

**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
«Работа со списками в языке Python»
по дисциплине «Основы программной инженерии»**

Выполнил студент группы

ПИЖ-б-о-21-1

Зиберов Александр

« » ноября 2022 г.

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.

Выполнение работы:

Создать общедоступный репозиторий на GitHub, в котором будет использована лицензия MIT, рисунок 1.

Ссылка: <https://github.com/afk552/lab2.4>

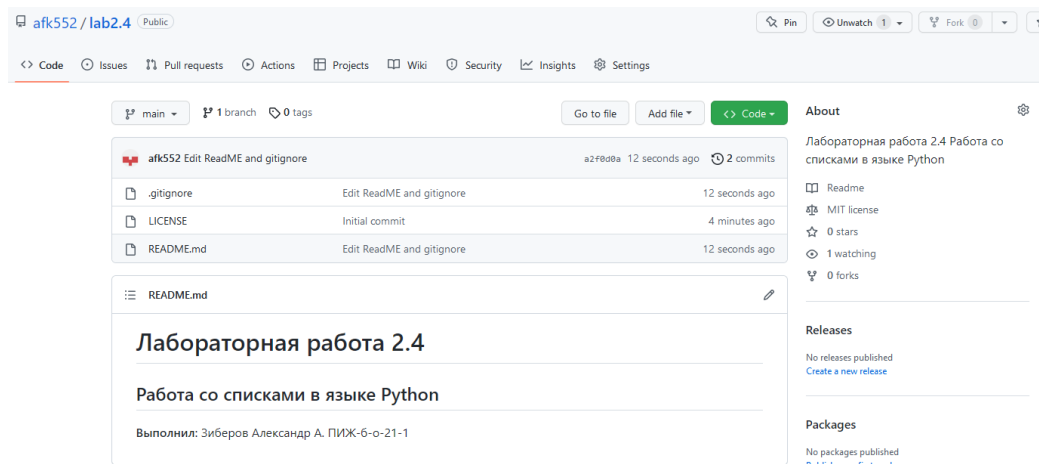


Рисунок 1 – Репозиторий GitHub

Дополните файл .gitignore необходимыми правилами для работы с IDE PyCharm, рисунок 2.

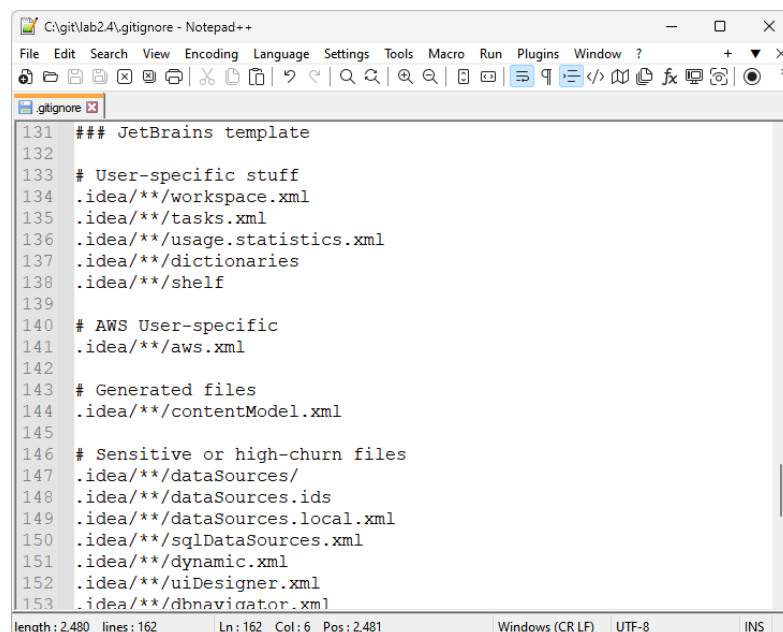


Рисунок 2 – Окно блокнота

Организируйте свой репозиторий в соответствии с моделью ветвления git-flow, рисунок 3.



```
C:\Windows\System32\cmd.exe

C:\git\lab2.4>git branch
* main

C:\git\lab2.4>git checkout -b develop
Switched to a new branch 'develop'

C:\git\lab2.4>git branch
* develop
  main

C:\git\lab2.4>git push origin develop
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'develop' on GitHub by visiting:
remote:   https://github.com/afk552/lab2.4/pull/new/develop
remote:
To https://github.com/afk552/lab2.4
 * [new branch]      develop -> develop

C:\git\lab2.4>
```

Рисунок 3 – Окно командной строки

Создайте проект PyCharm в папке репозитория, рисунок 4.

