

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 №3
дисциплины «Анализ данных»

Выполнил:
Говоров Егор Юрьевич
2 курс, группа ИВТ-б-о-22-1,
22.04.2024 «Информатика
и вычислительная
техника», направленность
(профиль)
«Программное обеспечение средств
вычислительной техники и
автоматизированных систем», очная
форма обучения

(подпись)

Руководитель практики:
Воронкин Р. А., канд. технических
наук, доцент кафедры
инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

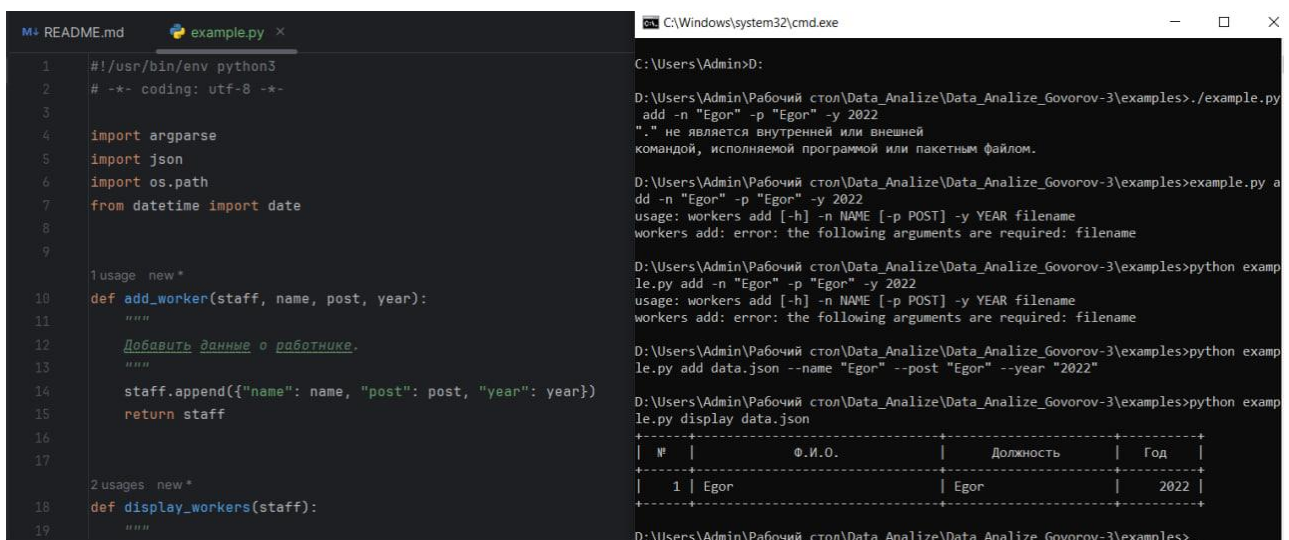
Ставрополь, 2024 г.

Тема: Разработка приложений с интерфейсом командной строки (CLI) в Python3
Цель работы: приобретение навыков по работе с данными формата JSON с помощью языка программирования Python версии 3.x.

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

Ход работы

1. Создал общедоступный репозиторий на GitHub, в котором использована лицензия MIT и язык программирования Python. Выполнил клонирование созданного репозитория.
2. Дополнил файл .gitignore необходимыми правилами.
3. Организовал созданный репозиторий в соответствии с необходимыми требованиями.
4. Проработал примеры лабораторной работы. Создал для них отдельные модули языка Python. Привел в отчете скриншоты результата выполнения программ примеров при различных исходных данных, вводимых с клавиатуры.



The screenshot shows a code editor on the left with a file named `example.py` and a terminal window on the right. The code in `example.py` is a CLI application for managing a list of workers. It uses `argparse` for command-line arguments and `json` for data storage. The terminal shows the execution of the script with various commands and their outputs, including a table of workers.

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import argparse
5 import json
6 import os.path
7 from datetime import date
8
9
10 usage new *
11 def add_worker(staff, name, post, year):
12     """
13     Добавить данные о работнике.
14     """
15     staff.append({"name": name, "post": post, "year": year})
16     return staff
17
18 2 usages new *
19 def display_workers(staff):
20     """
21     """
```

Terminal output:

```
C:\Users\Admin>D:
D:\Users\Admin\Рабочий стол\Data_Analyze\Data_Analyze_Govorov-3\examples>./example.py
add -n "Egor" -p "Egor" -y 2022
"." не является внутренней или внешней
командой, исполняемой программой или пакетным файлом.
D:\Users\Admin\Рабочий стол\Data_Analyze\Data_Analyze_Govorov-3\examples>example.py a
dd -n "Egor" -p "Egor" -y 2022
usage: workers add [-h] -n NAME [-p POST] -y YEAR filename
workers add: error: the following arguments are required: filename
D:\Users\Admin\Рабочий стол\Data_Analyze\Data_Analyze_Govorov-3\examples>python examp
le.py add -n "Egor" -p "Egor" -y 2022
usage: workers add [-h] -n NAME [-p POST] -y YEAR filename
workers add: error: the following arguments are required: filename
D:\Users\Admin\Рабочий стол\Data_Analyze\Data_Analyze_Govorov-3\examples>python examp
le.py add data.json --name "Egor" --post "Egor" --year "2022"
D:\Users\Admin\Рабочий стол\Data_Analyze\Data_Analyze_Govorov-3\examples>python examp
le.py display data.json
+-----+-----+-----+-----+
| № | Ф.И.О. | Должность | Год |
+-----+-----+-----+-----+
| 1 | Egor | Egor | 2022 |
+-----+-----+-----+-----+
```

Рисунок 1. Результат работы программы из примера 1

5. Выполнил индивидуальные задания, согласно варианту 5. Привёл в отчете скриншоты работы программ.

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

