

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙ-
СКОЙ ФЕДЕРАЦИИ**

**Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Кафедра инфокоммуникаций

Основы кроссплатформенного программирования

Отчет по лабораторной работе №2.5

Тема: «Работа с кортежами в языке Python»

Выполнил студент группы

ИВТ-б-о-21-1

Богадунов В.И. « » _____ 20__ г.

Подпись студента _____

Работа защищена « » _____ 20__ г.

Проверил доцент

Кафедры инфокоммуникаций, старший
преподаватель

Воронкин Р.А.

(подпись)

Ставрополь 2022

1. Создал репозиторий в GitHub, дополнил правила в .gitignore для работы с IDE PyCharm с ЯП Python, выбрал лицензию MIT, клонировал его на компьютер и организовал в соответствии с моделью ветвления git-flow.

Owner * Repository name *

ItsMyLife1337 / ForPrograms2..5 ✓

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

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:
Skip this step if you're importing an existing repository.

☒ Add a README file
This is where you can write a long description for your project. [Learn more.](#)

Add .gitignore
Choose which files not to track from a list of templates. [Learn more.](#)

.gitignore template: Python ▼

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

License: MIT License ▼

① You are creating a public repository in your personal account.

Create repository

Рисунок 1.1 – Создал репозиторий

```
280 lines (223 sloc) | 5.7 KB
1  .idea/
2  # Created by https://www.toptal.com/developers/gitignore/api/python,pycharm
3  # Edit at https://www.toptal.com/developers/gitignore?templates=python,pycharm
4
5  ### PyCharm ###
6  # Covers JetBrains IDEs: IntelliJ, RubyMine, PhpStorm, AppCode, PyCharm, CLion, Android Studio, WebStorm and Rider
7  # Reference: https://intellij-support.jetbrains.com/hc/en-us/articles/206544839
8
9  # User-specific stuff
10 .idea/**/workspace.xml
11 .idea/**/tasks.xml
12 .idea/**/usage.statistics.xml
13 .idea/**/dictionaries
14 .idea/**/shelf
15
16 # AWS User-specific
17 .idea/**/aws.xml
18
19 # Generated files
20 .idea/**/contentModel.xml
```

Рисунок 1.2 – Изменённый файл .gitignore

```

C:\Users\Admin>cd /d c:\users\admin\desktop\git

c:\Users\Admin\Desktop\git>git clone https://github.com/ItsMyLife1337/ForPrograms2.5.git
Cloning into 'ForPrograms2.5'...
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 7 (delta 1), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (7/7), 4.41 KiB | 2.20 MiB/s, done.
Resolving deltas: 100% (1/1), done.

c:\Users\Admin\Desktop\git>cd /d c:\users\admin\desktop\git\forprograms2.5

```

Рисунок 1.3 – Скопировал репозиторий на ПК

```

c:\Users\Admin\Desktop\git\ForPrograms2.5>git flow init

Which branch should be used for bringing forth production releases?
- main
Branch name for production releases: [main]
Branch name for "next release" development: [develop]

How to name your supporting branch prefixes?
Feature branches? [feature/]
Bugfix branches? [bugfix/]
Release branches? [release/]
Hotfix branches? [hotfix/]
Support branches? [support/]
Version tag prefix? []
Hooks and filters directory? [C:/Users/Admin/Desktop/git/ForPrograms2.5/.git/hooks]

c:\Users\Admin\Desktop\git\ForPrograms2.5>

```

Рисунок 1.4 – Организация репозитория в соответствии с моделью ветвления git-flow

