

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ  
ФЕДЕРАЦИИ**

**Федеральное государственное автономное образовательное учреждение  
высшего образования**

**«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Институт цифрового развития**

**Кафедра информационных систем и технологий**

Отчет по лабораторной работе №8.

Дисциплина: «Основы программной инженерии»

**Выполнил:**

Студент группы ПИЖ-б-о-22-1,  
направление подготовки: 09.03.04

«Программная инженерия»

ФИО: Джараян Арег Александрович

**Проверил:**

Воронкин Р. А.

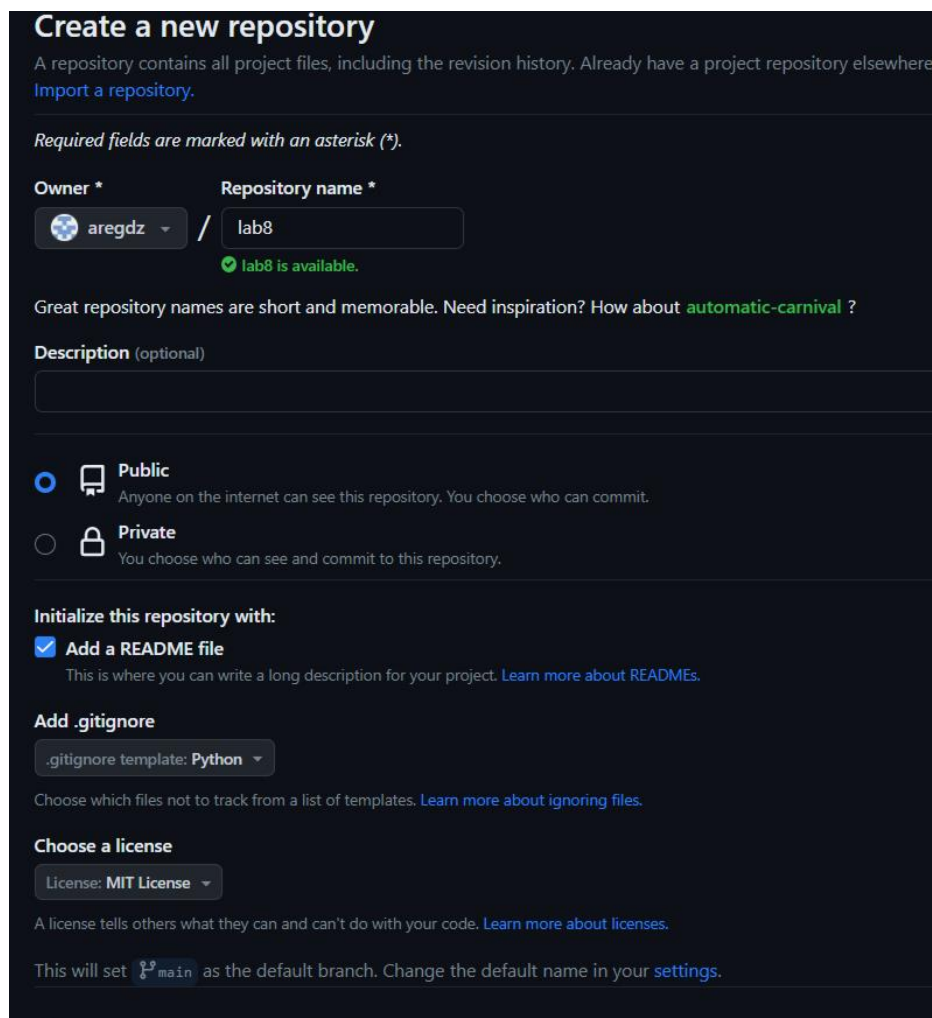
Ставрополь 2022

## Тема: Лабораторная работа 2.5 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..

Выполнение работы:

1. Изучил теоретический материал работы.
2. Создал репозиторий на git.hub.



