

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 №8**  
**дисциплины**  
**«Основы кроссплатформенного программирования»**

Выполнил:  
Ляш Денис Александрович  
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 \_\_\_\_\_

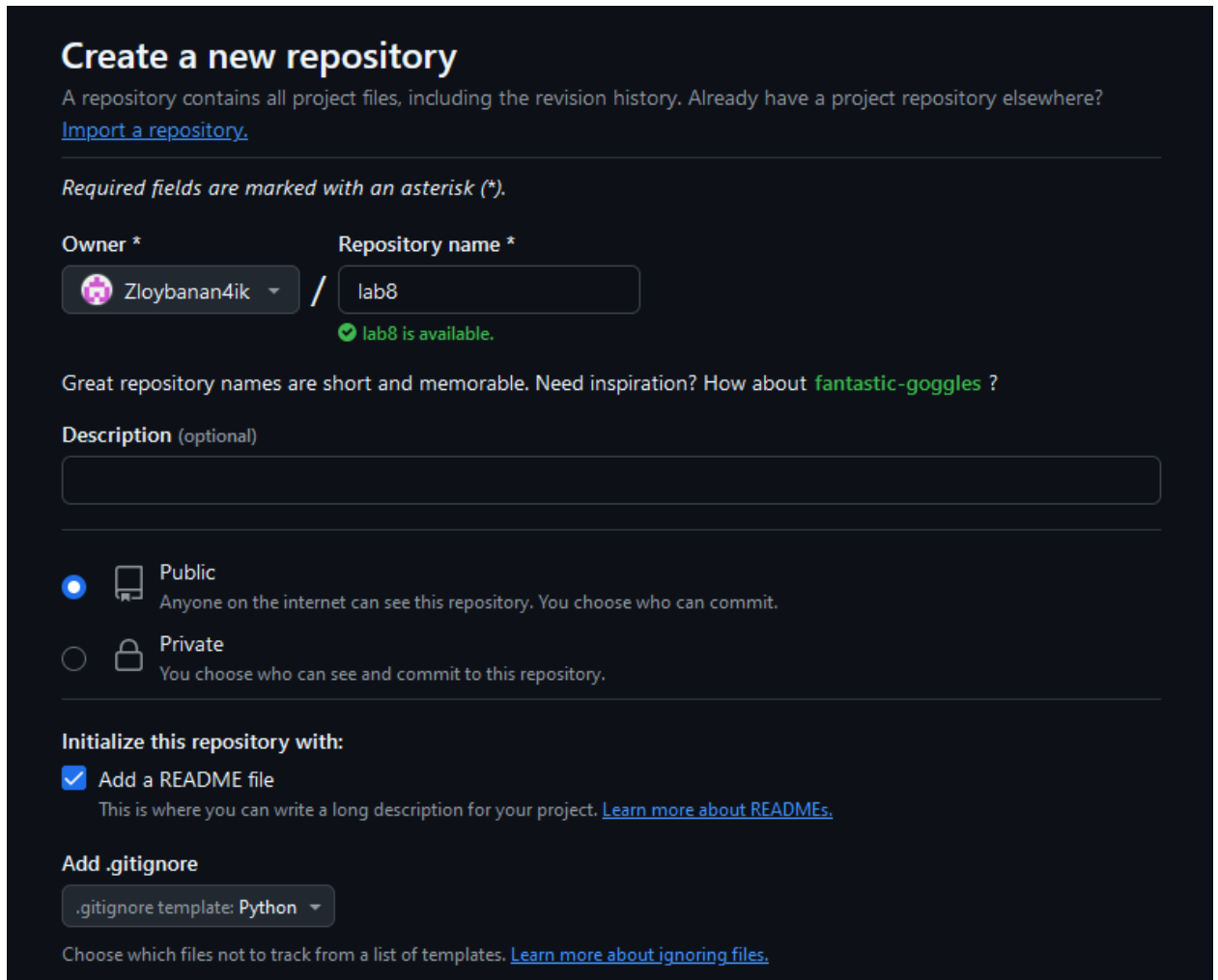
Ставрополь, 2024 г.

## Тема: Работа с кортежами в языке Python

**Цель:** приобретение навыков по работе с кортежами при написании программ с помощью языка программирования Python версии 3.x.

### Порядок выполнения работы:

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




**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 \*** **Repository name \***


 Zloybanan4ik / lab8

✔ lab8 is available.

Great repository names are short and memorable. Need inspiration? How about **fantastic-goggles** ?

**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.](#)

Рисунок 1. Репозиторий

### 4. Проработал примеры

```
import random
```

```
def sum_elements_less_than_5(tuple_A):  
    """
```

Вычисляет сумму элементов кортежа, модуль которых меньше 5.

Args:

tuple\_A: Кортеж чисел.

