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_____ **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.х.

Порядок выполнения работы:

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

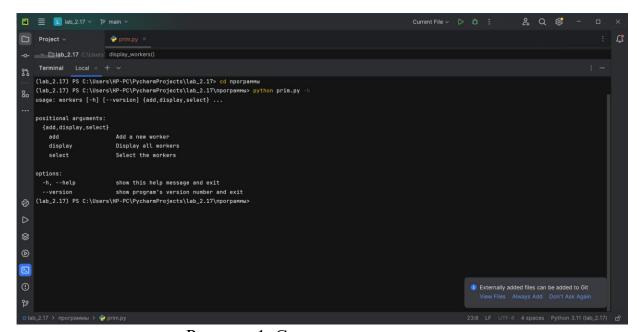


Рисунок 1. Страницы руководства

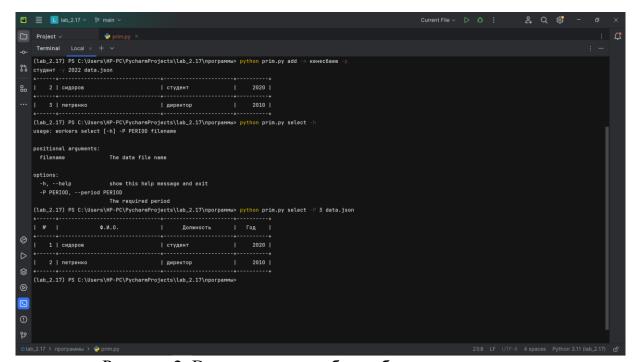


Рисунок 2. Ввод, вывод и выбор работников в консоли

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

```
(lab_2.17) PS C:\Users\HP-PC\PycharmProjects\lab_2.17\программы> python ind.py
Список команд:
add - добавить знак зодиака;
list - вывести список;
select <список знаков зодиака> - запросить данные о зодиаке;
help - отобразить справку;
load - загрузить данные из файла;
save - сохранить данные в файл;
exit - завершить работу с программой.
>>> load individ.json
JSON валиден по схеме.
>>> list
| № | Фамилия, имя | Знак Зодиака | Дата рождения |
+-----

      1 | Kenesbayev Hilol
      | Kozerog
      | 28.12.2002 |

      2 | Sidorov Ivan
      | Edinorog
      | 12.12.2012 |

+-----
```

Рисунок 3. Страницы руководства и результат работы программы

Код индивидуального задания №1:

```
print(line)
         print(
                 {:^4} | {:^30} | {:^20} | {:^14} | '.format(
         print(line)
         for idx, worker in enumerate (routes, 1):
             print(
                     \{:>4\} \mid \{:<30\} \mid \{:<20\} \mid \{:>14\} \mid '.format(
                      worker.get('start', ''),
worker.get('finish', ''),
worker.get('zodiac', '')
         print(line)
    else:
         print("Список пуст")
def select route(routes, period):
    result = []
    for employee in routes:
         if employee.get('finish') == period:
             result.append(employee)
    return result
def save routes(file name, staff):
    with open (file name, "w", encoding="utf-8") as fout:
         json.dump(staff, fout, ensure ascii=False, indent=4)
def load routes (file name):
                  "zodiac": {"type": "string"},
    with open(file_name, "r") as file_in:
```

```
data = json.load(file in) # Прочитать данные из файла
   try:
       print("JSON валиден по схеме.")
   except ValidationError as e:
       print(f"Ошибка валидации: {e.message}")
   return data # Вернуть загруженные и проверенные данные
def main():
   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'):
           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)
           save routes(file name, routes)
       elif command.startswith("load "):
           parts = command.split(maxsplit=1)
           routes = load routes(file name)
       elif command == 'help':
           print("Список команд:\n")
           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(f"Неизвестная команда {command}", file=sys.stderr)
   __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.

2) 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.

3) Какие существуют средства языка программирования Руthon 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).

4) 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()).

5) Какие особенности построения CLI с использованием модуля getopt?

Ответ: Модуль getopt в Python расширяет разделение входной строки проверкой параметров. Основанный на функции С 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.

6) Какие особенности построения CLI с использованием модуля argparse?

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

- а) Возможность создания подкоманд для более сложных CLI.
- b) Автоматическая генерация справки.
- с) Поддержка типизации аргументов и их ограничений.
- d) Гибкая конфигурация для обработки различных сценариев использования.
- е) 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.х.