

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

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №2.4
дисциплины «Основы программной инженерии»

Выполнил:
Юрьев Илья Евгеньевич
2 курс, группа ПИЖ-б-о-22-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка
и сопровождение программного
обеспечения», очная форма обучения

(подпись)

Руководитель практики:
Богданов С.С., ассистент кафедры
инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

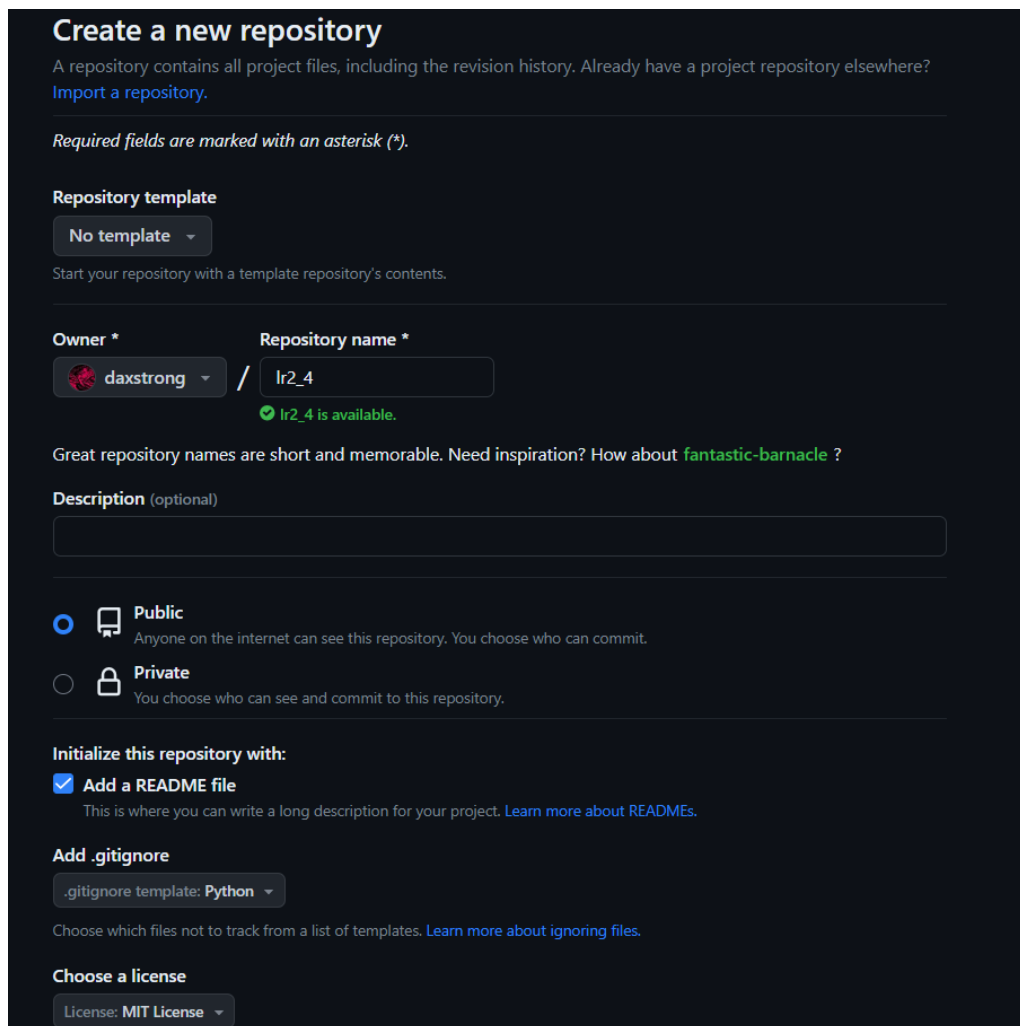
Ставрополь, 2023 г.

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, в котором будет использована лицензия 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 ().*

Repository template
No template ▾
Start your repository with a template repository's contents.

Owner * daxstrong ▾ / **Repository name *** lr2_4
✔ lr2_4 is available.