The screenshot shows the 'Create a new repository' page on GitHub. The form is titled 'Create a new repository' and includes a subtitle: 'A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)'. Below this, a note states: 'Required fields are marked with an asterisk (\*)'. The form has two main sections: 'Owner \*' and 'Repository name \*'. The 'Owner' is set to 'aregdz' with a dropdown arrow. The 'Repository name' is 'lab8', and a green checkmark indicates 'lab8 is available.'. Below this, a tip says: 'Great repository names are short and memorable. Need inspiration? How about [automatic-carnival](#) ?'. The 'Description (optional)' field is empty. The 'Visibility' section has two options: 'Public' (selected with a radio button) and 'Private'. The 'Initialize this repository with:' section has a checked checkbox for 'Add a README file'. Below this, there is a section for 'Add .gitignore' with a dropdown menu set to 'Python'. The 'Choose a license' section has a dropdown menu set to 'MIT License'. At the bottom, it says: 'This will set `main` as the default branch. Change the default name in your [settings](#).'

Рисунок 1 – создание репозитория

3. Клонировал репозиторий.

```
MINGW64:/c:/Users/aregd/OneDrive/Рабочий стол/git8
aregd@DESKTOP-5KV9QA9 MINGW64 ~
$ cd "C:\Users\aregd\OneDrive\Рабочий стол\git8"

aregd@DESKTOP-5KV9QA9 MINGW64 ~/OneDrive/Рабочий стол/git8
$ git clone https://github.com/aregdz/lab8.git
Cloning into 'lab8'...
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.

aregd@DESKTOP-5KV9QA9 MINGW64 ~/OneDrive/Рабочий стол/git8
$ |
```

Рисунок 2 – клонирование репозитория 4.

Дополнить файл gitignore необходимыми правилами.

```
.gitignore – Блокнот
Файл Правка Формат Вид Справка
# Created by .ignore support plugin (hsz.mobi)
### Python template
# Byte-compiled / optimized / DLL files
__pycache__/
*.py[cod]
*$py.class

# C extensions
*.so

# Distribution / packaging
.Python
env/
build/
develop-eggs/
dist/
downloads/
```

Рисунок 3 – .gitignore для IDE PyCharm

5. Организовать свой репозиторий в соответствии с моделью ветвления git-flow.

```
aregd@DESKTOP-5KV9QA9 MINGW64 ~/OneDrive/Рабочий стол/git 5/lab5 (main)
$ git checkout -b develop
Switched to a new branch 'develop'

aregd@DESKTOP-5KV9QA9 MINGW64 ~/OneDrive/Рабочий стол/git 5/lab5 (develop)
$ |
```

Рисунок 4 – создание ветки develop

6. Проработал примеры из методички.

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6  if __name__ == '__main__':
7      # Ввести кортеж одной строкой.
8      A = tuple(map(int, input().split()))
9      # Проверить количество элементов кортежа.
10     if len(A) != 10:
11         print("Неверный размер кортежа", 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)
```

Рисунок 5 – пример 1

```
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 = sum(a for a in A if abs(a) < 5)
16     print(s)
```

Рисунок 6 – пример 1(2)

```

C:\Users\aregd\AppData\Local
1 2 3 4 5 6 7 8 9 0
10
Process finished with exit c

```

Рисунок 7 – пример выполнения примера 1

7. Если в кортеже есть хотя бы одна пара одинаковых соседних элементов, то напечатать все элементы, следующие за элементами первой из таких пар.

```

1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import sys
5
6  ▶ if __name__ == '__main__':
7      # Ввести кортеж одной строкой.
8      a = tuple(map(int, input().split()))
9      💡
10     # Найти одинаковые соседние элементы
11     l = 0
12     for i, item in enumerate(a):
13         if i < len(a)-1:
14             if item == a[i+1]:
15                 l = i + 2
16                 break
17         else:
18             break
19
20     # вывести элементы от l до конца
21     print(a[l:])
22

```

Рисунок 8 – индивидуальное задание

```

C:\Users\aregd\AppData\Local\Programs\Python\Python3
1 3 4 5 6 7 8 9 0 0 8 6 4 4 3 5 7 5 3 5 6 3
(8, 6, 4, 4, 3, 5, 7, 5, 3, 5, 6, 3)

Process finished with exit code 0

```

Рисунок 9 – пример выполнения индивидуального задания

9.Зафиксировал все изменения в github в ветке develop.

