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 № 4
по дисциплине «Программирование на Python»

Выполнил студент Ромащенко Данил группы ИВТ-б-о-24-1

Подпись студента_____
Работа защищена « »_____20__г.
Проверил Воронкин Р.А. _____
(подпись)

Ставрополь 2025

Тема: «Работа со списками и кортежами в языке Python».

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

Ход работы

Вариант 18

Ссылка на репозиторий: https://github.com/l1N322/lab-4.git

1) Был создан общедоступный репозиторий, в котором использована лицензия MIT и язык программирование Python.

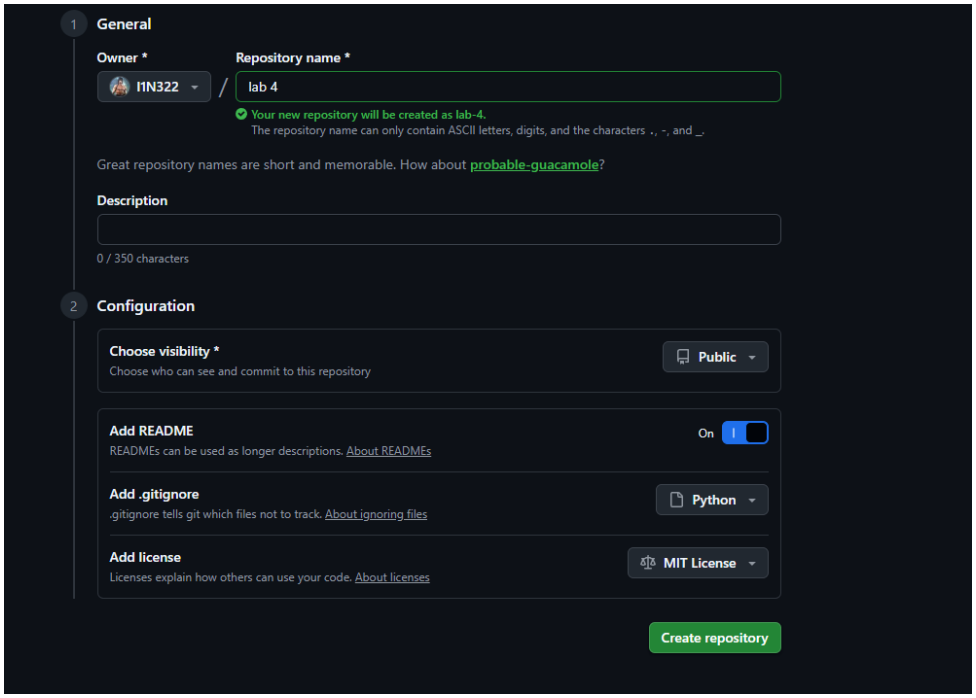

Рисунок 1. Созданный репозиторий

2) Был дополнен файл .gitignore необходимыми правилами для работы с IDE PyCharm.

