

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
дисциплины «Анализ данных»

Выполнил:
Данилецкий Дмитрий Витальевич
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. 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 соответствие 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.

```
pr1.py U X
progr > pr1.py > main
104 def main(command_line=None):
158     # Создать субпарсер для выбора работ
159     select = subparsers.add_parser(
160         "select",
161         parents=[file_parser],
162         help="Select the workers"
163     )
164     select.add_argument(
165         "-p",
166         "--period",
167         action="store",
168         type=int,
169         required=True,
170         help="The required period"
171     )
172
173     # Выполнить разбор аргументов команд
174     args = parser.parse_args(command_line)
175
176     # Загрузить всех работников из файла
177     is_dirty = False
178     if os.path.exists(args.filename):
179         workers = load_workers(args.filename)
180     else:
181         workers = []
182
183     # Добавить работника.
184     if args.command == "add":
185         workers = add_worker(
186             workers,
187             args.name,
188             args.post,
189             args.year
190         )
191         is_dirty = True
192
193     # Отобразить всех работников.
194     elif args.command == "display":
195         display_workers(workers)
196
197     # Командная строка
198     C:\универ2\git\DATA3\progr>python pr1.py display data.json
199
200     +-----+-----+-----+-----+
201     | № | Сидоров Сидор | Главный инженер | 2012 |
202     +-----+-----+-----+-----+
203     | 6 | Сидоров Сидор | Главный инженер | 100 |
204     +-----+-----+-----+-----+
205     | 7 | Иванов Иван | Директор | 2007 |
206     +-----+-----+-----+-----+
207     | 8 | Данилецкий Данил | Пушистый | 2004 |
208     +-----+-----+-----+-----+
209
210     C:\универ2\git\DATA3\progr>python pr1.py add data.json --name "Я" --post "Директор" --ye
211     usage: workers add [-h] -n NAME [-p POST] -y YEAR filename
212     workers add: error: the following arguments are required: -n/--name, -y/--year
213
214     C:\универ2\git\DATA3\progr>python pr1.py add data.json --name "Я" --post "Пушистый" --ye
215
216     C:\универ2\git\DATA3\progr>python pr1.py display data.json
217
218     +-----+-----+-----+-----+
219     | № | Ф.И.О. | Должность | Год |
220     +-----+-----+-----+-----+
221     | 1 | Иванов Иван | Директор | 2007 |
222     +-----+-----+-----+-----+
223     | 2 | Петров Петр | Бухгалтер | 2010 |
224     +-----+-----+-----+-----+
225     | 3 | Сидоров Сидор | Главный инженер | 2012 |
226     +-----+-----+-----+-----+
227     | 4 | Данилецкий Данил | Пушистый | 2004 |
228     +-----+-----+-----+-----+
229     | 5 | Я | Пушистый | 2004 |
230     +-----+-----+-----+-----+
231
232     C:\универ2\git\DATA3\progr>
```

Рисунок 1. Результат работы программы из примера 1

5. Выполнил индивидуальные задания, согласно варианту 8. Привёл в отчете скриншоты работы программ.

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

```
C:\универ2\git\DATA3\progr>python ind.py add ind.json --departure "Ставрополь" --number "222" --time "18:30" --destination "Ростов"
C:\универ2\git\DATA3\progr>python ind.py add ind.json --departure "Ростов" --number "222" --time "23:30" --destination "Ставрополь"
C:\универ2\git\DATA3\progr>python ind.py select ind.json --point "18:30"
Список поездов пуст.
C:\универ2\git\DATA3\progr>python ind.py select ind.json --point "Ставрополь"
Список поездов пуст.
C:\универ2\git\DATA3\progr>python ind.py display ind.json
+-----+-----+-----+-----+-----+
| № | Пункт отправления | Номер поезда | Время отправления | Пункт назначения |
+-----+-----+-----+-----+-----+
| 1 | Ставрополь | 222 | 18:30 | Ростов |
+-----+-----+-----+-----+-----+
| 2 | Ростов | 222 | 23:30 | Ставрополь |
+-----+-----+-----+-----+-----+
C:\универ2\git\DATA3\progr>python ind.py select ind.json --point "Ставрополь"
```

Рисунок 2. Результат работы программы из индивидуального задания 1

```
[
  {
    "departure_point": "Ставрополь",
    "number_train": "222",
    "time_departure": "18:30",
    "destination": "Ростов"
  },
  {
    "departure_point": "Ростов",
    "number_train": "222",
    "time_departure": "23:30",
    "destination": "Ставрополь"
  }
]
```

Рисунок 3. Файл ind.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`.

```
122
123
124 @command.command()
125 @click.argument("filename")
126 @click.option("-p", "--point_user", required=True, help="Destination train")
127 def select(filename, point_user):
128     """
129     Выбрать
130     """
131     filename = os.path.join("data", filename)
132     point_user = point_user.lower()
133     trains = load_trains(filename)
134     selected_trains = select_trains(trains, point_user)
135     display_trains(selected_trains)
136
137
138 @command.command()
139 @click.argument("filename")
140 def display(filename):
141     """
142     Отобразить
143     """
144     filename = os.path.join("data", filename)
145     trains = load_trains(filename)
146     display_trains(trains)
147
148
149 if __name__ == "__main__":
150     command()
```

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.