

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Кафедра инфокоммуникаций

**Отчет по лабораторной работе № 2.4
по дисциплине «Основы программной инженерии»**

Выполнил студент группы ПИЖ-б-о-22-1

Душин Александр Владимирович.

Подпись студента _____

Работа защищена « » _____ 20__ г.

Проверил Воронкин Р.А. _____
(подпись)

Ставрополь 2023

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.

Ход выполнения работы:


1. Создать общедоступный репозиторий на GitHub с использованием лицензии MIT и язык программирования Python:

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk ().*

Owner *

 MrPlatynum ▾

Repository name *

/ ProgrammEngineering7

✔ ProgrammEngineering7 is available.

Great repository names are short and memorable. Need inspiration? How about **animated-rotary-phone** ?

Description (optional)



Public

Anyone on the internet can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.

Initialize this repository with:



Add a README file

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: Python ▾

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: MIT License ▾

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)



You are creating a public repository in your personal account.

Create repository

Рисунок 1 – Создание общедоступного репозитория на GitHub с заданными настройками

```
Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents
$ git clone https://github.com/MrPlatynum/ProgrammEngineering7.git
Cloning into 'ProgrammEngineering7'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.
```

Рисунок 2 – Клонирование созданного репозитория на локальный компьютер



Рисунок 3 – файл .gitignore

```
Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering7 (main)
$ git checkout -b develop
Switched to a new branch 'develop'

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering7 (develop)
$
```

Рисунок 4 – организация репозитория в соответствии с моделью ветвления git flow

2. Проработать примеры лабораторной работы, оформляя код согласно PEP-8:

```
example1.py x
1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import sys
5
6 ▶ if __name__ == '__main__':
7     # Ввести список одной строкой.
8     A = list(map(int, input().split()))
9
10    # Проверить количество элементов списка.
11    if len(A) != 10:
12        print("Неверный размер списка", file=sys.stderr)
13        exit(1)
14
15    # Найти искомую сумму.
16    s = 0
17    for item in A:
18        if abs(item) < 5:
19            s += item
20
21    print(s)
22
```

Рисунок 5 – Сумма элементов, меньших по модулю 5 (задание №1)

```
"C:\Program Files\Python312\python.exe" C:/Users/Alexander/Documents/ProgrammEngineering7/example1.py
-4 2 3 4 5 6 1 2 3 6 7
10
```

Рисунок 6 – Вывод программы (задание №1)

```

example2.py x
1  ▶  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6
7  ▶  if __name__ == '__main__':
8      # Ввести список одной строкой.
9      a = list(map(int, input().split()))
10     # Если список пуст, завершить программу.
11     if not a:
12         print("Заданный список пуст", file=sys.stderr)
13         exit(1)
14
15     # Определить индексы минимального и максимального элементов.
16     a_min = a_max = a[0]
17     i_min = i_max = 0
18     for i, item in enumerate(a):
19         if item < a_min:
20             i_min, a_min = i, item
21         if item >= a_max:
22             i_max, a_max = i, item
23
24     # Проверить индексы и обменять их местами.
25     if i_min > i_max:
26         i_min, i_max = i_max, i_min
27
28     # Посчитать количество положительных элементов.
29     count = 0
30     for item in a[i_min + 1:i_max]:
31         if item > 0:
32             count += 1
33
34     print(count)
35

```

Рисунок 7 – Количество положительных элементов (задание №2)