```

D:\Users\Admin\Рабочий стол\Data_Analyze\Data_Analyze_Govorov-3\individ>python individ.py -a
Введите название пункта назначения: Москва-Таганрог
Введите номер рейса: 12
Введите тип самолета: Грузовой

D:\Users\Admin\Рабочий стол\Data_Analyze\Data_Analyze_Govorov-3\individ>python individ.py -a
Введите название пункта назначения: Ставрополь-Москва
Введите номер рейса: 34
Введите тип самолета: Пассажирский

D:\Users\Admin\Рабочий стол\Data_Analyze\Data_Analyze_Govorov-3\individ>python individ.py -p
+-----+-----+-----+
| Название пункта назначения | Номер рейса | Тип самолета |
+-----+-----+-----+
| Москва-Таганрог           | 12          | Грузовой     |
| Ставрополь-Москва         | 34          | Пассажирский |
+-----+-----+-----+

```

Рисунок 2. Результат работы программы из индивидуального задания 1

```

[
  {
    "название пункта назначения": "Москва-Таганрог",
    "номер рейса": "12",
    "тип самолета": "Грузовой"
  },
  {
    "название пункта назначения": "Ставрополь-Москва",
    "номер рейса": "34",
    "тип самолета": "Пассажирский"
  }
]

```

Рисунок 3. Файл individ.json

Задание. Самостоятельно изучите работу с пакетом click для построения интерфейса командной строки (CLI). Для своего варианта лабораторной работы 2.16 необходимо реализовать интерфейс командной строки с использованием пакета click .

```

@click.command()
@click.option('--add-flight', is_flag=True, help='Add a new flight')
@click.option('--print-flights', is_flag=True, help='Print the list of flights')
@click.option('--search-by-type', help='Search flights by aircraft type')
@click.option('--file', default='flights.json', help='JSON file to load/save flight data')
def main(add_flight, print_flights, search_by_type, file):
    if add_flight:
        destination = click.prompt(text='Введите название пункта назначения', type=str)
        flight_number = click.prompt(text='Введите номер рейса', type=str)
        aircraft_type = click.prompt(text='Введите тип самолета', type=str)
        flights_list = load_from_json(file)
        flights_list = add_flight(destination, flight_number, aircraft_type, flights_list)
        save_to_json(file, flights_list)

    elif print_flights:
        flights_list = load_from_json(file)
        print_flights(flights_list)

    elif search_by_type:
        flights_list = load_from_json(file)
        search_flights_by_aircraft_type(flights_list, search_by_type)

    else:
        click.echo("Пожалуйста, выберите действие из списка: --add-flight, --print-flights, or --search-by-type")

if __name__ == '__main__':
    main()

```

Рисунок 3. работа с пакетом click

Контрольные вопросы

1. Отличие между терминалом и консолью

Терминал и консоль – это термины, связанные с работой в командной строке операционной системы.

Терминал – это физическое устройство, которое позволяет пользователю взаимодействовать с компьютером посредством текстового интерфейса.

Консоль – это программное обеспечение, предоставляющее пользователю доступ к командной строке операционной системы.

2. Консольное приложение и его определение

Консольное приложение – это программа, которая работает в командной строке операционной системы. Она взаимодействует с пользователем через текстовый интерфейс, принимая команды и предоставляя результаты выполнения.

3. Средства языка программирования Python для построения приложений командной строки

Для построения приложений командной строки на языке программирования Python существуют несколько средств:

`sys.argv` - это список аргументов командной строки, передаваемых при запуске скрипта на Python.

`getopt` – модуль Python для парсинга аргументов командной строки.

`argparse` – модуль Python для создания гибких командных интерфейсов.

4. Особенности построения CLI с использованием модуля `sys`

Модуль `sys` в Python предоставляет доступ к некоторым переменным и функциям, связанным с интерпретатором Python и его окружением. Он позволяет работать с аргументами командной строки и другими системными параметрами.

5. Особенности построения CLI с использованием модуля `getopt`

Модуль `getopt` в Python предоставляет средства для парсинга аргументов командной строки. Он позволяет обрабатывать опции и аргументы командной строки, упрощая разработку приложений командной строки.

6. Особенности построения CLI с использованием модуля argparse

Модуль `argparse` в Python предоставляет более гибкие средства для создания командных интерфейсов. Он позволяет определять аргументы, опции и подкоманды, а также автоматически генерировать справку для пользователей.

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