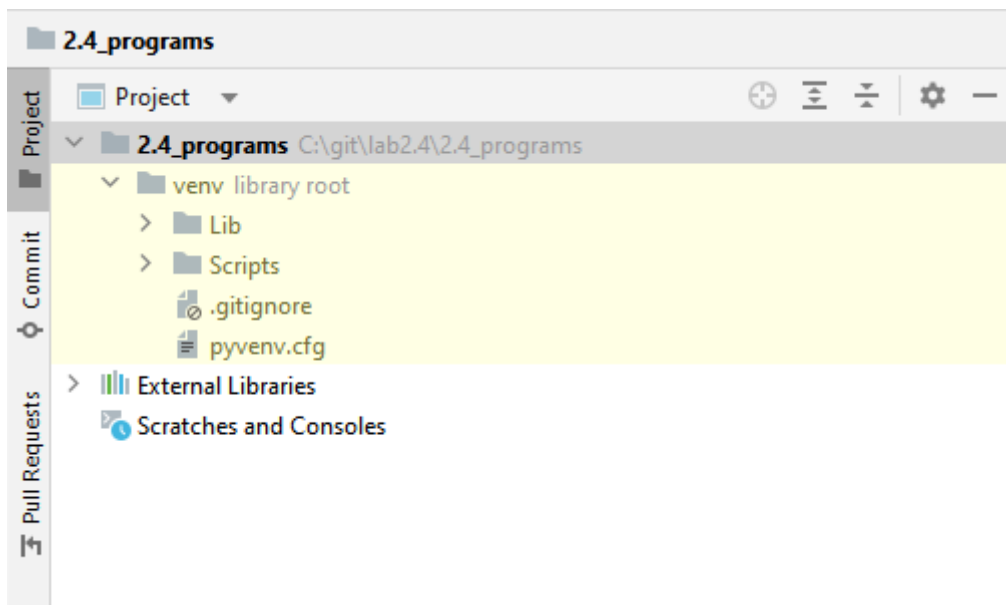


Рисунок 4 – Окно проекта в PyCharm

Проработайте примеры лабораторной работы. Создайте для каждого примера отдельный модуль языка Python. Зафиксируйте изменения в репозитории. Приведите в отчете скриншоты результатов выполнения каждой из программ примеров при различных исходных данных, вводимых с клавиатуры. Рисунки 5-11.

```

example1_cycle.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      # Проверить количество элементов списка.
10     if len(A) != 10:
11         print("Неверный размер списка", file=sys.stderr)
12         exit(1)
13
14     # Найти искомую сумму.
15     s = 0
16     for item in A:
17         if abs(item) < 5:
18             s += item
19     print(s)
20

```

Рисунок 5 – Код программы из примера 1 (через циклы)

```

Run: example1_cycle x
C:\git\lab2.4\2.4_programs\venv\Scripts\python.exe C:/git/lab2.4/2.4_programs/example1_cycle.py
1 2 3 4
Неверный размер списка
Process finished with exit code 1

Run: example1_cycle x
C:\git\lab2.4\2.4_programs\venv\Scripts\python.exe C:/git/lab2.4/2.4_programs/example1_cycle.py
1 2 3 4 4 3 2 1 9 10
20
Process finished with exit code 0

```

Рисунок 6 – Результат выполнения программы из примера 1 (через циклы)

```

example1_lc.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      # Проверить количество элементов списка
10     if len(A) != 10:
11         print("Неверный размер списка", file=sys.stderr)
12         exit(1)
13
14     # Найти искомую сумму
15     s = sum([a for a in A if abs(a) < 5])
16     print(s)
17

```

Рисунок 7 – Код программы из примера 1 (через списковое включение)

```
Run: example1_lc x
C:\git\lab2.4\2.4_programs\venv\Scripts\python.exe C:/git/lab2.4/2.4_programs/example1_lc.py
1 2 3 4
Неверный размер списка
Process finished with exit code 1

Run: example1_lc x
C:\git\lab2.4\2.4_programs\venv\Scripts\python.exe C:/git/lab2.4/2.4_programs/example1_lc.py
1 2 3 4 4 3 2 1 10 9
20
Process finished with exit code 0
```

Рисунок 8 – Результаты выполнения программы из примера 1 (через списковое включение)

```
example2.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      # Если список пуст, завершить программу.
10     if not a:
11         print("Заданный список пуст", file=sys.stderr)
12         exit(1)
13
14     # Определить индексы минимального и максимального элементов.
15     a_min = a_max = a[0]
16     i_min = i_max = 0
17     for i, item in enumerate(a):
18         if item < a_min:
19             i_min, a_min = i, item
20         if item >= a_max:
21             i_max, a_max = i, item
22
23     # Проверить индексы и обменять их местами
24     if i_min > i_max:
25         i_min, i_max = i_max, i_min
26
27     # Посчитать количество положительных элементов
28     count = 0
29     for item in a[i_min+1:i_max]:
30         if item > 0:
31             count += 1
32
33     print(count)
34
```