Great repository names are short and memorable. Need inspiration? How about [fantastic-barnacle](#) ?

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 ▾

Рисунок 1 – Создание репозитория с заданными настройками

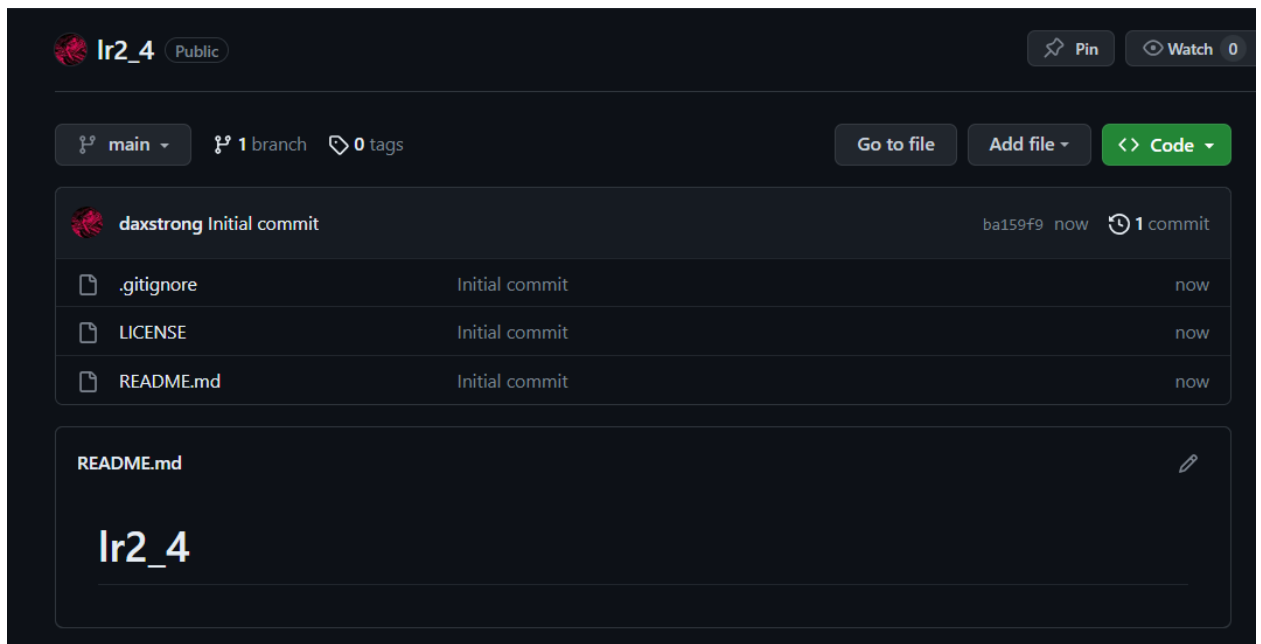


Рисунок 2 – Созданный репозиторий

```
ilyay@DESKTOP-FF1JT6S MINGW64 ~/OneDrive/Рабочий стол/Основы программной инженерии
$ git clone https://github.com/daxstrong/lr2_3.git
Cloning into 'lr2_3'...
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.
```

Рисунок 3 – Клонирование репозитория

```
ilyay@DESKTOP-FF1JT6S MINGW64 ~/OneDrive/Рабочий стол/Основы программной инженерии/lr2_4 (main)
$ git checkout -b develop
Switched to a new branch 'develop'
```

Рисунок 4 – Создание ветки develop

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

```
task1.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
20     print(s)
21
```

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

```
task1 x
C:\Users\ilyay\AppData\Local\Microsoft\WindowsApps\python3.11.exe
3 4 -1 2 3 7 9 8 6 4
15
```

Рисунок 6 – Вывод программы (задание №1)

```

task2.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 между максимальным и минимальным элементами (задание №2)

```

task2 x
C:\Users\ilyay\AppData\Local\Microsoft\WindowsApps\python3.11.exe
-2 1 2 3 4 5 6
5
Process finished with exit code 0

```

Рисунок 8 – Вывод программы (задание №2)