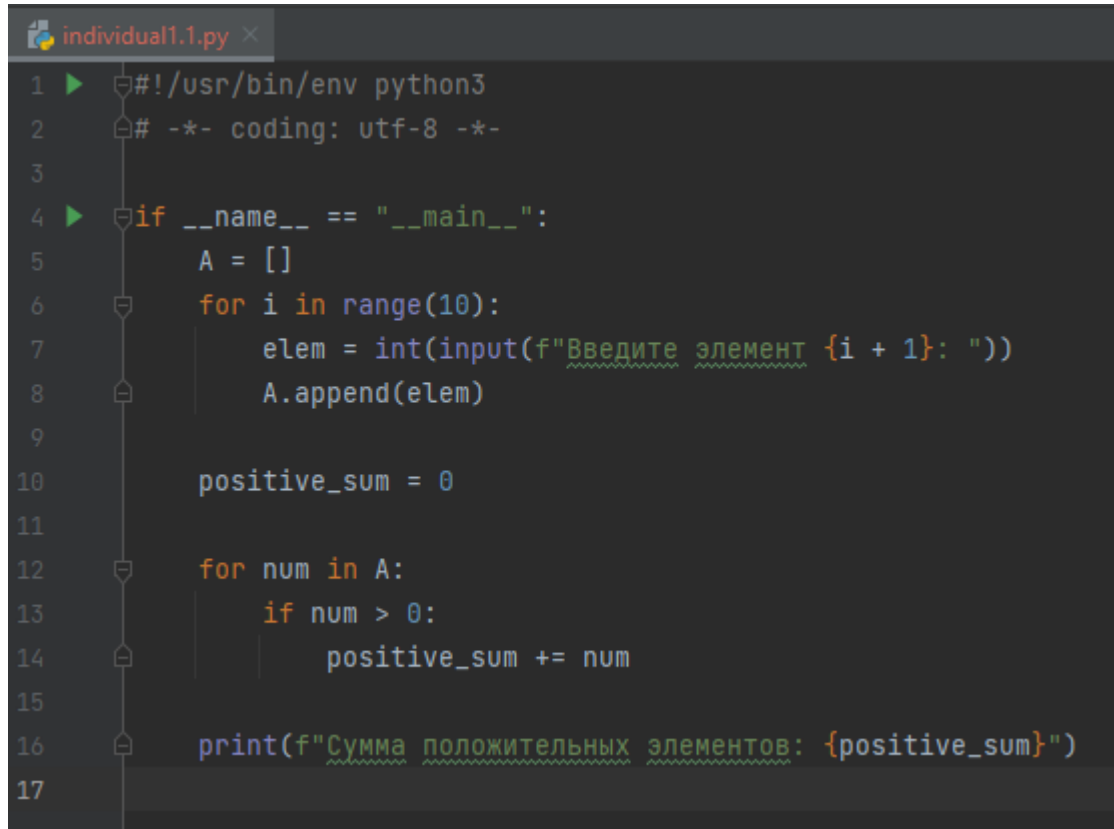
```

"C:\Program Files\Python312\python.exe" C:/Users/Alexander/Documents/ProgrammEngineering7/example2.py
1 -4 2 4 5 -2 66 4
3

```

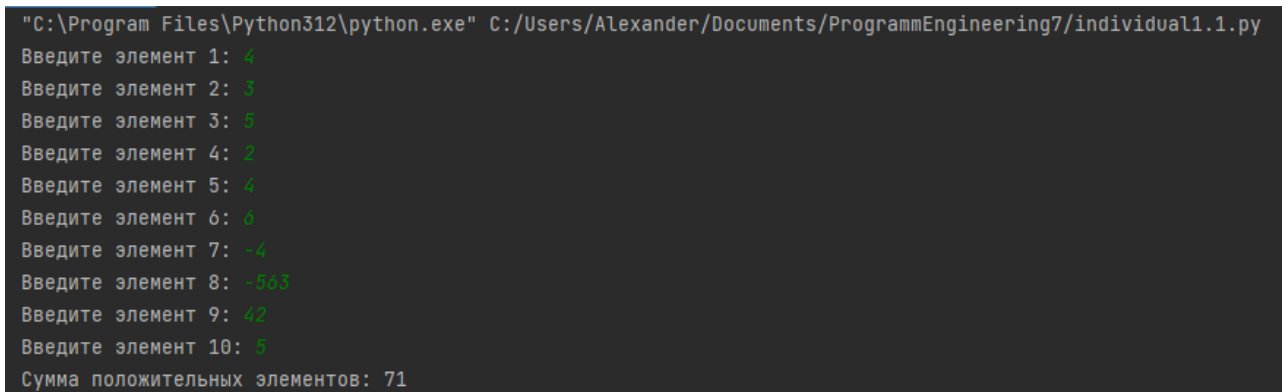
Рисунок 8 – Вывод программы (задание №2)

3. Выполним индивидуальные задания:



```
individual1.1.py x
1  ▶  #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3
4  ▶  if __name__ == "__main__":
5      A = []
6      for i in range(10):
7          elem = int(input(f"Введите элемент {i + 1}: "))
8          A.append(elem)
9
10     positive_sum = 0
11
12     for num in A:
13         if num > 0:
14             positive_sum += num
15
16     print(f"Сумма положительных элементов: {positive_sum}")
17
```

Рисунок 9 – Сумма положительных элементов (задание №1)



```
"C:\Program Files\Python312\python.exe" C:/Users/Alexander/Documents/ProgrammEngineering7/individual1.1.py
Введите элемент 1: 4
Введите элемент 2: 3
Введите элемент 3: 5
Введите элемент 4: 2
Введите элемент 5: 4
Введите элемент 6: 6
Введите элемент 7: -4
Введите элемент 8: -503
Введите элемент 9: 42
Введите элемент 10: 5
Сумма положительных элементов: 71
```