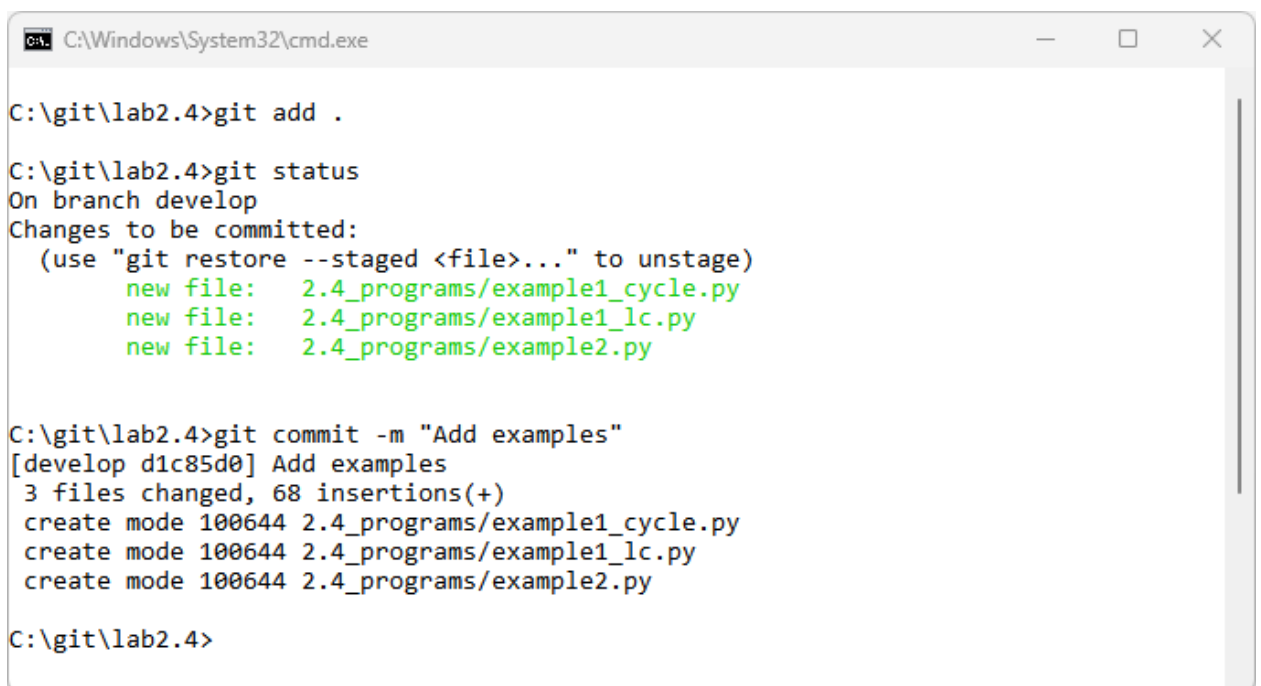
Рисунок 9 – Код программы из примера 2



```
Run: example2 x
C:\git\lab2.4\2.4_programs\venv\Scripts\python.exe C:/git/lab2.4/2.4_programs/example2.py
Заданный список пуст
Process finished with exit code 1

Run: example2 x
C:\git\lab2.4\2.4_programs\venv\Scripts\python.exe C:/git/lab2.4/2.4_programs/example2.py
200 1000 10 20 30 40 1 90
4
Process finished with exit code 0
```

Рисунок 10 – Результаты выполнения программы из примера 2



```
C:\Windows\System32\cmd.exe

C:\git\lab2.4>git add .

C:\git\lab2.4>git status
On branch develop
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   2.4_programs/example1_cycle.py
        new file:   2.4_programs/example1_lc.py
        new file:   2.4_programs/example2.py

C:\git\lab2.4>git commit -m "Add examples"
[develop d1c85d0] Add examples
 3 files changed, 68 insertions(+)
 create mode 100644 2.4_programs/example1_cycle.py
 create mode 100644 2.4_programs/example1_lc.py
 create mode 100644 2.4_programs/example2.py

C:\git\lab2.4>
```

Рисунок 11 – Результаты выполнения программы из примера 3

Выполните индивидуальные задания, согласно своего варианта.

