

**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 №2.4

Дисциплина: «Основы кроссплатформенного программирования»

Тема: «Работа со списками в языке Python»

Выполнил: студент 1 курса

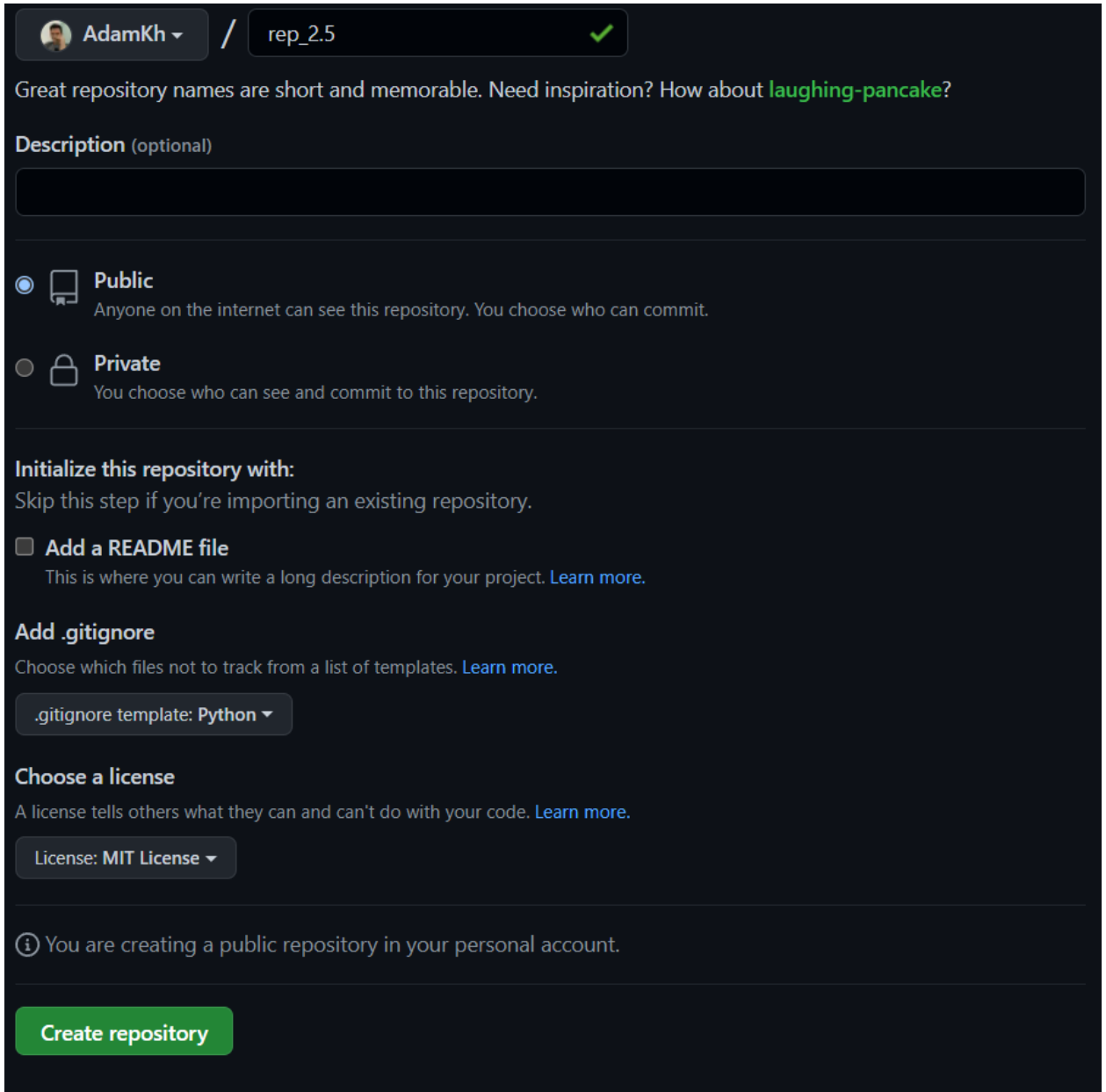
группы ИВТ-б-о-21-1

Хашиев Адам Мухарбекович

Ставрополь 2022

Выполнение работы:

1. Создал репозиторий в GitHub «rep 2.5» 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.



