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»

Выполнил:
Кожуховский Виктор Андреевич
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 Роман Александрович

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 г.

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.

Методика и порядок выполнения работы

1. Изучил теоретический материал работы.

2.Создал общедоступный репозиторий на GitHub, в котором использована лицензия MIT и язык программирования Python.

3. Выполнил клонирование созданного репозитория.

4. Дополнил файл .gitignore необходимыми правилами для работы с IDE PyCharm.

5. Организовал свой репозиторий в соответствие с моделью ветвления git-flow.

6. Создал проект в папке репозитория.

7. Проработал примеры лабораторной работы. Создал для каждого примера отдельный модуль языка Python. Зафиксировал изменения в репозитории.

```python
#!/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)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS

PS C:\Users\viktor> & "C:/Program Files/Python311/python.exe" "c:/Us
1 6 8 3 4 7 12 8 4 2
14
```

Рисунок 1. Код и выполнение примера 1

```python
#!/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)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS

PS C:\Users\viktor> & "C:/Program Files/Python311/python.exe" "c:/Us
1 6 8 3 4 7 12 8 4 2
14
```

Рисунок 2. Код и выполнение примера 1 с другой реализацией

```python
#!/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)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS

PS C:\Users\viktor> & "C:/Program Files/Python311/python.exe" "c:/Users/vikt
1 5 3 1 7 9 5 6
4
```

Рисунок 3. Код и выполнение примера 2

8. Выполнил индивидуальные задания, согласно своему варианту.

Вариант 16

Ввести список A из 10 элементов, найти сумму элементов кратных 2, их количество и вывести результаты на экран.

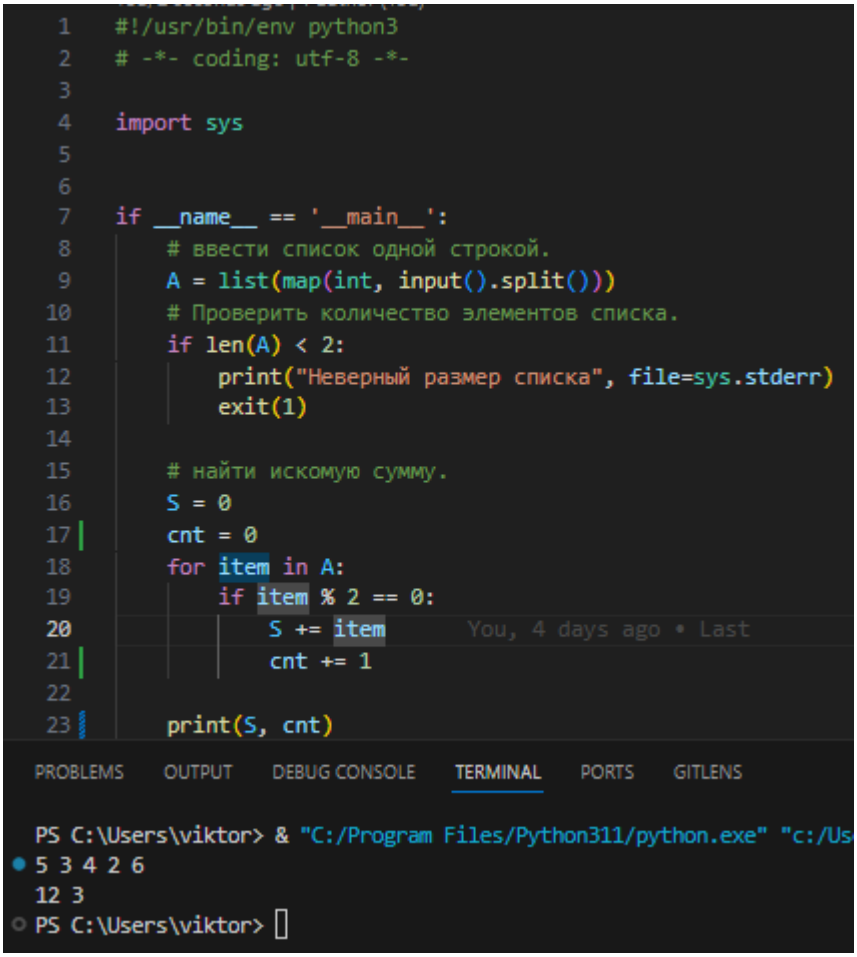

Рисунок 4. Код и выполнение индивидуального задания 1

В списке, состоящем из вещественных элементов, вычислить:

1. количество положительных элементов списка;

2. сумму элементов списка, расположенных после последнего элемента, равного нулю.

Преобразовать список таким образом, чтобы сначала располагались все элементы, целая часть которых не превышает 1, а потом - все остальные.

