

**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 программной инженерии»**

Выполнил студент группы ПИЖ-б-о-21-1
Коновалова В.Н. « » 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.

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 соответствие с моделью ветвления git-flow.
6. Создайте проект PyCharm в папке репозитория.
7. Проработайте примеры лабораторной работы. Создайте для каждого примера отдельный модуль языка Python. Зафиксируйте изменения в репозитории.

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

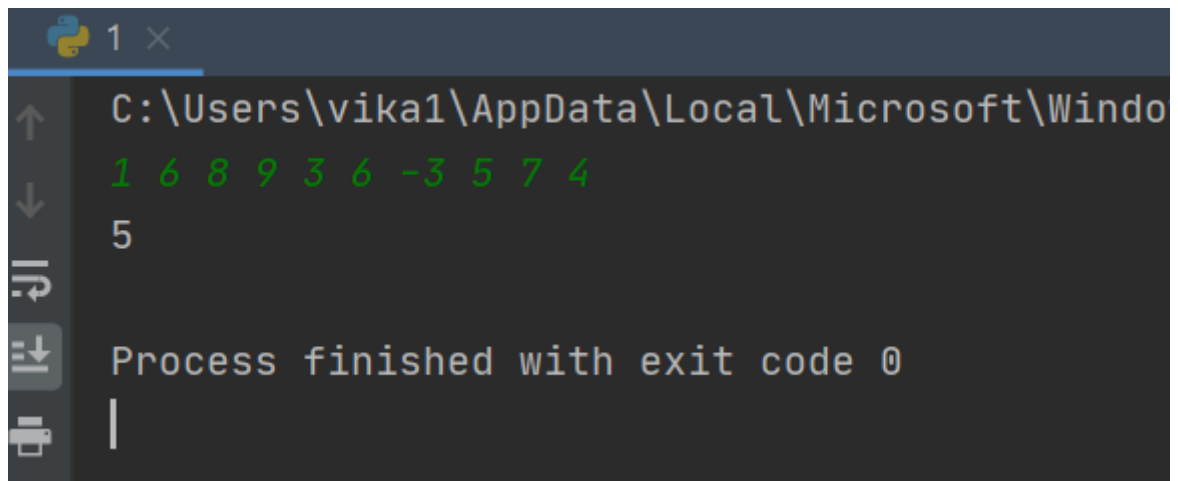
Код:

```
#!/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)
```



```
1 x
C:\Users\vika1\AppData\Local\Microsoft\Windows
1 6 8 9 3 6 -3 5 7 4
5
Process finished with exit code 0
```

Рисунок 6 – Результат работы программы

Решение задачи с помощью списковых включений:

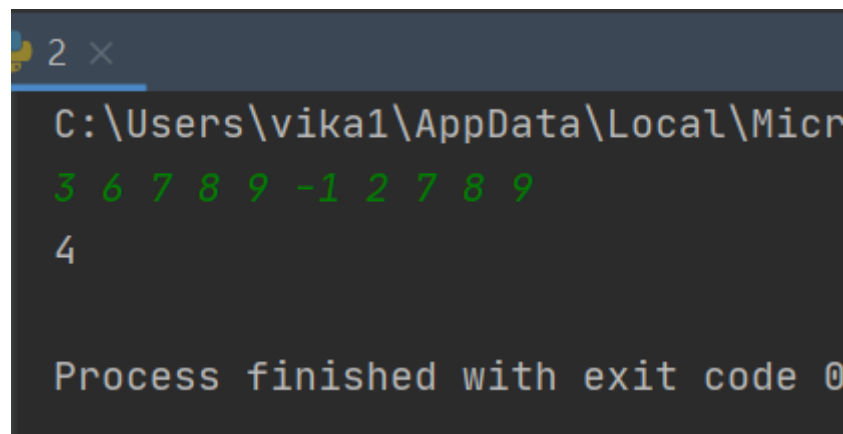
Код:

```
#!/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 = sum([a for a in A if abs(a) < 5])
    print(s)
```



```
2 x
C:\Users\vika1\AppData\Local\Microsoft\Windows
3 6 7 8 9 -1 2 7 8 9
4
Process finished with exit code 0
```

