

Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

ОТЧЁТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №3
дисциплины «Анализ данных»
Вариант 28

Выполнил:
Репкин Александр Павлович
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., канд. техн. наук,
доцент кафедры инфокоммуникаций

(подпись)

Отчет защищен с оценкой _____ Дата защиты _____

Ставрополь, 2024 г.

Тема: Разработка приложений с интерфейсом командной строки (CLI) в Python 3.

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

Порядок выполнения работы:

1. Выполнен пример лабораторной работы – модифицирован пример №1 из Лабораторной работы №3 дисциплины “Анализ данных”: в примере создаётся словарь, хранящий информацию о сотрудниках – ФИО, должность, год поступления на работу. Обработываемые команды: вывод всех сотрудников (display), вывод сводки о имеющихся командах (help), добавление сотрудника (add), вывод сотрудников относительно требуемой продолжительности работы (select --period), выход из программы (exit), сохранение данных о сотрудниках в файл (save), загрузка данных о сотрудниках из файла (load). Теперь к данной программе также добавлен интерфейс командной строки CLI.

```
93 def main(command_line=None):
94     # Создать родительский парсер для определения имени файла.
95     file_parser = argparse.ArgumentParser(add_help=False)
96     file_parser.add_argument(
97         "filename",
98         action="store",
99         help="The data file name"
100     )
101     # Создать основной парсер командной строки.
102     parser = argparse.ArgumentParser("workers")
103     parser.add_argument(
104         "--version",
105         action="version",
106         version="%(prog)s 0.1.0"
107     )
108     subparsers = parser.add_subparsers(dest="command")
109     # Создать субпарсер для добавления работника.
110     add = subparsers.add_parser(
111         "add",
112         parents=[file_parser],
113         help="Add a new worker"
114     )
115     add.add_argument(
116         "-n",
117         "--name",
118         action="store",
119         required=True,
120         help="The worker's name"
121     )
122     add.add_argument(
123         "-p",
124         "--post",
125         action="store",
126         help="The worker's post"
127     )
128     add.add_argument(
129         "-y",
```

Рисунок 1. Код примера

```
1 {
2   {
3     "name": "Кики К.К.",
4     "post": "Доставщик",
5     "year": 1989
6   },
7   {
8     "name": "Хаку К.",
9     "post": "Спасатель",
10    "year": 2001
11  }
12 }
```

Рисунок 2. Содержимое файла json до выполнения команды

```
1 {
2   {
3     "name": "Кики К.К.",
4     "post": "Доставщик",
5     "year": 1989
6   },
7   {
8     "name": "Хаку К.",
9     "post": "Спасатель",
10    "year": 2001
11  },
12  {
13    "name": "Бенедикт Б.",
14    "post": "Почтальон",
15    "year": 1910
16  }
17 }
```

Рисунок 3. Содержимое файла json после выполнения команды

```
(Data_Analysis) C:\Users\yabuz\GitHub\Data Analysis\Data_Analysis_3\Программы и результаты>python Example_1.py add observe.json --name="Бенедикт Б." --post="Почтальон" --year=1910
```

Рисунок 4. Команда, добавившая запись в json файл

```
(Data_Analysis) C:\Users\yabuz\GitHub\Data Analysis\Data_Analysis_3\Программы и результаты>python Example_1.py display observe.json
```

№	Ф.И.О.	Должность	Год
1	Кики К.К.	Доставщик	1989
2	Хаку К.	Спасатель	2001
3	Бенедикт Б.	Почтальон	1910

Рисунок 5 Вывод всех сотрудников

```
(Data_Analysis) C:\Users\yabuz\GitHub\Data Analysis\Data_Analysis_3\Программы и результаты>python Example_1.py select observe.json --period=12
```

№	Ф.И.О.	Должность	Год
1	Кики К.К.	Доставщик	1989
2	Хаку К.	Спасатель	2001
3	Бенедикт Б.	Почтальон	1910


```
(Data_Analysis) C:\Users\yabuz\GitHub\Data Analysis\Data_Analysis_3\Программы и результаты>python Example_1.py select observe.json --period=112
```

