

**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 №8

Дисциплина: «Основы кроссплатформенного программирования»

Тема: «Работа со списками в языке Python»

Выполнил: студент 1 курса

группы ИВТ-б-о-21-1

Уланбекова Айканыш Уланековна

Ставрополь 2022

Выполнение работы:

1. Создала репозиторий в GitHub «rep 2.5» в который добавила .gitignore, который дополнила правила для работы с IDE PyCharm с ЯП Python, выбрала лицензию MIT, клонировала его на лок. сервер и организовала в соответствии с моделью ветвления git-flow.

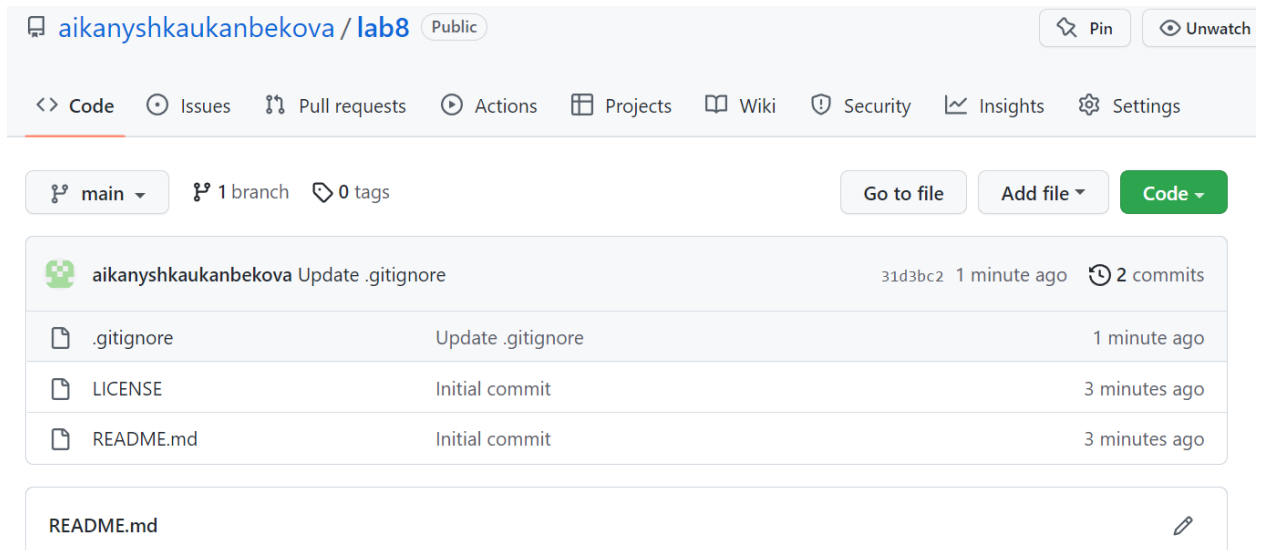


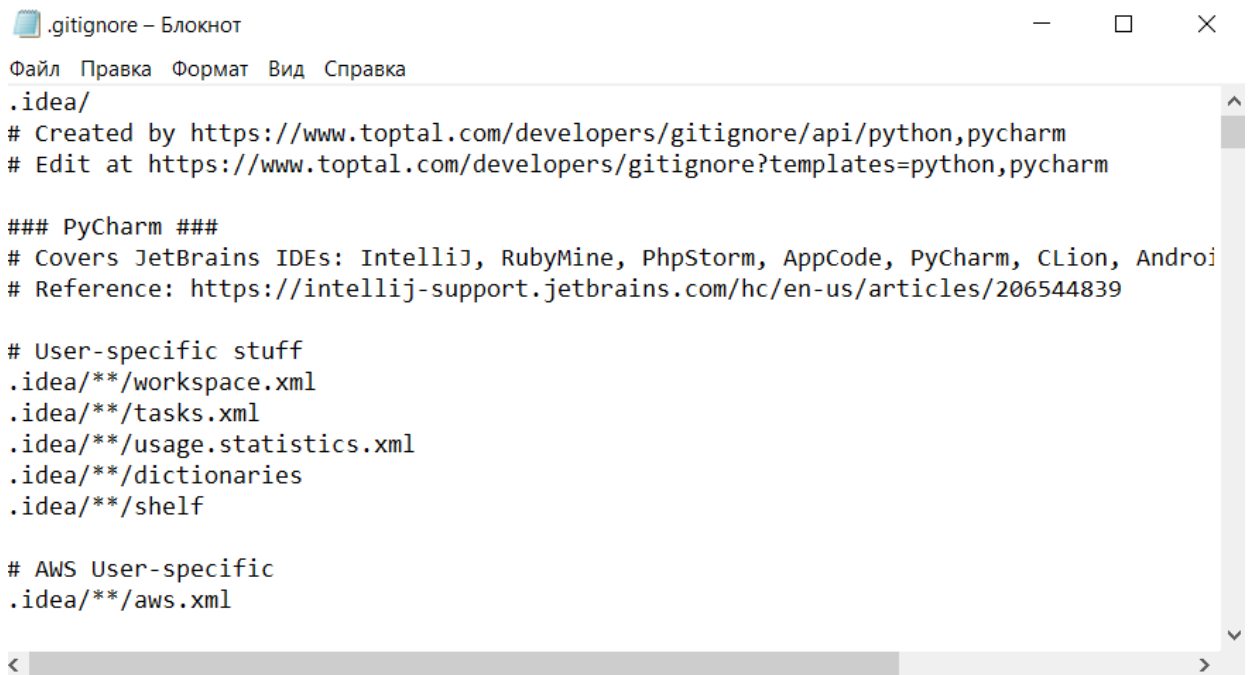
Рисунок 1.1 Создание репозитория

```
C:\Users\User>cd C:\Users\User\Desktop\lab8
C:\Users\User\Desktop\lab8>git clone https://github.com/aikanyshkauanbekova/lab8.git
Cloning into 'lab8'...
remote: Enumerating objects: 8, done.
remote: Counting objects: 100% (8/8), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 8 (delta 1), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (8/8), done.
Resolving deltas: 100% (1/1), done.
C:\Users\User\Desktop\lab8>_
```

