

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное
образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра инфокоммуникаций

Отчет по лабораторной работе №12
Разработка приложений с интерфейсом командной строки (CLI) в
Python3
по дисциплине «Технологии программирования и алгоритмизация»

Выполнил студент группы ИВТ-б-о-20-1

Бобров Н. В. « » _____ 20__ г.

Подпись студента _____

Работа защищена « » _____ 20__ г.

Проверил Воронкин Р. А. _____

(подпись)

Ставрополь 2021

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

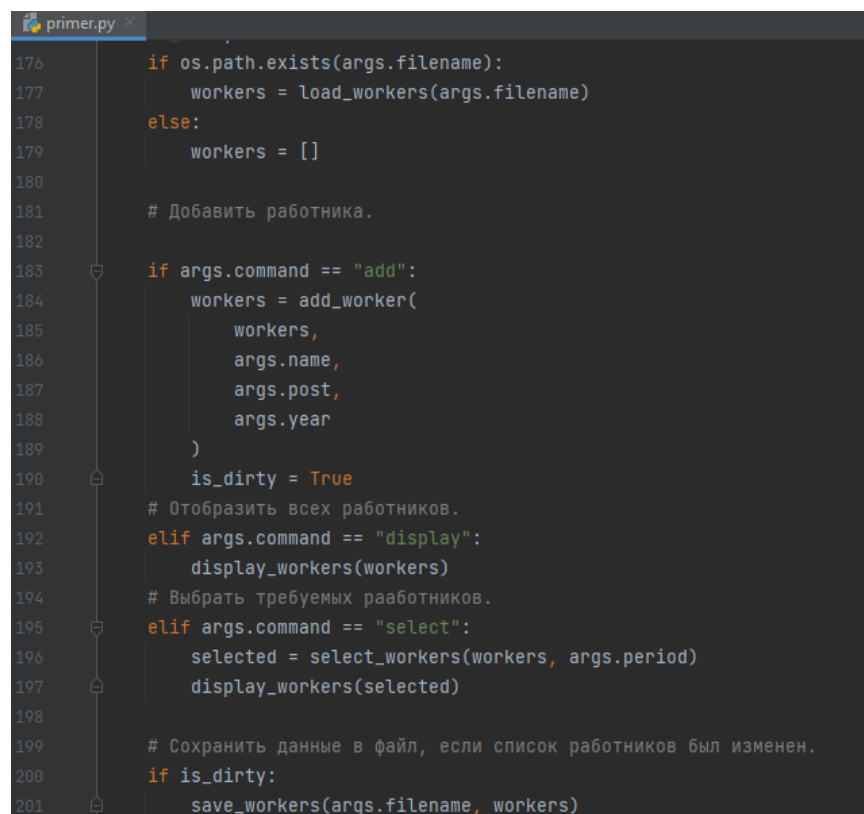
Ход работы:

1. Создал общедоступный репозиторий, клонировал его на локальный сервер.
2. Создал проект и новое виртуальное окружение, затем установил пакет библиотеки jsonschema. Сохранил список зависимостей.

```
PS C:\Users\DNS\PycharmProjects\pythonProject5> conda env export > environment.yml
PS C:\Users\DNS\PycharmProjects\pythonProject5> pip freeze > requirements.txt
```

Рисунок 1 – Сохранение данных виртуального окружения

3. Изучил теоретический материал и приступил к выполнению лабораторной работы.
4. Написал код из примера и выполнил его.



```
176     if os.path.exists(args.filename):
177         workers = load_workers(args.filename)
178     else:
179         workers = []
180
181     # Добавить работника.
182
183     if args.command == "add":
184         workers = add_worker(
185             workers,
186             args.name,
187             args.post,
188             args.year
189         )
190         is_dirty = True
191     # Отобразить всех работников.
192     elif args.command == "display":
193         display_workers(workers)
194     # Выбрать требуемых работников.
195     elif args.command == "select":
196         selected = select_workers(workers, args.period)
197         display_workers(selected)
198
199     # Сохранить данные в файл, если список работников был изменен.
200     if is_dirty:
201         save_workers(args.filename, workers)
```