№	Ф.И.О.	Должность	Год
1	Бенедикт Б.	Почтальон	1910

Рисунок 6. Вывод работников с заданным периодом работы из файла

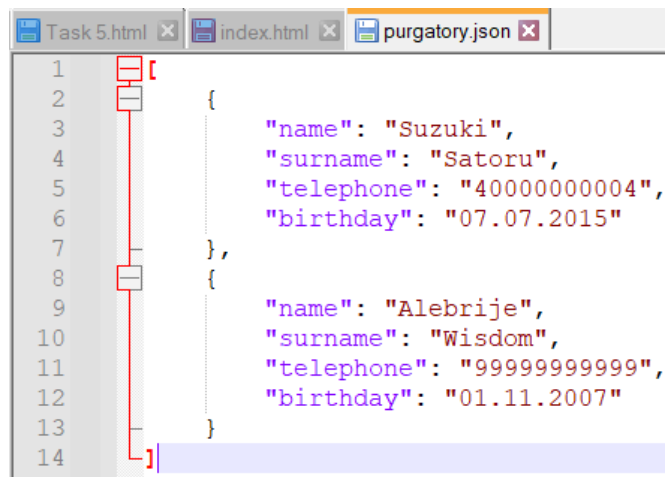
2. Выполнено индивидуальное задание – модифицировано задание №9 из Лабораторной работы №11 дисциплины “Программирование на Python”: в задании создаётся словарь, хранящий информацию о людях – фамилия, имя, телефонный номер, дата рождения. Требовалось обрабатывать несколько команд – вывод всех людей (list), вывод сводки о имеющихся командах (help), добавление человека (add), вывод людей относительно требуемого месяца рождения (select “месяц”), выход из программы (exit). Теперь данные о внесённых в список людях сохраняются и могут быть считаны в файл формата JSON (Команды save и load соответственно).

```
76 def main(command_line=None):
77     """Главная функция программы."""
78     # Создать родительский парсер для определения имени файла.
79     file_parser = argparse.ArgumentParser(add_help=False)
80     file_parser.add_argument(
81         "filename",
82         action="store",
83         help="The name of data file."
84     )
85     # Создать основной парсер командной строки.
86     parser = argparse.ArgumentParser("people")
87     parser.add_argument(
88         "--version",
89         action="version",
90         version="%(prog)s 0.1.0"
91     )
92     subparsers = parser.add_subparsers(dest="command")
93     # Создать субпарсер для добавления человека.
94     add = subparsers.add_parser(
95         "add",
96         parents=[file_parser],
97         help="Add a record about new human."
98     )
99     add.add_argument(
100         "-n",
101         "--name",
102         action="store",
103         required=True,
104         help="The human's name."
105     )
106     add.add_argument(
107         "-s",
108         "--surname",
109         action="store",
110         required=True,
111         help="The human's post."
112     )
```

Рисунок 7. Код индивидуального задания

```
(Data_Analysis) C:\Users\yabuz\GitHub\Data Analysis\Data_Analysis_3\Программы и результаты>python Task.py add purgatory.json --name="Angus" --surname="Bambi" --telephone="30403040304" --birthday="14.06.2011"
```

Рисунок 8. Команда, добавившая запись в json файл



```
1  [
2
3      {
4          "name": "Suzuki",
5          "surname": "Satoru",
6          "telephone": "40000000004",
7          "birthday": "07.07.2015"
8      },
9
10     {
11         "name": "Alebrije",
12         "surname": "Wisdom",
13         "telephone": "99999999999",
14         "birthday": "01.11.2007"
15     }
16 ]
```

Рисунок 9. Json файл до добавления новой записи



```
1  [
2
3      {
4          "name": "Suzuki",
5          "surname": "Satoru",
6          "telephone": "40000000004",
7          "birthday": "07.07.2015"
8      },
9
10     {
11         "name": "Alebrije",
12         "surname": "Wisdom",
13         "telephone": "99999999999",
14         "birthday": "01.11.2007"
15     },
16
17     {
18         "name": "Angus",
19         "surname": "Bambi",
20         "telephone": "30403040304",
21         "birthday": "14.06.2011"
22     }
23 ]
```

Рисунок 10. Json файл после добавления новой записи

```
(Data_Analysis) C:\Users\yabuz\GitHub\Data Analysis\Data_Analysis_3\Программы и результаты>python Task.py display purgatory.json
```