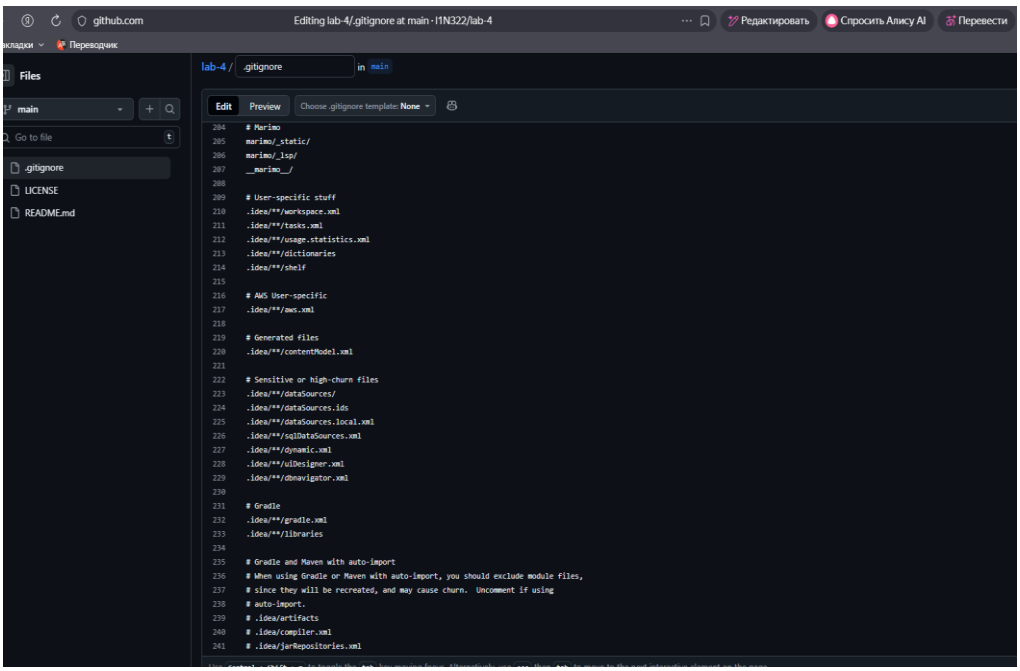

Рисунок 2. Добавленные правила в файл .gitignore

3) Было выполнено клонирование созданного репозитория на компьютер.

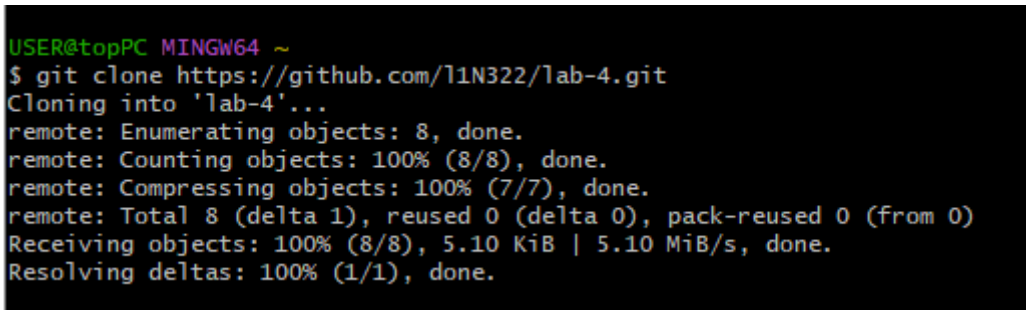

Рисунок 3. Клонирование репозитория

4) Был создан проект PyCharm в папке репозитория.

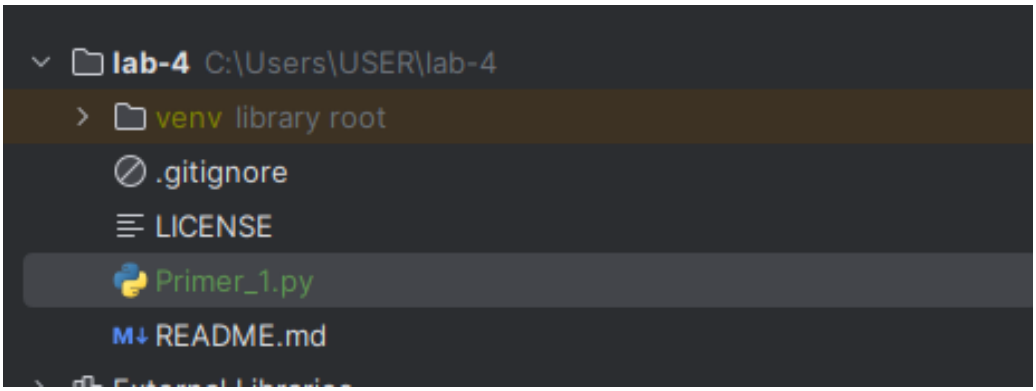

Рисунок 4. Созданный проект

5) Были проработаны примеры лабораторной работы и зафиксированы результаты выполнения каждой из программ при разных исходных данных, вводимых с клавиатуры.