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

import sys


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

    count = 0
    # Определить индексы минимального и максимального элементов.
    for i, item in enumerate(a):
        if item == 0:
            i_zero = i
        if item > 0:
            count += 1
        if int(item) <= 1:
            poped = a_modified.pop(i)
            a_modified.insert(0, poped)

    a_cut = a[i_zero + 1:]
    S = 0
    for item in a_cut:
        S += item

    print("Количество положительных элементов списка: ", count)
    print("Сумма элементов списка, расположенных после последнего элемента, равного нулю: ", S)
    print("Преобразованный список, где сначала располагаются все элементы, целая часть которых не превышает 1, а потом - все остальные: ", a_modified)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS                                Python

PS C:\Users\viktor> & "C:/Program Files/Python311/python.exe" "c:/Users/viktor/Desktop/скфу/python/gitstuff/pylab7/indiv 2.py"
1.2 4.5 8.9 5.6 7.8 0 53.3 4.2 1.3 1.5 0.3 0 0.4 0.9 8.9
Количество положительных элементов списка:  13
Сумма элементов списка, расположенных после последнего элемента, равного нулю:  66.4
Преобразованный список, где сначала располагаются все элементы, целая часть которых не превышает 1, а потом - все остальные:  [0.9, 0.4, 0.0, 0.3, 1.5, 1.3, 0.0, 1.2, 4.5, 8.9, 5.6, 7.8, 53.3, 4.2, 8.9]
```

Рисунок 5. Код и выполнение индивидуального задания 2

10. Зафиксировал сделанные изменения в репозитории.

11. Добавил отчет по лабораторной работе в формате PDF в папку doc репозитория. Зафиксировал изменения.

12. Выполнил слияние ветки для разработки с веткой main / master.

13. Отправил сделанные изменения на сервер GitHub.

Вопросы для защиты работы

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]

3. Как организовано хранение списков в оперативной памяти?

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. Каким образом можно перебрать все элементы 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 insert можно использовать, чтобы вставить элемент в список:

```
my_list = [1, 2, 3, 4, 5]
my_list.insert(1,  'Привет')
print(my_list)
```

Метод append можно использовать для добавления элемента в список:

```
my_list = ['один', 'два', 'три', 'четыре', 'пять']
my_list.append('ещё один')
print(my_list)
```

9. Как выполнить сортировку списка?

Для сортировки списка нужно использовать метод sort.

```
my_list = ['cde', 'fgh', 'abc', 'klm', 'opq']
list_2 = [3, 5, 2, 4, 1]
my_list.sort()
list_2.sort()
print(my_list)
print(list_2)
```

10. Как удалить один или несколько элементов из списка?

Удалить элемент можно, написав его индекс в методе pop:

```
my_list = ['один', 'два', 'три', 'четыре', 'пять']
removed = my_list.pop(2)
```

Элемент можно удалить с помощью метода remove.

```
my_list = ['один', 'два', 'три', 'четыре', 'пять']
my_list.remove('два')
```

Оператор del можно использовать для тех же целей:

```
my_list = ['один', 'два', 'три', 'четыре', 'пять']
del my_list[2]
```

Можно удалить несколько элементов с помощью оператора среза:

```
my_list = ['один', 'два', 'три', 'четыре', 'пять']
del my_list[1:3]
```

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.

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.

Пример с заменой функции map.

```
>>> a = [1, 2, 3, 4, 5, 6, 7]
>>> b = [i**2 for i in a]
>>> print('a = {}\nb = {}'.format(a, b))
a = [1, 2, 3, 4, 5, 6, 7]
b = [1, 4, 9, 16, 25, 36, 49]
```

Пример с заменой функции filter.

```
>>> a = [1, 2, 3, 4, 5, 6, 7]
>>> b = [i for i in a if i % 2 == 0]
>>> print('a = {}\nb = {}'.format(a, b))
a = [1, 2, 3, 4, 5, 6, 7]
b = [2, 4, 6]
```

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

Слайс задается тройкой чисел, разделенных двоеточием: 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.

```
>>> # Получить копию списка
>>> a[:]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> # Получить первые пять элементов списка
>>> a[0:5]
[0, 1, 2, 3, 4]
>>> # Получить элементы с 3-го по 7-ой
>>> a[2:7]
[2, 3, 4, 5, 6]
>>> # Взять из списка элементы с шагом 2
```

```
>>> a[::2]
[0, 2, 4, 6, 8]
>>> # Взять из списка элементы со 2-го по 8-ой с шагом 2
>>> a[1:8:2]
[1, 3, 5, 7]
```

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.

```
>>> a = [1, 2, 3, 4, 5]
>>> b = a.copy()
>>> b
[1, 2, 3, 4, 5]
>>> a == b
True
>>> a is b
False
>>> a is not b
True
```

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

Функция sorted в Python используется для получения нового отсортированного списка из элементов итерируемого объекта.

sorted(iterable, *, key=None, reverse=False)

Параметры:

iterable: Итерируемый объект, который вы хотите отсортировать.

key: (Необязательный) Одноаргументная функция, которая будет использоваться для извлечения ключа сравнения из каждого элемента в iterable. По умолчанию ключом является сам элемент.

reverse: (Необязательный) Булевое значение. Если True, список сортируется в обратном (убывающем) порядке. По умолчанию False (сортировка в возрастающем порядке).

Возвращаемое значение:

Функция возвращает новый отсортированный список, содержащий все элементы из iterable.

Отличия:

1. **Возвращаемое значение:**

sorted: Возвращает новый отсортированный список, не изменяя исходный итерируемый объект.

sort: Изменяет исходный список.

2. **Универсальность:**

sorted: Может принимать любой итерируемый объект, например списки, кортежи, словари, строки и т. д.

sort: Предназначен только для использования с объектами списка.

3. **Характер использования:**

sorted: Более универсальный, использует функциональный подход и полезен, когда сортировка нужна без изменения оригинала.

sort: Используется для внутренней сортировки списка; рассматривается как процедурный или объектно-ориентированный подход.