МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра инфокоммуникаций

Основы кроссплатформенного программирования

Отчет по лабораторной работе №2.17

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

Выполнил студент группы

ИВТ-б-о-21-1

Богадуров В.И. « »_____20__г.

Подпись студента_____

Работа защищена « »_____20__г.

Проверил доцент
Кафедры инфокоммуникаций, старший преподаватель
Воронкин Р.А.

_____
(подпись)

Ставрополь 2022

Цель работы: приобретение построения приложений с интерфейсом командной строки с помощью языка программирования Python версии 3.x.

Ход работы:

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.

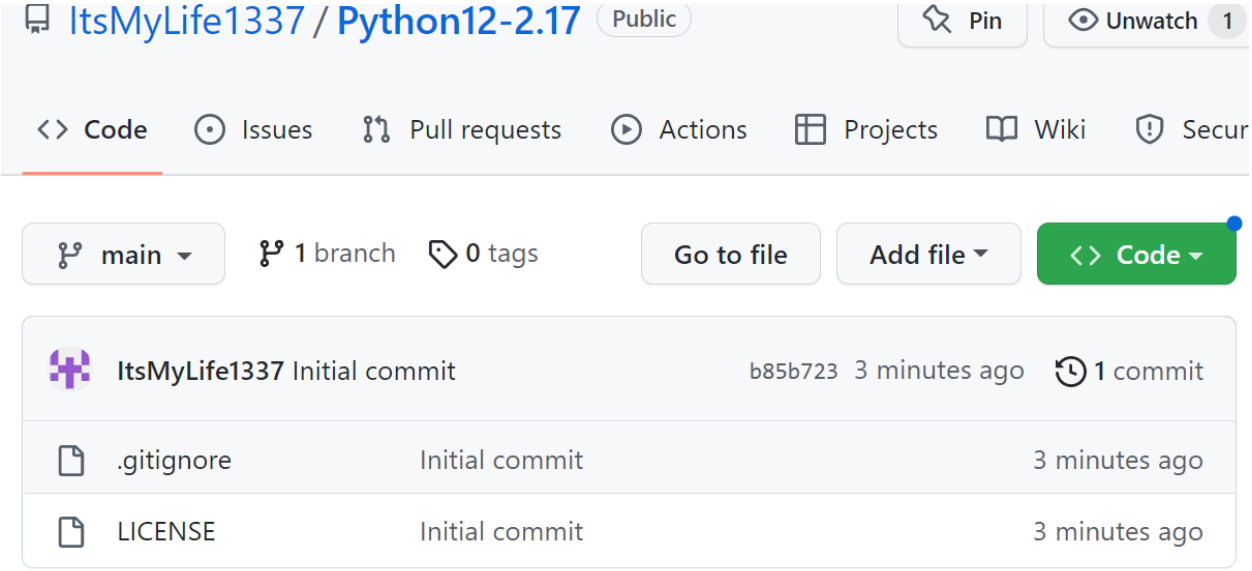

Рисунок 1.1 – Созданный репозиторий

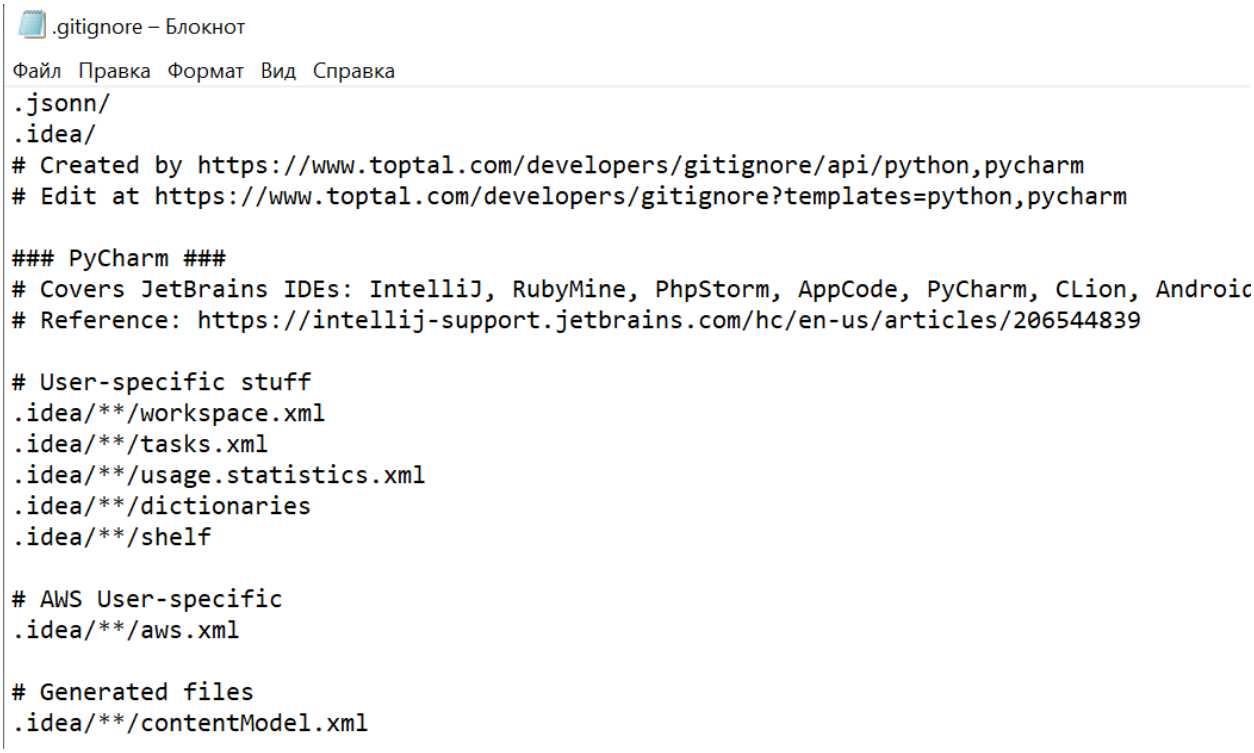

Рисунок 1.2 – Дополнил правила в .gitignore

```
c:\Users\Admin\Desktop\git\Python12-2.17>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/Admin/Desktop/git/Python12-2.17/.git/hooks]

c:\Users\Admin\Desktop\git\Python12-2.17>
```

