мІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

нАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «лЬВІВСЬКА ПОЛІТЕХНІКА»

ІНСТИТУТ КОМП’ЮТЕРНИХ НАУК ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

|  |
| --- |
| 2023 |



Звіт

Про виконання лабораторної роботи №7

З предмету «Спеціалізовані мови програмування»

Виконав

ст.гр ІТ-21

Капанайко А.Т

Прийняв:

доц.каф

Щербак С.С

**Мета роботи:** Робота з API та веб-сервісами. Створення консольного об’єктно - орієнтованого додатка з використанням API

**План роботи:**

**Завдання 1: Вибір провайдера API**

Виберіть надійний API, який надає через HTTP необхідні дані для віддаленого зберігання, вивантаження або реалізуйте свій. Для прикладу це може бути jsonplaceholder.org

**Завдання 2: Інтеграція API**

Виберіть бібліотеку для роботи з API та обробки HTTP запитів (для прикладу це може бути бібліотека Requests). Інтегруйте обраний API в ваш консольний додаток на Python. Ознайомтеся з документацією API та налаштуйте необхідний API-ключ чи облікові дані.

**Завдання 3: Введення користувача**

Розробіть користувальницький інтерфейс, який дозволяє користувачам візуалізувати всі доступні дані в табличному вигляді та у вигляді списку. Реалізуйте механізм для збору та перевірки введеного даних користувачем.

**Завдання 4: Розбір введення користувача**

Створіть розбірник для видобування та інтерпретації виразів користувача на основі регулярних виразів, наприклад, для візуалізації дат, телефонів, тощо. Переконайтеся, що розбірник обробляє різні формати введення та надає зворотний зв'язок про помилки.

**Завдання 5: Відображення результатів**

Реалізуйте логіку для візуалізації даних через API в консолі. Обробляйте відповіді API для отримання даних у вигляді таблиць, списків. Заголовки таблиць, списків мають виділяться кольором та шрифтом, які задається користувачем

**Завдання 6: Збереження даних**

Реалізуйте можливості збереження даних у чіткому та читабельному форматі JSON, CSV та TXT

**Завдання 7: Обробка помилок**

Розробіть надійний механізм обробки помилок для керування помилками API, некоректним введенням користувача та іншими можливими проблемами. Надавайте інформативні повідомлення про помилки**.**

**Завдання 8: Ведення історії обчислень**

Включіть функцію, яка реєструє запити користувача, включаючи введені запити та відповідні результати. Дозвольте користувачам переглядати та рецензувати історію своїх запитів.

**Завдання 9: Юніт-тести**

Напишіть юніт-тести для перевірки функціональності вашого додатку. Тестуйте різні операції, граничні випадки та сценарії помилок.

**Код програми:**

import requests

import json

import csv

from tabulate import tabulate

class ConsoleApp:

    def \_\_init\_\_(self, api\_url, api\_key):

        self.api\_url = api\_url

        self.api\_key = api\_key

        self.history = []

    def make\_api\_request(self, endpoint, params=None):

        headers = {"Authorization": f"Bearer {self.api\_key}"}

        response = requests.get(

            f"{self.api\_url}/{endpoint}", params=params, headers=headers

        )

        if response.status\_code == 200:

            return response.json()

        else:

            raise Exception(f"API Request Failed. Status Code: {response.status\_code}")

    def display\_data(self, data, headers\_color="green", headers\_font="bold"):

        if isinstance(data, list):

            self.display\_list\_data(data, headers\_color, headers\_font)

        elif isinstance(data, dict):

            self.display\_table\_data(data, headers\_color, headers\_font)

        else:

            print(f"Unsupported data format: {type(data)}")

    def display\_table\_data(self, data, headers\_color, headers\_font):

        headers = [key for key in data.keys()]

        rows = [list(data.values())]  # Кожне значення у словнику буде окремим рядком

        print(

            tabulate(

                rows,

                headers=headers,

                tablefmt="fancy\_grid",

                numalign=headers\_color,

                stralign=headers\_font,

            )

        )

    def display\_list\_data(self, data, headers\_color, headers\_font):

        print(

            tabulate(

                data,

                tablefmt="fancy\_grid",

                numalign=headers\_color,

                stralign=headers\_font,

            )

        )

    def save\_data(self, data, format):

        if format.lower() == "json":

            self.save\_json(data)

        elif format.lower() == "csv":

            self.save\_csv(data)

        elif format.lower() == "txt":

            self.save\_txt(data)

        else:

            print(f"Unsupported format: {format}")

    def save\_json(self, data):

        with open("output.json", "w") as f:

            json.dump(data, f, indent=2)

        print("Data saved to output.json")

    def save\_csv(self, data):

        with open("output.csv", "w", newline="") as f:

            writer = csv.writer(f)

            writer.writerows(data)

        print("Data saved to output.csv")

    def save\_txt(self, data):

        with open("output.txt", "w") as f:

            f.write(str(data))

        print("Data saved to output.txt")

    def parse\_user\_input(self, user\_input):

        # Простий приклад парсингу команди користувача

        # Припускаємо, що користувач вводить команду у вигляді 'get posts -c green -f json'

        parts = user\_input.split()

        endpoint = parts[1]

        params = {}

        format\_option = None

        headers\_color = "green"

        headers\_font = "bold"

        for i in range(2, len(parts), 2):

            if parts[i] == "-c":

                headers\_color = parts[i + 1]

            elif parts[i] == "-f":

                format\_option = parts[i + 1]

        return {

            "endpoint": endpoint,

            "params": params,

            "format": format\_option,

            "headers\_color": headers\_color,

            "headers\_font": headers\_font,

        }

    def handle\_errors(self, error\_message):

        print(f"Error: {error\_message}")

    def run(self):

        while True:

            user\_input = input("Enter a command: ")

            if user\_input.lower() == "exit":

                break

            try:

                parsed\_input = self.parse\_user\_input(user\_input)

                api\_data = self.make\_api\_request(

                    parsed\_input["endpoint"], parsed\_input["params"]

                )

                self.display\_data(

                    api\_data,

                    parsed\_input["headers\_color"],

                    parsed\_input["headers\_font"],

                )

                self.save\_data(api\_data, parsed\_input["format"])

                self.history.append({"input": user\_input, "output": api\_data})

            except Exception as e:

                self.handle\_errors(str(e))

if \_\_name\_\_ == "\_\_main\_\_":

    api\_url = "https://jsonplaceholder.typicode.com"

    api\_key = None

    app = ConsoleApp(api\_url, api\_key)

    app.run()

#<команда> <ресурс> -c <колір> -f <формат>

# де:

# <команда> - операція, наприклад, get, create, update, delete.

# <ресурс> - ресурс API, наприклад, posts, users, comments.

# -c <колір> - параметр для вибору кольору виведення (необов'язковий).

# -f <формат> - параметр для вибору формату збереження результатів (необов'язковий). json csv txt

**Посилання на GitHub репозиторій:** [**https://github.com/Senichkaa/lab1-on-python**](https://github.com/Senichkaa/lab1-on-python)

**Висновок:** Виконавши ці завдання, ви створите проект, який надасть вам цінний досвід роботи з API, дизайну користувацького інтерфейсу, валідації введення, обробки помилок та тестування. надійність і точність