Пример 1.py

```python
#!/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)

    s = 0
    for item in A:
        if abs(item) < 5:
            s += item

    print(s)
```

Рисунок 5. Пример 1

Пример 1

```
C:\Users\USER\lab-4\venv\Scripts\python.exe "C:/Users/USER/lab-4/Пример 1.py"
1 2 3 4 5 6 7 8 9 3
13

Process finished with exit code 0
```

Рисунок 6. Результат выполнения примера 1

Пример 1

```
C:\Users\USER\lab-4\venv\Scripts\python.exe "C:/Users/USER/lab-4/Пример 1.py"
1
Неверный размер списка

Process finished with exit code 1
```

Рисунок 7. Результат выполнения примера 1 с другими данными

Пример 1.py Пример 2.py

```
if __name__ == '__main__':
    a = list(map(int, input().split()))
    if not a:
        print("Заданный список пуст", file=sys.stderr)
        exit(1)

    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

    if i_min > i_max:
        i_min, i_max = i_max, i_min

    count = 0
    for item in a[i_min+1:i_max]:
        if item > 0:
            count += 1

    print(count)
```

Рисунок 8. Пример 2

Пример 2

```
C:\Users\USER\lab-4\venv\Scripts\python.exe "C:/Users/USER/lab-4/Пример 2.py"
1 9 2 4 6 2 7 0
5

Process finished with exit code 0
```

Рисунок 9. Результат выполнения примера 2

Рисунок 10. Результат выполнения примера 2 с другими данными

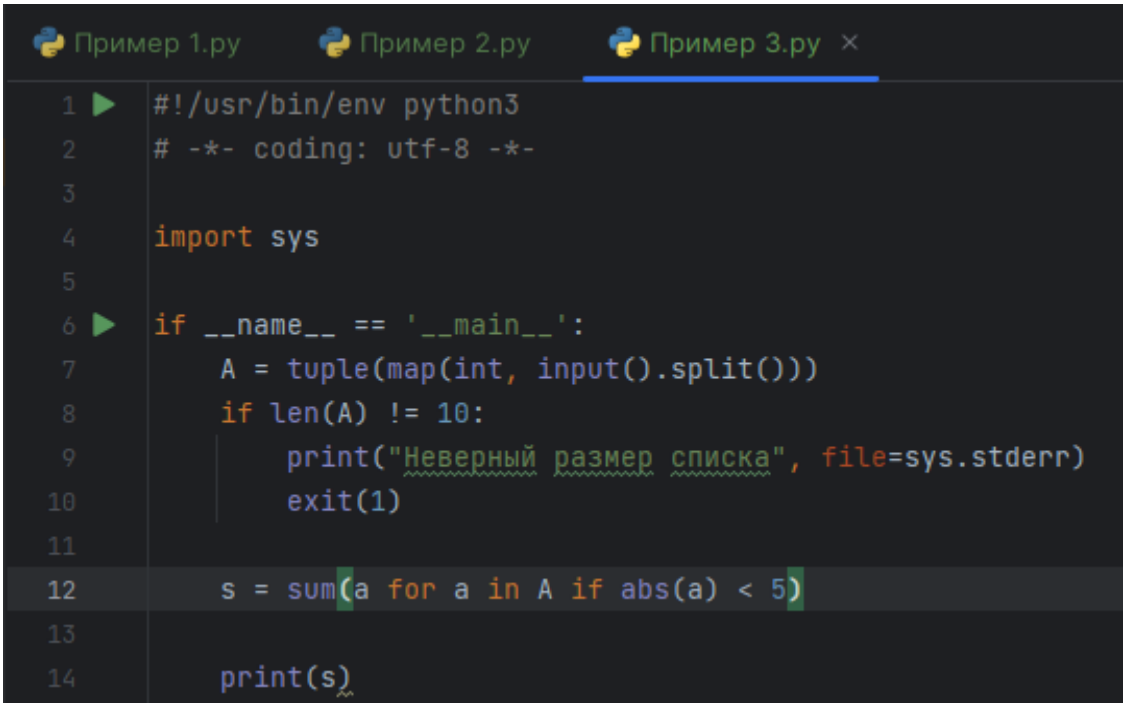

Рисунок 11. Пример 3

Рисунок 12. Результат выполнения примера 3

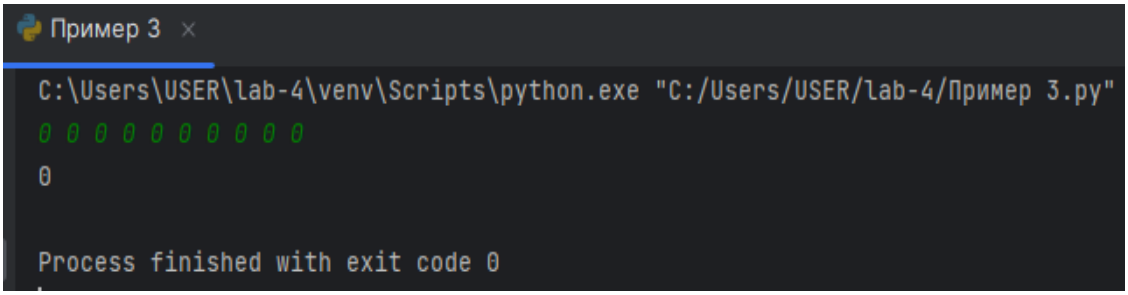

Рисунок 13. Результат выполнения примера 3 с другими данными

6) Были выполнены индивидуальные задания.