Рисунок 7 – Результат работы программы

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

Код:

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

import sys

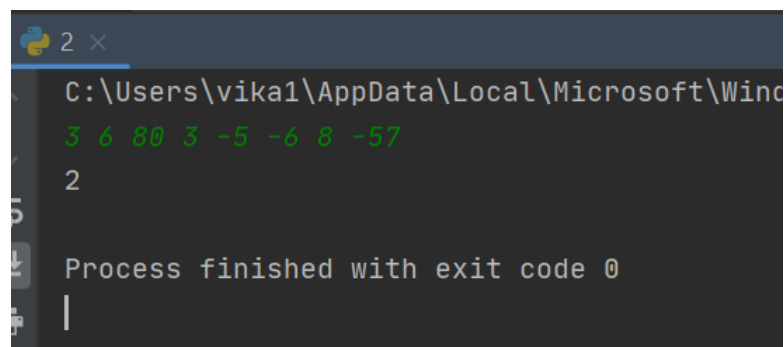
if __name__ == '__main__':
    # Ввести список одной строкой.
    a = list(map(int, input().split()))
    # Если список пуст, завершить программу.
    if not a:
        print("Заданный список пуст", file=sys.stderr)
        exit(1)

    # 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)
```



```
2 x
C:\Users\vika1\AppData\Local\Microsoft\Wind
3 6 80 3 -5 -6 8 -57
2
Process finished with exit code 0
|
```

Рисунок 8 – Результат работы программы

8. Выполните индивидуальные задания, согласно своему варианту. Для заданий повышенной сложности номер варианта должен быть получен у преподавателя.

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

Решение задачи с помощью цикла:

Код:

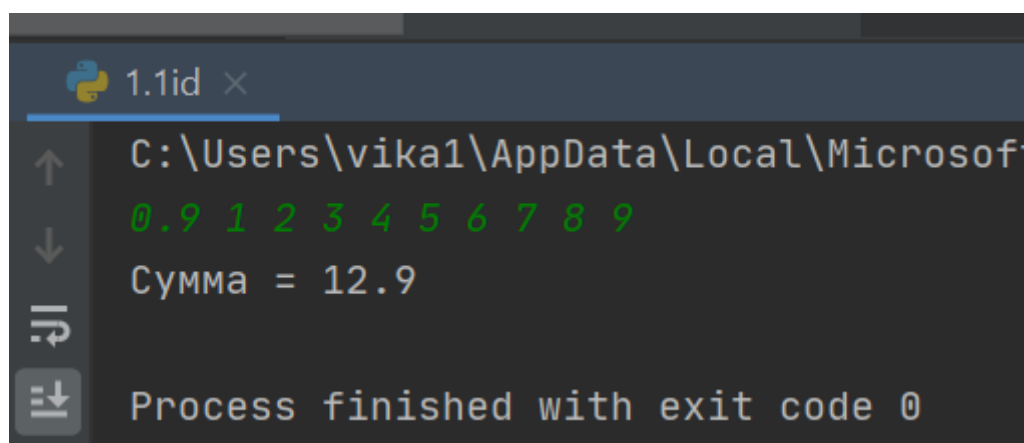
```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

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

import sys

if __name__ == '__main__':
    # Ввести список одной строкой.
    a = list(map(float, input().split()))
    # Проверить количество элементов списка.
    if len(a) != 10:
        print("Неверный размер списка", file=sys.stderr)
        exit(1)

    # Нахождение суммы отрицательных элементов
    summ = 0
    for i in a:
        if (i % 9 == 0) or (abs(3) > i):
            summ += i
    print(f"Сумма = {summ}")
```