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

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

```
c:\Users\Admin\Desktop\git\Python12-2.17\Primers>python primer1.py display data.json
+------+----------------------------------+----------------------+----------+
|  No  |             Ф.И.О.               |      Должность       |   Год    |
+------+----------------------------------+----------------------+----------+
|    1 | Сидоров Сидор                    | Главный инженер      |     2012 |
|    2 | Иванченко Никита                 | Старший слесарь      |     2008 |
+------+----------------------------------+----------------------+----------+

c:\Users\Admin\Desktop\git\Python12-2.17\Primers>
```

Рисунок 2 – Результат работы примера

Индивидуальное задание. Для своего варианта лабораторной работы 2.16 необходимо дополнительно реализовать интерфейс командной строки (CLI).

```
c:\Users\Admin\Desktop\git\Python12-2.17\Individual>python individual1.py add data.json --name="Черков БЛ" --group="6" --progress=3

c:\Users\Admin\Desktop\git\Python12-2.17\Individual>python individual1.py list data.json
+------+----------------------------------+----------------------+----------+
|  №   |             Ф.И.О.               |        Группа        |  Оценка  |
+------+----------------------------------+----------------------+----------+
|    1 | Иванов Антон                     | 1                    |        4 |
+------+----------------------------------+----------------------+----------+
|    2 | Черкассов ВЛ                     | 1                    |        2 |
+------+----------------------------------+----------------------+----------+
|    3 | Сидоров БК                       | 2                    |        5 |
+------+----------------------------------+----------------------+----------+
|    4 | Сидоров ВИ                       | 2                    |        2 |
+------+----------------------------------+----------------------+----------+
|    5 | Сидоров БК                       | 12                   |        5 |
+------+----------------------------------+----------------------+----------+
|    6 | Черков БЛ                        | 6                    |        3 |
+------+----------------------------------+----------------------+----------+

c:\Users\Admin\Desktop\git\Python12-2.17\Individual>
```

Рисунок 3 – Проверка работы программы(1)

```
c:\Users\Admin\Desktop\git\Python12-2.17\Individual>python individual1.py select data.json
+------+--------------------------------+----------------------+-----------------+
|  №   |             Ф.И.О.             |        Группа        |     Оценка      |
+------+--------------------------------+----------------------+-----------------+
|    1 | Иванов Антон                   | 1                    |               4 |
|    2 | Сидоров БК                     | 2                    |               5 |
|    3 | Сидоров БК                     | 12                   |               5 |
+------+--------------------------------+----------------------+-----------------+

c:\Users\Admin\Desktop\git\Python12-2.17\Individual>
```

Рисунок 4 – Проверка работы программы(2)

Задание повышенной сложности. Для своего варианта лабораторной работы 2.16 необходимо реализовать интерфейс командной строки с использованием пакета click.

```python
@click.command()
@click.option("-c", "--command")
@click.argument('file_name')
@click.option("-n", "--name")
@click.option("-g", "--group")
@click.option("-pr", "--progress")
def main(command, name, group, progress, file_name):
    students_load = load_students(file_name)
```

Рисунок 5 – Решение задачи с помощью пакета click

```
c:\Users\Admin\Desktop\git\Python12-2.17\Individual>python Individual2.py -c display data.json
+------+--------------------------------+----------------------+----------+
|  №   |             Ф.И.О.             |        Группа        |  Оценка  |
+------+--------------------------------+----------------------+----------+
|    1 | Иванов Антон                   | 1                    |        4 |
+------+--------------------------------+----------------------+----------+
|    2 | Черкассов ВЛ                   | 1                    |        2 |
+------+--------------------------------+----------------------+----------+
|    3 | Сидоров БК                     | 2                    |        5 |
+------+--------------------------------+----------------------+----------+
|    4 | Сидоров ВИ                     | 2                    |        2 |
+------+--------------------------------+----------------------+----------+
|    5 | Сидоров БК                     | 12                   |        5 |
+------+--------------------------------+----------------------+----------+

c:\Users\Admin\Desktop\git\Python12-2.17\Individual>python Individual2.py -c select data.json
+------+--------------------------------+----------------------+-----------------+
|  №   |             Ф.И.О.             |        Группа        |     Оценка      |
+------+--------------------------------+----------------------+-----------------+
|    1 | Иванов Антон                   | 1                    |               4 |
|    2 | Сидоров БК                     | 2                    |               5 |
|    3 | Сидоров БК                     | 12                   |               5 |
+------+--------------------------------+----------------------+-----------------+

c:\Users\Admin\Desktop\git\Python12-2.17\Individual>
```

Рисунок 6 – Результат работы программы

Вывод: в результате выполнения лабораторной работы были получены практические навыки и теоретические сведения для построения приложений с интерфейсом командной строки с помощью языка программирования Python версии 3.x.

Ответы на контрольные вопросы:

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 фасеты. 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.