Вариант 12

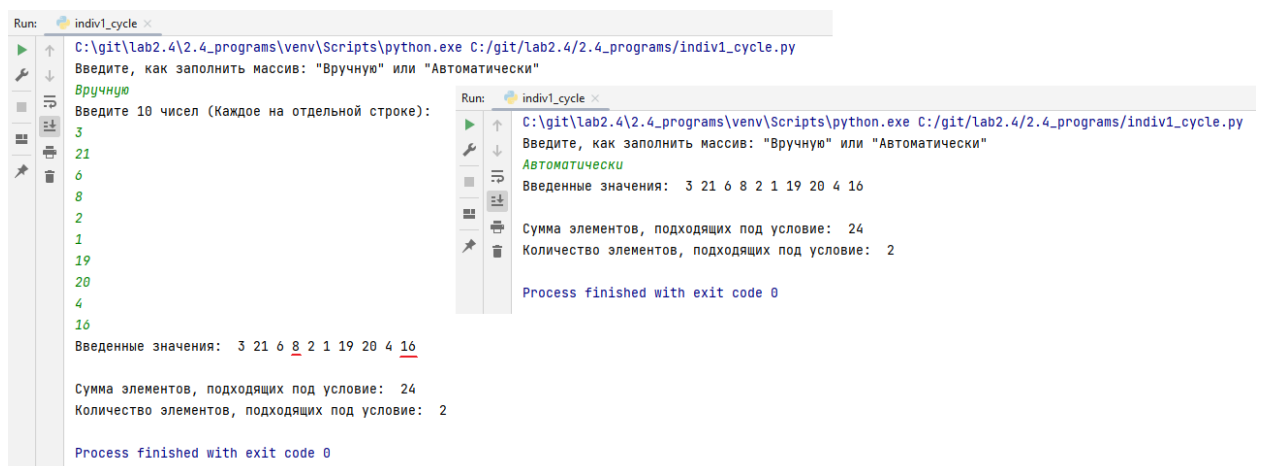
Задание 1. Составить программу с использованием одномерных массивов для решения задачи. Номер варианта необходимо получить у преподавателя. Решить индивидуальное задание как с использованием циклов, так и с использованием List Comprehensions. Рисунки 12-15.

```

1  ▶  #!/usr/bin/env python3
2  ▶  # -*- coding: utf-8 -*-
3
4  ▶  if __name__ == '__main__':
5      option = str(input(
6          "Введите, как заполнить список: \"Вручную\" или \"Автоматически\"\\n"))
7      A = []
8      if option.lower() == "вручную":
9          print("Введите 10 чисел (Каждое на отдельной строке): ")
10         for i in range(10):
11             A.append(int(input()))
12     else:
13         A = [3, 21, 6, 8, 2, 1, 19, 20, 4, 16]
14     print("Введенные значения: ", *A, "\\n")
15
16     summ = 0
17     cnt = 0
18     for i, elem in enumerate(A):
19         if 2 < A[i] < 20 and A[i] % 8 == 0:
20             summ += elem
21             cnt += 1
22             print(elem, end=" ")
23     print("\\nСумма элементов, подходящих под условие: ", summ)
24     print("Количество элементов, подходящих под условие: ", cnt)
25

```

Рисунок 12 – Код программы индивидуального задания 1 (через циклы)



```

Run: C:\git\lab2.4\2.4_programs\venv\Scripts\python.exe C:\git\lab2.4\2.4_programs\individ1_cycle.py
Введите, как заполнить массив: "Вручную" или "Автоматически"
Вручную
Введите 10 чисел (Каждое на отдельной строке):
3
21
6
8
2
1
19
20
4
16
Введенные значения: 3 21 6 8 2 1 19 20 4 16

Сумма элементов, подходящих под условие: 24
Количество элементов, подходящих под условие: 2

Process finished with exit code 0

```

Рисунок 13 – Результаты выполнения программы индивидуального задания 1 (через циклы)

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6  if __name__ == '__main__':
7      option = str(input(
8          "Введите, как заполнить список: \"Вручную\" или \"Автоматически\"\\n"))
9      A = []
10     if option.lower() == "вручную":
11         A = list(map(int, input("Введите 10 чисел через пробел: ").split()))
12     else:
13         A = [3, 21, 6, 8, 2, 1, 19, 20, 4, 16]
14     if not A:
15         print("Заданный список пуст", file=sys.stderr)
16         exit(1)
17     else:
18         print("Введенные значения: ", *A, "\\n")
19
20     cnt = 0
21     summ = 0
22     A = [x for x in A if x > 2 if x < 20 if x % 8 == 0]
23     print("Элементы, подходящие под условие: ", *A)
24     print("Сумма элементов, подходящие под условие: ", sum(A))
25     print("Количество элементов, подходящие под условие: ", len(A))
26

```

Рисунок 14 – Код программы индивидуального задания 1 (через списковое включение)

