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 практической работе №2.4

Дисциплины «Программирование на Python»

Выполнил:

Пустяков Андрей Сергеевич

2 курс, группа ИВТ-б-о-22-1,

09.03.01 «Информатика и вычислительная техника (профиль) «Программное обеспечение средств вычислительной
техники и автоматизированных систем», очная форма обучения

_____

(подпись)

Руководитель практики:

Воронкин Р. А. кандидат технических наук, доцент, доцент кафедры инфокоммуникаций

_____

(подпись)

Ставрополь, 2023 г.

Тема: Работа со списками в языке Python.

Цель: приобрести навыки по работе со списками при написании программ на языке программирования Python версии 3.x.

Ход работы:

Создание репозитория, клонирование репозитория, редактирование файла «.gitignore» и организация репозитория согласно модели ветвления «git-flow» (рис. 1).

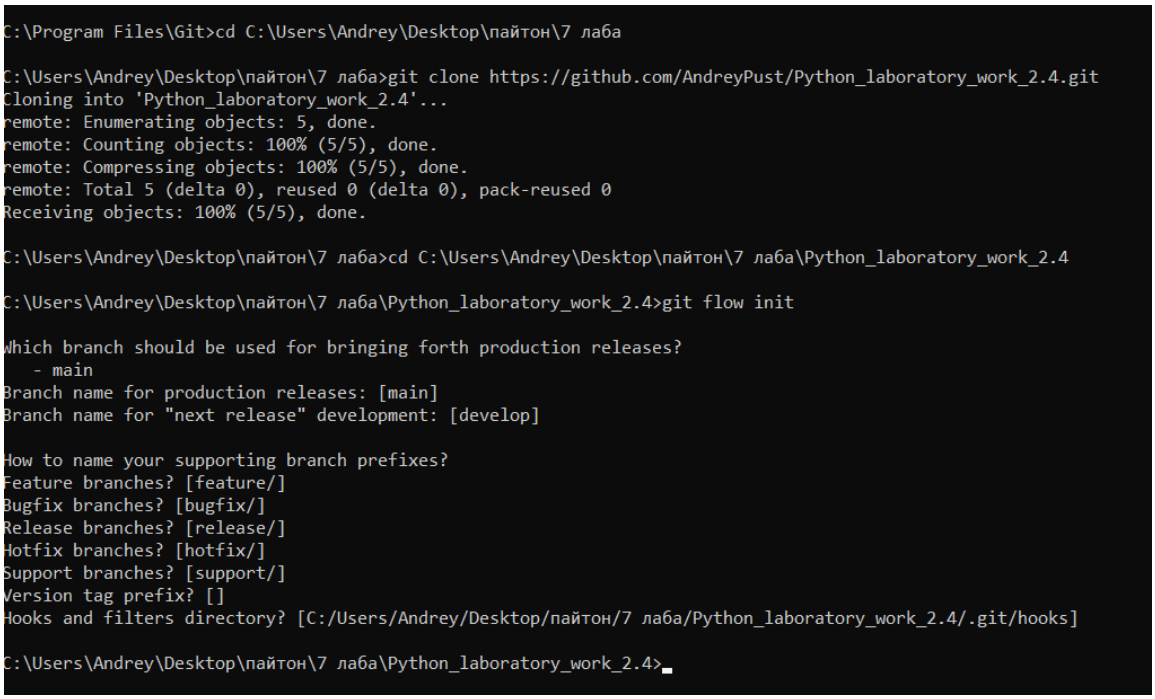

Рисунок 1 – Клонирование репозитория.

Проработка примеров лабораторной работы:

Пример 1.

Необходимо ввести список, состоящий из 10-ти элементов, найти сумму этих элементов меньших по модулю 5 и вывести ее на экран.

Код программы примера №1 и результаты работы программы с различными элементами списка (рис. 2, 3).

```python
import sys


if __name__ == '__main__':
    # Ввести список одной строкой.
    A = list(map(int, input("Введите элементы:").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 2 – Код программы примера №1.

