Министерство науки и высшего образования Российской Федерации Федеральное государственное автономное образовательное учреждение высшего образования

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития Кафедра инфокоммуникаций

# ОТЧЕТ

**ПО ЛАБОРАТОРНОЙ РАБОТЕ №2.17**

# дисциплины «Анализ данных»

|  |
| --- |
| Выполнил:  Кенесбаев Хилол Куат улы  2 курс, группа ИВТ-б-о-22-1,  09.03.01 «Информатика и вычислительная техника», направленность (профиль)  «Программное обеспечение средств вычислительной техники и автоматизированных систем», очная форма обучения  (подпись) |
| Руководитель практики: Воронкин Р.А., доцент кафедры инфокоммуникаций  (подпись) |

Отчет защищен с оценкой \_ Дата защиты

Ставрополь, 2024 г.

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

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

# Порядок выполнения работы:

1. Создал новый репозиторий, клонировал его, в нем создал ветку developer и перешел на нее.
2. Проработал пример лабораторной работы:

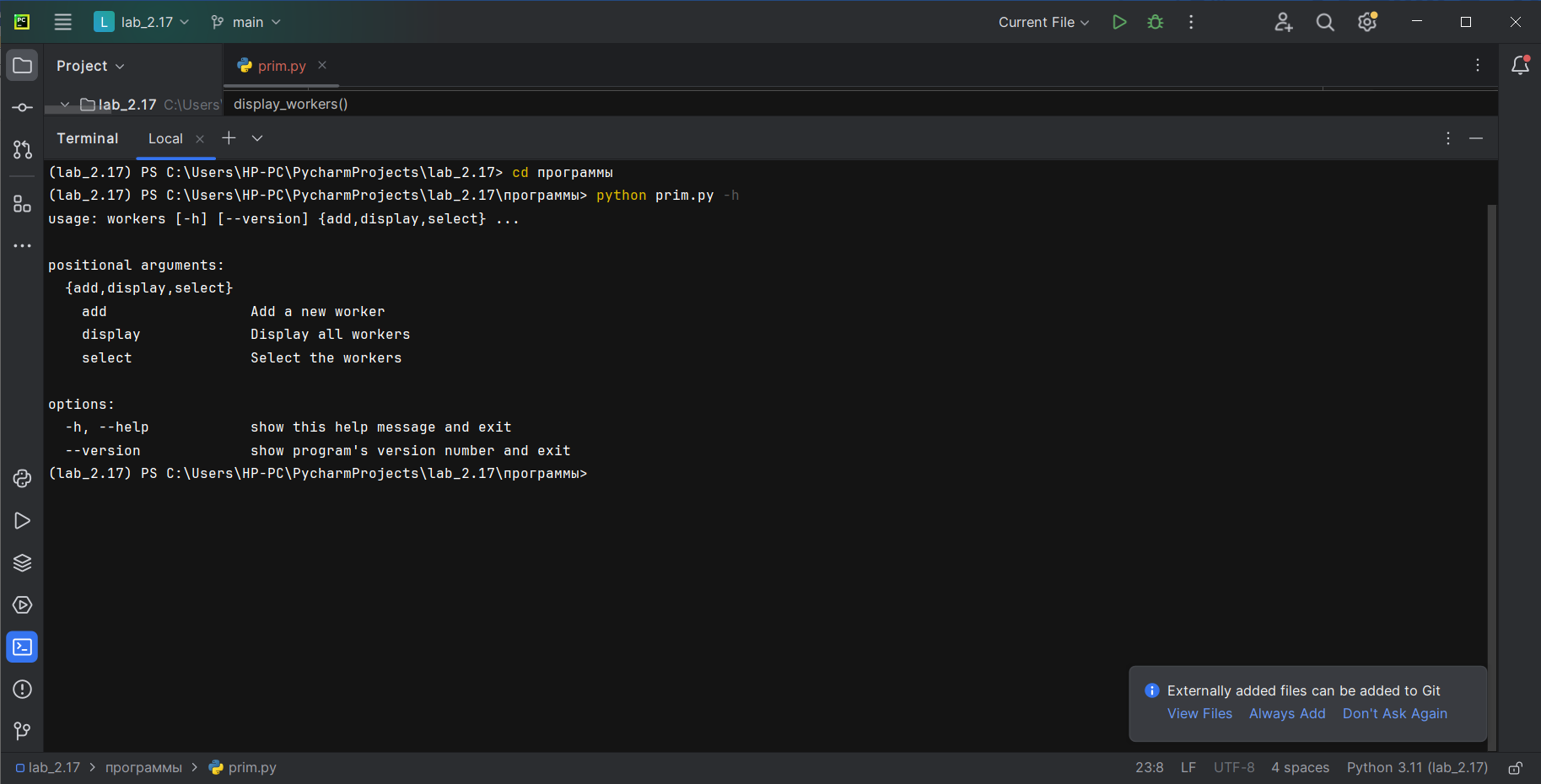


Рисунок 1. Страницы руководства

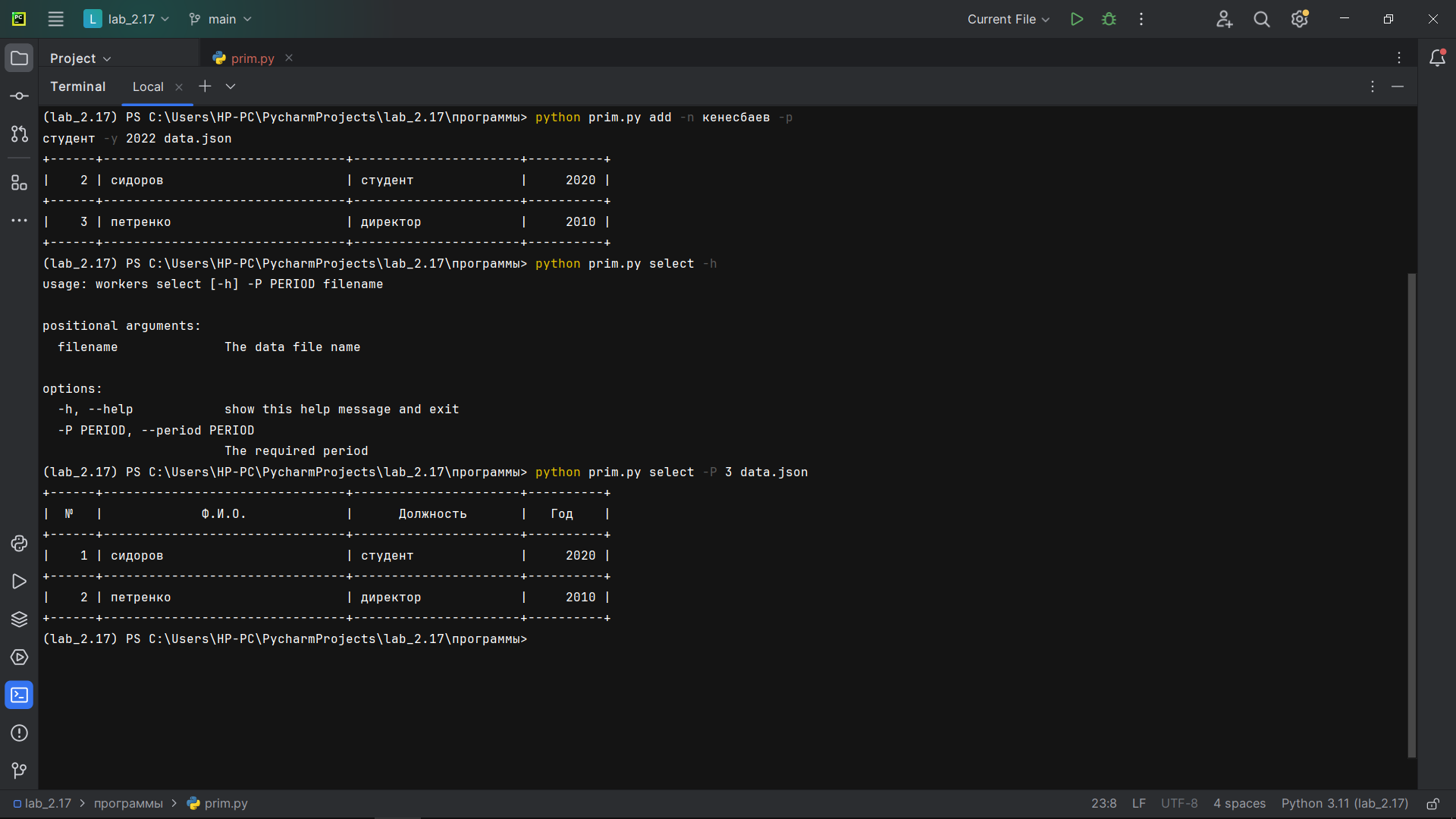


Рисунок 2. Ввод, вывод и выбор работников в консоли

1. Выполнил индивидуальное задание: для своего варианта лабораторной работы 2.16 необходимо дополнительно реализовать интерфейс командной строки (CLI).

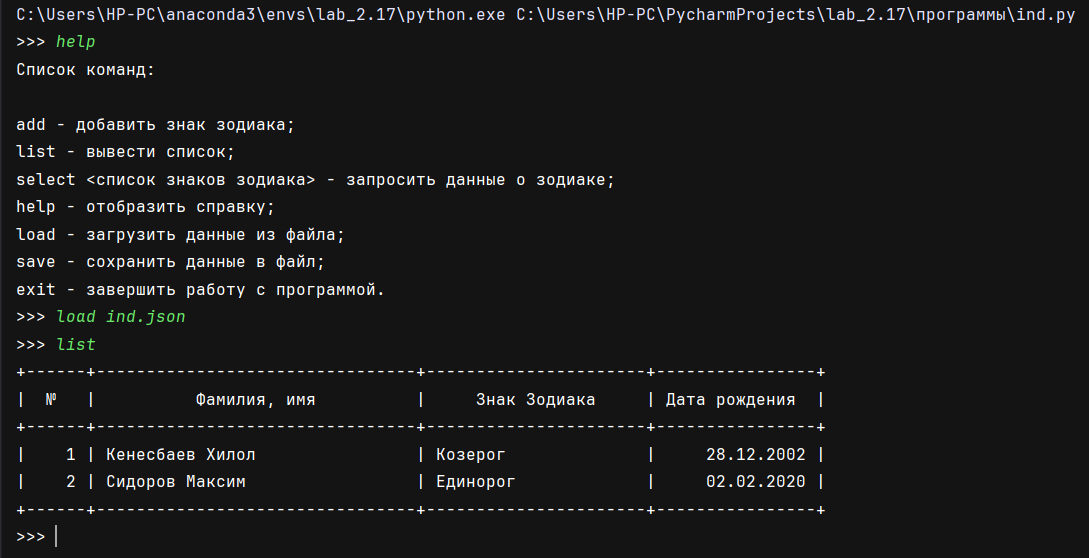


Рисунок 3. Страницы руководства и результат работы программы

# Код индивидуального задания №1:

import json  
import sys  
  
  
def get\_route():  
 *"""  
 Запросить данные о списке  
 """* start = input("Ведите фамилию, имя ")  
 finish = input("Введите знак Зодиака ")  
 zodiac = (input("Введите дату рождения "))  
  
 return {  
 'start': start,  
 'finish': finish,  
 'zodiac': zodiac,  
 }  
  
  
def display\_route(routes):  
 *"""  
 Отобразить список  
 """* if routes:  
 line = '+-{}-+-{}-+-{}-+-{}-+'.format(  
 '-' \* 4,  
 '-' \* 30,  
 '-' \* 20,  
 '-' \* 14  
 )  
 print(line)  
 print(  
 '| {:^4} | {:^30} | {:^20} | {:^14} |'.format(  
 "№",  
 "Фамилия, имя",  
 "Знак Зодиака",  
 "Дата рождения"  
 )  
 )  
 print(line)  
  
 for idx, worker in enumerate(routes, 1):  
 print(  
 '| {:>4} | {:<30} | {:<20} | {:>14} |'.format(  
 idx,  
 worker.get('start', ''),  
 worker.get('finish', ''),  
 worker.get('zodiac', 0)  
 )  
 )  
 print(line)  
 else:  
 print("Список пуст")  
  
