

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

Кафедра инфокоммуникаций

Институт цифрового развития

ОТЧЁТ

по лабораторной работе №2.5

Дисциплина: «Основы кроссплатформенного программирования»

Тема: «Работа со кортежами в языке Python»

Выполнил: студент 1 курса

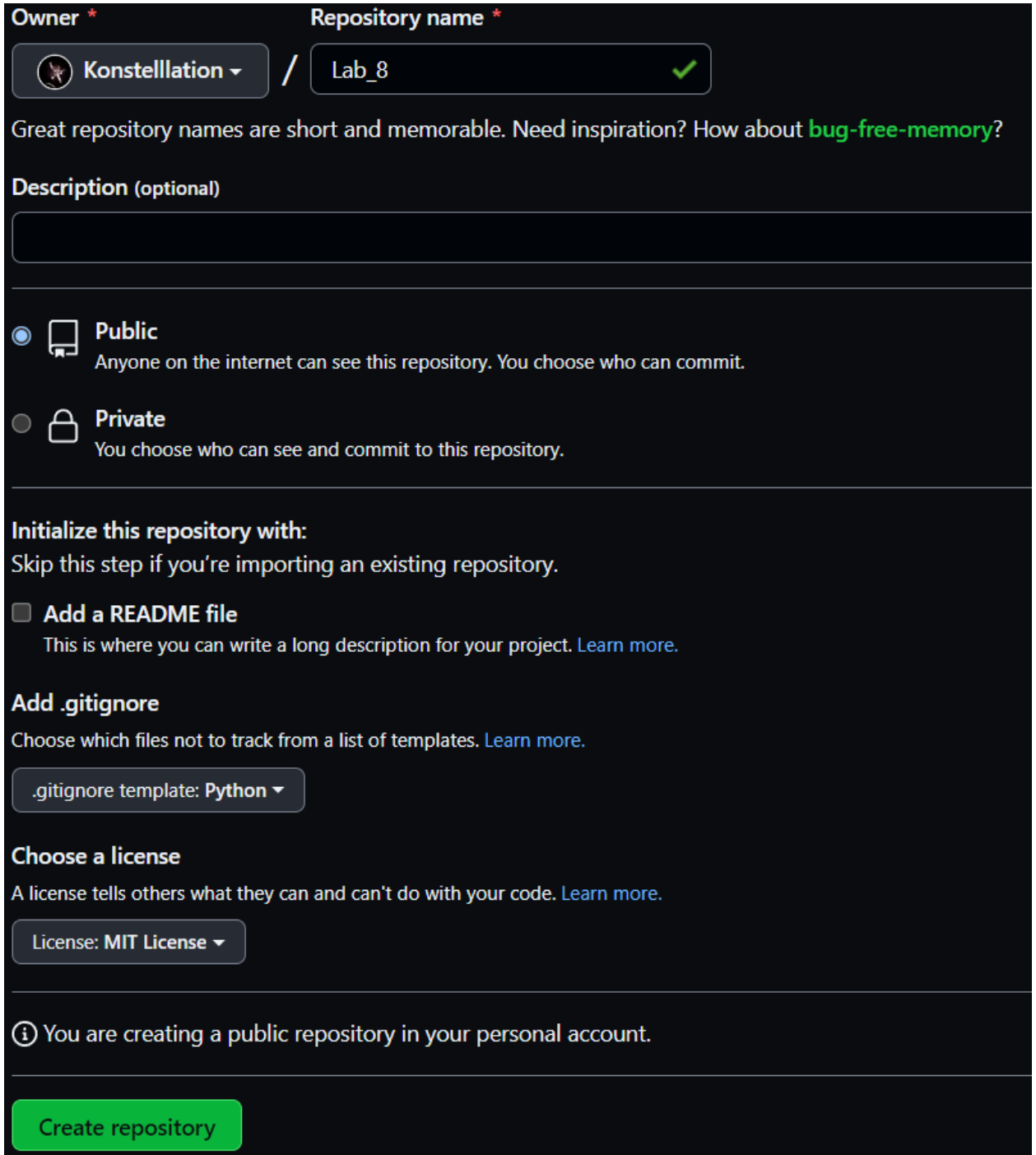
группы ИВТ-б-о-21-1

Харченко Богдан Романович

Ставрополь 2022

Выполнение работы:

1. Создал репозиторий в GitHub «Lab_8» в который добавил .gitignore, который дополнил правила для работы с IDE PyCharm с ЯП Python, выбрал лицензию MIT, клонировал его на лок. сервер и организовал в соответствии с моделью ветвления git-flow.