```

aregd@DESKTOP-5KV9QA9 MINGW64 ~/OneDrive/Рабочий стол/git8/lab8 (develop)
$ git add .

aregd@DESKTOP-5KV9QA9 MINGW64 ~/OneDrive/Рабочий стол/git8/lab8 (develop)
$ git commit -m"j"
[develop 07f6b03] j
3 files changed, 44 insertions(+)
create mode 100644 PyCharm/1task.py
create mode 100644 PyCharm/primer1(2).py
create mode 100644 PyCharm/primer1.py

aregd@DESKTOP-5KV9QA9 MINGW64 ~/OneDrive/Рабочий стол/git8/lab8 (develop)
$ git push origin develop
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 1.12 KiB | 1.12 MiB/s, done.
Total 6 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
remote:
remote: Create a pull request for 'develop' on GitHub by visiting:
remote: https://github.com/aregdz/lab8/pull/new/develop

```

Рисунок 10 – фиксация изменений в ветку develop

10.Слил ветки.

```

aregd@DESKTOP-5KV9QA9 MINGW64 ~/OneDrive/Рабочий стол/git8/lab8 (develop)
$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

aregd@DESKTOP-5KV9QA9 MINGW64 ~/OneDrive/Рабочий стол/git8/lab8 (main)
$ git merge develop
Updating bf72f30..07f6b03
Fast-forward
 PyCharm/1task.py      | 15 +++++
 PyCharm/primer1(2).py | 13 +++++
 PyCharm/primer1.py    | 16 +++++
3 files changed, 44 insertions(+)
create mode 100644 PyCharm/1task.py
create mode 100644 PyCharm/primer1(2).py
create mode 100644 PyCharm/primer1.py

```

Рисунок 11 – сливание ветки develop в ветку main

Контрольные вопросы:

## 1. Что такое списки в языке Python?

Список (list) – это структура данных для хранения объектов различных типов. Список очень похож на массив, только, как было уже сказано выше, в нем можно хранить объекты различных типов. Размер списка 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. Каково назначение кортежей в языке Python?

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 придется как нельзя кстати. Используя их в данной задаче, мы дополнительно получаем сразу несколько бонусов – во-первых, это экономия места

## 3. Как осуществляется создание кортежей?

```
>>> a = ()
>>> b = tuple()
>>> a = (1, 2, 3, 4, 5)
>>> a = tuple([1, 2, 3, 4])
>>> tuple = (42,)
```

## 4. Как осуществляется доступ к элементам кортежа?

Доступ к элементам кортежа осуществляется также как к элементам списка – через указание индекса.

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

Распаковка (или деструктуризация) кортежа в программировании позволяет извлечь отдельные элементы кортежа и присвоить их переменным. Это полезная операция, потому что она позволяет удобно работать с данными в кортеже, делая их доступными для дальнейшего использования.

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

Используя множественное присваивание, можно провернуть интересный трюк: обмен значениями между двумя переменными.

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

```
T2 = T1[i:j]
```

T2 – новый кортеж, который получается из кортежа T1;

T1 – исходный кортеж, для которого происходит срез;

i, j – соответственно нижняя и верхняя границы среза. Фактически берутся ко вниманию элементы, лежащие на позициях i, i+1, ..., j-1. Значение j определяет позицию за последним элементом среза.

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

Во время конкатенации складываются два кортежа, а во время повторения, они повторяются  $n$  раз.

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

Элементы кортежа можно последовательно просмотреть с помощью операторов цикла `while` или `for`.

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

С помощью оператора `in`.

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

Метод `index()` - поиск позиции элемента в кортеже

Метод `count()` - количество вхождений элемента в кортеж

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

Допустимо использование.

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

`my_tuple = tuple(i for i in A)`