Рисунок 10 – Вывод программы (задание №1)

```
individual1.2.py x
1 ▶ #!/usr/bin/env python3
2   # -*- coding: utf-8 -*-
3
4 ▶ if __name__ == "__main__":
5     A = [int(input(f"Введите элемент {i + 1}: ")) for i in range(10)]
6
7     # Вычисляем сумму положительных элементов с использованием List Comprehensions
8     positive_sum = sum(num for num in A if num > 0)
9
10    print(f"Сумма положительных элементов: {positive_sum}")
11
12
13
14
```

Рисунок 11 – Сумма положительных элементов через с использованием List Comprehensions (задание №1)

```
"C:\Program Files\Python312\python.exe" C:/Users/Alexander/Documents/ProgrammEngineering7/individual1.2.py
Введите элемент 1: 4
Введите элемент 2: 3
Введите элемент 3: 5
Введите элемент 4: 2
Введите элемент 5: 4
Введите элемент 6: 6
Введите элемент 7: -4
Введите элемент 8: -503
Введите элемент 9: 42
Введите элемент 10: 5
Сумма положительных элементов: 71
```

Рисунок 12 – Вывод программы (задание №1)

6. В списке, состоящем из целых элементов, вычислить:

1. номер максимального элемента списка;
2. произведение элементов списка, расположенных между первым и вторым нулевыми элементами.

Преобразовать список таким образом, чтобы в первой его половине располагались элементы, стоявшие в нечетных позициях, а во второй половине - элементы, стоявшие в четных позициях.

```
individual2.py x
1 ▶ #!/usr/bin/env python3
2   # -*- coding: utf-8 -*-
3
4 ▶ if __name__ == "__main__":
5     numbers = list(map(int, input("Enter a list of integer elements separated by space: ").split()))
6
7     max_index = numbers.index(max(numbers))
8
9     # 2. Find the product of elements between the first and second zero elements.
10    if 0 in numbers:
11        first_zero_index = numbers.index(0)
12    else:
13        first_zero_index = -1
14    if first_zero_index != -1:
15        second_zero_index = numbers.index(0, first_zero_index + 1)
16    else:
17        second_zero_index = -1
18    product_between_zeros = 1
19
20    if first_zero_index != -1 and second_zero_index != -1:
21        for num in numbers[first_zero_index + 1:second_zero_index]:
22            product_between_zeros *= num
23
24    # Transform the list.
25    transformed_list = numbers[1::2] + numbers[0::2]
26
27    print("Index of the maximum element:", max_index)
28    print("Product of elements between the first and second zero elements:", product_between_zeros)
29    print("Transformed list:", transformed_list)
30
```

Рисунок 13 – Выполнение задания в соответствии с требованиями
(задание №2)

```
"C:\Program Files\Python312\python.exe" C:/Users/Alexander/Documents/ProgrammEngineering7/individual2.py
Enter a list of integer elements separated by space: 1 2 3 4 0 4 5 3 4 0 2 4
Index of the maximum element: 6
Product of elements between the first and second zero elements: 240
Transformed list: [2, 4, 4, 3, 0, 4, 1, 3, 0, 5, 4, 2]
```

Рисунок 14 – Запуск программы (задание №2)

4. Зафиксируем проделанные изменения, сольем ветки и отправим на удаленный репозиторий:


```

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering7 (develop)
$ git log
commit 8887d384bc3d0fb7b618beb7da8e44d26be2501f (HEAD -> develop)
Author: MrPlatynum <dushinsasha4@gmail.com>
Date: Sat Nov 25 17:11:24 2023 +0300

    финальные изменения

commit 55d04a89a3ca2877ac066cfca2c5b27c15bc526e (origin/main, origin/HEAD, main)
Author: MrPlatynum <71084177+MrPlatynum@users.noreply.github.com>
Date: Fri Nov 24 23:38:13 2023 +0300

    Initial commit