```

task3.py x
1  ▶  #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3
4      import sys
5
6  ▶  if __name__ == '__main__':
7      s = input("Введите предложение: ")
8      n = int(input("Введите длину: "))
9      # Проверить требуемую длину.
10     if len(s) >= n:
11         print(
12             "Заданная длина должна быть больше длины предложения",
13             file=sys.stderr
14         )
15         exit(1)
16     # Разделить предложение на слова.
17     words = s.split(' ')
18     # Проверить количество слов в предложении.
19     if len(words) < 2:
20         print(
21             "Предложение должно содержать несколько слов",
22             file=sys.stderr
23         )
24         exit(1)
25     # Количество пробелов для добавления.
26     delta = n
27     for word in words:
28         delta -= len(word)
29     # Количество пробелов на каждое слово.

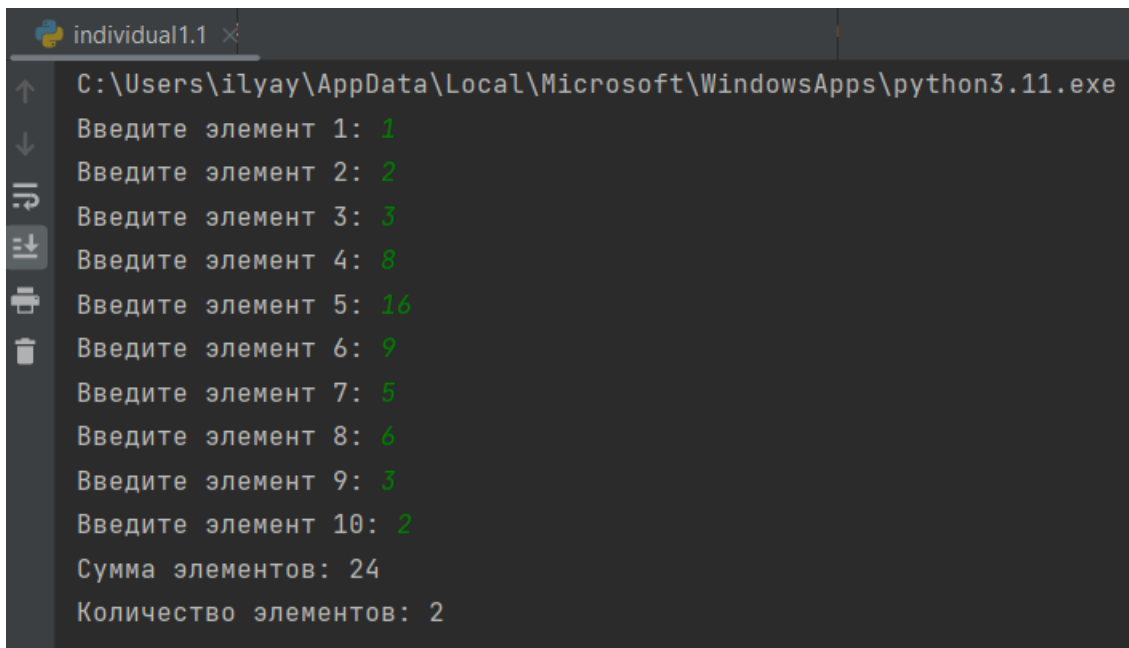
```

Рисунок 9 – Изменение длины строки (задание №3)

3. Выполним индивидуальные задания:

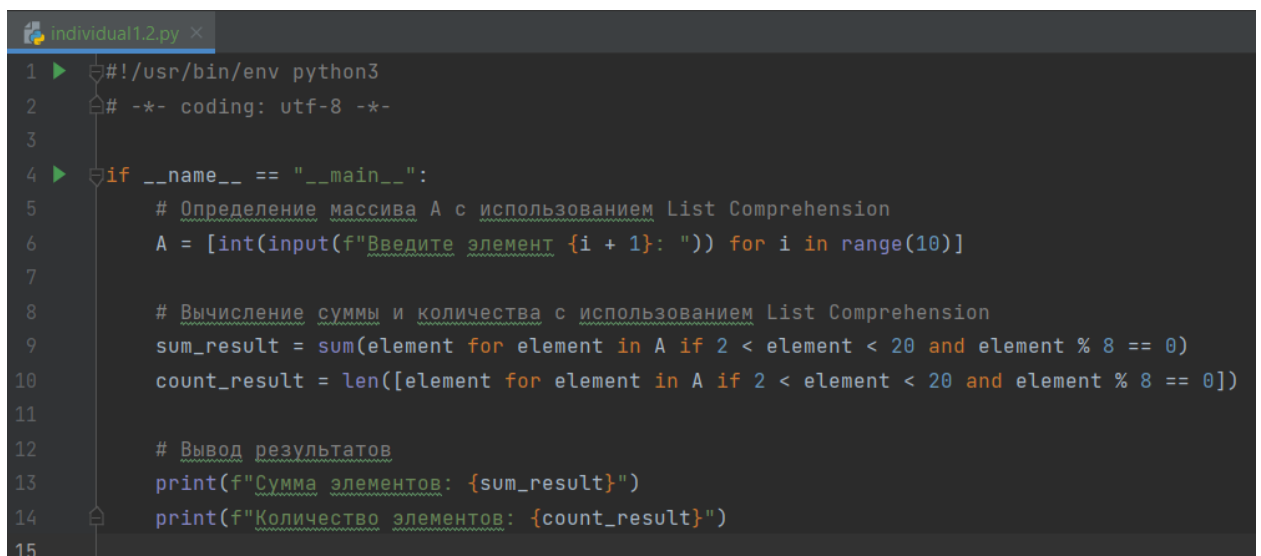
```
individual1.1.py x
1  ▶  #!/usr/bin/env python3
2    # -*- coding: utf-8 -*-
3