```
C:\Users\Andrey\AppData\Local\Programs\Python\Pyt
Введите элементы:1 2 3 4 5 6 7 8 9 10
10
```

```
C:\Users\Andrey\AppData\Local\Programs\Python\Python39\
Введите элементы:1 -23 -4 -4 -6 -2 5 7 10 2
-7
```

```
C:\Users\Andrey\AppData\Local\Programs\Python\Python39\p
Введите элементы:1 2 3
Неверный размер списка

Process finished with exit code 1
```

Рисунок 3 – Результаты работы программы примера №1.

Код программы примера №1 и результаты работы программы с использование списковых включений (рис. 4, 5).

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

import sys


if __name__ == '__main__':
    # Ввести список одной строкой.
    A = list(map(int, input("Введите элементы: ").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)
```

Рисунок 3 – Код программы примера №1 со списковыми включениями.

```
C:\Users\Andrey\AppData\Local\Programs\Python\Python39\
Введите элементы: 1 2 3 4 5 6 7 8 9 10
10
```

```
C:\Users\Andrey\AppData\Local\Programs\Python\Python39\pyt
Введите элементы: 9 -7 -2 0 -2 -5 -8 80 90 -100
-4

Process finished with exit code 0
```

```
C:\Users\Andrey\AppData\Local\Programs\Python\Python39\p
Введите элементы: 1 2 3
Неверный размер списка

Process finished with exit code 1
```

Рисунок 4 – Результаты работы программы примера №1 со списковыми включениями.

Пример 2.

Необходимо написать программу, которая для целочисленного списка определяет, сколько положительных элементов располагается между его максимальным и минимальным элементами.

Код программы примера №2 и результаты работы программы с различными исходными данными (рис. 5, 6).

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

import sys


if __name__ == '__main__':
    # Ввести список одной строкой.
    a = list(map(int, input("Введите элементы: ").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 – Код программы примера №2.

```
C:\Users\Andrey\AppData\Local\Programs\Python\Python39\python.
Введите элементы: 1 2 3 4 5 6 7 8 9 10
8


Process finished with exit code 0
```

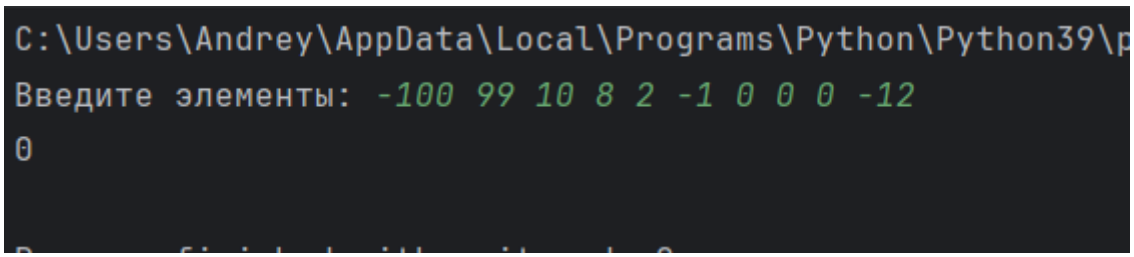

Рисунок 5 – Результаты работы программы примера №2.

Выполнение индивидуальных заданий:

Задание 1.

Необходимо написать программу, которая из списка целых чисел составляет три других списка. В первый записать числа, кратные 5, во второй – числа, кратные 7, а в третий – оставшиеся числа (Вариант 26).

Код программы индивидуального задания №1 и результаты работы программы с различными входными данными (рис. 6, 7).

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

# Необходимо написать программу, которая из списка целых чисел составляет
# три других списка. В первый записать числа, кратные 5, во второй - числа,
# кратные 7, а в третий - оставшиеся числа (Вариант 26).

if __name__ == '__main__':
    int_list = list(map(int, input("Введите список целых чисел: ").split()))
    multiple_five = []
    multiple_seven = []
    remaining_list = []

    # Создадим из певого списка трое новых
    for i in int_list:
        if i % 5 == 0:
            multiple_five += [i]
        if i % 7 == 0:
            multiple_seven += [i]
        if i % 5 != 0:
            if i % 7 != 0:
                remaining_list += [i]

    print("Числа, кратные 5:", multiple_five)
    print("Числа, кратные 7:", multiple_seven)
    print("Оставшиеся числа:", remaining_list)
```

Рисунок 6 – Код программы индивидуального задания №1.

