

**РОССИЙСКОЙ ФЕДЕРАЦИИ**  
**Федеральное государственное автономное**  
**образовательное учреждение высшего образования**  
**«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Кафедра инфокоммуникаций**  
**«Разработка приложений с интерфейсом командной строки (CLI) в**  
**Python3»**

**Отчет по лабораторной работе № 2.17**  
**по дисциплине «Анализ Данных»**

Выполнил студент группы ИВТ-б-о-22-1

Сумин Никита Сергеевич.

« » \_\_\_\_\_ 2024г.

Подпись студента \_\_\_\_\_

Работа защищена « » \_\_\_\_\_ 20\_\_ г.

Проверил Воронкин Р.А. \_\_\_\_\_  
(подпись)

Ставрополь 2024

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

**Порядок выполнения работы:**


1. Создал общедоступный репозиторий на GitHub, в котором использована лицензия MIT и язык программирования Python.

### Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (\*).



Owner \*      Repository name \*

 BrandooDi / DataAn3

✔ DataAn3 is available.

Great repository names are short and memorable. Need inspiration? How about [animated-couscous](#) ?

Description (optional)

- ☒  **Public**  
Anyone on the internet can see this repository. You choose who can commit.
- ☐  **Private**  
You choose who can see and commit to this repository.

Initialize this repository with:

- ☒ **Add a README file**  
This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore

.gitignore template: Python ▾

Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: MIT License ▾

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

This will set  **main** as the default branch. Change the default name in your [settings](#).

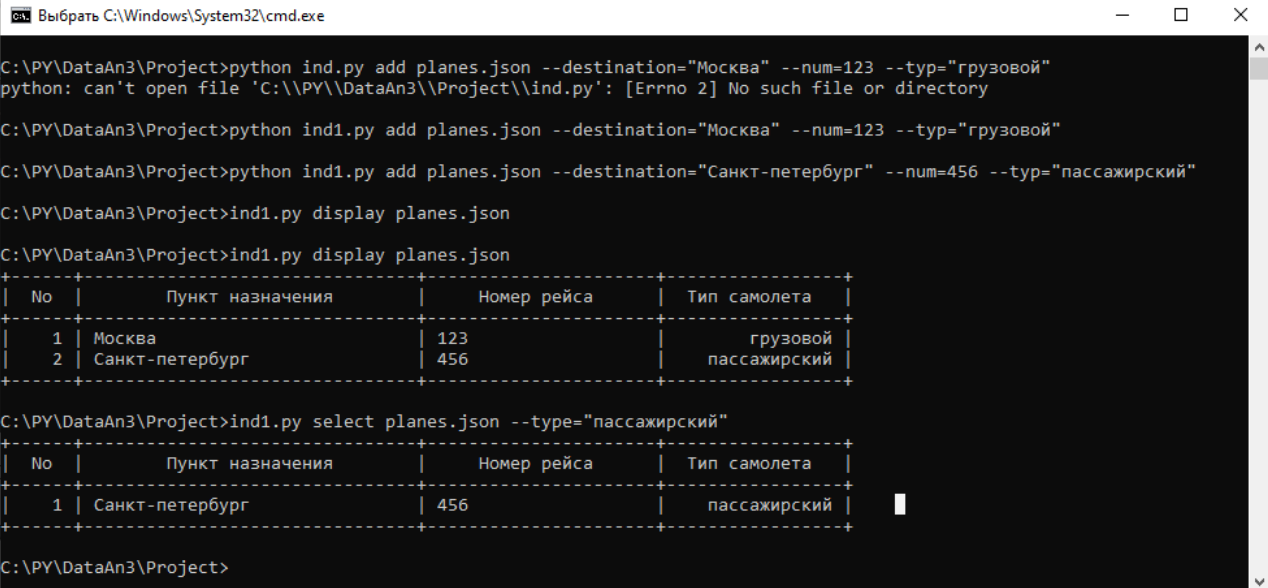
 You are creating a public repository in your personal account.