4  ▶  if __name__ == "__main__":
5      A = []
6      for i in range(10):
7          elem = int(input(f"Введите элемент {i + 1}: "))
8          A.append(elem)
9
10     # Инициализация суммы, количества итераций
11     sum_result = 0
12     count_result = 0
13
14     # Цикл для обхода элементов массива
15     for element in A:
16         if 2 < element < 20 and element % 8 == 0:
17             sum_result += element
18             count_result += 1
19
20     # Вывод результатов
21     print(f"Сумма элементов: {sum_result}")
22     print(f"Количество элементов: {count_result}")
23
```

Рисунок 5 – Нахождение суммы и количества элементов списка, больших 2 и меньших 20 и кратных 8 с использованием цикла (задание №1)



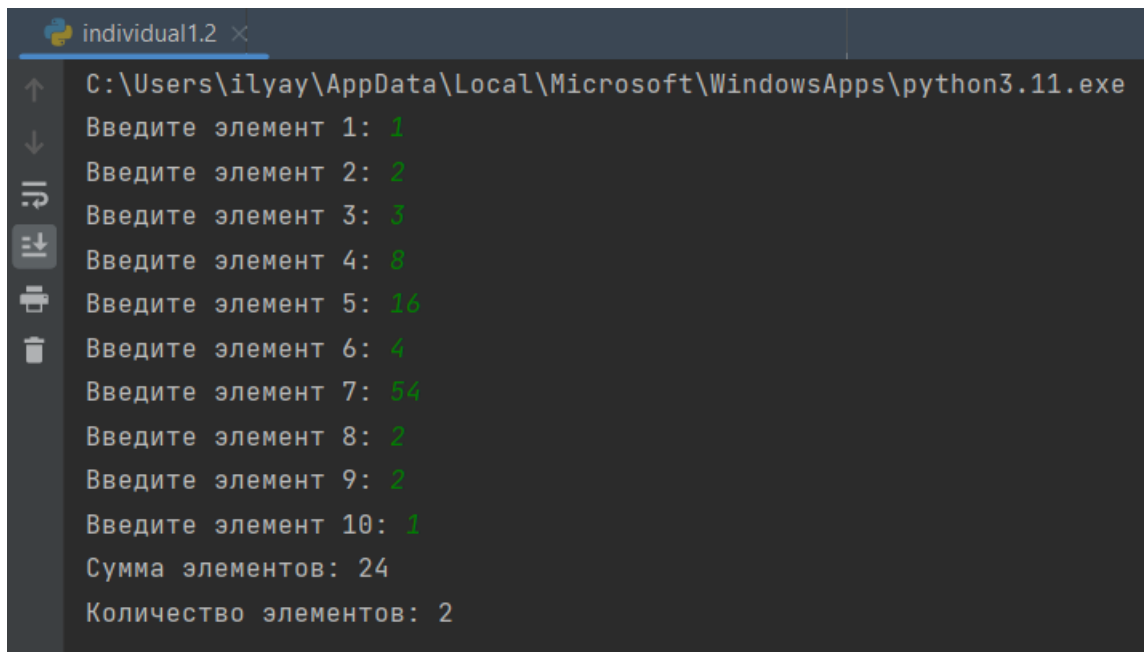
```
individual1.1 x
C:\Users\ilyay\AppData\Local\Microsoft\WindowsApps\python3.11.exe
Введите элемент 1: 1
Введите элемент 2: 2
Введите элемент 3: 3
Введите элемент 4: 8
Введите элемент 5: 16
Введите элемент 6: 9
Введите элемент 7: 5
Введите элемент 8: 6
Введите элемент 9: 3
Введите элемент 10: 2
Сумма элементов: 24
Количество элементов: 2
```

Рисунок 6 – Вывод программы (задание №1)



