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 ПРАКТИЧЕСКОЙ РАБОТЕ №3
дисциплины «Анализ данных»

Выполнил:
Лейс Алексей Вячеславович
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»

_____
(подпись)

Руководитель практики: кандидат тех. наук доцент кафедры инфокоммуникаций: Воронкин Р. А.

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2024 г.

Тема: Разработка приложений с интерфейсом командной строки (CLI) в Python3

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

Порядок выполнения работы:

Пример из работы:

I: > ИВТ > Анализ данных > 3 > Andate_3 > primer1.py > add_worker

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import argparse
import json
import os.path
from datetime import date


def add_worker(staff, name, post, year):
    """
    Добавить данные о работнике.
    """
    staff.append({"name": name, "post": post, "year": year})
    return staff


def display_workers(staff):
    """
    Отобразить список работников.
    """
    # Проверить, что список работников не пуст.
    if staff:
        # Заголовок таблицы.
        line = "+-{}-+-{}-+-{}-+-{}-+".format("-" * 4, "-" * 30, "-" * 20, "-" * 8)
        print(line)
        print(
            "| {:^4} | {:^30} | {:^20} | {:^8} |".format(
                "No", "Ф.И.О.", "Должность", "Год"
            )
        )
        print(line)

        # Вывести данные о всех сотрудниках.
        for idx, worker in enumerate(staff, 1):
            print(
                "| {:>4} | {:<30} | {:<20} | {:>8} |".format(
                    idx,
                    worker.get("name", ""),
                    worker.get("post", ""),
                    worker.get("year", 0),
                )
            )
        print(line)
```

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

```
I: > ИВТ > Анализ данных > 3 > Andate_3 > individual1.py > ...
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

import argparse
import json
import os.path


def get_poezd(poezd, name, no, time):

    poezd.append({"name": name, "no": no, "time": time})
    return poezd


def list(poezd):
    if poezd:
        line = "+-{}-+-{}-+-{}-+".format(
            "-" * 10,
            "-" * 20,
            "-" * 8,
        )
        print(line)
        print("| {:^10} | {:^20} | {:^8} |".format(" No ", "Название", "Время"))
        print(line)

        for idx, po in enumerate(poezd, 1):
            print(
                "| {:>10} | {:<20} | {"
                "} |".format(po.get("no", ""), po.get("name", ""), po.get("time", ""))
            )
        print(line)

    else:
        print("Список поездов пуст.")


def select_poezd(poezd, nom):
    rezult = []
    for idx, po in enumerate(poezd, 1):
        if po["no"] == str(nom):
            rezult.append(po)

    return rezult
```

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.

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-


import json
import click


@click.group()
def cli():
    pass


@cli.command("add")
@click.argument("filename")
@click.option("-n", "--name")
@click.option("-no", "--nomer", type=int)
@click.option("-t", "--time")
def add(filename, name, nomer, time):
    """
    Добавить данные о поезде
    """
    # Запросить данные о поезде.
    poezd = load_poezd(filename)
    poezd.append(
        {
            "nomer": nomer,
            "name": name,
            "time": time,
        }
    )
    with open(filename, "w", encoding="utf-8") as fout:
        json.dump(poezd, fout, ensure_ascii=False, indent=4)
    click.secho("Поезд добавлен")
```

Ответы на 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. У каждого из этих способов есть свои плюсы и минусы, поэтому стоит
оценить каждый, чтобы увидеть, какой из них лучше всего соответствует вашим потребностям.

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() . Позже мы покажем это на примере кода.

5. Какие особенности построение CLI с использованием модуля getopt ?

Как вы могли заметить ранее, модуль sys разбивает строку командной строки только на отдельные фасеты. 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 ?

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.
Одним из аргументов противников включения argparse в Python был довод о том, что в стандартных модулях и без этого содержится две библиотеки для семантической обработки (парсинга) параметров командной строки. Однако, как заявляют разработчики argparse , библиотеки getopt и optparse уступают argparse по нескольким причинам:
 обладая всей полнотой действий с обычными параметрами командной строки, они не умеют обрабатывать позиционные аргументы (positional arguments). Позиционные аргументы — это аргументы, влияющие на работу программы, в зависимости от порядка, в котором они в эту программу передаются. Простейший пример — программа cp, имеющая минимум 2 таких аргумента («cp source destination»).
 argparse дает на выходе более качественные сообщения о подсказке при минимуме затрат (в этом плане при работе с optparse часто можно наблюдать некоторую избыточность кода);
argparse дает возможность программисту устанавливать для себя, какие символы являются параметрами, а какие нет. В отличие от него, optparse считает опции с синтаксисом наподобие "-pf, -file, +rgb, /f и т.п. «внутренне противоречивыми» и «не поддерживается optpars 'ом и никогда не будет»;
argparse даст Вам возможность использовать несколько значений переменных у одного аргумента командной строки (nargs);
argparse поддерживает субкоманды (subcommands). Это когда основной парсер отсылает к другому (субпарсеру), в зависимости от аргументов на входе.

Вывод: на основе выполненной работы приобрел навыки построения приложений с интерфейсом командной строки с помощью языка программирования Python версии 3.x.