Рисунок 1.2 Клонирование репозитория

```
C:\Users\User\Desktop\lab8\lab8>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/User/Desktop/lab8/lab8/.git/hooks]
```

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



The image shows a Notepad window titled ".gitignore – Блокнот". The menu bar includes "Файл", "Правка", "Формат", "Вид", and "Справка". The content of the .gitignore file is as follows:

```
.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. Создала проект PyCharm в папке репозитория, проработала примеры ЛР.

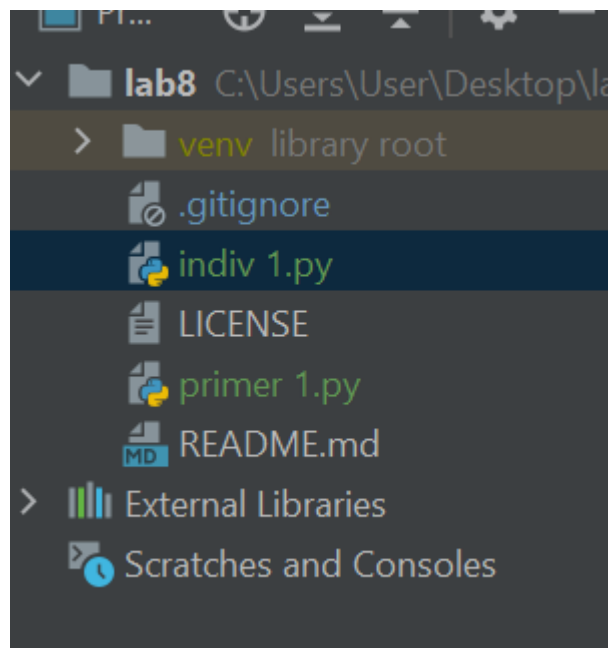
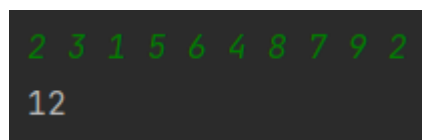


Рисунок 2.1 Создание проекта в PyCharm



The image shows a terminal window with the following output:

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

Рисунок 2.2 Рез-т выполнения программы

3. Выполнил индивидуальное задание. Вариант 23

```
primer 1.py x indiv 1.py x
1 ▶ #!/usr/bin/env python3
2   # -*- coding: utf-8 -*-
3
4 ▶ if __name__ == '__main__':
5     a = ()
6     b = ()
7     a_li = list(a)
8     b_li = list(b)
9
10    n = int(input('Введите количество элементов кортежа: '))
11    print('Ведите элементы списка:\n')
12    for i in range(n):
13        a_li.append(int(input()))
14        if (i+1) % 2 == 0:
15            b_li.append(a_li[i]**2)
16        else:
17            b_li.append(a_li[i]*2)
18    a = tuple(a_li)
19    b = tuple(b_li)
20    print('a = ', a, '\nb = ', b)
```

```
Введите количество элементов кортежа: 6
Ведите элементы списка:

2
1
5
9
-8
-4

a = (2, 1, 5, 9, -8, -4)
b = (4, 1, 10, 81, -16, 16)
```

Рисунок 3.1 Вывод программы индивидуального задания

4. Сделала коммит, выполнила слияние с веткой main, и запустила изменения в уд. репозиторий.

