

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт перспективной инженерии
Департамент цифровых, робототехнических систем и электроники

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №7
дисциплины
«Основы кроссплатформенного программирования»

Выполнил:
Пугачев Кирилл Дмитриевич
2 курс, группа ИТС-б-о-23-1,
11.03.02«Инфокоммуникационные
технологии и системы связи», очная
форма обучения

(подпись)

Проверил:
Доцент департамента цифровых,
робототехнических систем и
электроники Воронкин Р.А.

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2024 г.

ТЕМА: РАБОТА СО СПИСКАМИ В ЯЗЫКЕ PYTHON

Цель: приобретение навыков по работе со списками при написании программ с помощью языка программирования версии 3.x.

Порядок выполнения работы:

1. Изучил теоретический материал.
2. Приступил к выполнению заданий.
3. Создание репозитория

Ссылка на репозиторий: <https://github.com/chillkirill/LABA7.git>

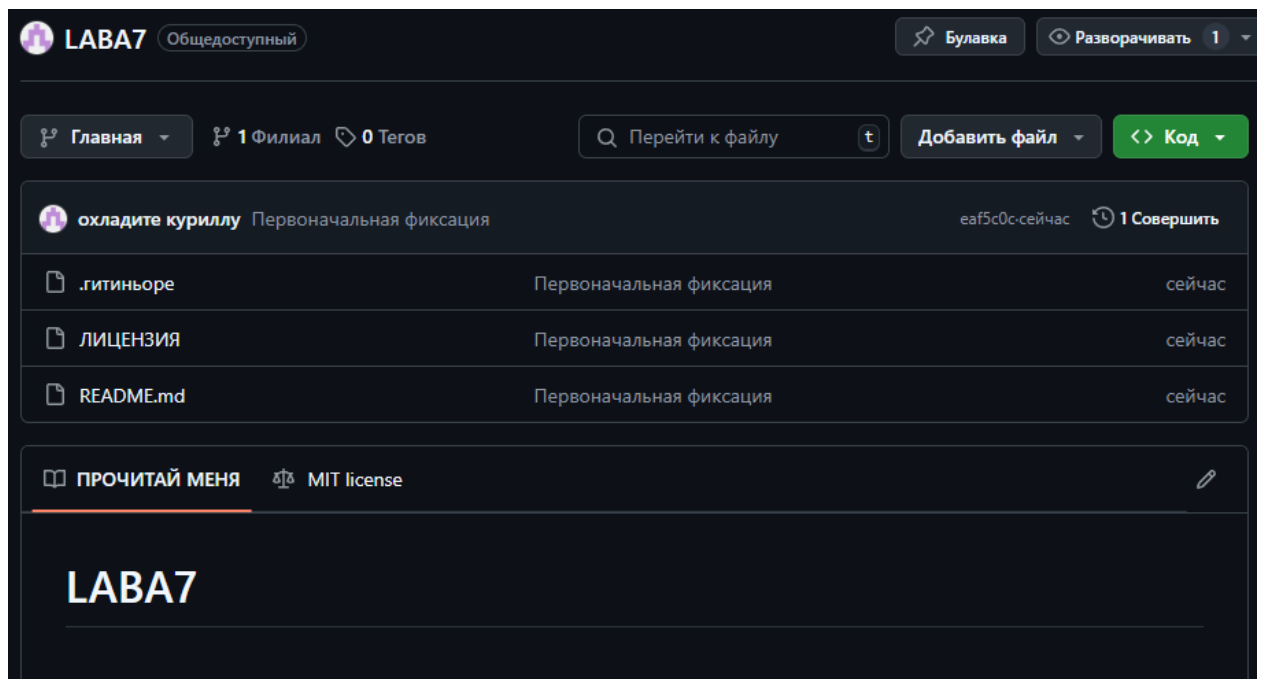


Рисунок 1. Репозиторий

4. Клонирование репозитория

```
C:\Users\user>git clone https://github.com/chillkirill/LABA7.git
Cloning into 'LABA7'...
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 (from 0)
Receiving objects: 100% (5/5), done.
```

