Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт цифрового развития
Кафедра инфокоммуникаций

«Лабораторная работа 2.17. Разработка приложений с интерфейсом командной строки (CLI) в Python3»

ОТЧЕТ
по лабораторной работе №20
дисциплины
«Основы программной инженерии»

Выполнил:
Луценко Дмитрий Андреевич
2 курс, группа ПИЖ-б-о-21-1,
09.03.04 «Программная инженерия»,
направленность (профиль) «Разработка и сопровождение программного обеспечения», очная форма обучения

_____
(подпись)

Проверил:

_____
(подпись)

Отчет защищен с оценкой _____ Дата защиты_____

Ставрополь, 2023 г.

Лабораторная работа 2.17. 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.

Ход работы:

Ниже представлен код индивидуального задания:

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

import sys
import json
import jsonschema
from jsonschema import validate
import argparse
import os.path


schema = {
    "type": "object",
    "properties": {
        "product": {
            "type": "string"
        },
        "shop": {
            "type": "string"
        },
        "cost": {
            "type": "number"
        }
    }
}


def add_product(products, prod, shop, cost):
    """
    Ввод информации о товарах.
    """
    products.append(
        {
            "product": prod,
            "shop": shop,
            "cost": cost
        }
    )
    return products


def save_products(file_name, products):
    """
    Сохранить список всех товаров в формате JSON
    """
    with open(file_name, "w", encoding="utf-8") as fout:
        json.dump(products, fout, ensure_ascii=False, indent=4)
```

```
def load_products(file_name):
    """
    Загрузить список всех товаров из файла JSON
    """
    with open(file_name, "r", encoding="utf-8") as fin:
        return json.load(fin)


def product_list(products):
    """
    Вывод списка товаров
    """
    if products:
        line = '+-{}-+-{}-+-{}-+'.format(
            '-' * 4,
            '-' * 30,
            '-' * 20
        )
        print(line)
        print(
            '| {:^25} | {:^15} | {:^14} |'.format(
                "Товар",
                "Магазин",
                "Стоимость"
            )
        )
        print(line)

        for product in products:
            print(
                '| {:^25} | {:^15} | {:^14} |'.format(
                    product.get('product', ''),
                    product.get('shop', ''),
                    product.get('cost', 0)
                )
            )
            print(line)


def select(products, shop):
    """
    Выбрать товары из конкретного магазина.
    """
    result = []
    for product in products:
        if product.get('shop', '') == shop:
            result.append(product)
    return result


def validation(json_data):
    """
    Валидация данных
    """
    try:
        validate(instance=json_data, schema=schema)
    except:
        raise jsonschema.exceptions.ValidationError("Данные недействительны")

    msg = "Данные успешно загружены"
    return True, msg


def main(command_line=None):
```

```
    """
    Главная функция программы.
    """
    file_parser = argparse.ArgumentParser(add_help=False)
    file_parser.add_argument(
        "filename",
        action="store",
        help="Имя файла"
    )

    parser = argparse.ArgumentParser("products")
    parser.add_argument(
        "--version",
        action="version",
        version="%(prog)s 0.1.0"
    )
    subparsers = parser.add_subparsers(dest="command")

    add = subparsers.add_parser(
        "add",
        parents=[file_parser],
        help="Добавить новый продукт"
    )
    add.add_argument(
        "-n",
        "--name",
        action="store",
        required=True,
        help="Название продуктов"
    )
    add.add_argument(
        "-s",
        "--shop",
        action="store",
        help="Название магазина"
    )
    add.add_argument(
        "-c",
        "--cost",
        action="store",
        type=float,
        required=True,
        help="Стоимость товара"
    )

    list = subparsers.add_parser(
        "list",
        parents=[file_parser],
        help="Отобразить список товаров"
    )

    select = subparsers.add_parser(
        "select",
        parents=[file_parser],
        help="Выбрать товары из магазина"
    )
    select.add_argument(
        "-s",
        "--shop",
        action="store",
        type=str,
        required=True,
        help="Название магазина"
    )
```

```
    args = parser.parse_args(command_line)

    is_dirty = False
    if os.path.exists(args.filename):
        products = load_products(args.filename)
    else:
        products = []

    if args.command == 'add':
        products = add_product(
            products,
            args.name,
            args.shop,
            args.cost
        )
        is_dirty = True

    elif args.command == 'list':
        product_list(products)

    elif args.command == 'select':
        selected = select(products, args.shop)
        product_list(selected)

    if is_dirty:
        save_products(args.filename, products)


if __name__ == '__main__':
    main()
```

```
(laba20) E:\GitHub\laba20>individual.py add test.json --name="салфетки" --shop="fixprice" --cost="49.99"

(laba20) E:\GitHub\laba20>individual.py list test.json
+------+-------------------------------+----------------------+
|          Товар          |     Магазин     |   Стоимость    |
+------+-------------------------------+----------------------+
|          кола           |     магнит      |     100.0      |
+------+-------------------------------+----------------------+
|        макароны         |    пятерочка    |      44.0      |
+------+-------------------------------+----------------------+
|         котлеты         |     магнит      |      249       |
+------+-------------------------------+----------------------+
|        салфетки         |    fixprice     |     49.99      |
+------+-------------------------------+----------------------+
```

Рисунок 1 – Результат работы программs

Вывод: были приобретены навики построения приложений с интерфейсом командной строки с помощью языка программирования Python версии 3.x.

Ответы на 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? Модуль sys, getopt, argparse,

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 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 ? Вообще, argparse — довольно мощная и легкая библиотека,

предоставляющая очень удобный интерфейс для работы с параметрами командной строки.