```

Run: indiv1_lc
C:\git\lab2.4\2.4_programs\venv\Scripts\python.exe C:/git/lab2.4/2.4_programs/indiv1_lc.py
Введите, как заполнить массив: "Вручную" или "Автоматически"
вручную
Введите 10 чисел через пробел:
Заданный список пуст
Process finished with exit code 1

Run: indiv1_lc
C:\git\lab2.4\2.4_programs\venv\Scripts\python.exe C:/git/lab2.4/2.4_programs/indiv1_lc.py
Введите, как заполнить массив: "Вручную" или "Автоматически"
вручную
Введите 10 чисел через пробел: 3 21 6 8 2 1 19 20 4 16
Введенные значения: 3 21 6 8 2 1 19 20 4 16

Элементы, подходящие под условие: 8 16
Сумма элементов, подходящие под условие: 24
Количество элементов, подходящие под условие: 2

Process finished with exit code 0

Run: indiv1_lc
C:\git\lab2.4\2.4_programs\venv\Scripts\python.exe C:/git/lab2.4/2.4_programs/indiv1_lc.py
Введите, как заполнить массив: "Вручную" или "Автоматически"
Автоматически
Введенные значения: 3 21 6 8 2 1 19 20 4 16

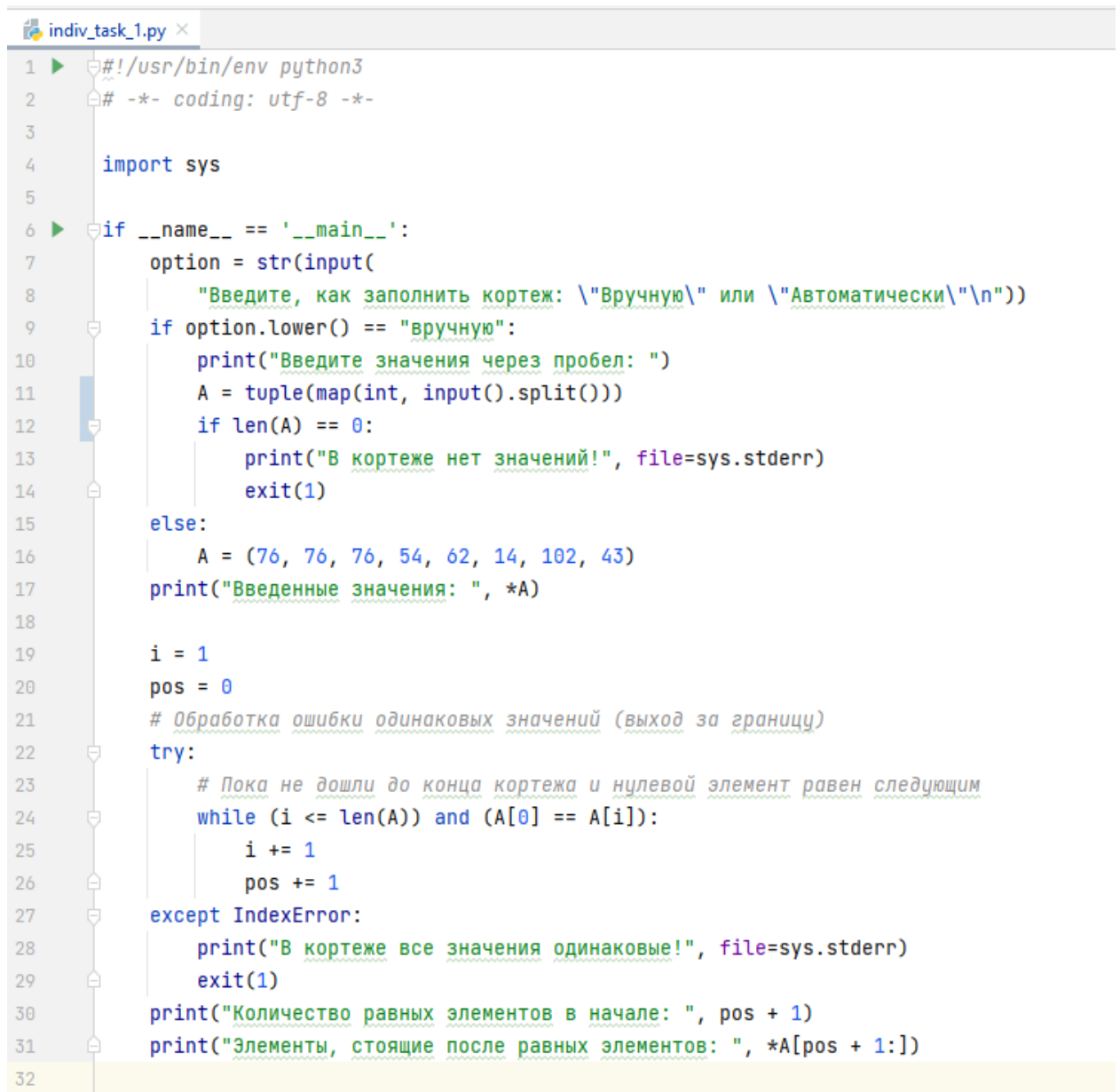
Элементы, подходящие под условие: 8 16
Сумма элементов, подходящие под условие: 24
Количество элементов, подходящие под условие: 2

