

Отчет защищен с оценкой \_\_\_\_\_\_\_\_\_\_\_ Дата защиты\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Ставрополь, 2024

**Тема:** Работа со списками в языке Python

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

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

**Листинг к примеру №1:**

#!/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)

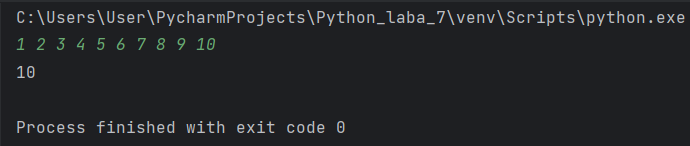


Рисунок 1. Тест №1

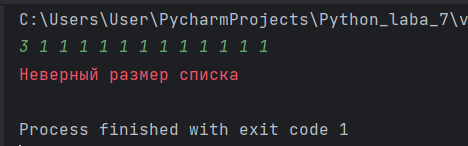


Рисунок 2. Тест №2

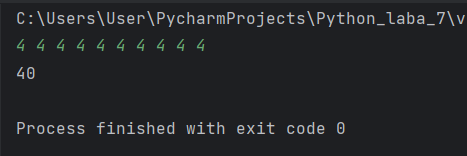


Рисунок 3. Тест №3

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

**Листинг к примеру №2:**

#!/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)

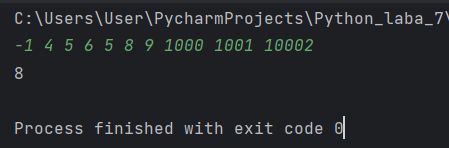


Рисунок 4. Тест №1

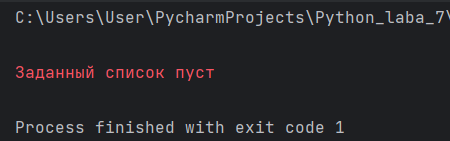


Рисунок 5. Тест №2

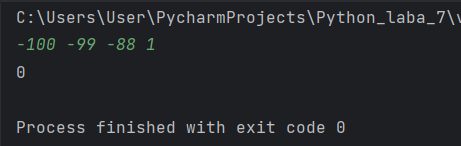


Рисунок 6. Тест №6

**Индивидуальное задание №1.** Ввести список А из 10 элементов, найти произведение отрицательных элементов и вывести его на экран.

**Листинг к индивидуальному заданию №1:**

#!/usr/bin/env python3  
# -\*- coding: utf-8 -\*-  
  
import sys  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 A = list(map(int, input().split()))  
 if len(A) != 10:  
 print('Неверный размер списка', file=sys.stderr)  
 exit(1)  
  
 total = 1  
 for item in A:  
 if item < 0:  
 total \*= item  
  
 print(total)

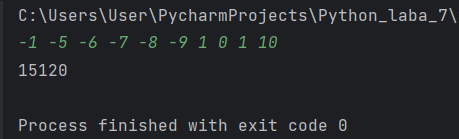


Рисунок 7. Тест №1

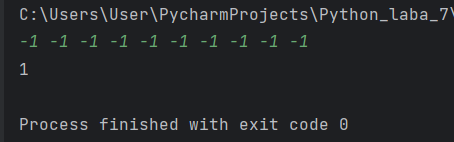


Рисунок 8. Тест №2

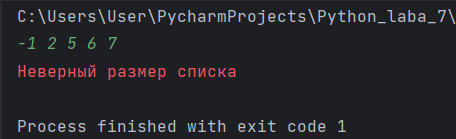


Рисунок 9. Тест №3

**Второй способ решения данного задания (с использованием lambda функции):**

#!/usr/bin/env python3  
# -\*- coding: utf-8 -\*-  
  
import sys  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 A = list(map(int, input().split()))  
 if len(A) != 10:  
 print('Неверный размер списка', file=sys.stderr)  
 exit(1)  
  
 total = 1  
 B = list(filter(lambda x: x < 0, A))  
 for item in B:  
 total \*= item  
  
 print(total)

**Индивидуальное задание №2.** В списке, состоящем из вещественных элементов, вычислить:

1. номер минимального элемента списка;

2. сумму элементов списка, расположенных между первым и вторым отрицательными элементами.

**Листинг к индивидуальному заданию №2:**

#!/usr/bin/env python3  
# -\*- coding: utf-8 -\*-  
  
import sys  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 lst = list(map(float, input().split()))  
  
 # Find the number of the minimum list item  
 min\_index = sorted(range(len(lst)), key=lambda i: lst[i])[0]  
  
 # Find the sum of the list items located between the first and second negative elements  
 neg\_indexes = sorted([i for i in range(len(lst)) if lst[i] < 0])  
  
 if len(neg\_indexes) >= 2:  
 sum\_between = sum(lst[neg\_indexes[0] + 1:neg\_indexes[1]])  
 print(f"The number of the minimum element: {min\_index}")  
 print(f"The sum of the elements between the first and second negative: {sum\_between}")  
 else:  
 print("There are no two negative elements in the list!", file=sys.stderr)