```
C:\Users\User\Desktop\lab8\lab8>git add .  
C:\Users\User\Desktop\lab8\lab8>git commit -m "d"  
[develop 1049896] d  
3 files changed, 190 insertions(+), 3 deletions(-)  
create mode 100644 ""\320\230\320\275\320\264\320\270\320\262\320\270\320\264\321\203\320\260\320\273\321\214\320\275\3  
0\276\320\265 \320\267\320\260\320\264\320\260\320\275\320\270\320\265\indiv 1.py"  
create mode 100644 ""\320\237\321\200\320\270\320\274\320\265\321\200\321\213\primer 1.py"  
C:\Users\User\Desktop\lab8\lab8>git checkout main  
Switched to branch 'main'  
Your branch is up to date with 'origin/main'.
```

Рисунок 4.1 коммит изменений и переход на ветку main

```
C:\Users\adamk\OneDrive\Рабочий стол\rep_2.5>git merge develop
Updating 307f1de..1a66e01
Fast-forward
 .gitignore      | 1 +
 ind/ind1.py     | 20 +++++
 primers/pr1.py  | 21 +++++
 3 files changed, 42 insertions(+)
 create mode 100644 ind/ind1.py
 create mode 100644 primers/pr1.py
```

Рисунок 4.2 Слияние ветки main с develop

```
C:\Users\User\Desktop\lab8\lab8>git push
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 1.13 KiB | 1.13 MiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/aikanyshkauanbekova/lab8.git
   d962e52..3f9cb2c  main -> main

C:\Users\User\Desktop\lab8\lab8>
```

Рисунок 4.3 Пуш изменений на удаленный сервер

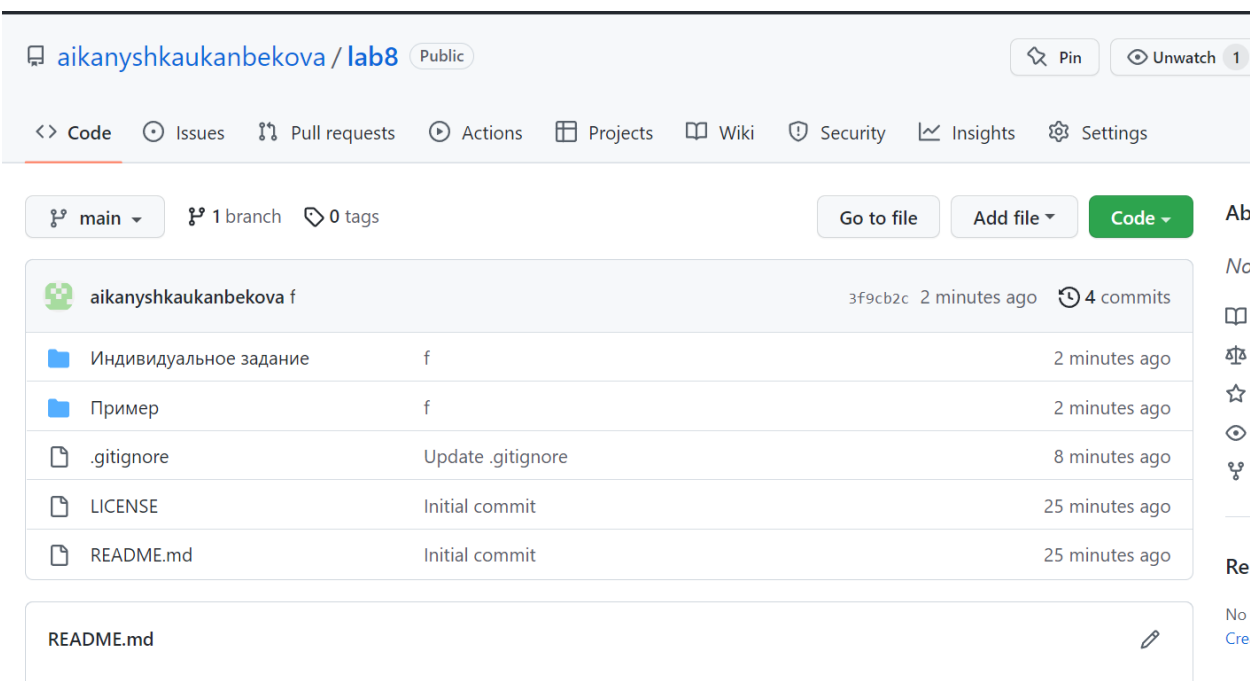


Рисунок 4.4 Изменения на удаленном сервере

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 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 $i, i+1, \dots, j-1$. 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.