Process finished with exit code 0

```

Рисунок 15 – Результаты выполнения программы индивидуального задания 1 (через списковое включение)

Задание 2. Составить программу с использованием одномерных массивов для решения задачи на переупорядочивание элементов массива. Для сортировки допускается использовать метод `sort` с заданным параметром `key` (<https://docs.python.org/3/howto/sorting.html>) и объединение нескольких списков. Номер варианта необходимо получить у преподавателя. Рисунки 16, 17.



```
1  ▶  #!/usr/bin/env python3
2  ▶  # -*- coding: utf-8 -*-
3
4  import sys
5
6  ▶  if __name__ == '__main__':
7      option = str(input(
8          "Введите, как заполнить кортеж: \"Вручную\" или \"Автоматически\"\\n"))
9      if option.lower() == "вручную":
10         print("Введите значения через пробел: ")
11         A = tuple(map(int, input().split()))
12         if len(A) == 0:
13             print("В кортеже нет значений!", file=sys.stderr)
14             exit(1)
15     else:
16         A = (76, 76, 76, 54, 62, 14, 102, 43)
17     print("Введенные значения: ", *A)
18
19     i = 1
20     pos = 0
21     # Обработка ошибки одинаковых значений (выход за границу)
22     try:
23         # Пока не дошли до конца кортежа и нулевой элемент равен следующим
24         while (i <= len(A)) and (A[0] == A[i]):
25             i += 1
26             pos += 1
27     except IndexError:
28         print("В кортеже все значения одинаковые!", file=sys.stderr)
29         exit(1)
30     print("Количество равных элементов в начале: ", pos + 1)
31     print("Элементы, стоящие после равных элементов: ", *A[pos + 1:])
32
```

Рисунок 16 – Код программы индивидуального задания 2

```
Run: indiv2 <
C:\git\lab2.4\2.4_programs\venv\Scripts\python.exe C:/git/lab2.4/2.4_programs/indiv2.py
Введите, как заполнить массив: "Вручную" или "Автоматически"
Автоматически
Введите количество чисел: 10
Введенные значения: 280.84 -28.32 -72.18 -46.21 111.71 -70.79 236.56 200.36 97.46 294.61

Введите левую границу: -30.1
Введите правую границу: 200.0
Элементы, удовлетворяющие [-30.1 , 200.0] >> [-28.32, 111.71, 97.46]
Количество элементов, удовлетворяющие условию: 3
Сумма чисел списка, начиная от максимального значения: 294.61
Отсортированный по убыванию модулей исходный список: [294.61, 280.84, 236.56, 200.36, 111.71, 97.46, -72.18, -70.79, -46.21, -28.32]

Process finished with exit code 0

Run: indiv2 <
C:\git\lab2.4\2.4_programs\venv\Scripts\python.exe C:/git/lab2.4/2.4_programs/indiv2.py
Введите, как заполнить массив: "Вручную" или "Автоматически"
Вручную
Введите числа через пробел: 10.2 -54.6 3.4 6.3 -0.3
Введенные значения: 10.2 -54.6 3.4 6.3 -0.3

Введите левую границу: -1.2
Введите правую границу: 50.0
Элементы, удовлетворяющие [-1.2 , 50.0] >> [10.2, 3.4, 6.3, -0.3]
Количество элементов, удовлетворяющие условию: 4
Сумма чисел списка, начиная от максимального значения: -35.00000000000001
Отсортированный по убыванию модулей исходный список: [-54.6, 10.2, 6.3, 3.4, -0.3]

Process finished with exit code 0

Run: indiv2 <
C:\git\lab2.4\2.4_programs\venv\Scripts\python.exe C:/git/lab2.4/2.4_programs/indiv2.py
Введите, как заполнить массив: "Вручную" или "Автоматически"
вручную
Введите числа через пробел:
Заданный список пуст

Process finished with exit code 1
```

Рисунок 17 – Результаты выполнения программы индивидуального задания 2

```
C:\Windows\System32\cmd.exe

C:\git\lab2.4>git add .