```
1.1id x
C:\Users\vika1\AppData\Local\Microsoft
0.9 1 2 3 4 5 6 7 8 9
Сумма = 12.9
Process finished with exit code 0
```

Рисунок 10 – Результат работы программы

Решение задачи с помощью списковых включений:

Код:

```
#!/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)

    # Нахождение суммы отрицательных элементов
    summ = sum(i for i in a if i < 0)
    print(f"Сумма отрицательных элементов: {summ}")
```

The screenshot shows a Python IDE window titled '1.1id x'. The command prompt shows the file path 'C:\Users\vika1\AppData\Local\Microsoft...'. The input '0.9 1 2 3 4 5 6 7 8 9' is entered in green. The output 'Сумма = 12.9' is displayed in orange. The status bar at the bottom indicates 'Process finished with exit code 0'.

Рисунок 11 – Результат работы программы

Задание 2.

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.

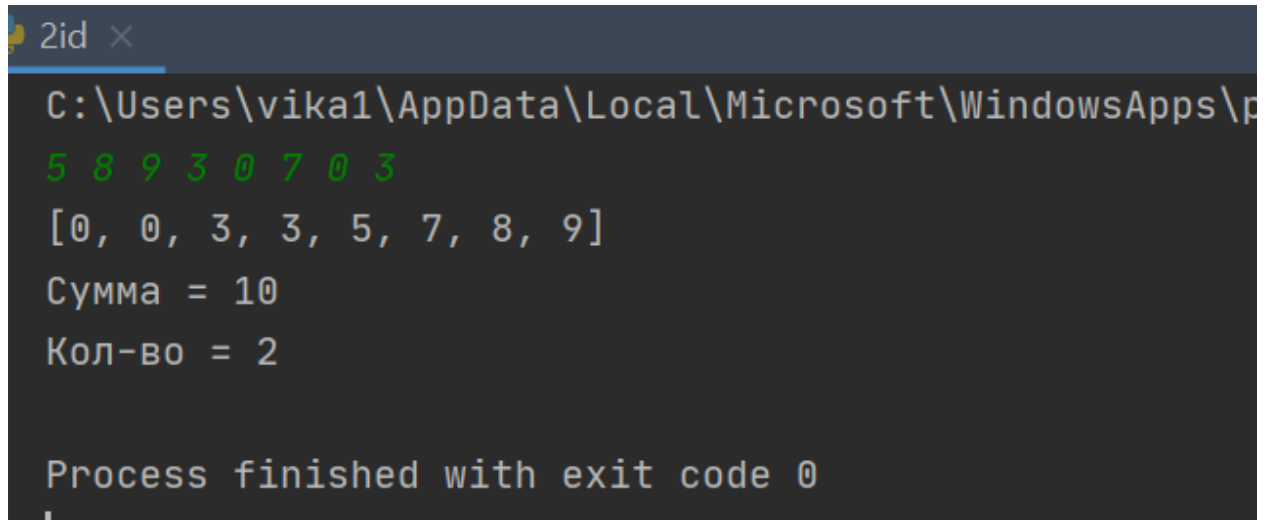
Код:

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

# 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.

if __name__ == '__main__':
    a = list(map(int, input().split()))
```

```
count = 0
a_min = a[0]
i_min = 0
for index, value in enumerate(a):
    if value == 0:
        count += 1
    if value < a_min:
        i_min, a_min = index, value
summ = sum(value for value in a[i_min + 1:])
a.sort()
print(a)
print(f"Сумма = {summ}\nКол-во = {count}")
```



```
2id x
C:\Users\vika1\AppData\Local\Microsoft\WindowsApps\p
5 8 9 3 0 7 0 3
[0, 0, 3, 3, 5, 7, 8, 9]
Сумма = 10
Кол-во = 2

Process finished with exit code 0
```

Рисунок 12 – Результат работы программы

Вопросы для защиты работы

1. Что такое списки в языке Python?

Список — это изменяемый упорядоченный тип данных предоставляющий возможность хранения объектов разных типов.

2. Как осуществляется создание списка в Python?

