

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 №2.5
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 с кортежами в языке Python.

Цель работы: приобретение навыков по работе с кортежами 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_5

✓ lr2_5 is available.

Great repository names are short and memorable. Need inspiration? How about [fluffy-octo-bassoon](#) ?

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

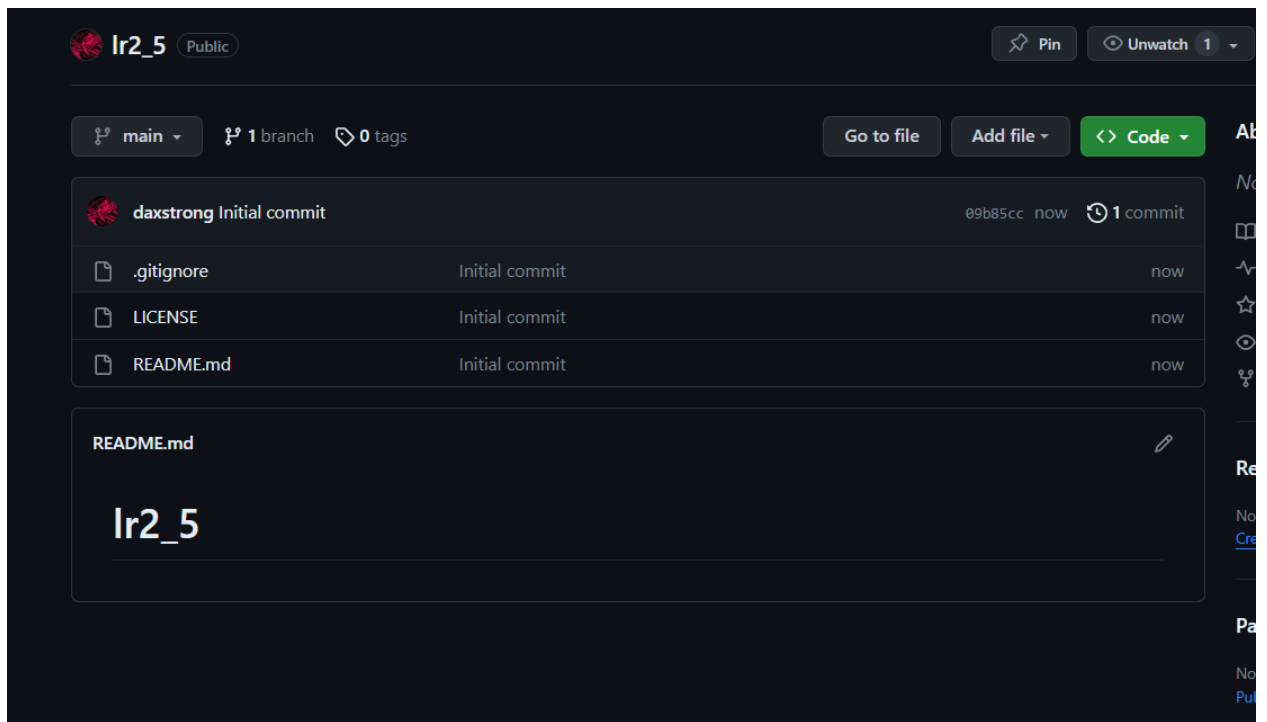


Рисунок 2 – Созданный репозиторий

```
ilyay@DESKTOP-FF1JT6S MINGW64 ~/OneDrive/Рабочий стол/Основы программной инженерии
$ git clone https://github.com/daxstrong/lr2_5.git
Cloning into 'lr2_5'...
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_5 (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
7  ▶  if __name__ == '__main__':
8      # Ввести кортеж одной строкой.
9      A = tuple(map(int, input().split()))
10     # Проверить количество элементов кортежа.
11     if len(A) != 10:
12         print("Неверный размер кортежа", file=sys.stderr)
13         exit(1)
14
15     # Найти искомую сумму.
16     s = 0
17     for item in A:
18         if abs(item) < 5:
19             s += item
20
21     print(s)
22
```

Рисунок 5 – Сумма элементов, меньших по модулю 5 с использованием кортежа (задание №1)

```
task1 x
C:\Users\ilyay\AppData\Local\Microsoft\WindowsApps\python3.11.exe
2 3 4 5 -1 -2 7 8 9 4
10
```