The screenshot shows the GitHub 'Create repository' page for user AdamKh. The repository name is 'rep_2.5', which is marked as valid with a green checkmark. A tip suggests repository names should be short and memorable, with an example 'laughing-pancake?'. The 'Description' field is empty. Under 'Visibility', the 'Public' option is selected, indicating that anyone on the internet can see the repository. The 'Initialize this repository with:' section has 'Skip this step if you're importing an existing repository.' checked. Below this, 'Add a README file' is unchecked, with a note that it's for a long description. The 'Add .gitignore' section is active, showing a dropdown menu with '.gitignore template: Python' selected. The 'Choose a license' section shows a dropdown menu with 'License: MIT License' selected. A note at the bottom states, 'You are creating a public repository in your personal account.' A green 'Create repository' button is at the bottom left.

Рисунок 1.1 Создание репозитория

```
C:\Users\adamk\OneDrive\Рабочий стол>git clone https://github.com/AdamKh/rep_2.5.git
Cloning into 'rep_2.5'...
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

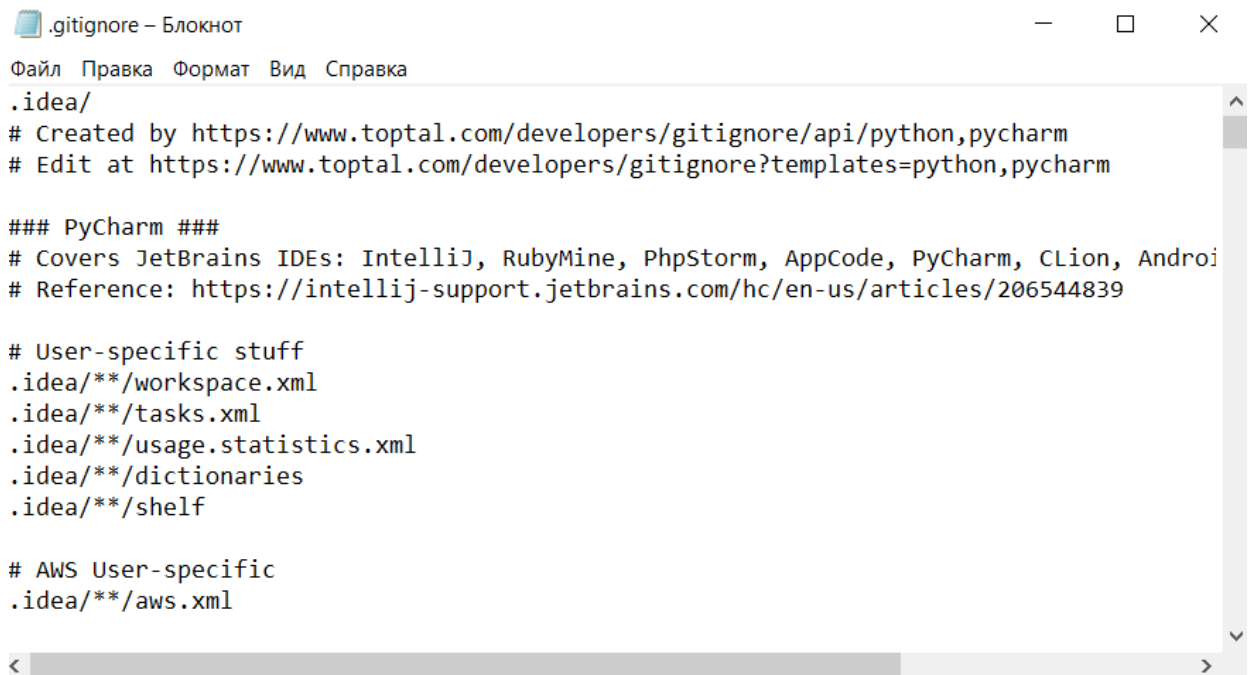
```

C:\Users\adamk\OneDrive\Рабочий стол\rep_2.5>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/adamk/OneDrive/Рабочий стол/rep_2.5/.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.