Owner * Repository name *

Konstellation / Lab_8 ✓

Great repository names are short and memorable. Need inspiration? How about [bug-free-memory?](#)

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 ▾

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

Create repository

Рисунок 1.1 Создание репозитория

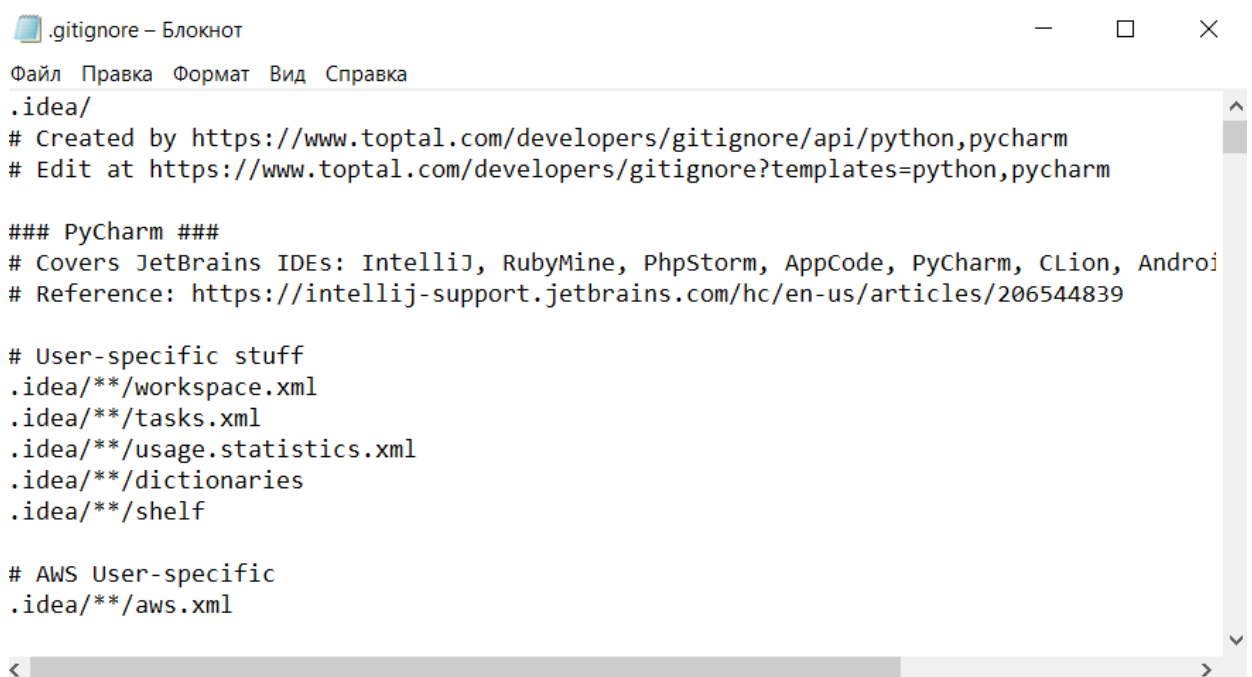
```
C:\>git clone https://github.com/Konstellation/Lab_8.git
Cloning into 'Lab_8'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (4/4), done.
```

Рисунок 1.2 Клонирование репозитория

```
C:\Lab_8>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:/Lab_8/.git/hooks]
```

Рисунок 1.3 Организация репозитория в соответствии с моделью ветвления
git-flow



```
.gitignore – Блокнот
Файл Правка Формат Вид Справка
.idea/
# Created by https://www.toptal.com/developers/gitignore/api/python,pycharm
# Edit at https://www.toptal.com/developers/gitignore?templates=python,pycharm

### PyCharm ###
# Covers JetBrains IDEs: IntelliJ, RubyMine, PhpStorm, AppCode, PyCharm, CLion, Android Studio
# Reference: https://intellij-support.jetbrains.com/hc/en-us/articles/206544839

# User-specific stuff
.idea/**/workspace.xml
.idea/**/tasks.xml
.idea/**/usage.statistics.xml
.idea/**/dictionaries
.idea/**/shelf

# AWS User-specific
.idea/**/aws.xml
```