```
C:\Users\Andrey\AppData\Local\Programs\Python\Python39\python.exe "C:\
Введите список целых чисел: 5 7 70 1 11
Числа, кратные 5: [5, 70]
Числа, кратные 7: [7, 70]
Оставшиеся числа: [1, 11]
```

```
C:\Users\Andrey\AppData\Local\Programs\Python\Python39\python.exe "C:\Users\Andrey\D
Введите список целых чисел: 11 372 9137 1271 2847 215 31386 31937 0 631 215 210
Числа, кратные 5: [215, 0, 215, 210]
Числа, кратные 7: [0, 210]
Оставшиеся числа: [11, 372, 9137, 1271, 2847, 31386, 31937, 631]
```

Рисунок 7 – Результаты работы программы индивидуального задания №1.

Код программы индивидуального задания №1 с использованием списковых включений и результаты работы программы с различными входными данными (рис. 8, 9).

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

# Необходимо написать программу, которая из списка целых чисел составляет
# три других списка. В первый записать числа, кратные 5, во второй - числа,
# кратные 7, а в третий - оставшиеся числа (Вариант 26).

if __name__ == '__main__':
    int_list = list(map(int, input("Введите список целых чисел: ").split()))
    multiple_five = [i for i in int_list if i % 5 == 0]
    multiple_seven = [i for i in int_list if i % 7 == 0]
    remaining_list = [i for i in int_list if i % 5 != 0 if i % 7 != 0]

    print("Числа, кратные 5:", multiple_five)
    print("Числа, кратные 7:", multiple_seven)
    print("Оставшиеся числа:", remaining_list)
```

Рисунок 8 – Код программы индивидуального задания №1 со списковыми включениями.

```
C:\Users\Andrey\AppData\Local\Programs\Python\Python39\py
Введите список целых чисел: 5 7 70 99
Числа, кратные 5: [5, 70]
Числа, кратные 7: [7, 70]
Оставшиеся числа: [99]
```

```
C:\Users\Andrey\AppData\Local\Programs\Python\Python39\python.exe "C:\Users\Andrey\Desktop\n
Введите список целых чисел: 23 3726 3263 21 2390 23 0 2863 2357 236 34 16 34 15 2876 0 9
Числа, кратные 5: [2390, 0, 15, 0]
Числа, кратные 7: [21, 0, 2863, 0]
Оставшиеся числа: [23, 3726, 3263, 23, 2357, 236, 34, 16, 34, 2876, 9]
```

Рисунок 9 – Результаты работы программы индивидуального задания №1 с использованием списковых включений.

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.

Преобразовать список таким образом, чтобы сначала располагались все элементы, модуль которых не превышает 1, а потом - все остальные (допустимо использовать метод «sort» с параметром «key») (Вариант 26 (7)).

Код программы индивидуального задания №2 и результаты работы программы с различными входными данными (рис. 10, 11).

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

# В списке, состоящем из вещественных элементов, вычислить:
# 1. номер минимального элемента списка;
# 2. сумму элементов списка, расположенных между первым
# и вторым отрицательными элементами.
# Преобразовать список таким образом, чтобы сначала располагались
# все элементы, модуль которых не превышает 1, а потом - все остальные.

if __name__ == '__main__':
    float_list = list(map(float, input("Введите список целых чисел:
    «).split()))

    # Отсортируем список согласно условию задачи
    abs_list = []
    more_list = []
    for i in float_list:
```

```
        if abs(i) < 1:
            abs_list.append(i)
        else:
            more_list.append(i)
    float_list = abs_list + more_list
    print(«Отсортированный список :», float_list)

    # Вычислим номер минимального элемента списка
    num_min = float_list.index(min(float_list))
    print(«Номер минимального элемента списка: «, num_min+1)

    # Вычислим сумму элементов списка между
    # первыми отрицательными элементами
    first = False
    sum_num = 0
    for i in float_list:
        if i < 0:
            first = True
        if first:
            if float_list[float_list.index(i) + 1] < 0:
                first = False
                break
        if first:
            sum_num += float_list[float_list.index(i) + 1]

    print(«Сумма элементов между первыми отрицательными числами: «, sum_num)
```

Рисунок 9 – Код программы индивидуального задания №2.