Рисунок 2. Клонирование

5. Проработал примеры

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

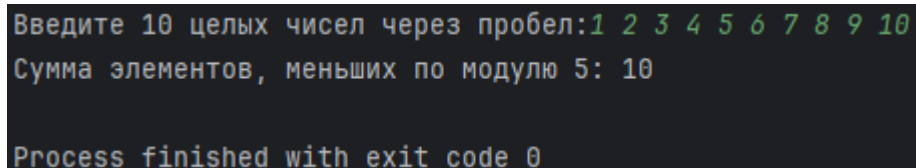
if __name__ == '__main__':
    # Ввод списка из 10 элементов
    A = list(map(int, input("Введите 10 целых чисел через пробел:").split()))

    # Проверка, что введено именно 10 элементов
    if len(A) != 10:
        print("Ошибка: необходимо ввести ровно 10 целых чисел.", file=sys.stderr)
        exit(1)

    # Вычисление суммы элементов, меньших по модулю 5
    total_sum = sum(item for item in A if abs(item) < 5)

    # Вывод суммы
    print("Сумма элементов, меньших по модулю 5:", total_sum)
```

Рисунок 3. Код к 1 примеру



```
Введите 10 целых чисел через пробел:1 2 3 4 5 6 7 8 9 10
Сумма элементов, меньших по модулю 5: 10

Process finished with exit code 0
```

Рисунок 4. Пример 1

```
def count_positive_between_min_max(lst):
    if not lst:
        return 0 # Если список пустой, возвращаем 0

    min_value = min(lst)
    max_value = max(lst)

    # Находим индексы минимального и максимального элементов
    min_index = lst.index(min_value)
    max_index = lst.index(max_value)

    # Убедимся, что индексы корректные (первый индекс - меньший)
    start_index = min(min_index, max_index) + 1
    end_index = max(min_index, max_index)

    # Считаем количество положительных элементов между ними
    positive_count = sum(1 for x in lst[start_index:end_index] if x > 0)

    return positive_count

if __name__ == '__main__':
    # Ввод списка целых чисел
    A = list(map(int, input("Введите целые числа через пробел: ").split()))

    # Получаем количество положительных элементов между минимальным и максимальным
    result = count_positive_between_min_max(A)

    print("Количество положительных элементов между максимальным и минимальным:", result)
```

Рисунок 5. Код к 2 примеру

```
Введите целые числа через пробел: 1 5 47 32 6
Количество положительных элементов между максимальным и минимальным: 1
Process finished with exit code 0
```

Рисунок 6. Пример 2

6. Зафиксировал изменения

```
C:\Users\user\LABA7>git add .
C:\Users\user\LABA7>git commit -m "Добавление примеров"
[main 823b816] Добавление примеров
2 files changed, 45 insertions(+)
create mode 100644 primer1.py.txt
create mode 100644 primer2.py.txt
C:\Users\user\LABA7>git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 1.35 KiB | 1.35 MiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/chillkirill/LABA7.git
 eaf5c0c..823b816  main -> main
```

Рисунок 7. Изменения

7. Приступил к выполнению индивидуального задания

```
# Ввод списка A из 10 элементов
A = [int(input(f"Введите элемент {i+1}: ")) for i in range(10)]

# Находим сумму положительных элементов, кратных 5, и их количество
positive_multiples_of_5 = [x for x in A if x > 0 and x % 5 == 0]

# Результаты
sum_of_elements = sum(positive_multiples_of_5)
count_of_elements = len(positive_multiples_of_5)

# Выводим результаты
print(f"Сумма положительных элементов, кратных 5: {sum_of_elements}")
print(f"Количество положительных элементов, кратных 5: {count_of_elements}")
```

Рисунок 8. Код к 1 заданию

```
Введите элемент 1: 1
Введите элемент 2: 2
Введите элемент 3: 3
Введите элемент 4: 4
Введите элемент 5: 5
Введите элемент 6: 6
Введите элемент 7: 7
Введите элемент 8: 8
Введите элемент 9: 9
Введите элемент 10: 10
Сумма положительных элементов, кратных 5: 15
Количество положительных элементов, кратных 5: 2
```

Рисунок 9. Задание 1

```
# Ввод списка вещественных чисел
A = list(map(float, input("Введите вещественные числа через пробел: ").split()))