2. Создал проект PyCharm в папке репозитория, проработал примеры ЛР

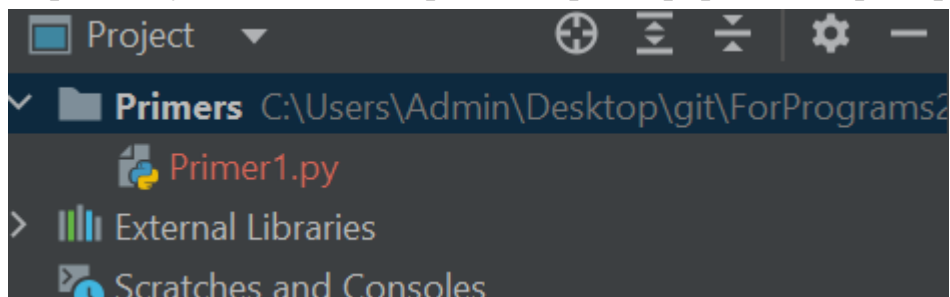


Рисунок 2.1 – Созданные проекты

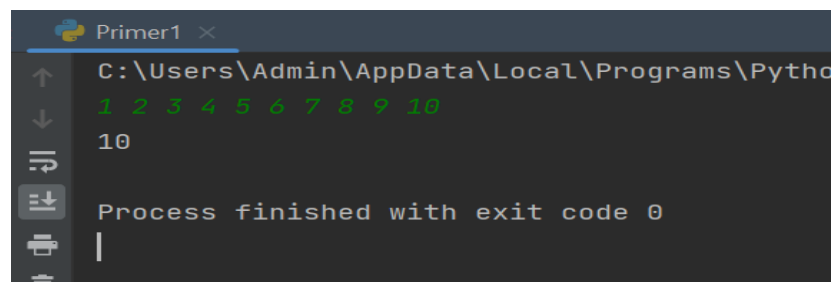
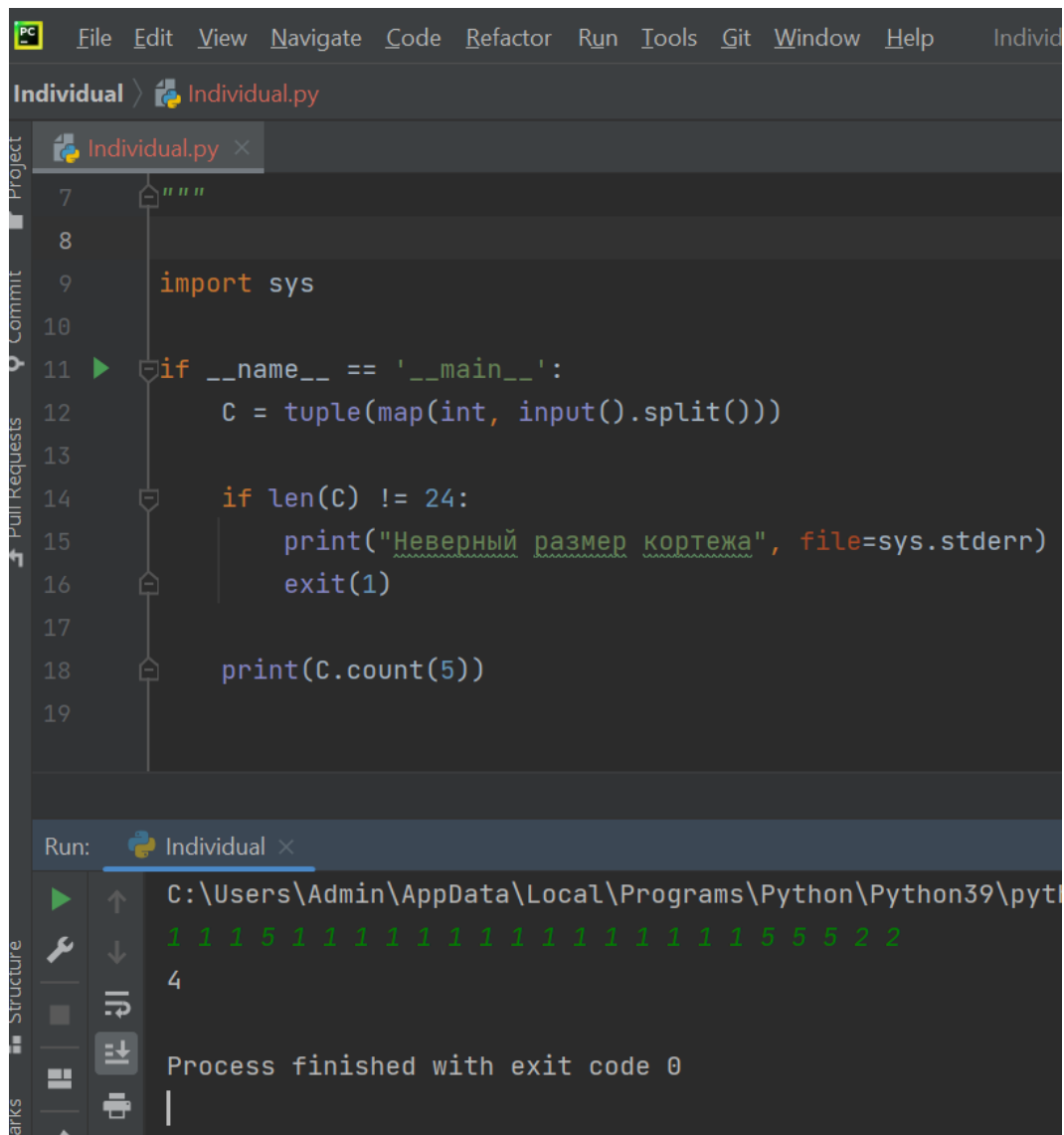