Рисунок 2 – Код для выполнения примера

```

PS C:\Users\DNS\PycharmProjects\pythonProject5> python primer.py add data.json --name="Bobrov Nikolay" --post="Student" --year=2019
PS C:\Users\DNS\PycharmProjects\pythonProject5> python primer.py add data.json --name="Sergeev Sergey" --post="Developer" --year=2010
PS C:\Users\DNS\PycharmProjects\pythonProject5> python primer.py display data.json
+-----+-----+-----+-----+
| № |          Ф.И.О.          |      Должность      |      Год      |
+-----+-----+-----+-----+
| 1 | Bobrov Nikolay          |      Student        |      2019     |
+-----+-----+-----+-----+
| 2 | Sergeev Sergey          |      Developer       |      2010     |
+-----+-----+-----+-----+
PS C:\Users\DNS\PycharmProjects\pythonProject5> python primer.py select data.json --period=12
Список работников пуст.

```

Рисунок 3 – Результат выполнения программы

5. Приступил к выполнению индивидуального задания.
6. Для своего варианта из лабораторной 2.16 дополнительно реализовал интерфейс командной строки, дополнив код необходимыми параметрами.

```

123 def main(command_line=None):
124     # Создать родительский парсер для определения имени файла.
125     file_parser = argparse.ArgumentParser(add_help=False)
126     file_parser.add_argument(
127         "filename",
128         action="store",
129         help="The data file name"
130     )
131
132     # Создать основной парсер командной строки.
133     parser = argparse.ArgumentParser("students")
134     parser.add_argument(
135         "--version",
136         action="version",
137         version="%s 0.1.0"
138     )
139
140     subparsers = parser.add_subparsers(dest="command")
141
142     # Создать субпарсер для добавления студента.
143     add = subparsers.add_parser(
144         "add",
145         parents=[file_parser],
146         help="Add a new student"
147     )
148     add.add_argument(

```

Рисунок 2 – Код, дополненный необходимыми параметрами

```

PS C:\Users\DNS\PycharmProjects\pythonProject5> python individual.py add students.json --name="Ivanov Ivan" --group=3 --grade="2 2 3 4 2"
Successfully!
PS C:\Users\DNS\PycharmProjects\pythonProject5> python individual.py add students.json --name="Smirnov Andrey" --group=7 --grade="5 4 5 5 5"
Successfully!
PS C:\Users\DNS\PycharmProjects\pythonProject5> python individual.py display students.json
Successfully!
+-----+-----+-----+-----+
| № |          Ф.И.О.          |      Группа      |      Успеваемость      |
+-----+-----+-----+-----+
| 1 | Bobrov Nikolay          |      5           |      5 4 5 5 4 |
| 2 | Ivanov Ivan             |      3           |      2 2 3 4 2 |
| 3 | Smirnov Andrey          |      7           |      5 4 5 5 5 |
+-----+-----+-----+-----+

```

Рисунок 3 – Результат добавление данных и вывода

7. Проверил выполнение команды, которая выводит список студентов со средним баллов 4.0 и выше.

```
PS C:\Users\DNS\PycharmProjects\pythonProject5> python individual.py select students.json --select=1
Successfully!
* Bobrov Nikolay группа № 5
* Smirnov Andrew группа № 7
```

Рисунок 4 – Вывод студентов с баллов выше 4.0

8. Выполнил задание повышенной сложности с использованием библиотеки click.

```
ind2.py
78 @click.command()
79 @click.option("-c", "--command")
80 @click.argument('filename')
81 @click.option("-n", "--name")
82 @click.option("-g", "--group")
83 @click.option("-gr", "--grade")
84 def main(command, filename, name, group, grade):
85     students = load_students(filename)
86     line = '+-{}-+-{}-+-{}-+-{}-+'.format(
87         '-' * 4,
88         '-' * 30,
89         '-' * 20,
90         '-' * 15
91     )
92     if command == 'add':
93         add_student(students, name, group, grade, filename)
94         click.secho('Студент добавлен', fg='green')
95     elif command == 'display':
96         show_list(line, students)
97     elif command == 'select':
98         show_selected(line, students)
99
100
101 if __name__ == '__main__':
102     main()
```