```
individual1.2.py x
1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 ▶ if __name__ == "__main__":
5     # Определение массива A с использованием List Comprehension
6     A = [int(input(f"Введите элемент {i + 1}: ")) for i in range(10)]
7
8     # Вычисление суммы и количества с использованием List Comprehension
9     sum_result = sum(element for element in A if 2 < element < 20 and element % 8 == 0)
10    count_result = len([element for element in A if 2 < element < 20 and element % 8 == 0])
11
12    # Вывод результатов
13    print(f"Сумма элементов: {sum_result}")
14    print(f"Количество элементов: {count_result}")
15
```

Рисунок 7 – Нахождение суммы и количества элементов списка, больших 2 и меньших 20 и кратных 8 с использованием List Comprehension (задание №1)

A screenshot of a Python terminal window titled 'individual1.2'. The window shows the execution of a program that prompts the user to enter 10 elements. The user has entered the following values: 1, 2, 3, 8, 16, 4, 54, 2, 2, 1. The program then calculates the sum of these elements (24) and the number of elements (10).

```
individual1.2 ×
C:\Users\ilyay\AppData\Local\Microsoft\WindowsApps\python3.11.exe
Введите элемент 1: 1
Введите элемент 2: 2
Введите элемент 3: 3
Введите элемент 4: 8
Введите элемент 5: 16
Введите элемент 6: 4
Введите элемент 7: 54
Введите элемент 8: 2
Введите элемент 9: 2
Введите элемент 10: 1
Сумма элементов: 24
Количество элементов: 10
```

Рисунок 8 – Вывод программы (задание №1)

12. В списке, состоящем из вещественных элементов, вычислить:

1. количество элементов списка, лежащих в диапазоне от A до B ;
2. сумму элементов списка, расположенных после максимального элемента.

Упорядочить элементы списка по убыванию модулей элементов.

