

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

Кафедра инфокоммуникаций

Отчет по лабораторной работе №2.17

Разработка приложений с интерфейсом командной строки (CLI) в Python3
по дисциплине «Технологии программирования и алгоритмизация»

Выполнил

студент группы ИВТ-б-о-20-1

Дыбов Д.В. « » _____ 20__ г.

Работа защищена « » _____ 20__ г.

Проверил Воронкин Р.А. _____

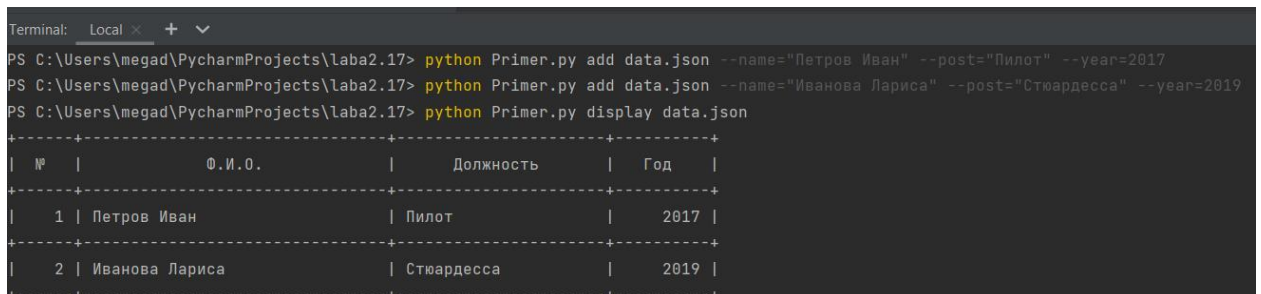
(подпись)

Ставрополь 2021

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

Ход работы

1. Создал новый репозиторий для лабораторной работы №2.17;
2. Клонировал созданный репозиторий на компьютер;
3. Создал новый PyCharm проект в папке репозитория;
4. Проработал пример;
5. Проверил пример на работоспособность:

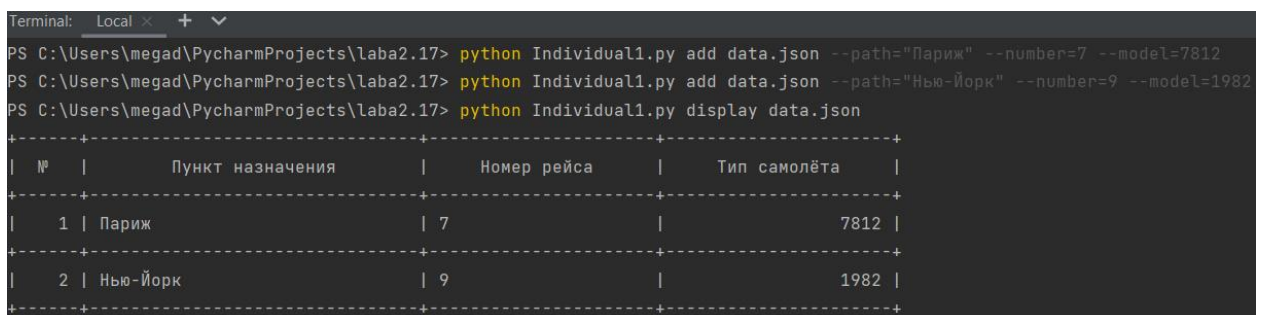


```
Terminal: Local x + v
PS C:\Users\megad\PycharmProjects\laba2.17> python Primer.py add data.json --name="Петров Иван" --post="Пилот" --year=2017
PS C:\Users\megad\PycharmProjects\laba2.17> python Primer.py add data.json --name="Иванова Лариса" --post="Стюардесса" --year=2019
PS C:\Users\megad\PycharmProjects\laba2.17> python Primer.py display data.json
```

№	Ф.И.О.	Должность	Год
1	Петров Иван	Пилот	2017
2	Иванова Лариса	Стюардесса	2019

Рисунок 1 – Результат выполнения примера

6. Выполнил индивидуальное задание;
7. Проверил индивидуальное задание на работоспособность:



```
Terminal: Local x + v
PS C:\Users\megad\PycharmProjects\laba2.17> python Individual1.py add data.json --path="Париж" --number=7 --model=7812
PS C:\Users\megad\PycharmProjects\laba2.17> python Individual1.py add data.json --path="Нью-Йорк" --number=9 --model=1982
PS C:\Users\megad\PycharmProjects\laba2.17> python Individual1.py display data.json
```

№	Пункт назначения	Номер рейса	Тип самолёта
1	Париж	7	7812
2	Нью-Йорк	9	1982

Рисунок 2 – Результат выполнения индивидуального задания

8. Выполнил задание повышенной сложности;
9. Проверил на работоспособность:

```

PS C:\Users\megad\PycharmProjects\laba2.17> python Individual2.py add dataa.json --path="Сидней" --number=5 --model=1938
Рейс добавлен
PS C:\Users\megad\PycharmProjects\laba2.17> python Individual2.py add dataa.json --path="Токио" --number=9 --model=9831
Рейс добавлен
PS C:\Users\megad\PycharmProjects\laba2.17> python Individual2.py display dataa.json
+-----+-----+-----+
| Пункт назначения | Номер рейса | Тип самолёта |
+-----+-----+-----+
| Сидней          | 5           | 1938          |
| Токио           | 9           | 9831          |
+-----+-----+-----+

```

Рисунок 3 – Результат задания повышенной сложности

10. С помощью сайта проверил первый пример на наличие ошибок;
11. Результат не выдал ошибок;

Python code

```

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 def add_worker(staff, name, post, year):
11     staff.append(
12         {
13             "name": name,

```

No syntax errors detected :)

Рисунок 4 – Проверка примера на наличие ошибок

12. С помощью сайта проверил индивидуальное задание на наличие ошибок;
13. Результат не выдал ошибок;

Python code

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import argparse
5 import json
6 import os.path
7
8
9 def add_airplane(race, path, number, model):
10     race.append(
11         {
12             "path": path,
13             "number": number,
```

No syntax errors detected :)

Рисунок 5 – Проверка индивидуального задания на наличие ошибок

14. С помощью сайта проверил задание повышенной сложности на наличие ошибок;

15. Результат не выдал ошибок;

Python code

```
1 #!/usr/bin/env python3
2 # -*- coding: utf-8 -*-
3
4 import json
5 import click
6 import os
7
8
9 @click.group()
10 def cli():
11     pass
12
13
```

No syntax errors detected :)

Рисунок 6 – Проверка задания повышенной сложности на наличие ошибок

16. Отправил все изменения на репозиторий.

Контрольные вопросы

1. В чем отличие терминала и консоли?

Терминал – 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`). Вторым способом – 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.

`argparse` предлагает:

- 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 по построению приложений с интерфейсом командной строки с помощью языка программирования Python версии 3.x.