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

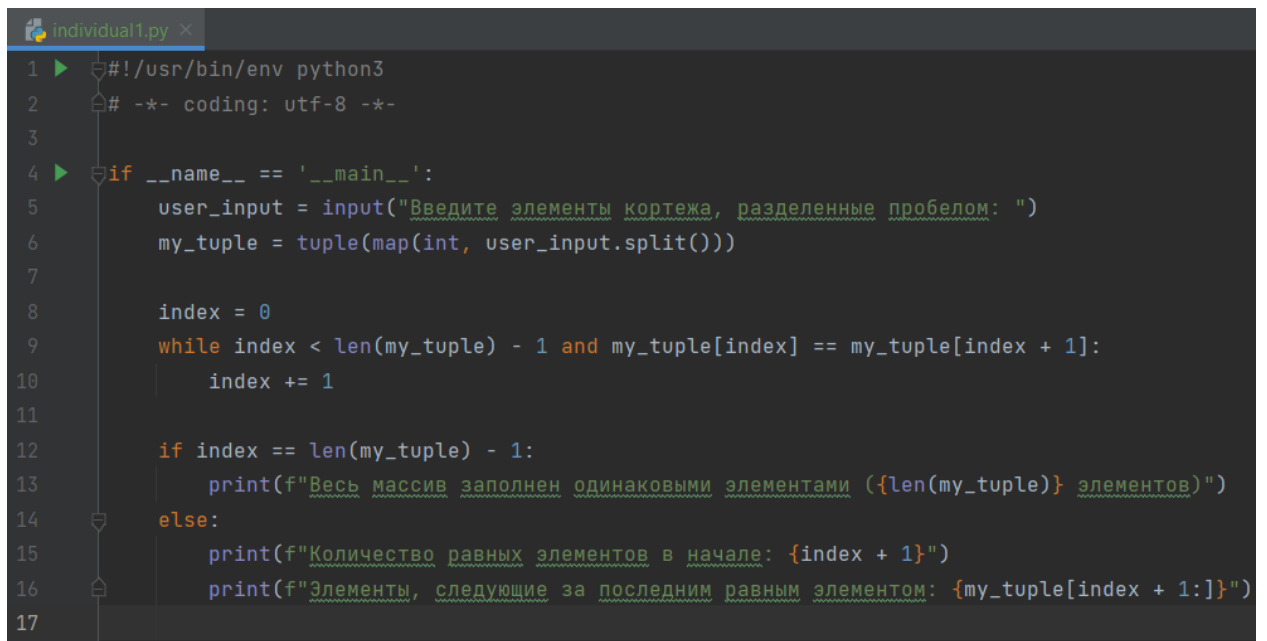
Рисунок 7 — Сумма элементов, меньших по модулю 5 при помощи списковых включений (задание №2)

```
task2 x
C:\Users\ilyay\AppData\Local\Microsoft\WindowsApps\python3.11.exe
2 3 -4 3 2 1 7 8 1 -3
5
```

Рисунок 8 – Вывод программы (задание №2)

3. Выполним индивидуальные задания:

12. В начале кортежа записано несколько равных между собой элементов. Определить количество таких элементов и вывести все элементы, следующие за последним из них. Рассмотреть возможность того, что весь массив заполнен одинаковыми элементами. Условный оператор не использовать.

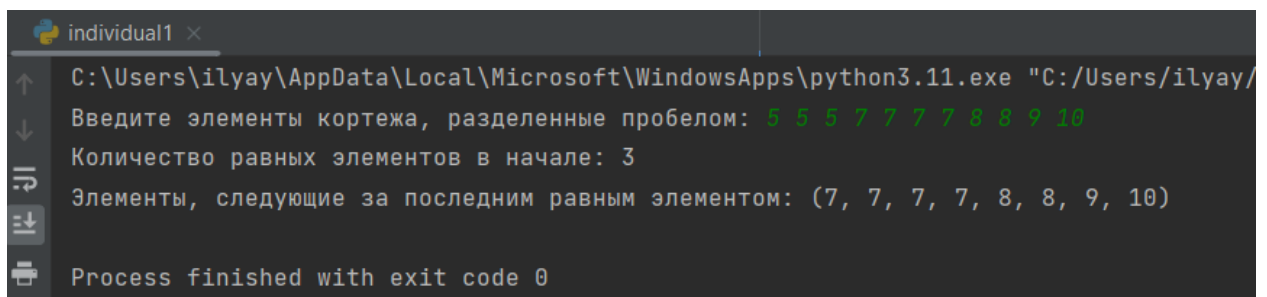


```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  if __name__ == '__main__':
5      user_input = input("Введите элементы кортежа, разделенные пробелом: ")
6      my_tuple = tuple(map(int, user_input.split()))
7
8      index = 0
9      while index < len(my_tuple) - 1 and my_tuple[index] == my_tuple[index + 1]:
10         index += 1
11
12     if index == len(my_tuple) - 1:
13         print(f"Весь массив заполнен одинаковыми элементами ({len(my_tuple)} элементов)")
14     else:
15         print(f"Количество равных элементов в начале: {index + 1}")
16         print(f"Элементы, следующие за последним равным элементом: {my_tuple[index + 1:]}")
17

