

**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 №2.17

Дисциплина: «Программирование на Python»

Тема: «Разработка приложений с интерфейсом командной строки (CLI) в
Python3»

Выполнил: студент 2 курса

группы ИВТ-б-о-21-1

Хашиев Адам Мухарбекович

Ставрополь 2022

Выполнение работы:

1. Создал репозиторий в GitHub «rep 2.6» 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.

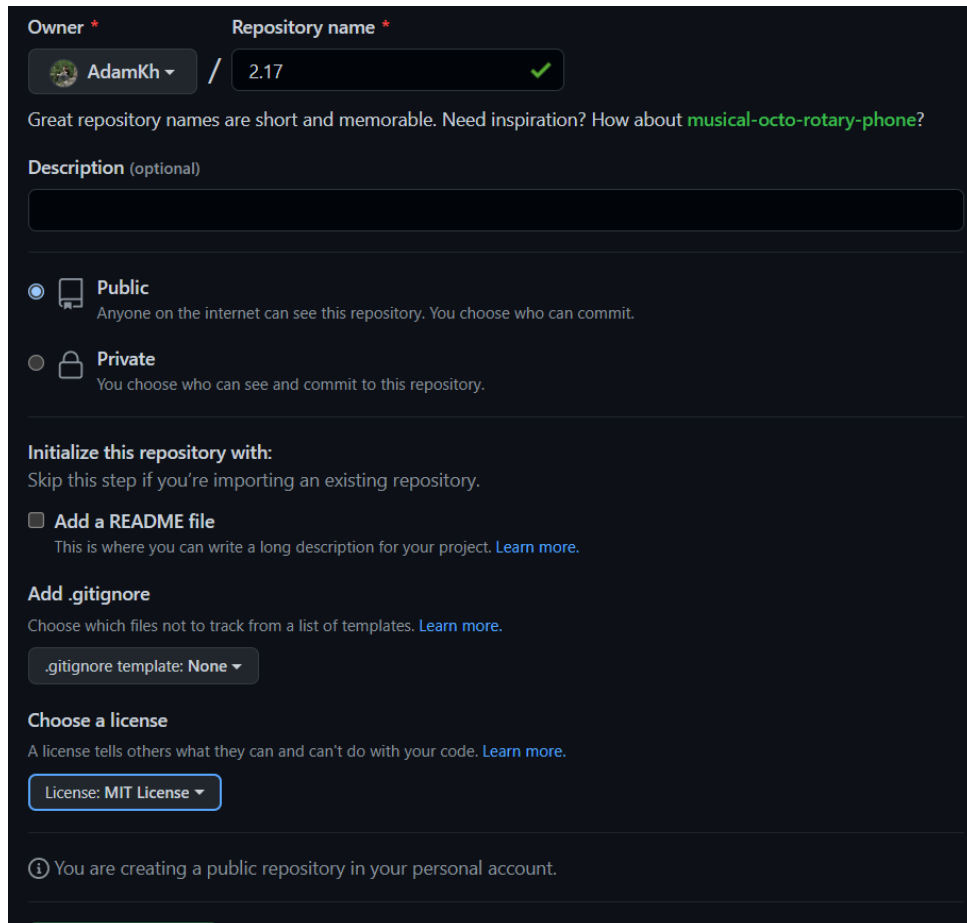


Рисунок 1.1 Создание репозитория

```
C:\Users\adamkh\Desktop\3sem\Python\2.17>git clone https://github.com/AdamKh/2.17.git
Cloning into '2.17'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
```

