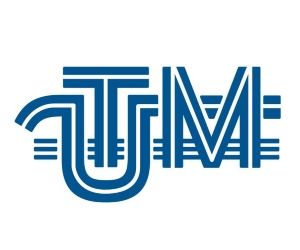
Ministerul Educației și Cercetării al Republicii Moldova

Universitatea Tehnică a Moldovei

Facultatea Calculatoare, Informatică și Microelectronică

Departamentul Ingineria Software și Automatică

**Raport**

la disciplina **”Programarea Declarativa”**

Tema:[**Colectarea automată a datelorAssignment**](https://else.fcim.utm.md/mod/assign/view.php?id=45186)

Efectuat de: studentul/studenta gr.**TI-216 Vlasitchi Stefan**

Verificat de: asis.univ **Viorel Rusu**

Chișinău-2023

**Exercitiul 1.**

Alegeți un subiect ce vă interesează pe wikipedia.org și îndepliniți următoarele sarcini:

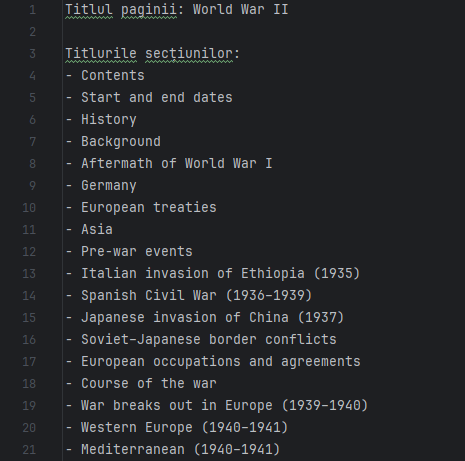
• capturați titlul paginii;

• capturați toate titlurile secțiunilor;

• obțineți minim o imagine de pe acel site

from urllib.parse import urljoin  
import requests  
from bs4 import BeautifulSoup  
  
# URL-ul paginii Wikipedia  
url = "https://en.wikipedia.org/wiki/World\_War\_II"  
  
# Faceți o cerere GET la URL  
response = requests.get(url)  
  
# Analizați răspunsul cu BeautifulSoup  
soup = BeautifulSoup(response.text, 'html.parser')  
  
# Capturați titlul paginii  
page\_title = soup.find('h1').text  
  
# Capturați toate titlurile secțiunilor  
section\_titles = [header.text for header in soup.find\_all(['h2', 'h3', 'h4', 'h5', 'h6'])]  
  
# Găsiți toate elementele de imagine pe pagină  
img\_tags = soup.find\_all('img')  
  
# Creați o listă pentru a stoca URL-urile imaginilor  
image\_urls = []  
  
# Parcurgeți toate elementele de imagine și adăugați URL-urile lor la listă  
for img in img\_tags:  
 image\_url = img.get('src')  
 if image\_url:  
 # Construiți un URL absolut folosind urljoin  
 absolute\_image\_url = urljoin(url, image\_url)  
 image\_urls.append(absolute\_image\_url)  
  
# Salvare într-un fișier text  
with open("informatii\_wikipedia.txt", "w", encoding="utf-8") as file:  
 file.write(f"Titlul paginii: {page\_title}\n\n")  
 file.write("Titlurile secțiunilor:\n")  
 for section\_title in section\_titles:  
 file.write(f"- {section\_title}\n")  
 file.write("\nURL-urile imaginilor:\n")  
 for image\_url in image\_urls:  
 file.write(f"- {image\_url}\n")  
  
print("Informațiile au fost salvate în fișierul 'informatii\_wikipedia.txt'.")

Figura 1. Codul exercitiului 1

****

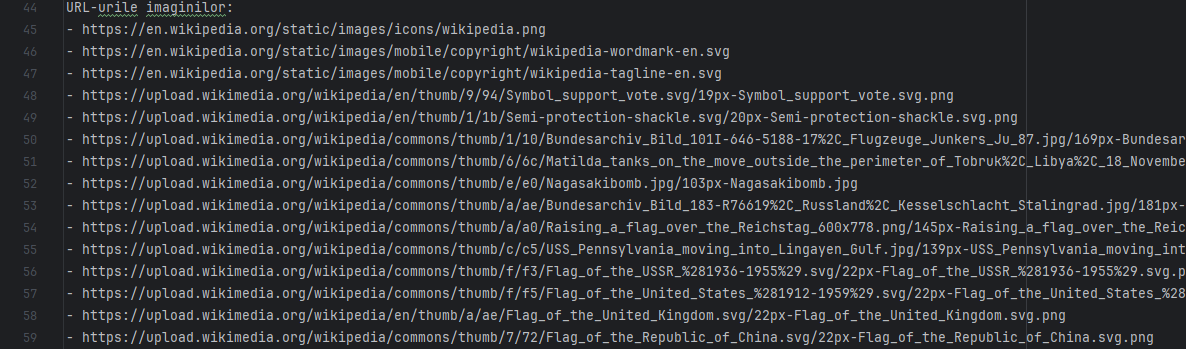
 Figura 2.1. Rezultatul exercitiul 2

Figura 2.2. Rezultatul exercitiul 2

**Exercitiul 2.**

Accesați site-ul web: http://books.toscrape.com/index.html care este conceput special pentru testarea web scraping. Obțineți titlul fiecărei cărți care are o evaluare de 2 stele și, la sfârșit, să aveți doar o listă Python cu toate titlurile lor.