# 1. Произведение отрицательных элементов списка
negative_elements = [x for x in A if x < 0]
product_of_negatives = 1
for num in negative_elements:
    product_of_negatives *= num if num != 0 else 1 # Учитываем, что произведение на 0

# 2. Сумма положительных элементов до максимального элемента
max_element = max(A)
max_index = A.index(max_element)

positive_elements_before_max = [x for x in A[:max_index] if x > 0]
sum_of_positives_before_max = sum(positive_elements_before_max)

# 3. Изменяем порядок следования элементов в списке на обратный
A.reverse()

# Вывод результатов
print(f"Произведение отрицательных элементов: {product_of_negatives}")
print(f"Сумма положительных элементов до максимального элемента: {sum_of_positives_befo")
print(f"Список в обратном порядке: {A}")
```

Рисунок 10. Код ко 2 заданию

```
Введите вещественные числа через пробел: 1.5 -2.3 4.6 -0.5
Произведение отрицательных элементов: 1.15
Сумма положительных элементов до максимального элемента: 1.5
Список в обратном порядке: [-0.5, 4.6, -2.3, 1.5]
```

Рисунок 11. Задание 2

8. Зафиксировал изменения

```
C:\Users\user\LABA7>git add .  
  
C:\Users\user\LABA7>git commit -m "Добавление индивидуальных заданий"  
[main bb36961] Добавление индивидуальных заданий  
2 files changed, 36 insertions(+)  
create mode 100644 individual1.py.txt  
create mode 100644 individual2.py.txt  
  
C:\Users\user\LABA7>git push\  
git: 'push\' is not a git command. See 'git --help'.  
  
The most similar command is  
    push  
  
C:\Users\user\LABA7>git push  
Enumerating objects: 5, done.  
Counting objects: 100% (5/5), done.  
Delta compression using up to 8 threads  
Compressing objects: 100% (4/4), done.  
Writing objects: 100% (4/4), 1.20 KiB | 1.21 MiB/s, done.  
Total 4 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)  
remote: Resolving deltas: 100% (1/1), completed with 1 local object.  
To https://github.com/shillini11/LABA7.git
```

Рисунок 12. Изменение

Ответы на контрольные вопросы:

1. Что такое списки в языке Python?

Списки в Python — это изменяемые последовательности, которые могут содержать элементы различных типов, включая числа, строки и другие объекты. Списки позволяют хранить и управлять коллекциями данных.

2. Как осуществляется создание списка в Python?

Список создается с помощью квадратных скобок [], также можно использовать функцию list.

3. Как организовано хранение списков в оперативной памяти?

Списки хранятся в виде массивов указателей на объекты. Каждый элемент списка хранит ссылку на объект, а не сам объект. Это позволяет спискам содержать элементы разных типов.

4. Каким образом можно перебрать все элементы списка?

Перебор элементов списка можно осуществить с помощью цикла for.

5. Какие существуют арифметические операции со списками?

Основные арифметические операции со списками: конкатенация — объединяет два списка. Повторение — повторяет список n раз.

6. Как проверить есть ли элемент в списке?

Через оператора in if element in my list

7. Как определить число вхождений заданного элемента в списке?

Через метод count

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

Через метод append для добавления в конец списка или insert для вставки по индексу.

9. Как выполнить сортировку списка?

Через sort для сортировки списка на месте или функцию sorted для создания нового отсортированного списка

10. Как удалить один или несколько элементов из списка?

Через remove для удаления первого вхождения элемента, pop(index) для удаления элемента по индексу, или del для удаления по индексу

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

Списковое включение — это способ создания нового списка на основе существующего.

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

Срезы позволяют извлекать подсписки. Синтаксис: `my list[start:end]`, где `start` — начальный индекс, а `end` — конечный индекс.

13. Какие существуют функции агрегации для работы со списками?

Основные функции: `sum` — сумма элементов. `min` — минимальный элемент. `max` — максимальный элемент. `len` — количество элементов.

14. Как создать копию списка?

Через метод `copy`

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

Функция `sorted` создает новый отсортированный список и не изменяет оригинальный, а метод `sort` сортирует список на месте и возвращает `None`

Вывод: в ходе работы исследовал базовые возможности системы контроля версий Git для работы с локальными репозиториями.