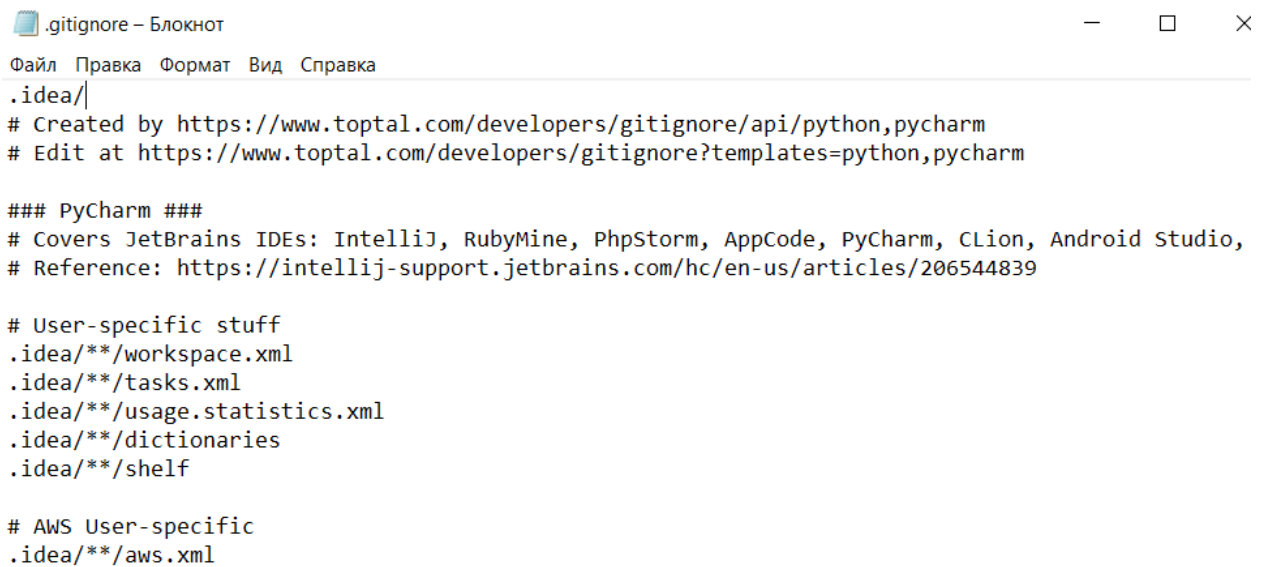
Рисунок 1.2 Клонирование репозитория

```

C:\Users\adamkh\Desktop\3sem\Python\2.17\2.17>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/adamkh/Desktop/3sem/Python/2.17/2.17/.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.

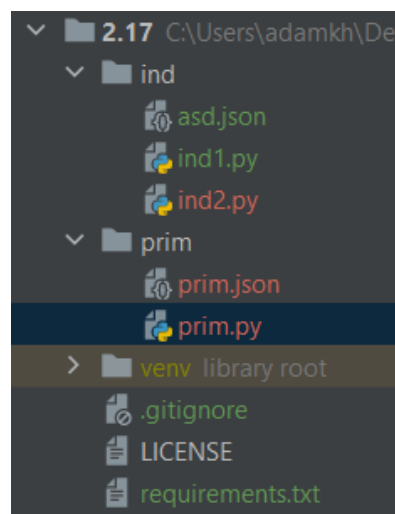


Рисунок 2.1 Создание проекта в PyCharm

```
(venv) C:\Users\adamkh\Desktop\3sem\Python\2.17\2.17\prim>python prim.py display prim.json
```

No	Ф.И.О.	Должность	Год
1	asd	ert	0

```
(venv) C:\Users\adamkh\Desktop\3sem\Python\2.17\2.17\prim>
```

Рисунок 2.2 Рез-т выполнения программы

3. (15 вариант). Выполнил индивидуальное задание.

```
(venv) C:\Users\adamkh\Desktop\3sem\Python\2.17\2.17\ind>python ind1.py add asd.json -n "ASdqw" -sn "Bsdsaaa" -z "Asdwq" -d "02/01/1995"
```

```
(venv) C:\Users\adamkh\Desktop\3sem\Python\2.17\2.17\ind>python ind1.py display asd.json
```

№	Фамилия и имя	Знак Зодиака	Дата рождения
1	qwe	asd	11/11/1111111
2	Skalette	Vito	22/11/1943
3	Bsdsaaa	ASdqw	02/01/1995

```
(venv) C:\Users\adamkh\Desktop\3sem\Python\2.17\2.17\ind>
```

Рисунок 3.1 Вывод программы индивидуального задания №1

```
(venv) C:\Users\adamkh\Desktop\3sem\Python\2.17\2.17\ind>python ind2.py add asd.json -n "Vito" -sn "Skalette" -z "Oven" -d "22/11/1943"
```

данные добавлены

```
(venv) C:\Users\adamkh\Desktop\3sem\Python\2.17\2.17\ind>python ind2.py display asd.json
```

№	Фамилия и имя	Знак Зодиака	Дата рождения
1	qwe	asd	11/11/1111111
2	Skalette	Vito	22/11/1943

```
(venv) C:\Users\adamkh\Desktop\3sem\Python\2.17\2.17\ind>
```

Рисунок 3.2 Вывод программы индивидуального задания №2

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

```
C:\rep_2.6>git add .
```

```
C:\rep_2.6>git commit -m "added programs + modidied .gitignore"
```

```
[develop 2582c62] added programs + modidied .gitignore
```

```
4 files changed, 379 insertions(+), 3 deletions(-)
```

```
create mode 100644 ind.py
```

```
create mode 100644 prim.py
```

```
create mode 100644 zadaniya.py
```

```
C:\rep_2.6>git checkout main
```

```
Switched to branch 'main'
```

```
Your branch is up to date with 'origin/main'.
```

```
C:\rep_2.6>
```

```
C:\rep_2.6>git push
```

```
Everything up-to-date
```

Рисунок 4.1 коммит и пуш изменений и переход на ветку main

```

C:\rep_2.6>git merge develop
Updating 5d4b8d1..2582c62
Fast-forward
 .gitignore | 157 ++++++
--
 ind.py      | 105 ++++++
 prim.py    | 99  ++++++
 zadaniya.py | 21  ++++++
4 files changed, 379 insertions(+), 3 deletions(-)
create mode 100644 ind.py
create mode 100644 prim.py
create mode 100644 zadaniya.py
C:\rep_2.6>_

```

Рисунок 4.2 Слияние ветки main с develop