Returns:

Сумма элементов, модуль которых меньше 5. Возвращает 0, если кортеж пуст.

```
"""
```

```
if not tuple_A: # проверка на пустой кортеж
```

```
    return 0
```

```
sum_elements = sum(x for x in tuple_A if abs(x) < 5)
```

```
return sum_elements
```

#Пример использования:

# Создаем кортеж из 10 случайных чисел

```
tuple_A = tuple(random.randint(-10, 10) for _ in range(10))
```

```
print("Кортеж:", tuple_A)
```

# Вычисляем сумму элементов

```
sum_less_than_5 = sum_elements_less_than_5(tuple_A)
```

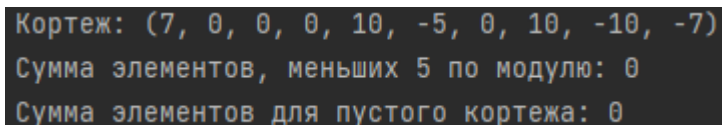
```
print("Сумма элементов, меньших 5 по модулю:", sum_less_than_5)
```

#Пример с пустым кортежем

```
tuple_B = ()
```

```
sum_less_than_5_B = sum_elements_less_than_5(tuple_B)
```

```
print("Сумма элементов для пустого кортежа:", sum_less_than_5_B)
```



```
Кортеж: (7, 0, 0, 0, 10, -5, 0, 10, -10, -7)
Сумма элементов, меньших 5 по модулю: 0
Сумма элементов для пустого кортежа: 0
```

Рисунок 3. Пример

```
data = tuple(map(int, input("Enter elements of the tuple separated by spaces: ").split()))
```

```
first_element = data[0]
```

```
count = sum(1 for i in data if i == first_element)
```

```
count = min(count, len(data))
```

```
remaining_elements = data[count:]
```

```
print(f'Count of identical elements at the beginning: {count}')
```

```
print(f'Elements after the last identical element: {remaining_elements}')
```

```
Enter elements of the tuple separated by spaces: 5 5 5 5 54 6 44
Count of identical elements at the beginning: 4
Elements after the last identical element: (54, 6, 44)
```

Рисунок 4. Задание

## 5. Зафиксировал изменения

```
user@DESKTOP-VI4I94J MINGW64 ~/workspace/lab8 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   .gitignore
        new file:   example.py
        new file:   individual.py

user@DESKTOP-VI4I94J MINGW64 ~/workspace/lab8 (main)
$ git commit -m "commit"
[main 707c2e8] commit
3 files changed, 42 insertions(+)
create mode 100644 example.py
create mode 100644 individual.py
```

Рисунок 5. Изменения

## Ответы на контрольные вопросы:

### 1. Что такое кортежи в языке Python?

Кортежи — это неизменяемые последовательности, которые могут содержать элементы различных типов. Они используются для хранения коллекций данных, аналогично спискам, но с тем отличием, что их содержимое нельзя изменить после создания.

### 2. Каково назначение кортежей в языке Python?

Кортежи используются для хранения фиксированных наборов данных, которые не должны изменяться. Они могут быть полезны для группировки связанных данных и передачи их в функции.

### 3. Как осуществляется создание кортежей?

Кортежи создаются с помощью круглых скобок (), также можно создать кортеж без скобок, просто перечислив элементы через запятую. Для создания кортежа с одним элементом необходимо добавить запятую.

#### **4. Как осуществляется доступ к элементам кортежа?**

Доступ к элементам кортежа осуществляется с помощью индексов, аналогично спискам

#### **5. Зачем нужна распаковка (деструктуризация) кортежа?**

Распаковка кортежа позволяет присвоить значения его элементам переменным в одном выражении, что делает код более читаемым и удобным.

#### **6. Какую роль играют кортежи в множественном присваивании?**

Кортежи позволяют удобно присваивать несколько значений нескольким переменным одновременно, что упрощает код и делает его более понятным.

#### **7. Как выбрать элементы кортежа с помощью среза?**

Срезы для кортежей работают так же, как и для списков: `my tuple[start:end]` вернет элементы с индексами от `start` до `end-1`.

#### **8. Как выполняется конкатенация и повторение кортежей?**

Конкатенация осуществляется с помощью оператора `+`, а повторение — с помощью оператора `*`

#### **9. Как выполняется обход элементов кортежа?**

Обход элементов кортежа можно осуществить с помощью цикла `for`.

#### **10. Как проверить принадлежность элемента кортежу?**

Используя оператор `in` `if element in my tuple`

#### **11. Какие методы работы с кортежами Вам известны?**

Кортежи имеют ограниченное количество методов, `count` — подсчитывает количество вхождений элемента. `index` — возвращает индекс первого вхождения элемента.

**12. Допустимо ли использование функций агрегации таких как `len`, `sum` и тд при работе с кортежами?**

Да, функции агрегации, такие как `len`, `sum`, `min`, `max`, могут использоваться с кортежами так же, как и со списками.

### **13. Как создать кортеж с помощью спискового включения?**

Кортеж можно создать с помощью генератора кортежей, используя круглые скобки

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

**Ссылка на репозиторий:** <https://github.com/Zloybanan4ik/lab8.git>