7) 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: Ввести список A из 10 элементов, найти сумму положительных элементов кратных 5, их количество и вывести результаты на экран.

Пример 1.py | Пример 2.py | Задание 1.py

```
#!/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)

    sum = 0
    for item in A:
        if item > 0 and item % 5 ==0:
            sum += item

    print(sum)
```

Рисунок 14. Задание 1

```
C:\Users\USER\lab-4\venv\Scripts\python.exe "C:/Users/USER/lab-4/Задание 1.py"
1 4 -2 0 5 8 10 5 9 -5
20

Process finished with exit code 0
```

Рисунок 15. Результат выполнения задания 1

```
C:\Users\USER\lab-4\venv\Scripts\python.exe "C:/Users/USER/lab-4/Задание 1.py"
1 2 3 -1 -4 0 8 2 9 0
0

Process finished with exit code 0
```

Рисунок 16. Результат выполнения задания 1 с другими данными

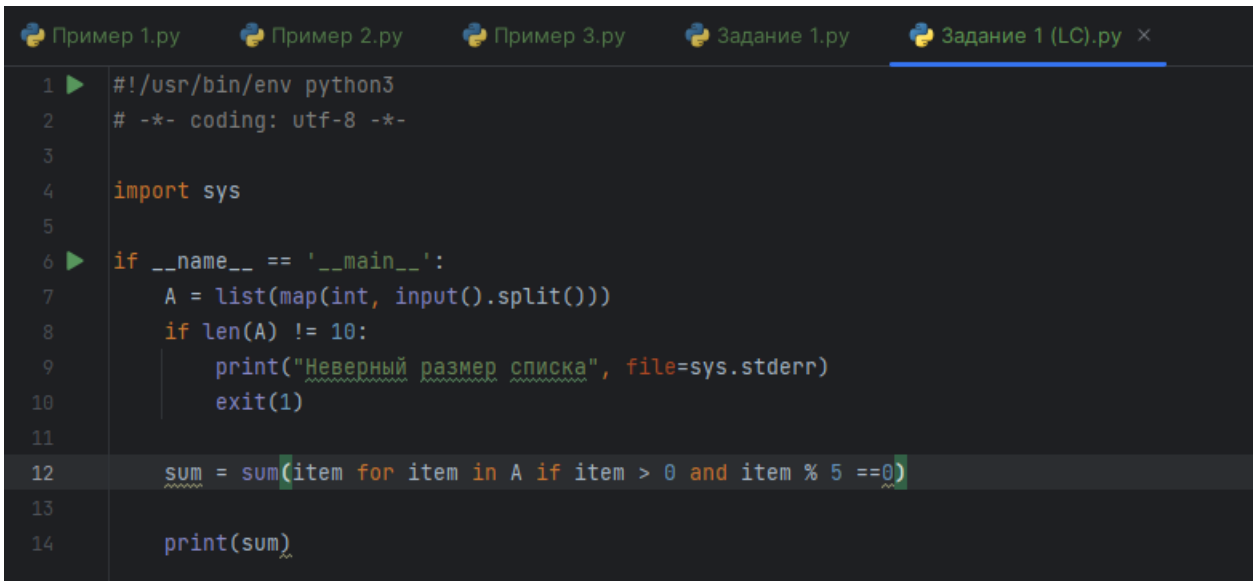

Рисунок 17. Задание 1 с использованием List Comprehensions

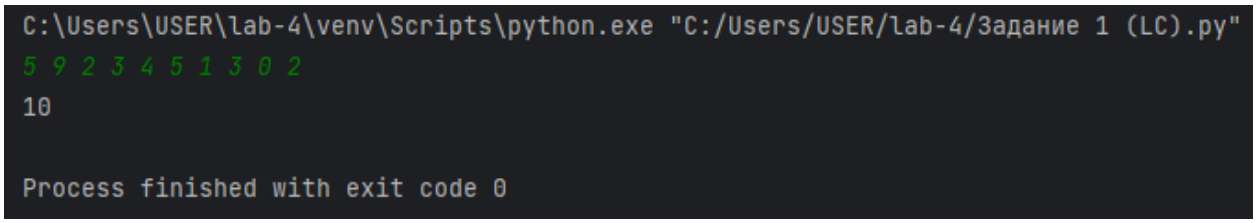

Рисунок 18. Результат выполнения задания 1 с использованием List Comprehensions

Рисунок 19. Результат выполнения задания 1 с другими данными

8) Задание 2: Составить программу с использованием одномерных массивов для решения задачи на переупорядочивание элементов списка. Для сортировки допускается использовать метод sort с заданным параметром key и объединение нескольких списков. Номер варианта необходимо получить у преподавателя: В списке, состоящем из вещественных элементов, вычислить: произведение отрицательных элементов списка; сумму положительных элементов списка, расположенных до максимального элемента. Изменить порядок следования элементов в списке на обратный.

```python
import sys

if __name__ == '__main__':
    a = list(map(int, input().split()))
    if not a:
        print("Заданный список пуст", file=sys.stderr)
        exit(1)

    neg = 1
    has_neg = False
    for num in a:
        if num < 0:
            neg *= num
            has_neg = True
    if not has_neg:
        neg = 0

    max = a[0]
    i_max = 0
    for i, num in enumerate(a):
        if num > max:
            max = num
            i_max = i

    sum_pos = sum(num for num in a[:i_max] if num > 0)

    reverse = a[::-1]

    print(f"Произведение отрицательных элементов: {neg}")
    print(f"Сумма положительных элементов до максимума: {sum_pos}")
    print(f"Список в обратном порядке: {reverse}")
```

Рисунок 20. Задание 2

```
C:\Users\USER\lab-4\venv\Scripts\python.exe "C:/Users/USER/lab-4/Задание 2.py"
2 -3 0 5 8 9 -4 5 2 7
Произведение отрицательных элементов: 12
Сумма положительных элементов до максимума: 15
Список в обратном порядке: [7, 2, 5, -4, 9, 8, 5, 0, -3, 2]

Process finished with exit code 0
```

Рисунок 21. Результат выполнения задания 2

```
C:\Users\USER\lab-4\venv\Scripts\python.exe "C:/Users/USER/lab-4/Задание 2.py"
1 2 0 9 5 2 4 1 2 2
Произведение отрицательных элементов: 0
Сумма положительных элементов до максимума: 3
Список в обратном порядке: [2, 2, 1, 4, 2, 5, 9, 0, 2, 1]

Process finished with exit code 0
```

Рисунок 22. Результат выполнения задания 2 с другими данными

9) Задание 3: Имеется информация о количестве осадков, выпавших за каждый день месяца, и о температуре воздуха в эти дни. Определить, какое количество осадков выпало в виде снега и какое - в виде дождя. (Считать, что идет дождь, если температура воздуха выше 0 °C.)

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

import sys

if __name__ == '__main__':
    count = tuple(map(int, input().split()))
    temperature = tuple(map(int, input().split()))
    if not count or not temperature:
        print("Неверные данные", file=sys.stderr)
        exit(1)

    rain = 0
    snow = 0

    for c, temp in zip(count, temperature):
        if temp > 0:
            rain += c
        else:
            snow += c