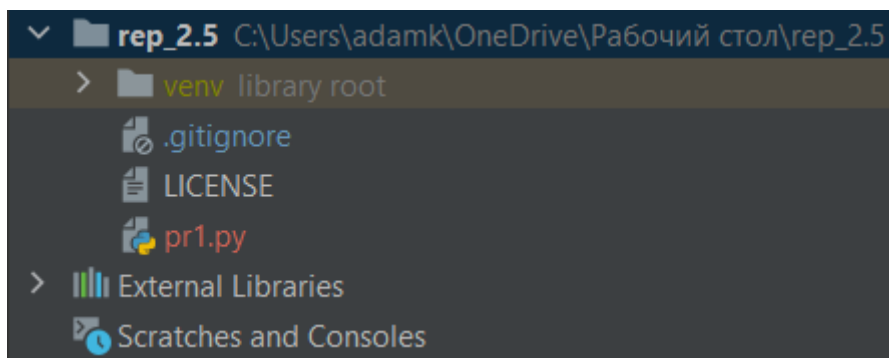


Рисунок 2.1 Создание проекта в PyCharm

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

Рисунок 2.2 Рез-т выполнения программы

3. (22 вариант). Выполнил индивидуальное задание.

```
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)
21
```

```

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

```

C:\Users\adamk\OneDrive\Рабочий стол\rep_2.5>git add .
C:\Users\adamk\OneDrive\Рабочий стол\rep_2.5>git commit -m "added progs"
[develop 1a66e01] added progs
3 files changed, 42 insertions(+)
create mode 100644 ind/ind1.py
create mode 100644 primers/pr1.py
C:\Users\adamk\OneDrive\Рабочий стол\rep_2.5>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\adamk\OneDrive\Рабочий стол\rep_2.5>git push
info: please complete authentication in your browser...
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), 1.10 KiB | 562.00 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/AdamKh/rep_2.5.git
307f1de..1a66e01 main -> main
```

Рисунок 4.3 Пуш изменений на удаленный сервер

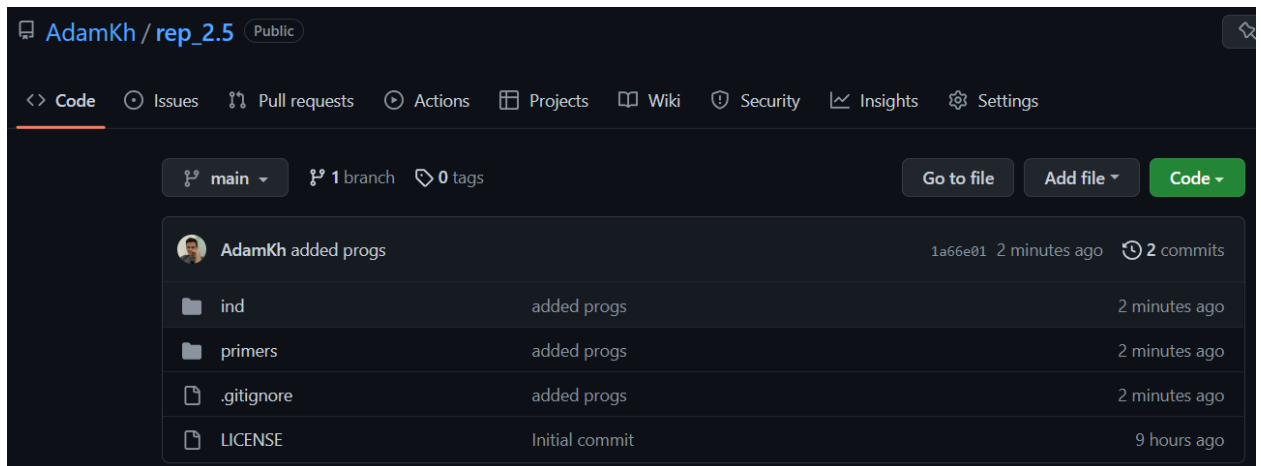


Рисунок 4.4 Изменения на удаленном сервере

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 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?

Доступно.

13. Как создать кортеж с помощью спискового включения.

Так же как и список.