№	Name	Surname	Telephone	Birthday
1	Suzuki	Satoru	40000000004	07.07.2015
2	Alebrije	Wisdom	99999999999	01.11.2007
3	Angus	Bambi	30403040304	14.06.2011

Рисунок 11. Вывод всех добавленных людей

```
(Data_Analysis) C:\Users\yabuz\GitHub\Data Analysis\Data_Analysis_3\Программы и результаты>python Task.py select purgato
ry.json --period=02
There are no people in list!

(Data_Analysis) C:\Users\yabuz\GitHub\Data Analysis\Data_Analysis_3\Программы и результаты>python Task.py select purgato
ry.json --period=11
```

№	Name	Surname	Telephone	Birthday
1	Alebrije	Wisdom	99999999999	01.11.2007

Рисунок 12. Вывод людей, родившихся в требуемом месяце

3. Выполнено задание повышенной сложности – использован пакет `click` для построения интерфейса командной строки. Производилась модификация задания из работы №2 дисциплины “Анализ Данных”.

```
1  #!/usr/bin/env python3
2  # -*- coding: utf-8 -*-
3
4  import click
5  import json
6  import jsonschema
7
8
9  @click.group()
10 @click.version_option(version="0.1.0")
11 def cli():
12     pass
13
14
15 @cli.command()
16 @click.argument('filename')
17 @click.option('-n', '--name', required=True, help="The human's name.")
18 @click.option('-s', '--surname', required=True, help="The human's surname.")
19 @click.option('-t', '--telephone', required=True, help="The human's telephone number.")
20 @click.option('-b', '--birthday', required=True, help="The human's birthday.")
21 def add(filename, name, surname, telephone, birthday):
22     """Добавить запись в новом человеке."""
23     people = load_people(filename)
24     people.append({
25         "name": name,
26         "surname": surname,
27         "telephone": telephone,
28         "birthday": birthday
29     })
30     save_people(filename, people)
31
32
33 @cli.command()
34 @click.argument('filename')
35 def display(filename):
36     """Вывести список всех людей."""
37     people = load_people(filename)
```

Рисунок 13. Код усложнённого задания

```
(Data_Analysis) C:\Users\yabuz\GitHub\Data Analysis\Data_Analysis_3\Программы и результаты>python Hard_Task.py --help
Usage: Hard_Task.py [OPTIONS] COMMAND [ARGS]...

Options:
  --version Show the version and exit.
  --help Show this message and exit.

Commands:
  add Добавить запись о новом человеке.
  display Вывести список всех людей.
  select Выбрать людей по требуемому месяцу рождения.

(Data_Analysis) C:\Users\yabuz\GitHub\Data Analysis\Data_Analysis_3\Программы и результаты>python Hard_Task.py add --help
Usage: Hard_Task.py add [OPTIONS] FILENAME

Добавить запись о новом человеке.

Options:
  -n, --name TEXT The human's name. [required]
  -s, --surname TEXT The human's surname. [required]
  -t, --telephone TEXT The human's telephone number. [required]
  -b, --birthday TEXT The human's birthday. [required]
  --help Show this message and exit.

(Data_Analysis) C:\Users\yabuz\GitHub\Data Analysis\Data_Analysis_3\Программы и результаты>python Hard_Task.py display observe.json
>>> Data's structure is invalid. Please, check your JSON file. Error: 'surname' is a required property
There are no people in list!

(Data_Analysis) C:\Users\yabuz\GitHub\Data Analysis\Data_Analysis_3\Программы и результаты>python Hard_Task.py display purgatory.json
>>> Data is obtained!
```

№	Name	Surname	Telephone	Birthday
1	Suzuki	Satoru	40000000004	07.07.2015
2	Alebrije	Wisdom	99999999999	01.11.2007
3	Angus	Bambi	30403040304	14.06.2011

Рисунок 14. Выполнение программы через командную строку

4. Ответы на 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 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).

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 `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`.

6) Какие особенности построения CLI с использованием модуля `argparse`?

Ответ: особенности построения CLI с использованием модуля `argparse`:

- a) Поддержка создания позиционных аргументов и флагов.
- b) Возможность создания подкоманд для более сложных CLI.
- c) Автоматическая генерация справки.
- d) Поддержка типизации аргументов и их ограничений.
- e) Гибкая конфигурация для обработки различных сценариев использования.
- f) 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.