```

C:\rep_2.6>git push
info: please complete authentication in your browser...
Enumerating objects: 8, done.
Counting objects: 100% (8/8), done.
Delta compression using up to 8 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (6/6), 5.16 KiB | 2.58 MiB/s, done.
Total 6 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/AdamKh/rep_2.6.git
5d4b8d1..2582c62 main -> main

```

Рисунок 4.3 Пуш изменений на удаленный сервер

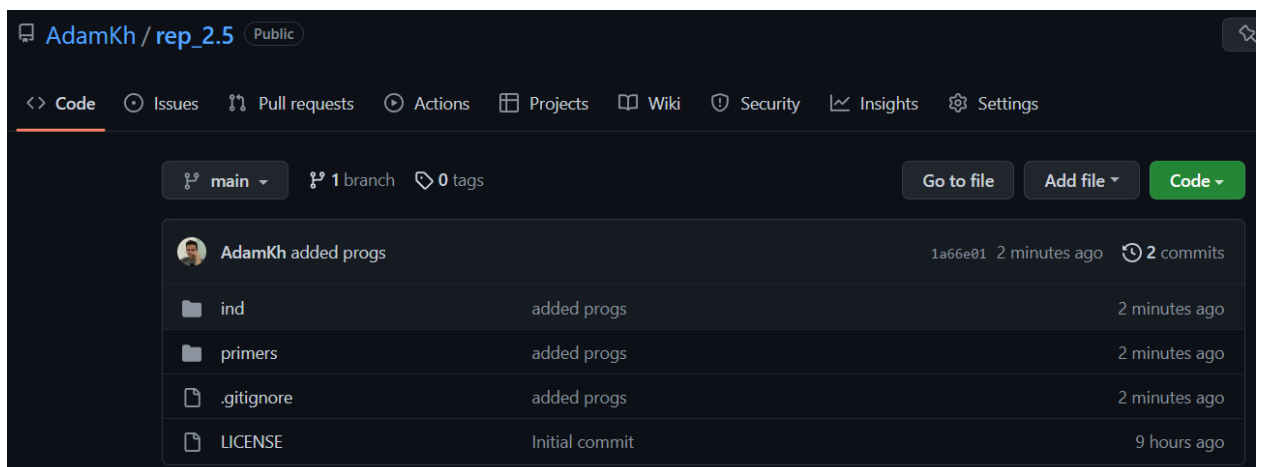


Рисунок 4.4 Изменения на удаленном сервере

Контр. вопросы и ответы на них:

1. В чем отличие терминала и консоли?

Терминал (от лат. terminus — граница) — устройство или ПО, выступающее посредником между человеком и вычислительной системой.

Обычно данный термин используется, когда точка доступа к системе вынесена в отдельное физическое устройство и предоставляет свой

пользовательский интерфейс на основе внутреннего интерфейса (например, сетевых протоколов).

Консоль `console` — исторически реализация терминала с клавиатурой и текстовым дисплеем. В настоящее время это слово часто используется как синоним сеанса работы или окна оболочки командной строки. В том же смысле иногда применяется и слово “терминал”.

2. Что такое консольное приложение?

Консольное приложение `console application` — вид ПО, разработанный с расчётом на работу внутри оболочки командной строки, т.е. опирающийся на текстовый ввод-вывод.

3. Какие существуют средства языка программирования Python для построения приложений командной строки?

Python 3 поддерживает несколько различных способов обработки аргументов командной строки.

Встроенный способ — использовать модуль `sys`. С точки зрения имен и использования, он имеет прямое отношение к библиотеке C (`libc`). Второй способ — это модуль `getopt`, который обрабатывает как короткие, так и длинные

параметры, включая оценку значений параметров.

4. Какие особенности построение CLI с использованием модуля `sys`?

Это базовый модуль, который с самого начала поставлялся с Python. Он использует подход, очень похожий на библиотеку C, с использованием `argc` и `argv` для доступа к аргументам. Модуль `sys` реализует аргументы командной строки в простой структуре списка с именем `sys.argv`

5. Какие особенности построение CLI с использованием модуля `getopt`?

Как вы могли заметить ранее, модуль `sys` разбивает строку командной строки только на отдельные фасы. Модуль `getopt` в Python идет немного дальше и расширяет разделение входной строки проверкой параметров.

Основанный на функции C getopt, он позволяет использовать как короткие, так и длинные варианты, включая присвоение значений.

6. Какие особенности построение CLI с использованием модуля argparse?

Начиная с версий Python 2.7 и Python 3.2, в набор стандартных библиотек была включена библиотека argparse для обработки аргументов (параметров, ключей) командной строки.

Для начала рассмотрим, что интересного предлагает argparse:

- * анализ аргументов sys.argv;
- * конвертирование строковых аргументов в объекты вашей программы и работа с ними;
- * форматирование и вывод информативных подсказок.