  
def select\_route(routes, period):  
 *"""  
 Выбрать зодиак  
 """* result = []  
 for employee in routes:  
 if employee.get('finish') == period:  
 result.append(employee)  
  
 return result  
  
  
def save\_routes(file\_name, staff):  
 *"""  
 Сохранить данные в файл JSON  
 """* with open(file\_name, "w", encoding="utf-8") as fout:  
 json.dump(staff, fout, ensure\_ascii=False, indent=4)  
  
  
def load\_routes(file\_name):  
 *"""  
 Загрузить данные из файла JSON  
 """* with open(file\_name, "r", encoding="utf-8") as fin:  
 return json.load(fin)  
  
  
def main():  
 *"""  
 Главная функция программы  
 """* routes = []  
  
 while True:  
 command = input(">>> ").lower()  
 if command == 'exit':  
 break  
  
 elif command == 'add':  
 route = get\_route()  
 routes.append(route)  
 routes.sort(key=lambda item: int(item.get('zodiac', '').split('.')[2]))  
  
 elif command == 'list':  
 display\_route(routes)  
  
 elif command.startswith('select'):  
 parts = command.split(' ', maxsplit=1)  
 period = parts[1].strip() *# Получаем название знака Зодиака* selected = select\_route(routes, period)  
 if selected:  
 display\_route(selected)  
 else:  
 print("Нет людей с таким знаком Зодиака.")  
  
 elif command.startswith("save "):  
 parts = command.split(maxsplit=1)  
 file\_name = parts[1]  
 save\_routes(file\_name, routes)  
  
 elif command.startswith("load "):  
 parts = command.split(maxsplit=1)  
 file\_name = parts[1]  
 routes = load\_routes(file\_name)  
  
 elif command == 'help':  
 print("Список команд:\n")  
 print("add - добавить знак зодиака;")  
 print("list - вывести список;")  
 print("select <список знаков зодиака> - запросить данные о зодиаке;")  
 print("help - отобразить справку;")  
 print("load - загрузить данные из файла;")  
 print("save - сохранить данные в файл;")  
 print("exit - завершить работу с программой.")  
 else:  
 print(f"Неизвестная команда {command}", file=sys.stderr)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

# Ответы на контрольные вопросы:

1. Чем отличаются терминал и консоль?

**Ответ:** терминал – программа-оболочка, запускающая оболочку и позволяющая вводить команды. Консоль – разновидность терминала, это окно, в котором активны программы текстового режима.

1. Что такое консольное приложение?

**Ответ:** консольное приложение – программа, не имеющая графического интерфейса (окон), и которая работает в текстовом режиме в консоли. Команды в такой программе нужно вводить с клавиатуры, результаты работы консольные приложения также выводят на экран в текстовом виде.

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

**Ответ:** модуль sys (предоставляет доступ к некоторым переменным и функциям, взаимодействующим с интерпретатором Python) и модуль argparse (Позволяет создавать красивые и гибкие интерфейсы командной строки с

автоматической генерацией справки и поддержкой нескольких параметров командной строки).

1. Какие особенности построения CLI с использованием модуля sys?

**Ответ:** sys.argv – позволяет получить список аргументов командной строки. Эквивалент argc – количество элементов в списке (Получается от len()).

1. Какие особенности построения CLI с использованием модуля getopt?

**Ответ:** Модуль getopt в Python расширяет разделение входной строки проверкой параметров. Основанный на функции C getopt, он позволяет использовать как короткие, так и длинные варианты, включая присвоение значений. Удобен для простых CLI, но может быть не так гибок и мощен, как argparse.

1. Какие особенности построения CLI с использованием модуля argparse?

**Ответ:** особенности построения CLI с использованием модуля argparse: Поддержка создания позиционных аргументов и флагов.

* 1. Возможность создания подкоманд для более сложных CLI.
  2. Автоматическая генерация справки.
  3. Поддержка типизации аргументов и их ограничений.
  4. Гибкая конфигурация для обработки различных сценариев использования.
  5. Часто используется для создания профессиональных и гибких CLI-интерфейсов.

**Вывод**: в ходе выполнения лабораторной работы, приобретены навыки построения приложений с интерфейсом командной строки с помощью языка программирования Python версии 3.x.