Рисунок 1.4 Изменение .gitignore

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

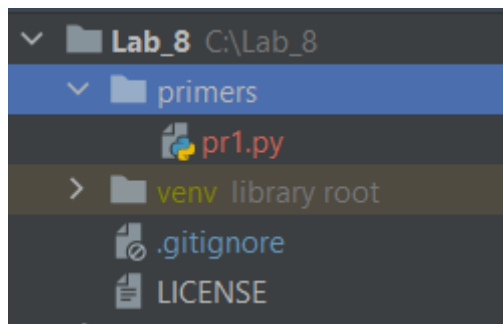


Рисунок 2.1 Создание проекта в PyCharm

```
2 3 1 5 6 4 8 7 9 2
12
```

Рисунок 2.2 Рез-т выполнения программы

3. (21 вариант). Выполнил индивидуальное задание.

```
1  ▶  #!/usr/bin/env python3
2      # -*- coding: utf-8 -*-
3
4  ▶  if __name__ == '__main__':
5      c = [(10, 12), (8, 6), (9, 12), (9, 9), (13, 13)]
6      all_x = []
7      all_y = []
8      for i in c:
9          all_x.append(i[0])
10         all_y.append(i[1])
11
12         fp = (min(all_x), min(all_y))
13         sp = (max(all_x), max(all_y))
14         print(fp)
15         print(sp)
16
```

```
(8, 6)
(13, 13)
```

Рисунок 3.1 Вывод программы индивидуального задания

4. Сделал коммит, выполнил слияние с веткой main, и запустил изменения в уд. репозиторий.

```
C:\Lab_8>git add .

C:\Lab_8>git commit -m "added project"
[develop b3aba6f] added project
 3 files changed, 37 insertions(+)
 create mode 100644 ind/ind1.py
 create mode 100644 primers/pr1.py

C:\Lab_8>git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.

C:\Lab_8>git merge develop
```

Рисунок 4.1 коммит изменений и переход на ветку main

```
C:\Lab_8>git merge develop
Updating e8f0cf5..b3aba6f
Fast-forward
 .gitignore      | 1 +
 ind/ind1.py     | 15 ++++++++
 primers/pr1.py  | 21 ++++++++
 3 files changed, 37 insertions(+)
 create mode 100644 ind/ind1.py
 create mode 100644 primers/pr1.py
```

Рисунок 4.2 Слияние ветки main с develop

```
C:\Lab_8>git push
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Delta compression using up to 8 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (7/7), 1007 bytes | 251.00 KiB/s, done.
Total 7 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/Konstellation/Lab_8.git
 e8f0cf5..b3aba6f main -> main
```

Рисунок 4.3 Пуш изменений на удаленный сервер

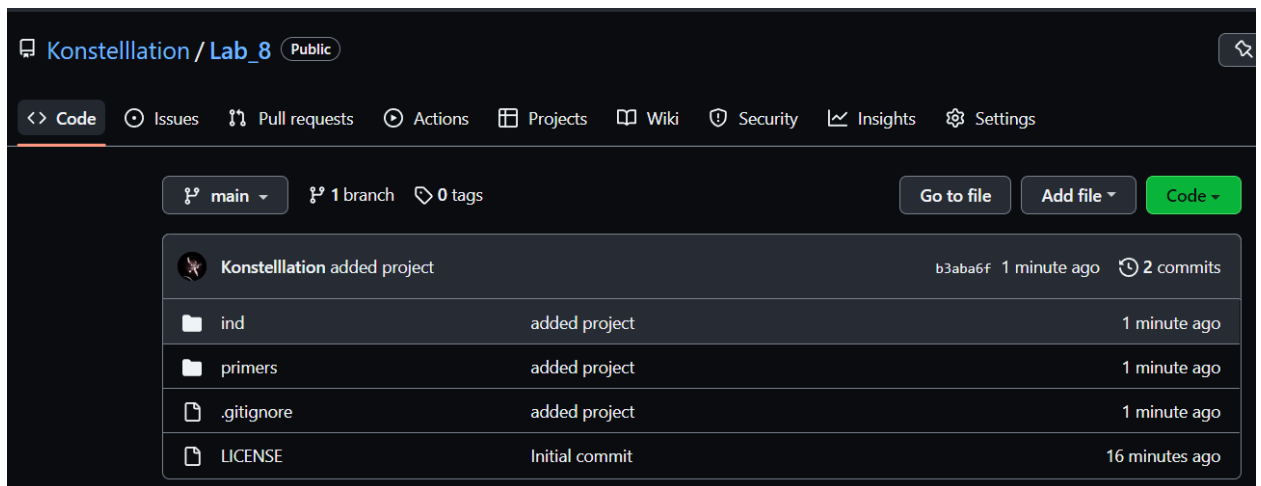


Рисунок 4.4 Изменения на удаленном сервере

Контр. вопросы и ответы на них:

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 проверить 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 во внимание элементы, лежащие на позициях $i, i+1, \dots, j-1$. 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?

Доступно.

13. Как создать кортеж с помощью спискового включения.

Так же как и список.