```
individual2.py x
1  ▶  #!/usr/bin/env python3
2  ▢  #- coding: utf-8 -*-
3
4  ▶  if __name__ == "__main__":
5      # Ввод списка вещественных элементов
6      float_list = []
7      num_elements = int(input("Введите количество элементов списка: "))
8      for i in range(num_elements):
9          elem = float(input(f"Введите элемент {i + 1}: "))
10         float_list.append(elem)
11
12     # Ввод диапазона [A, B]
13     A = float(input("Введите значение A: "))
14     B = float(input("Введите значение B: "))
15
16     # Количество элементов в диапазоне [A, B]
17     count_in_range = 0
18     for elem in float_list:
19         if A <= elem <= B:
20             count_in_range += 1
21
22     # Сумма элементов после максимального элемента
23     max_value = max(float_list)
24     max_index = float_list.index(max_value)
25     sum_after_max = 0
26     if max_index < num_elements - 1:
27         sum_after_max = sum(float_list[max_index + 1:])
28
29     # Упорядочивание элементов по убыванию модулей
30     float_list.sort(key=lambda x: abs(x), reverse=True)
31
32     # Вывод результатов
33     print(f"1. Количество элементов в диапазоне от {A} до {B}: {count_in_range}")
34     print(f"2. Сумма элементов списка, расположенных после максимального элемента: {sum_after_max}")
35     print("Упорядоченные элементы по убыванию модулей:", float_list)
36
```

Рисунок 9 – Выполнение задания в соответствии с условиями (задание №2)

```
individual2 x
C:\Users\ilyay\AppData\Local\Microsoft\WindowsApps\python3.11.exe "C:/Users/
Введите количество элементов списка: 5
Введите элемент 1: 1.2
Введите элемент 2: 34.4
Введите элемент 3: 43.4
Введите элемент 4: 3.4
Введите элемент 5: 8.9
Введите значение A: 2
Введите значение B: 5
1. Количество элементов в диапазоне от 2.0 до 5.0: 1
2. Сумма элементов списка, расположенных после максимального элемента: 12.3
Упорядоченные элементы по убыванию модулей: [43.4, 34.4, 8.9, 3.4, 1.2]

Process finished with exit code 0
```

Рисунок 10 – Вывод программы (задание №2)

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

```
ilyay@DESKTOP-FF1JT6S MINGW64 ~/OneDrive/Рабочий стол/Основы программной инженерии/lr2_4 (develop)
$ git log
commit 789f64de007fae471d78a178d7763de758cf9aca (HEAD -> develop)
Author: dexstrong <ilya.yurev.04@inbox.ru>
Date: Mon Nov 27 19:18:19 2023 +0300

    final changes

commit ba159f9147ecc816d9627c0cbb3f3c6fa9889aae (origin/main, origin/HEAD, main)
Author: Ilya Yurev <112946692+daxstrong@users.noreply.github.com>
Date: Mon Nov 27 18:17:43 2023 +0300

    Initial commit
```

Рисунок 11 – Коммиты ветки develop

```
ilyay@DESKTOP-FF1JT6S MINGW64 ~/OneDrive/Рабочий стол/Основы программной инженерии/lr2_4 (main)
$ git merge develop
Updating ba159f9..789f64d
Fast-forward
 .idea/.gitignore           | 8 ++++++
 .idea/inspectionProfiles/profiles_settings.xml | 6 +++++
 .idea/lr2_4.iml           | 8 ++++++
 .idea/misc.xml            | 4 +++
 .idea/modules.xml         | 8 ++++++
 .idea/vcs.xml             | 6 +++++
 individual1.1.py          | 22 +++++++++++++++++++++
 individual1.2.py          | 14 ++++++++
 individual2.py            | 35 ++++++++++++++++++++++
 task1.py                  | 20 ++++++++
 task2.py                  | 34 ++++++++
11 files changed, 165 insertions(+)
create mode 100644 .idea/.gitignore
create mode 100644 .idea/inspectionProfiles/profiles_settings.xml
create mode 100644 .idea/lr2_4.iml
create mode 100644 .idea/misc.xml
create mode 100644 .idea/modules.xml
create mode 100644 .idea/vcs.xml
create mode 100644 individual1.1.py
create mode 100644 individual1.2.py
create mode 100644 individual2.py
create mode 100644 task1.py
create mode 100644 task2.py
```

Рисунок 12 – Слияние веток main и develop