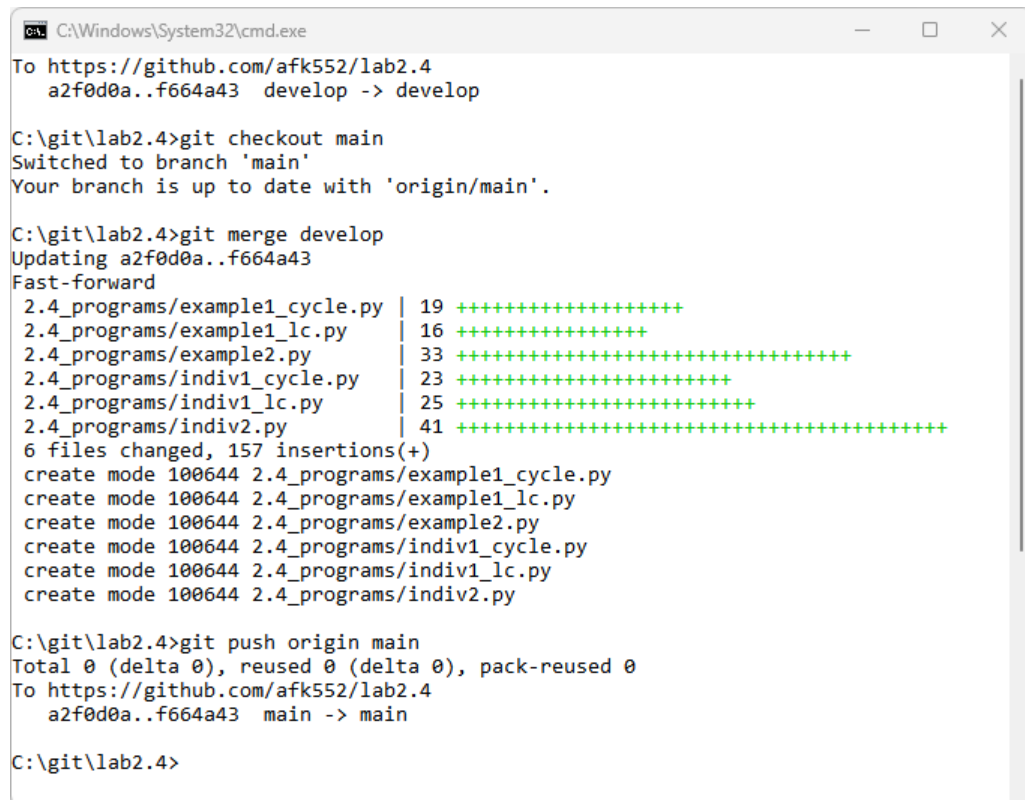
C:\git\lab2.4>git status
On branch develop
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   2.4_programs/indiv1_cycle.py
    new file:   2.4_programs/indiv1_lc.py
    new file:   2.4_programs/indiv2.py

C:\git\lab2.4>git commit -m "Add individual tasks"
[develop f664a43] Add individual tasks
3 files changed, 89 insertions(+)
create mode 100644 2.4_programs/indiv1_cycle.py
create mode 100644 2.4_programs/indiv1_lc.py
create mode 100644 2.4_programs/indiv2.py

C:\git\lab2.4>
```

Рисунок 18 – Окно командной строки

Выполните слияние ветки для разработки с веткой main / master.
Отправьте сделанные изменения на сервер GitHub. Рисунки 19, 20.