Create repository

Рисунок 1 - Создание репозитория

## 2. Задание:

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



```
C:\PY\DataAn3\Project>python ind.py add planes.json --destination="Москва" --num=123 --typ="грузовой"
python: can't open file 'C:\PY\DataAn3\Project\ind.py': [Errno 2] No such file or directory

C:\PY\DataAn3\Project>python ind1.py add planes.json --destination="Москва" --num=123 --typ="грузовой"

C:\PY\DataAn3\Project>python ind1.py add planes.json --destination="Санкт-петербург" --num=456 --typ="пассажирский"

C:\PY\DataAn3\Project>ind1.py display planes.json

C:\PY\DataAn3\Project>ind1.py display planes.json
+-----+-----+-----+-----+
| No | Пункт назначения | Номер рейса | Тип самолета |
+-----+-----+-----+-----+
| 1 | Москва | 123 | грузовой |
| 2 | Санкт-петербург | 456 | пассажирский |
+-----+-----+-----+-----+

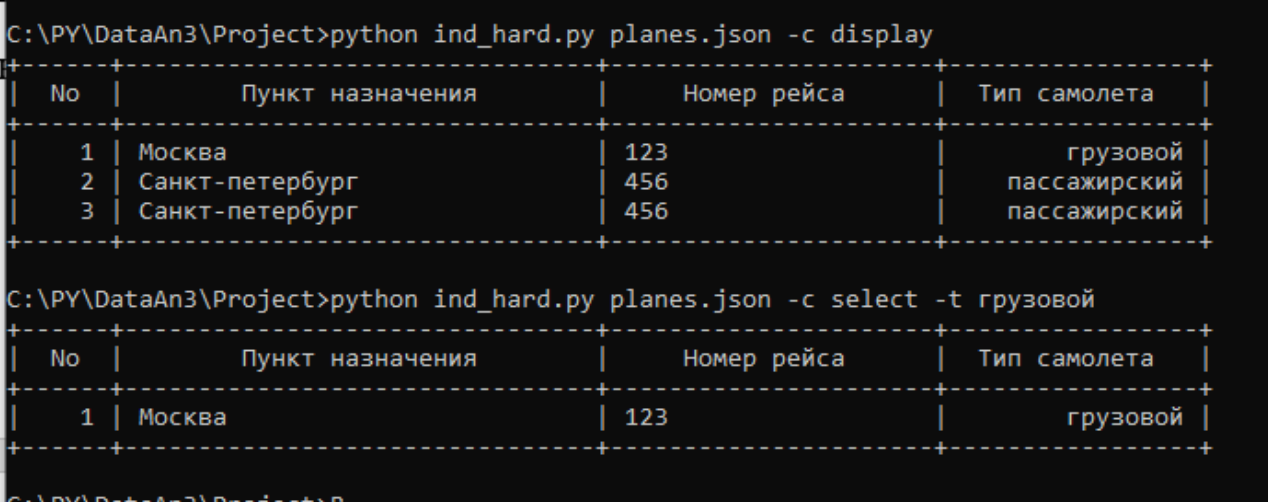
C:\PY\DataAn3\Project>ind1.py select planes.json --type="пассажирский"
+-----+-----+-----+-----+
| No | Пункт назначения | Номер рейса | Тип самолета |
+-----+-----+-----+-----+
| 1 | Санкт-петербург | 456 | пассажирский |
+-----+-----+-----+-----+

C:\PY\DataAn3\Project>
```