    print(f"Осадки в виде дождя: {rain} мм")
    print(f"Осадки в виде снега: {snow} мм")
```

Рисунок 23. Задание 3

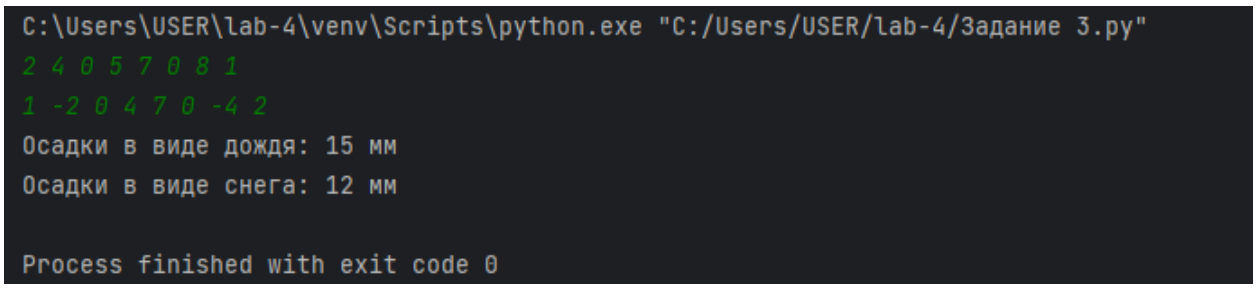

Рисунок 24. Результат выполнения задания 3

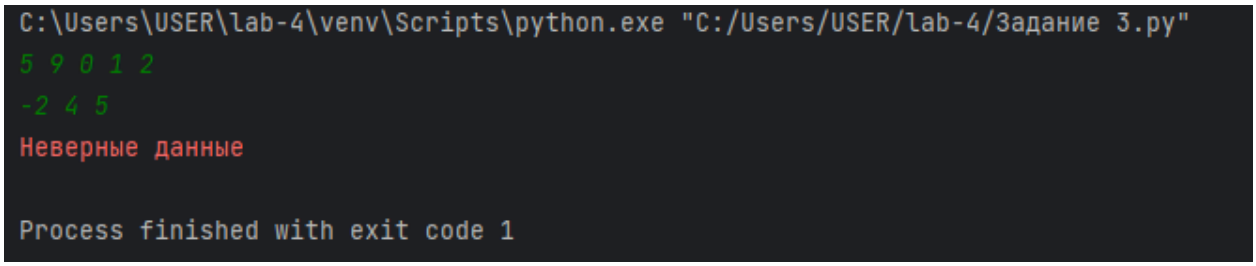

Рисунок 25. Результат выполнения задания 3 с другими данными

10) Были зафиксированы и отправлены сделанные изменения на сервер GitHub.

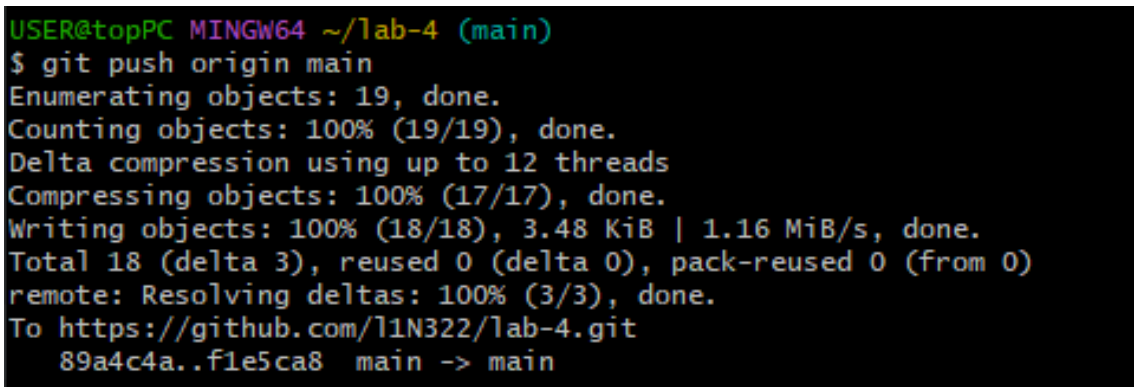

Рисунок 26. Отправка изменений

Ответы на контрольные вопросы:

1) Список (list) – это структура данных для хранения объектов различных типов. Размер списков 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) Для создания списка нужно элементы заключить в квадратные скобки: my_list = [1, 2, 3, 4, 5]

3) При создании списка 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) Перебрать все элементы списка можно с помощью следующего цикла:

```
my_list = [‘Один’, ‘Два’, ‘Три’, ‘Четыре’, ‘Пять’]
for elem in my_list:
    print(elem)
```

5) Арифметические операции со списками: + (объединение списков), * (повтор списка).

6) Для того, чтобы проверить, есть ли заданный элемент в списке Python необходимо использовать оператор in:

7) Метод count можно использовать для определения числа вхождений заданного элемента в списке.

8) Метод insert можно использовать для вставки элемента с список. Метод append можно использовать для добавления элемента в список.

9) Для сортировки списка нужно использовать метод sort.

10) Удалить метод можно, написав его индекс в методе pop.

11) List Comprehensions (абстракция списков или списковое включение) является частью синтаксиса языка, которая предоставляет простой способ построения списков.

В языке Python есть 2 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.

12) Срезы 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) Для работы со списками существуют следующие функции агрегации:

– len(L) – получить число элементов в списке

– min(L) – получить минимальный элемент списка

– max(L) – получить максимальный элемент списка

– sum(L) – получить сумму элементов списка, если список содержит только числовые значения.

14) Для создания копии списка необходимо использовать либо метод copy, либо оператор среза.

15) Функция sorted() в Python создаёт новый отсортированный список, не изменяя исходный. Это полезно, когда необходимо сохранить оригинальный порядок элементов.

Отличие от метода sort() списков заключается в том, что метод sort() изменяет исходный список на месте и возвращает None. Этот подход эффективен по памяти, поскольку не создаёт новый список. Ещё одно отличие в том, что метод list.sort() определён только для списков, в то время как функция sorted() работает со всеми итерируемыми объектами (списки, кортежи, строки, множества, словари и т. д.).

16) Кортеж (tuple) – это неизменяемая структура данных, которая по своему подобию очень похожа на список.

17) Существует несколько причин, по которым стоит использовать кортежи вместо списков. Одна из них - это обезопасить данные от случайного изменения. Если мы получили откуда-то массив данных, и у нас есть желание поработать с ним, но при этом непосредственно менять данные мы не собираемся, тогда, это как раз тот случай, когда кортежи придутся как нельзя кстати. Используя их в данной задаче, мы дополнительно получаем сразу несколько бонусов - во-первых, это экономия места. Дело в том, что кортежи в памяти занимают меньший объем по сравнению со списками. Во-вторых - прирост производительности, который связан с тем, что кортежи работают быстрее, чем списки (т. е. на операции перебора элементов и т. п. будет тратиться меньше времени). Важно также отметить, что кортежи можно использовать в качестве ключа у словаря.

18) Кортеж создается также как список, только вместо квадратных скобок используют круглые. При желании можно воспользоваться функцией tuple(). Для создания пустого кортежа, не нужно ничего писать в скобках.

19) Доступ к элементам кортеж осуществляется также как и к элементам списка – через указание индекса. Но изменять элементы кортежа нельзя.

20) Обращение по индексу, это не самый удобный способ работы с кортежами. Дело в том, что кортежи часто содержат значения разных типов, и помнить, по какому индексу что лежит – очень непросто. Для этого существует деструктуризация.

21) Благодаря тому, что кортежи легко собирать и разбирать, в Python удобно делать такие вещи, как множественное присваивание: (a, b, c) = (1, 2, 3)

22) С помощью операции взятия среза можно получить другой кортеж: T2 = T1[i:j].

23) Для кортежей можно выполнять операцию конкатенации, которая обозначается символом +. В простейшем случае общая форма операции следующая: T3 = T1 + T2.

Кортеж может быть образован путем операции повторения, обозначаемой символом *. Общая форма: T2 = T1 * n (где n – количество повторений кортежа T1).

24) Элементы кортежа можно последовательно просмотреть с помощью операторов цикла while или for, например:

```
A = (“abc”, “abcd”, “bcd”, “cde”)
for item in A:
    print(item)
```

25) Для проверки вхождения элемента в кортеж используется операция in:

```
A = (“abc”, “abcd”, “bcd”, “cde”)
item = str(input(“s = “))
if (item in A):
```

print(item, “ in “, A, “ = True”)

26) Методы работы с кортежем:

– index() – поиск позиции элемента в кортеже

count() – количество вхождения элемента в кортеж

27) Да, использование функций агрегации (таких как len(), sum(), max(), min() и др.) с кортежами в Python полностью допустимо и корректно. Кортежи поддерживают те же операции, что и другие итерируемые объекты (например, списки), так как они являются последовательностями (sequence type).

28) Создание кортежа с помощью спискового включения: a = tuple(map(int, input().split())).

Вывод: в ходе работы были приобретены навыки по работе со списками и кортежами при написании программ с помощью языка программирования Python версии 3.x.