

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ  
Федеральное государственное автономное  
образовательное учреждение высшего образования  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Кафедра инфокоммуникаций**

**Отчет по лабораторной работе № 2.5  
по дисциплине «Основы программной инженерии»**

Выполнил студент группы ПИЖ-б-о-22-1

Душин Александр Владимирович.

Подпись студента \_\_\_\_\_

Работа защищена « » \_\_\_\_\_ 20\_\_ г.

Проверил Воронкин Р.А. \_\_\_\_\_

(подпись)

Ставрополь 2023

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 с использованием лицензии 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 (\*).

Owner \*

 MrPlatynum ▾

Repository name \*

ProgrammEngineering8

✓ ProgrammEngineering8 is available.

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

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 ▾

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

 You are creating a public repository in your personal account.

Create repository

Рисунок 1 – Создание общедоступного репозитория на GitHub с заданными настройками

```
Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents
$ git clone https://github.com/MrPlatinum/ProgrammEngineering8.git
Cloning into 'ProgrammEngineering8'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.
```

Рисунок 2 – Клонирование созданного репозитория на локальный компьютер



Рисунок 3 – файл .gitignore

```
Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering8 (main)
$ git checkout -b develop
Switched to a new branch 'develop'

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering8 (develop)
$
```

Рисунок 4 – организация репозитория в соответствии с моделью ветвления git flow

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

```

example1.1.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 (задание №1)

```

"C:\Program Files\Python312\python.exe" C:/Users/Alexander/Documents/ProgrammEngineering8/example1.1.py
1 2 3 4 5 6 7 8 9 0
10

```

Рисунок 6 – Вывод программы – 1 данные (задание №1.1)

```

"C:\Program Files\Python312\python.exe" C:/Users/Alexander/Documents/ProgrammEngineering8/example1.1.py
0 0 1 1 2 2 3 3 4 4
20

```

Рисунок 7 – Вывод программы – 2 данные (задание №1.1)

```

"C:\Program Files\Python312\python.exe" C:/Users/Alexander/Documents/ProgrammEngineering8/example1.1.py
0 0 0 0 0 0 0 0 0 0
0

```

Рисунок 8 – Вывод программы – 3 данные (задание №1.1)

```
example1.2.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 9 – Сумма элементов, меньших по модулю 5 при помощи списковых включений (задание №2)

```
"C:\Program Files\Python312\python.exe" C:/Users/Alexander/Documents/ProgrammEngineering8/example1.2.py
1 2 3 4 5 6 7 8 9 0
10
```

Рисунок 10 – Вывод программы – 1 данные (задание №1.2)

```
"C:\Program Files\Python312\python.exe" C:/Users/Alexander/Documents/ProgrammEngineering8/example1.2.py
0 0 1 1 2 2 3 3 4 4
20
```

Рисунок 11 – Вывод программы – 2 данные (задание №1.2)

```
"C:\Program Files\Python312\python.exe" C:/Users/Alexander/Documents/ProgrammEngineering8/example1.2.py
0 0 0 0 0 0 0 0 0 0
0
```

Рисунок 12 – Вывод программы – 3 данные (задание №1.2)

3. Выполним индивидуальные задания:

```
individual1.py x
1 ▶ #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import sys
5
6 ▶ if __name__ == '__main__':
7     numbers = tuple(map(int, input("Введите кортеж целых чисел через пробел: ").split()))
8
9     # Проверить количество элементов кортежа.
10    if len(numbers) < 2:
11        print("Кортеж должен содержать как минимум два элемента", file=sys.stderr)
12        exit(1)
13
14    found = False
15    for i, num in enumerate(numbers[:-1]):
16        if num % 2 != 0 and numbers[i + 1] % 2 != 0:
17            print(f"Первая пара соседних нечетных чисел найдена в позициях {i} и {i + 1}")
18            found = True
19            break
20
21    # Вывод результата, если пара не найдена.
22    if not found:
23        print("В кортеже нет соседних нечетных чисел")
24
```

Рисунок 13 – Пара соседних нечётных чисел (задание №1)

```
"C:\Program Files\Python312\python.exe" C:/Users/Alexander/Documents/ProgrammEngineering8/individual1.py
Введите кортеж целых чисел через пробел: 0 0 1 2 3 4 5 7 8 9
Первая пара соседних нечетных чисел найдена в позициях 6 и 7
```

Рисунок 14 – Вывод программы (задание №1)

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

```
Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering8 (develop)
$ git log
commit f1f5327134c3616751e05ecef50e13f2f2071589 (HEAD -> develop)
Author: MrPlatynum <dushinsasha4@gmail.com>
Date: Sun Nov 26 14:03:40 2023 +0300

    финальные изменения

commit 632bdf0e447f60cccead3eb518631b4d4a366dd0 (origin/main, origin/HEAD, main)
Author: MrPlatynum <71084177+MrPlatynum@users.noreply.github.com>
Date: Sun Nov 26 12:26:47 2023 +0300

    Initial commit
```

Рисунок 15 – Коммиты ветки develop во время выполнения лабораторной работы

```

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering8 (develop)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering8 (main)
$ git merge develop
Updating 632bdf0..f1f5327
Fast-forward
 .idea/.gitignore           | 8 ++++++
 .idea/ProgrammEngineering8.iml | 8 ++++++
 .idea/inspectionProfiles/profiles_settings.xml | 6 +++++
 .idea/misc.xml             | 4 ++++
 .idea/modules.xml          | 8 ++++++
 .idea/vcs.xml              | 6 +++++
 example1.1.py              | 21 ++++++
 example1.2.py              | 17 ++++++
 individual1.py             | 23 ++++++
9 files changed, 101 insertions(+)
create mode 100644 .idea/.gitignore
create mode 100644 .idea/ProgrammEngineering8.iml
create mode 100644 .idea/inspectionProfiles/profiles_settings.xml
create mode 100644 .idea/misc.xml
create mode 100644 .idea/modules.xml
create mode 100644 .idea/vcs.xml
create mode 100644 example1.1.py
create mode 100644 example1.2.py
create mode 100644 individual1.py

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering8 (main)
$

```

Рисунок 16 – Слияние ветки develop в ветку main

```

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering8 (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), 2.58 KiB | 2.58 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/MrPlatinum/ProgrammEngineering8.git
  632bdf0..f1f5327  main -> main

Alexander@DESKTOP-IUJLQQ3 MINGW64 ~/Documents/ProgrammEngineering8 (main)
$

```

Рисунок 17 – Отправка на удаленный репозиторий

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))`.