

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ
ФЕДЕРАЦИИ**

**Федеральное государственное автономное образовательное учреждение
высшего образования**

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития

Кафедра информационных систем и технологий

Отчет по лабораторной работе №6.

Дисциплина: «Основы программной инженерии»

Выполнил:

Студент группы ПИЖ-б-о-22-1,

направление подготовки: 09.03.04

«Программная инженерия»

ФИО: Рядская Мария Александровна

Проверил:

Воронкин Р. А.


Ставрополь 2022

Выполнение работы:

Создайте новый репозиторий



Репозиторий содержит все файлы проекта, включая историю изменений. У вас уже есть репозиторий проекта в другом месте? [Импортируйте репозиторий](#).

Обязательные поля отмечены звездочкой (*).

Владелец *	Название репозитория *
 мрядская ▾	/ lab7
✔ доступен lab7.	

Отличные названия репозитория короткие и запоминающиеся. Вам нужно вдохновение? Как насчет **автоматической вычислительной машины?**

Описание (необязательно)

- ☒  **Общедоступный**
Этот репозиторий может видеть любой пользователь Интернета. Вы сами выбираете, кто может совершать коммиты.
- ☐  **Приватный**
Вы сами выбираете, кто может просматривать этот репозиторий и вносить в него изменения.

Инициализируйте этот репозиторий с помощью:

- ☒ **Добавьте файл README**
Здесь вы можете написать длинное описание вашего проекта. [Узнайте больше о READMEs.](#)

Добавить файл .gitignore

.шаблон gitignore: Отсутствует ▾

Изучил теоретический материал работы.

Создала репозиторий на git.hub.

```
pa
C:\>cd git1

C:\git1>git clone https://github.com/mryadskaya/lab7.git
Cloning into 'lab7'...
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 5 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (5/5), done.

C:\git1>
```

Клонировала репозиторий

```
#!/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)
```

ame__ == '__main__'

pr1 x

:

C:\Users\ADMIN\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\git1\lab4\PyCharm\pr1.py

15

Неверный размер списка

Process finished with exit code 1

|

- Организовать свой репозиторий в соответствии с моделью ветвления git-flow.

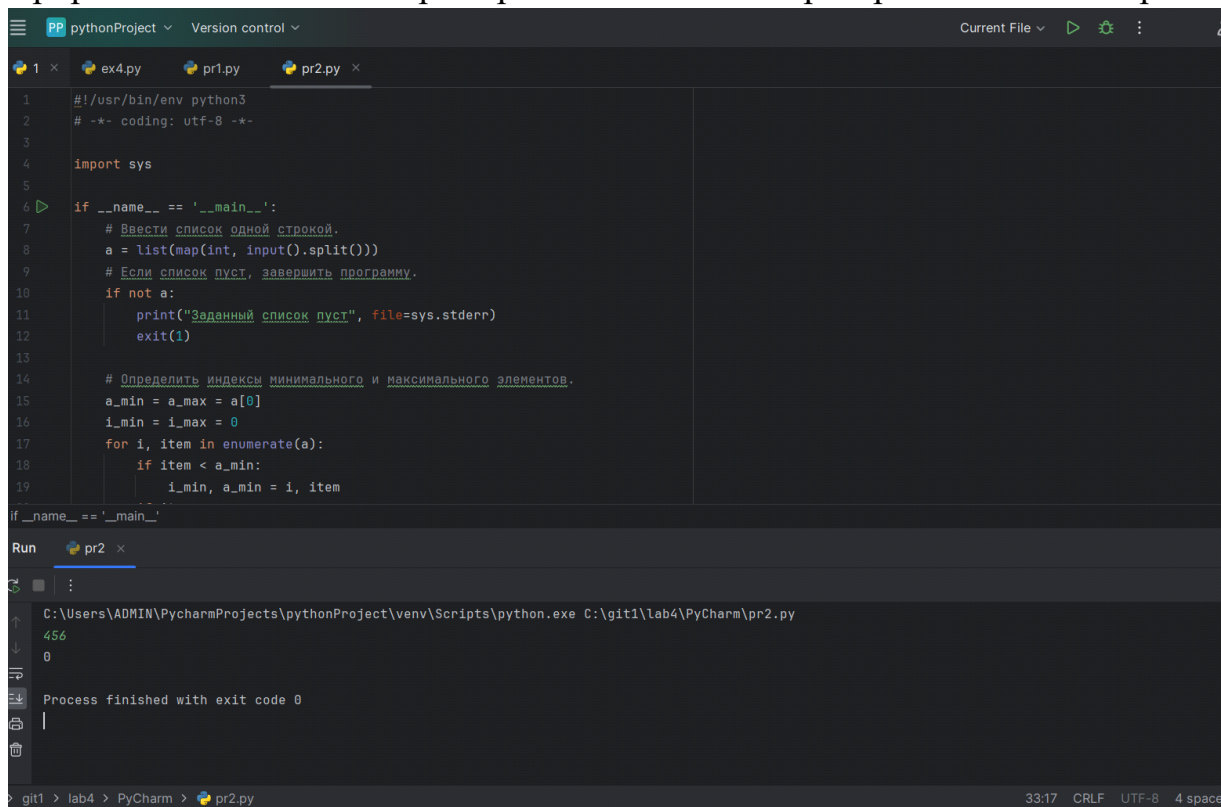
```
C:\git1\lab6>git branch develop

C:\git1\lab6>git push -u 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/mryadskaya/lab6/pull/new/develop
remote:
To https://github.com/mryadskaya/lab6.git
 * [new branch]      develop -> develop
branch 'develop' set up to track 'origin/develop'.

C:\git1\lab6>git checkout develop
Switched to branch 'develop'
Your branch is up to date with 'origin/develop'.

C:\git1\lab6>
```