```

Рисунок 15 – Коммиты ветки develop во время выполнения лабораторной работы

```

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering7 (develop)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering7 (main)
$ git merge develop
Updating 55d04a8..8887d38
Fast-forward
 .idea/.gitignore      | 8 ++++++
 .idea/.name           | 1 +
 .idea/ProgrammEngineering7.iml | 8 ++++++
 .idea/inspectionProfiles/profiles_settings.xml | 6 +++++
 .idea/misc.xml        | 4 +++
 .idea/modules.xml     | 8 ++++++
 .idea/vcs.xml         | 6 +++++
 example1.py           | 21 +++++++++++++++++++++
 example2.py           | 34 +++++++++++++++++++++++++++++++++++++
 individual1.1.py       | 16 ++++++++
 individual1.2.py       | 13 ++++++++
 individual2.py         | 29 +++++++++++++++++++++
12 files changed, 154 insertions(+)
create mode 100644 .idea/.gitignore
create mode 100644 .idea/.name
create mode 100644 .idea/ProgrammEngineering7.iml
create mode 100644 .idea/inspectionProfiles/profiles_settings.xml
create mode 100644 .idea/misc.xml
create mode 100644 .idea/modules.xml
create mode 100644 .idea/vcs.xml
create mode 100644 example1.py
create mode 100644 example2.py
create mode 100644 individual1.1.py
create mode 100644 individual1.2.py
create mode 100644 individual2.py

```

Рисунок 16 – Слияние ветки develop в ветку main

```
Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering7 (main)
$ git push origin main
Enumerating objects: 17, done.
Counting objects: 100% (17/17), done.
Delta compression using up to 12 threads
Compressing objects: 100% (14/14), done.
Writing objects: 100% (16/16), 3.54 KiB | 3.54 MiB/s, done.
Total 16 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/MrPlatynum/ProgrammEngineering7.git
   55d04a8..8887d38  main -> main

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering7 (main)
$
```

Рисунок 17 – Отправка на удаленный репозиторий

Ответы на контрольные вопросы:

1. Что такое списки в языке Python?

Список в Python - это упорядоченная изменяемая коллекция объектов различных типов данных. Они могут содержать элементы любых типов и быть изменены после создания.

2. Как осуществляется создание списка в Python?

Список создается с помощью квадратных скобок [], в которых перечисляются элементы списка через запятую: `my_list = [1, 2, 3, 'a', 'b', 'c']`.

3. Как организовано хранение списков в оперативной памяти?

Списки в Python хранятся в виде массива указателей на объекты, что позволяет легко изменять их размер и содержимое.

4. Каким образом можно перебрать все элементы списка?

Элементы списка можно перебрать с помощью цикла for:

```
for element in my_list:
```

```
    # делайте что-то с элементом
```

5. Какие существуют арифметические операции со списками?

Списки поддерживают операции сложения (+) для конкатенации списков и умножения на число (*) для повторения списка.

6. Как проверить есть ли элемент в списке?

Используйте оператор in: `element in my_list` вернет True, если element содержится в my_list.

7. Как определить число вхождений заданного элемента в списке?

Метод count() позволяет узнать количество вхождений элемента в список: `my_list.count(element)`.

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

Для добавления элемента в конец списка используется метод append(): `my_list.append(new_element)`. Для вставки элемента по индексу используется метод insert(): `my_list.insert(index, element)`.

9. Как выполнить сортировку списка?

Метод sort() сортирует список на месте: `my_list.sort()`. Функция sorted()

возвращает новый отсортированный список: `sorted_list = sorted(my_list)`.

10. Как удалить один или несколько элементов из списка?

`del` оператор удаляет элемент по индексу: `del my_list[index]`. Метод `remove()` удаляет первое вхождение элемента: `my_list.remove(element)`. Метод `pop()` удаляет элемент по индексу и возвращает его: `my_list.pop(index)`.

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

Списковое включение - это компактный способ создания списка с помощью выражения в квадратных скобках: `[expression for item in iterable]`. Это позволяет применять выражение к каждому элементу итерируемого объекта.

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

Срезы позволяют получать подсписки из списка. Используются квадратные скобки и индексы: `my_list[start:stop:step]`.

13. Какие существуют функции агрегации для работы со списками?

Функции агрегации, такие как `sum()`, `max()`, `min()`, применяются к спискам для вычисления суммы элементов, максимального и минимального значения соответственно.

14. Как создать копию списка?

Чтобы создать копию списка, используйте срез: `new_list = my_list[:]` или метод `copy()`: `new_list = my_list.copy()`.

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

`sorted()` - это встроенная функция Python, которая возвращает новый отсортированный список из переданного итерируемого объекта, не изменяя исходный. `sort()` - метод списка, который сортирует список на месте, изменяя исходный список. Таким образом, различие между ними заключается в том, что `sorted()` не изменяет исходный список, в то время как `sort()` изменяет его напрямую.