```
ilyay@DESKTOP-FF1JT6S MINGW64 ~/OneDrive/Рабочий стол/Основы программной инженерии/lr2_4 (main)
$ git push origin main
Enumerating objects: 16, done.
Counting objects: 100% (16/16), done.
Delta compression using up to 12 threads
Compressing objects: 100% (14/14), done.
Writing objects: 100% (15/15), 3.91 KiB | 3.91 MiB/s, done.
Total 15 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/daxstrong/lr2_4.git
ba159f9..789f64d main -> main
```

Рисунок 17 – Отправка изменений на удаленный репозиторий

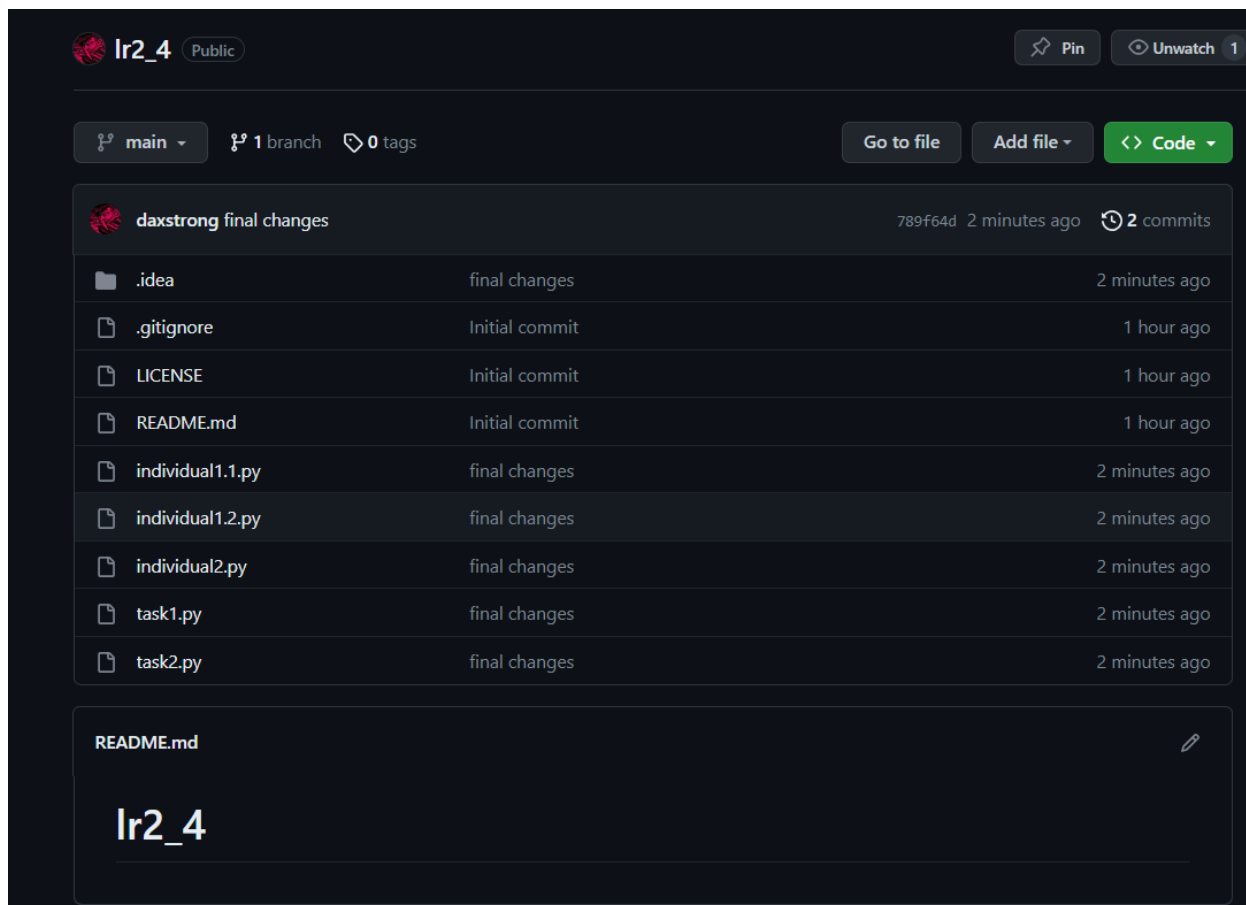


Рисунок 18 – Изменения удаленного репозитория

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 подписки 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.