```
C:\Windows\System32\cmd.exe
To https://github.com/afk552/lab2.4
a2f0d0a..f664a43 develop -> develop

C:\git\lab2.4>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

C:\git\lab2.4>git merge develop
Updating a2f0d0a..f664a43
Fast-forward
 2.4_programs/example1_cycle.py | 19 ++++++
 2.4_programs/example1_lc.py    | 16 ++++++
 2.4_programs/example2.py       | 33 ++++++
 2.4_programs/indiv1_cycle.py   | 23 ++++++
 2.4_programs/indiv1_lc.py      | 25 ++++++
 2.4_programs/indiv2.py         | 41 ++++++
6 files changed, 157 insertions(+)
create mode 100644 2.4_programs/example1_cycle.py
create mode 100644 2.4_programs/example1_lc.py
create mode 100644 2.4_programs/example2.py
create mode 100644 2.4_programs/indiv1_cycle.py
create mode 100644 2.4_programs/indiv1_lc.py
create mode 100644 2.4_programs/indiv2.py

C:\git\lab2.4>git push origin main
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/afk552/lab2.4
a2f0d0a..f664a43 main -> main

C:\git\lab2.4>
```

Рисунок 19 – Окно командной строки

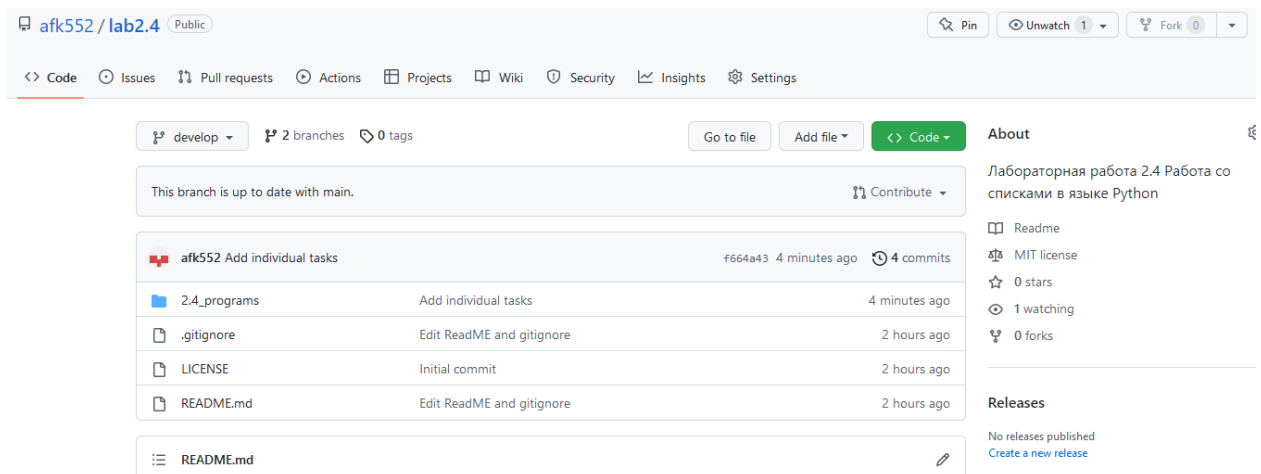


Рисунок 20 – Удаленный репозиторий на GitHub

Вывод: В результате выполнения работы были изучены списки в языке программирования Python 3, работа с ними и написание программ с их использованием.

Контрольные вопросы:

1. Что такое списки в языке Python?

Список (list) – это структура данных для хранения объектов различных типов. Список очень похож на массив, только в нем можно хранить объекты 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. Как осуществляется создание списка в Python?

Для создания списка нужно заключить элементы в квадратные скобки.

```
A = []
```

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.

4. Каким образом можно перебрать все элементы списка?

```
for elem in my_list:  
    print(elem)
```

5. Какие существуют арифметические операции со списками?

Объединение списков с помощью операции +, повторение с помощью *.

6. Как проверить есть ли элемент в списке?

```
if elem in list:  
    do_something()
```

7. Как определить число вхождений заданного элемента в списке?

Через метод count:

```
lst = [1, 2, 2, 3, 3]
print(lst.count(2))
```

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

Добавление нового элемента реализуется методом append():

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

Вставка реализуется методом insert():

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

9. Как выполнить сортировку списка?

Сортировку можно выполнить методом sort()

```
list_2.sort()
list_2.sort(reverse=True) (аргумент для сортировки в порядке убывания)
```

10. Как удалить один или несколько элементов из списка?

Метод pop(<позиция>)

Метод remove(<значение элемента>)

Функция del my_list[2]

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

List Comprehensions (списковое включение) является частью синтаксиса языка, которая предоставляет простой способ построения списков. Также, есть две 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.

```
a = [i for i in range(n)]
```

```
b = [i**2 for i in a]
```

```
b = [i for i in a if i % 2 == 0]
```

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

Так же, как и строки: A[start:stop:step]

13. Какие существуют функции агрегации для работы со списками?

len(), min(), max(), sum()

14. Как создать копию списка?

Через метод copy(): list2 = list1.copy()

Через срезы: list2 = list1[:]

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

Метод sorted() – сортирует, возвращая новый список.

sort() – сортирует, работает с текущим списком.