Рисунок 2.2 – Результат выполнения Примера 1

3. Выполнил 1 индивидуальное задание. Вариант – №3. Известны оценки по геометрии каждого из 24 учеников класса. Сколько учеников имеет по геометрии оценку «5»? Условный оператор не использовать.



The screenshot shows an IDE with a Python file named `Individual.py`. The code is as follows:

```
7 """
8
9 import sys
10
11 if __name__ == '__main__':
12     C = tuple(map(int, input().split()))
13
14     if len(C) != 24:
15         print("Неверный размер кортежа", file=sys.stderr)
16         exit(1)
17
18     print(C.count(5))
19
```

The Run window shows the execution output:

```
Run: Individual x
C:\Users\Admin\AppData\Local\Programs\Python\Python39\pyt
1 1 1 5 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 5 5 5 2 2
4
Process finished with exit code 0
```

Рисунок 3.1 – Индивидуальное задание №1

```
c:\Users\Admin\Desktop\git\ForPrograms2.5>git commit -m "old but gold"
[develop cb144b7] old but gold
2 files changed, 38 insertions(+)
create mode 100644 Individual/Individual.py
create mode 100644 Primers/Primer1.py
```

Рисунок 4.1 – Сделал коммит всех изменений

```

c:\Users\Admin\Desktop\git\ForPrograms2.5>git branch
* develop
  main

c:\Users\Admin\Desktop\git\ForPrograms2.5>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

c:\Users\Admin\Desktop\git\ForPrograms2.5>git merge develop
Updating 761ce16..cb144b7
Fast-forward
 Individual/Individual.py | 18 +++++
 Primers/Primer1.py       | 20 +++++
 2 files changed, 38 insertions(+)
 create mode 100644 Individual/Individual.py
 create mode 100644 Primers/Primer1.py

c:\Users\Admin\Desktop\git\ForPrograms2.5>

```

Рисунок 4.2 – Переход на ветку main и последующее её слияние с develop


```

c:\Users\Admin\Desktop\git\ForPrograms2.5>git push
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 4 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 1.15 KiB | 589.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/ItsMyLife1337/ForPrograms2.5.git
 761ce16..cb144b7  main -> main


c:\Users\Admin\Desktop\git\ForPrograms2.5>

```


Рисунок 4.3 – Отправка изменений на удалённый репозиторий



main



1 branch




0 tags

Go to file

Add file

Code




ItsMyLife1337


old but gold

cb144b7

6 minutes ago




3 commits



Individual

old but gold


6 minutes ago



Primers

old but gold


6 minutes ago



.gitignore

Update .gitignore

22 hours ago



LICENSE

Initial commit

22 hours ago

Рисунок 4.4 – Проверка изменений

Контрольные вопросы:

1. Что такое кортежи в языке Python?

Кортеж (tuple) – это неизменяемая структура данных, которая по-своему подобию очень похожа на список.

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 придутся как нельзя кстати. Кортежи в памяти занимают меньший объем по сравнению со списками. Кортежи работают быстрее, чем списки

3. Как осуществляется создание кортежей?

```
a = ()
```

```
b = tuple()
```

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?

Обращение по индексу, это не самый удобный способ работы с кортежами. Дело в том, что кортежи часто содержат значения разных типов, и помнить, по какому индексу что лежит — очень непросто.

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?

С помощью операции взятия среза можно получить другой кортеж.

Общая форма операции взятия среза для кортежа следующая

$T2 = T1[i:j]$ здесь:

- 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?

Для кортежей можно выполнять операцию конкатенации, которая обозначается символом +. $T3 = T1 + T2$

9. Как выполняется обход элементов кортежа?

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

10. Как проверить принадлежность элемента кортежу?

Проверка вхождения элемента в кортеж - оператор in.

11. Какие методы работы с кортежами Вам известны?

index(), count().

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

Допустимо.

13. Как создать кортеж с помощью спискового включения.

Так же, как и список.

Вывод: научился работать с кортежами в языке программирования Python.