Рисунок 2- Результат выполнения индивидуального задания

### Задание повышенной сложности

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 .



```
C:\PY\DataAn3\Project>python ind_hard.py planes.json -c display
+-----+-----+-----+-----+
| No | Пункт назначения | Номер рейса | Тип самолета |
+-----+-----+-----+-----+
| 1 | Москва | 123 | грузовой |
| 2 | Санкт-петербург | 456 | пассажирский |
| 3 | Санкт-петербург | 456 | пассажирский |
+-----+-----+-----+-----+

C:\PY\DataAn3\Project>python ind_hard.py planes.json -c select -t грузовой
+-----+-----+-----+-----+
| No | Пункт назначения | Номер рейса | Тип самолета |
+-----+-----+-----+-----+
| 1 | Москва | 123 | грузовой |
+-----+-----+-----+-----+

C:\PY\DataAn3\Project>
```

Рисунок 3 - Результат выполнения задания повышенной сложности

### 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.

Кроме того, существуют два других общих метода. Это модуль *argparse*, производный от модуля *optparse*, доступного до Python 2.7. Другой метод – использование модуля *docopt*, доступного на GitHub.

#### **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`.

Каждый элемент списка представляет собой единственный аргумент. Первый элемент в списке `sys.argv [0]` – это имя скрипта Python. Остальные элементы списка, от `sys.argv [1]` до `sys.argv [n]`, являются аргументами командной строки с 2 по n. В качестве разделителя между аргументами используется пробел. Значения аргументов, содержащие пробел, должны быть заключены в кавычки, чтобы их правильно проанализировал `sys`.

Эквивалент `argc` – это просто количество элементов в списке. Чтобы получить это значение, используйте оператор `len()`.

#### **5. Какие особенности построение CLI с использованием модуля getopt?**

Основанный на функции C `getopt`, он позволяет использовать как короткие, так и длинные варианты, включая присвоение значений. На практике для правильной обработки входных данных требуется модуль `sys`. Для этого необходимо заранее загрузить как модуль `sys`, так и модуль `getopt`. Затем из списка входных параметров мы удаляем первый элемент списка (см. код ниже) и сохраняем оставшийся список аргументов командной строки в переменной с именем `arguments_list`.

Аргументы в списке аргументов теперь можно анализировать с помощью метода `getopts()`. Но перед этим нам нужно сообщить `getopts()` о том, какие параметры допустимы.

Для метода `getopt()` необходимо настроить три параметра – список фактических аргументов из `argv`, а также допустимые короткие и длинные параметры.

Сам вызов метода хранится в инструкции `try - catch`, чтобы скрыть ошибки во время оценки. Исключение возникает, если обнаруживается аргумент, который не является частью списка, как определено ранее. Скрипт в Python выведет сообщение об ошибке на экран и выйдет с кодом ошибки 2.

Наконец, аргументы с соответствующими значениями сохраняются в двух переменных с именами `arguments` и `values`. Теперь вы можете легко оценить эти переменные в своем коде. Мы можем использовать цикл `for` для перебора списка распознанных аргументов, одна запись за другой.

## **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 работы с `argparse` необходимо задать парсер.

Далее, парсеру стоит указать, какие объекты Вы от него ждете.

Если действие (`action`) для данного аргумента не задано, то по умолчанию он будет сохраняться (`store`) в `namespace`, причем мы также можем указать тип этого аргумента (`int`, `boolean` и тд). Если имя возвращаемого аргумента (`dest`) задано, его значение будет сохранено в соответствующем атрибуте `namespace`.

Остановимся на действиях (`actions`). Они могут быть следующими:

`store`: возвращает в пространство имен значение (после необязательного приведения типа). Как уже говорилось, `store` — действие по умолчанию;

`store_const`: в основном используется для флагов. Либо вернет Вам значение, указанное в `const`, либо (если ничего не указано), `None`.

store\_true / store\_false: аналог store\_const , но для булевых True и False ;

append: возвращает список путем добавления в него значений

агругментов.

append\_const: возвращение значения, определенного в спецификации аргумента, в список.

count: как следует из названия, считает, сколько раз встречается значение данного аргумента.

**Вывод:** были приобретены навыки по работе с данными формата JSON при написании программ с помощью языка программирования Python версии 3.x.