```

Рисунок 9 – Определение количества равных элементов и вывод последующих за последним равным элементом в кортеже



```

C:\Users\ilyay\AppData\Local\Microsoft\WindowsApps\python3.11.exe "C:/Users/ilyay/
Введите элементы кортежа, разделенные пробелом: 5 5 5 7 7 7 7 8 8 9 10
Количество равных элементов в начале: 3
Элементы, следующие за последним равным элементом: (7, 7, 7, 7, 8, 8, 9, 10)
Process finished with exit code 0

```

Рисунок 10 – Вывод программы

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

```

ilyay@DESKTOP-FF1JT6S MINGW64 ~/OneDrive/Рабочий стол/Основы программной инженерии/lr2_5 (main)
$ git merge develop
Updating 09b85cc..27f87f1
Fast-forward
 .idea/.gitignore | 8 ++++++++
 .idea/inspectionProfiles/profiles_settings.xml | 6 ++++++
 .idea/lr2_5.iml | 8 ++++++++
 .idea/misc.xml | 4 +++++
 .idea/modules.xml | 8 ++++++++
 .idea/vcs.xml | 6 ++++++
 individual1.py | 16 ++++++
 task1.py | 21 ++++++
 task2.py | 17 ++++++
9 files changed, 94 insertions(+)
create mode 100644 .idea/.gitignore
create mode 100644 .idea/inspectionProfiles/profiles_settings.xml
create mode 100644 .idea/lr2_5.iml
create mode 100644 .idea/misc.xml
create mode 100644 .idea/modules.xml
create mode 100644 .idea/vcs.xml
create mode 100644 individual1.py
create mode 100644 task1.py
create mode 100644 task2.py

```

Рисунок 11 – Слияние веток main и develop

```

ilyay@DESKTOP-FF1JT6S MINGW64 ~/OneDrive/Рабочий стол/Основы программной инженерии/lr2_5 (main)
$ git push origin main
Enumerating objects: 14, done.
Counting objects: 100% (14/14), done.
Delta compression using up to 12 threads
Compressing objects: 100% (12/12), done.
Writing objects: 100% (13/13), 2.45 KiB | 2.45 MiB/s, done.
Total 13 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/daxstrong/lr2_5.git
    09b85cc..27f87f1  main -> main