Для этого необходимо воспользоваться следующей конструкцией:

имя_переменной = [перечисление элементов через запятую]

или

имя_переменной = []

3. Как организовано хранение списков в оперативной памяти?

Объект списка хранит указатели на объекты, а не на сами объекты, при этом элементы могут быть «разбросаны» по памяти.

4. Каким образом можно перебрать все элементы списка?

С помощью цикла, например:

```
s = [1, 2, 3, 5]
for i in s:
    print(i)
```

Результат:

```
1
2
3
5
```

5. Какие существуют арифметические операции со списками?

1) Объединение списков (+)

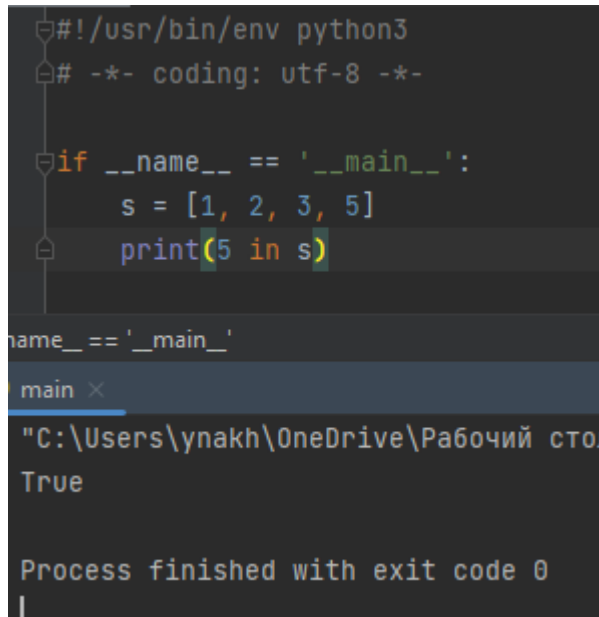
2) Умножение на число (*)

6. Как проверить есть ли элемент в списке?

Для этого можно использовать оператор in/not in. Например:

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

if __name__ == '__main__':
    s = [1, 2, 3, 5]
    print(5 in s)
```



7. Как определить число вхождений заданного элемента в списке?

Для этого используется метод count (имя_списка.count(элемент))

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

Существует несколько методов:

имя_списка.append(элемент) – добавляет в конец

имя_списка.insert(индекс, элемент) – добавляет по индексу со смещением всех последующих элементов.

9. Как выполнить сортировку списка?

Для сортировки списка нужно использовать метод sort (имя_списка.sort()) и sort(reverse=True) для сортировки в порядке убывания.

10. Как удалить один или несколько элементов из списка?

Для этого существуют методы .pop(индекс) – удаляет по индекс и возвращает удаленное значение; .remove(элемент) – удаляет первое вхождение. Также можно использовать оператор del имя_списка[индекс],

если поместить срез, удалиться несколько элементов. Удалить все элементы можно с помощью метода `.clear()`.

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

Списковое включение – это некий синтаксический сахар, позволяющий упростить генерацию последовательностей (списков, множеств, словарей, генераторов).

новый_список = [«операция» for «элемент списка» in «список»]

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

Срез имеет вид: имя_списка[start:stop:step], где start – индекс первого элемента, stop – индекс крайнего элемента (сам он не включается), step – шаг. При этом start, stop, step необязательно должны принимать значения, так отсутствие start означает срез с начала, stop – до конца, step – каждый элемент.

Также они могут принимать отрицательные значения, тогда -1 = последний элемент, -2 = предпоследний, отрицательный шаг = шаг назад. Важно, что элементы должны идти «в направлении» шага.

13. Какие существуют функции агрегации для работы со списками?

`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.

Важно, что для `min` и `max` элементы должны быть сравнимы

14. Как создать копию списка?

Это можно сделать с помощью срезов типа `a[:]`

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

Если `sort()` изменяет список, ничего не возвращая, то `sorted` возвращает измененный список, при этом не меняя исходный.