```
C:\Users\Andrey\AppData\Local\Programs\Python\Python39\python.exe "C:\Users\Andrey\Desktop\пайтон\7 л
Введите список целых чисел: 1.2 1.3 -2 10 20 30 -5 80 -99 -0.1 0.1 0.5 -0.7
Отсортированный список : [-0.1, 0.1, 0.5, -0.7, 1.2, 1.3, -2.0, 10.0, 20.0, 30.0, -5.0, 80.0, -99.0]
Номер минимального элемента списка:  13
Сумма элементов между первыми отрицательными числами:  0.6

Process finished with exit code 0
```

```
C:\Users\Andrey\AppData\Local\Programs\Python\Python39\python.exe "C:\Use
Введите список целых чисел: 0.1 -3.3 10 111 222 -8 0.3 0.6
Отсортированный список : [0.1, 0.3, 0.6, -3.3, 10.0, 111.0, 222.0, -8.0]
Номер минимального элемента списка:  8
Сумма элементов между первыми отрицательными числами:  343.0
```

Рисунок 9 – Результаты работы программы индивидуального задания №2.

Ответы на контрольные вопросы:

1. Списки – 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. Для создания списка нужно заключить элементы в квадратные скобки, либо ввести элементы списка с клавиатуры.

3. 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 отличии от таких типов данных как число или строка, содержимое «контейнера» списка можно менять.

4. Перебрать элементы списка можно с помощью цикла «for» и оператора «in», например: «for i in list». Также, можно проходить по индексам списка: for i in range(0, len(list)): list[i].

5. Списки можно объединять (list_1 + list_2) и повторять (list * число повторений).

6. Проверить есть ли конкретный элемент в списке можно, используя оператор «in».

7. Функция count() возвращает число повторений требуемого элемента в переданном списке.

8. Метод insert(индекс, «элемент») позволяет вставить требуемый элемент в список по указанному индексу, сдвигая последующие элементы вправо. Если нужно просто добавить элемент в конец списка, то нужен метод append().

9. Для сортировки списка есть метод «sort()», которому можно передать параметр «key» (по какому принципу будет идти сортировка), и параметр «reverse» (Если True, то список будет задом-наперёд).

10. Метод «pop(индекс)» позволяет удалить элемент в списке. Очистка списка – метод «clear()».

11. Списковое включение является частью синтаксиса языка, которая предоставляет простой способ построения списков. Благодаря нему не нужно писать отдельные циклы для заполнения списков, достаточно организовать перебор элементов используя списковые включения: a = [i for i in range(n)].

12. Слайсы (срезы) являются 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. Для работы со списками Python предоставляет следующие функции: len(list) - получить число элементов в списке list; min(list) - получить минимальный элемент списка list; max(list) - получить максимальный элемент списка list ; sum(list) - получить сумму элементов списка list , если список list содержит только числовые значения.

14. Для создания копии списка есть метод «copy». Также, можно использовать оператор среза. Для создания копии списка нельзя просто присвоить один список другому, в таком случае один список будет ссылаться на другой список.

15. Функция «sorted(list)» создаёт новый отсортированный список, не меняя старый, в то время как «sort» изменяет сам переданный ей список.

Вывод: в ходе лабораторной работы была изучена работа со списками в языке программирования Python 3. Были изучены способы создания списка, особенности хранения списка в памяти, способы работы с индексами списков, способы обращения к элементам списка, основные арифметические операции над списками, способы поиска элемента в списке, способы сортировки элементов в списке, способы изменения списков. Также была изучена такая особенность списков как псевдонимы(алиасинг).