```

Рисунок 12 – Отправка изменений на удаленный репозиторий

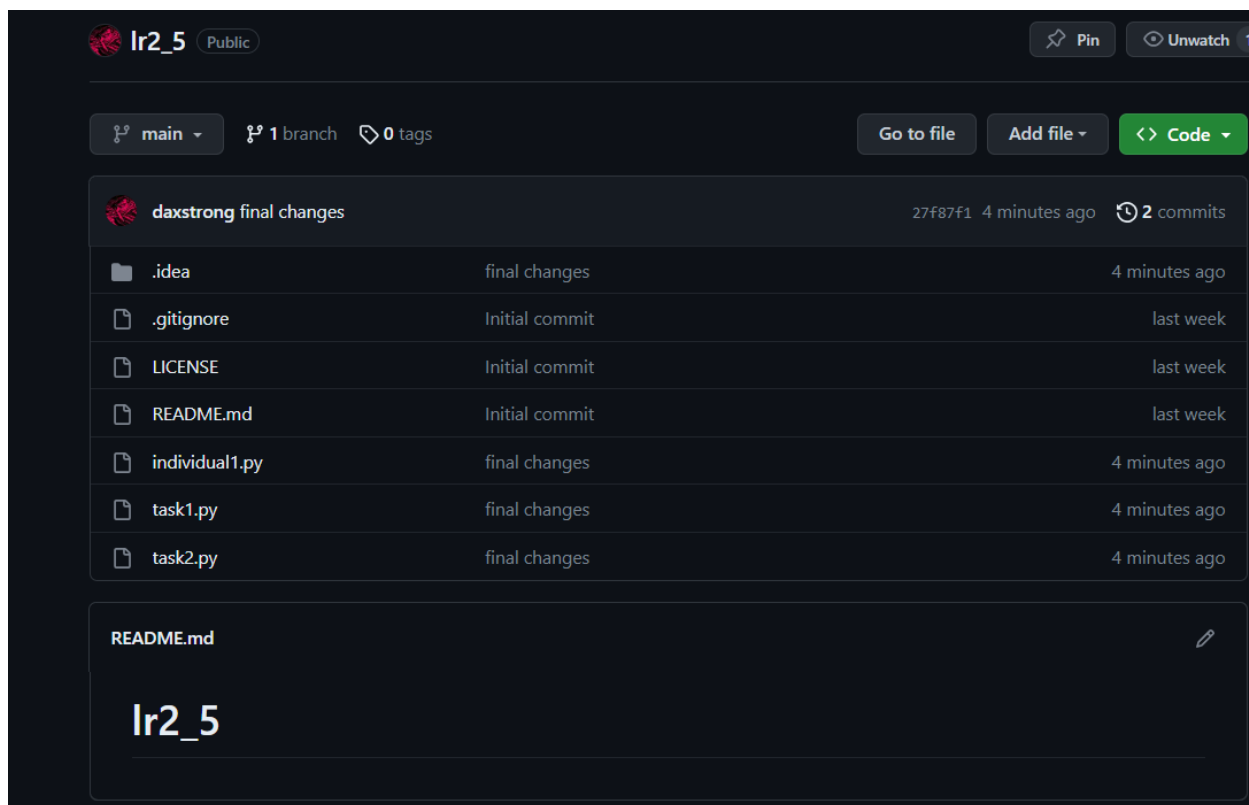


Рисунок 13 – Изменения удаленного репозитория

Ответы на контрольные вопросы:

1. Что такое списки в языке Python?

Список в Python – это упорядоченная коллекция элементов, которая позволяет хранить различные типы данных. Он создается с использованием квадратных скобок [] и элементы списка разделяются запятыми.

2. Каково назначение кортежей в языке Python?

Кортеж – это структура данных, похожая на список, но неизменяемая. Основное их предназначение - хранить неизменяемые коллекции объектов.

3. Как осуществляется создание кортежей?

Кортеж создается с использованием круглых скобок () и элементы кортежа разделяются запятыми. Например: `my_tuple = (1, 2, 3)`

4. Как осуществляется доступ к элементам кортежа?

Элементы кортежа доступны по индексам, начиная с 0. Например, `my_tuple[0]` вернет первый элемент кортежа.

5. Зачем нужна распаковка (деструктуризация) кортежа?

Распаковка кортежа позволяет присвоить значения элементов кортежа переменным одновременно. Например: `a, b, c = my_tuple`.

6. Какую роль играют кортежи в множественном присваивании?

Кортежи позволяют одновременно присваивать значения нескольким переменным, что удобно при обмене значениями переменных или при работе с функциями, возвращающими кортеж.

7. Как выбрать элементы кортежа с помощью среза?

Элементы кортежа могут быть выбраны с использованием срезов, похожих на списки. Например: `my_tuple[1:3]` вернет подкортеж с элементами с индексами от 1 до 2.

8. Как выполняется конкатенация и повторение кортежей?

Кортежи могут быть сконкатенированы с помощью оператора +, а также повторены с помощью оператора *.

9. Как выполняется обход элементов кортежа?

Элементы кортежа можно перебирать с помощью цикла for. Например:
for item in my_tuple:

```
    print(item)
```

10. Как проверить принадлежность элемента кортежу?

Для проверки принадлежности элемента кортежу можно использовать оператор in. Например:

```
if 1 in my_tuple:
```

```
    print(1)
```

11. Какие методы работы с кортежами Вам известны?

В отличие от списков, кортежи являются неизменяемыми, поэтому у них меньше методов. Некоторые из них: count() для подсчета вхождений элемента и index() для поиска индекса элемента.

12. Допустимо ли использование функций агрегации таких как len(), sum() и т. д. при работе с кортежами?

Да, функции агрегации, такие как len(), sum(), min(), max() и другие, могут быть использованы с кортежами для получения информации о них.

13. Как создать кортеж с помощью спискового включения.

В Python можно создать кортеж с помощью генератора кортежей. Например: my_tuple = tuple(x for x in range(5)).