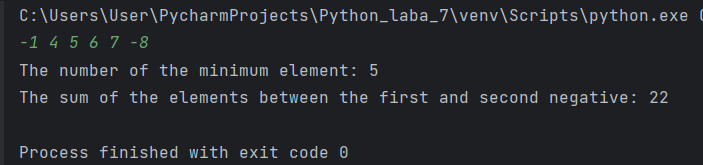


Рисунок 10. Тест №1

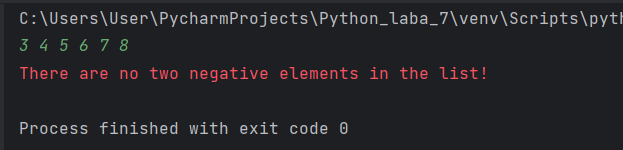


Рисунок 11. Тест №2

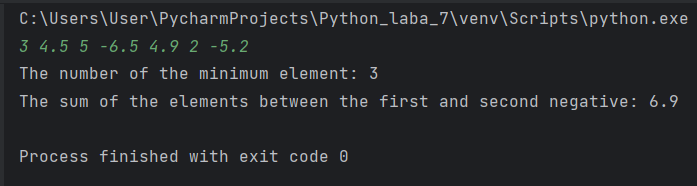


Рисунок 12. Тест №3

**Вывод:** в ходе выполнения данной лабораторной работы были приобретены навыки взаимодействия со списками при написании программ с помощью языка программирования Python версии 3.x.

**Ответы на контрольные вопросы**

1. Список в языке Python - это упорядоченная изменяемая коллекция объектов.

2. Создание списка осуществляется путем перечисления элементов в квадратных скобках, разделенных запятыми.

3. Списки в оперативной памяти хранятся как последовательность указателей на объекты.

4. Все элементы списка можно перебрать с помощью цикла for или с использованием генераторов списков.

5. Списки поддерживают операции сложения, умножения на число, извлечение срезов, проверку на вхождение элемента.

6. Для проверки наличия элемента в списке можно использовать оператор in.

7. Число вхождений заданного элемента в список можно определить с помощью метода count.

8. Добавление (вставка) элемента в список осуществляется с помощью методов append, insert или расширения списка с помощью метода extend.

9. Сортировка списка осуществляется с помощью метода sort или функции sorted.

10. Удаление одного или нескольких элементов из списка можно выполнить с помощью методов remove, pop или оператора del.

11. Списковое включение (list comprehension) - это способ создания нового списка на основе уже существующего, применяя к его элементам определенное выражение.

12. Доступ к элементам списков с помощью срезов осуществляется путем указания начального и конечного индексов в квадратных скобках, разделенных двоеточием.

13. Функции агрегации для работы со списками включают в себя sum, max, min, len и другие.

14. Копию списка можно создать с помощью метода copy или путем присваивания одного списка другому.

15. Функция sorted возвращает отсортированную копию списка, не изменяя исходный список, в то время как метод sort изменяет сам список, сортируя его элементы.

**Ответы на контрольные вопросы**

1. Списки в языке Python - это упорядоченные изменяемые коллекции элементов, которые могут быть различных типов данных.

2. Список можно создать, перечисляя элементы в квадратных скобках, разделяя их запятыми. Например: my\_list = [1, 2, 3, 'строка'].

3. Списки в оперативной памяти организованы как последовательные блоки памяти, где каждый элемент списка представляет отдельный объект, а его позиция в памяти определяется его индексом.

4. Чтобы перебрать все элементы списка, можно использовать цикл for. Например: for item in my\_list: print(item)

5. Со списками можно выполнять операции сложения (+) для объединения списков, умножения (\*) для повторения списков, и индексации ([]) для доступа к определенному элементу.

6. Чтобы проверить, содержится ли элемент в списке, можно использовать оператор in. Например: if item in my\_list: print("Элемент найден")

7. Для определения числа вхождений заданного элемента в списке можно использовать метод count(). Например: count = my\_list.count(item)

8. Для добавления элемента в список можно использовать метод append() для добавления в конец списка, или метод insert() для вставки элемента на определенную позицию. Например: my\_list.append(item) или my\_list.insert(index, item).

9. Сортировку списка можно выполнить с помощью метода sort() для сортировки в порядке возрастания или метода sorted() для создания нового отсортированного списка. Например: my\_list.sort() или sorted\_list = sorted(my\_list).

10. Чтобы удалить один или несколько элементов из списка, можно использовать методы remove() для удаления по значению, del для удаления по индексу или срезу, или pop() для удаления и возврата элемента по индексу.

11. Списковое включение (list comprehension) - это конструкция, позволяющая создать новый список, выполняя операции над элементами старого списка или другого итерируемого объекта. Например: new\_list = [item \* 2 for item in my\_list].

12. Доступ к элементам списков с помощью срезов осуществляется с использованием двоеточия (:). Например: my\_list[start:end:step]. start - индекс начала среза (включительно), end - индекс конца среза (не включая), step - шаг.

13. Для работы со списками существуют функции агрегации, такие как sum() для вычисления суммы всех элементов, len() для определения количества элементов, min() и max() для нахождения минимального и максимального значений.

14. Чтобы создать копию списка, можно использовать метод copy() или срез списка. Например: new\_list = my\_list.copy() или new\_list = my\_list[:]

15. Функция sorted() языка Python возвращает отсортированную версию списка без изменения исходного списка, в то время как метод sort() изменяет исходный список, сортируя его на месте.