Проработать примеры лабораторной работы



Пример 1

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

import sys

if __name__ == '__main__':
    A = list(map(int, input().split()))
    if len(A) != 10:
        print("Неверный размер списка", file=sys.stderr)
        exit(1)

    # Найти и вывести сумму элементов
    s = sum([a for a in A if a > 3 and a < 8 and i % 10 == 0])
    print(s)
```

zd1 x

C:\Users\ADMIN\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\git1\lab4\PyCharm\zd1.py

15 45 78 10 1 456 23 78 94 66

0

Process finished with exit code 0

Пример 2

Ввести список A из 10 элементов, найти произведение элементов, больших 8 и меньших 18

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import sys
5
6 if __name__ == '__main__':
7     A = list(map(int, input().split()))
8     if len(A) != 10:
9         print("Неверный размер списка", file=sys.stderr)
10        exit(1)
11
12    # Найти и вывести сумму элементов
13    s = sum([a for a in A if a > 3 and a < 8 and a % 10 == 0])
14    print(s)
15
16__name__ == '__main__'
```

un zd1 x

C:\Users\ADMIN\PycharmProjects\pythonProject\venv\Scripts\python.exe C:\git1\lab4\PyCharm\zd1.py

123 156 741 78 1 2 3 4 12 1314

0

Process finished with exit code 0

и кратных 10, их количество и вывести результаты на экран.

В списке, состоящем из вещественных элементов,
вычислить:

1 количество отрицательных элементов списка;

2 сумму модулей элементов списка, расположенных после минимального по модулю элемента.

Заменить все отрицательные элементы списка их квадратами и упорядочить элементы списка по возрастанию.

```
main.py
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6  if __name__ == '__main__':
7      lst = [-5, 3, -2, 7, -8, 10, -4]
8      # количество отрицательных элементов списка
9      negative_count = len(list(filter(lambda x: x < 0, lst)))
10
11     # находим индекс минимального по модулю элемента
12     min_abs_index = lst.index(min(lst, key=abs))
13
14     # сумма модулей элементов списка, расположенных после минимального по модулю элемента
15     sum_after_min_abs = sum(
16         map(abs, filter(lambda x: lst.index(x) > min_abs_index, lst)))
17
18     # заменяем отрицательные элементы списка их квадратами
19     lst = [x ** 2 if x < 0 else x for x in lst]
20
21     # упорядочиваем элементы списка по возрастанию
22     lst.sort()
23
24
25
26     # пример использования
27
28     print("Количество отрицательных элементов:", negative_count)
29     print("Сумма модулей элементов после минимального по модулю элемента:",
30           sum_after_min_abs)
31     print("Упорядоченный список с квадратами отрицательных элементов:",
32           lst)
```

input

Количество отрицательных элементов: 4
Сумма модулей элементов после минимального по модулю элемента: 29
Упорядоченный список с квадратами отрицательных элементов: [3, 4, 7, 10, 16, 25, 64]

Program finished with exit code 0

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, как было уже сказано выше, 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.

3. Как организовано хранение списков в оперативной памяти?

При создании списка в памяти резервируется область, которую можно условно назвать некоторым “контейнером”, в котором хранятся ссылки на другие

элементы данных в памяти. В отличие от таких типов данных как число или строка, содержимое “контейнера” списка можно менять.

4. Каким образом можно перебрать все элементы списка?

С помощью цикла:

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

```
for elem in my_list:
```

```
print(elem)
```

5. Какие существуют арифметические операции со списками?

Списки можно сложить используя “+”.

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?

Можно указать индекс списка, куда нужно вставить новый элемент.

Также можно воспользоваться `append`(один элемент) и `extend`(сразу несколько элементов).

9. Как выполнить сортировку списка?

Можно воспользоваться методом `sort`.

10. Как удалить один или несколько элементов из списка?

Удалить элемент можно, написав его индекс в методе `pop`. Если не указывать индекс, то функция удалит последний элемент. Элемент можно удалить

с помощью метода `remove`. Можно удалить несколько элементов с помощью оператора среза и `del`. Можно удалить все элементы из списка с помощью метода

`clear`.

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.

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

Слайсы (срезы) являются очень мощной 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. 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.

14. Как создать копию списка?

Используя метод `copy()`.

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

Основное отличие между `sorted` и `sort` заключается в том, что `sorted` создает новый отсортированный список, оставляя исходный список без изменений, в то

время как `sort` сортирует сам список, изменяя его. Выбор между ними зависит от

ваших потребностей и того, нужно ли вам сохранить оригинальный порядок элементов.