Рисунок 5 – Решение задачи с помощью пакета click

```
PS C:\Users\DNS\PycharmProjects\pythonProject5> python ind2.py -c add students.json -n "Nikolay Bobrov" -g 10 -gr "5 5 5 4 5"
Студент добавлен
PS C:\Users\DNS\PycharmProjects\pythonProject5> python ind2.py -c display students.json
+-----+-----+-----+-----+
| № | Ф.И.О. | Группа | Успеваемость |
+-----+-----+-----+-----+
| 1 | Bobrov Nikolay | 5 | 5 4 5 5 4 |
| 2 | Ivanov Ivan | 3 | 2 2 3 4 2 |
| 3 | Smirnov Andrew | 7 | 5 4 5 5 5 |
| 4 | Nikolay Bobrov | 10 | 5 5 5 4 5 |
+-----+-----+-----+-----+
PS C:\Users\DNS\PycharmProjects\pythonProject5> python ind2.py -c select students.json
+-----+-----+-----+-----+
| № | Ф.И.О. | Группа | Успеваемость |
+-----+-----+-----+-----+
* Bobrov Nikolay группа № 5
* Smirnov Andrew группа № 7
* Nikolay Bobrov группа № 10
+-----+-----+-----+-----+
PS C:\Users\DNS\PycharmProjects\pythonProject5>
```

Рисунок 6 – Результат выполнения программы

Контрольные вопросы:

1. В чем отличие терминала и консоли?

Терминал (от лат. *terminus* — граница) — устройство или ПО, выступающее посредником между человеком и вычислительной системой. Обычно данный термин используется, когда точка доступа к системе вынесена в отдельное физическое устройство и предоставляет свой пользовательский интерфейс на основе внутреннего интерфейса (например, сетевых протоколов).

Консоль *console* — исторически реализация терминала с клавиатурой и текстовым дисплеем. В настоящее время это слово часто используется как синоним сеанса работы или окна оболочки командной строки. В том же смысле иногда применяется и слово “терминал”.

2. Что такое консольное приложение?

Консольное приложение *console application* — вид ПО, разработанный с расчётом на работу внутри оболочки командной строки, т.е. опирающийся на текстовый ввод-вывод.

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

Python 3 поддерживает несколько различных способов обработки аргументов командной строки.

Встроенный способ – использовать модуль *sys*. С точки зрения имен и использования, он имеет прямое отношение к библиотеке C (*libc*). Второй способ – это модуль *getopt*, который обрабатывает как короткие, так и длинные параметры, включая оценку значений параметров.

4. Какие особенности построение CLI с использованием модуля sys?

Это базовый модуль, который с самого начала поставлялся с Python. Он использует подход, очень похожий на библиотеку C, с использованием *argc* и *argv* для доступа к аргументам.

Модуль *sys* реализует аргументы командной строки в простой структуре списка с именем *sys.argv*

5. Какие особенности построение CLI с использованием модуля getopt?

Как вы могли заметить ранее, модуль `sys` разбивает строку командной строки только на отдельные фасы. Модуль `getopt` в Python идет немного дальше и расширяет разделение входной строки проверкой параметров.

Основанный на функции `C getopt`, он позволяет использовать как короткие, так и длинные варианты, включая присвоение значений.

6. Какие особенности построение CLI с использованием модуля argparse

Начиная с версий Python 2.7 и Python 3.2, в набор стандартных библиотек была включена библиотека `argparse` для обработки аргументов (параметров, ключей) командной строки.

Для начала рассмотрим, что интересного предлагает `argparse`:

- анализ аргументов `sys.argv`;
- конвертирование строковых аргументов в объекты вашей программы и работа с ними;
- форматирование и вывод информативных подсказок.

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