* găsiți structura URL-ului pentru a parcurge fiecare pagină ;
* parsați fiecare pagină din catalog;
* găsiți ce etichetă/clasă reprezintă evaluarea cu stele ;
* filtrați cu if evaluarea cu stele;
* stocați rezultatele într-o listă.

import requests  
from bs4 import BeautifulSoup  
import json  
  
  
def retrieve\_books\_with\_two\_stars():  
 base\_url = "http://books.toscrape.com/catalogue/page-{}.html"  
 two\_star\_titles = []  
 for n in range(1, 51): # There are 50 pages in the catalogue  
 scrape\_url = base\_url.format(n)  
 res = requests.get(scrape\_url)  
  
 soup = BeautifulSoup(res.text, 'html.parser')  
 books = soup.select(".product\_pod")  
  
 for book in books:  
 if len(book.select('.star-rating.Two')) != 0: # If the book has a 2-star rating  
 two\_star\_titles.append(book.select('a')[1]['title']) # Add the title to our list  
 return two\_star\_titles  
  
  
# Deschideți un fișier .txt în modul de scriere  
with open('Final.txt', 'w', encoding='utf-8') as file:  
 file.write(f"Rezultatul final este: ")  
 # Scrieți titlul paginii  
 file.write(json.dumps(retrieve\_books\_with\_two\_stars(), indent=4, ensure\_ascii=False))

Figura 3. Codul exercitiului 2

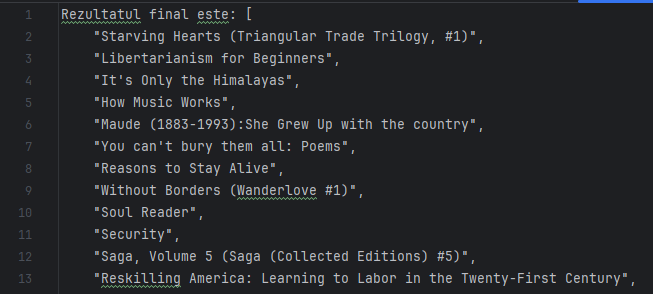


Figura 4. Rezultatul exercitiului 2

**Exercitiul 3.**

Faceți cereri către minim 3 site-uri pentru a obține informația dorită (la alegere: date meteo, curs valutar, produse la reducere, rating etc.). Salvați rezultatele în fișier .csv. Notă: asigurați-vă că site-urile permit web scraping.

import requests  
import csv  
  
# Replace 'YOUR\_API\_KEYS' with the actual API keys obtained from the respective services  
openweathermap\_api\_key = '87f7b694f19833470445a51c738250e2'  
weatherbit\_api\_key = 'a9831d17cd3a45a5b72f5c33780b0447'  
weatherstack\_api\_key = 'fa701df7e1ff52c8257577b9ab7a5b66'  
  
# Define cities for which you want weather data  
cities = ['London']  
  
# Set up OpenWeatherMap API request  
openweathermap\_base\_url = 'http://api.openweathermap.org/data/2.5/weather'  
openweathermap\_data = []  
  
# Set up Weatherbit API request  
weatherbit\_base\_url = 'https://api.weatherbit.io/v2.0/current'  
weatherbit\_data = []  
  
# Set up Weatherstack API request  
weatherstack\_base\_url = 'http://api.weatherstack.com/current'  
weatherstack\_data = []  
  
# Fetch weather data for each city from OpenWeatherMap  
for city in cities:  
 openweathermap\_params = {'q': city, 'appid': openweathermap\_api\_key}  
 openweathermap\_response = requests.get(openweathermap\_base\_url, params=openweathermap\_params)  
  
 if openweathermap\_response.status\_code == 200:  
 data = openweathermap\_response.json()  
 openweathermap\_data.append({  
 'City': city,  
 'Temperature (Celsius)': data['main']['temp'] - 273.15,  
 'Weather Description': data['weather'][0]['description']  
 })  
 else:  
 print(f"Failed to fetch OpenWeatherMap data for {city}. HTTP status code: {openweathermap\_response.status\_code}")  
  
# Fetch weather data for each city from Weatherbit  
for city in cities:  
 weatherbit\_params = {'city': city, 'key': weatherbit\_api\_key}  
 weatherbit\_response = requests.get(weatherbit\_base\_url, params=weatherbit\_params)  
  
 if weatherbit\_response.status\_code == 200:  
 data = weatherbit\_response.json()  
 current\_weather = data['data'][0]  
 weatherbit\_data.append({  
 'City': city,  
 'Temperature (Celsius)': current\_weather['temp'],  
 'Weather Description': current\_weather['weather']['description']  
 })  
 else:  
 print(f"Failed to fetch Weatherbit data for {city}. HTTP status code: {weatherbit\_response.status\_code}")  
  
# Fetch weather data for each city from Weatherstack  
for city in cities:  
 weatherstack\_params = {'access\_key': weatherstack\_api\_key, 'query': city}  
 weatherstack\_response = requests.get(weatherstack\_base\_url, params=weatherstack\_params)  
  
 if weatherstack\_response.status\_code == 200:  
 data = weatherstack\_response.json()  
 current\_weather = data['current']  
 weatherstack\_data.append({  
 'City': city,  
 'Temperature (Celsius)': current\_weather['temperature'],  
 'Weather Description': current\_weather['weather\_descriptions'][0]  
 })  
 else:  
 print(f"Failed to fetch Weatherstack data for {city}. HTTP status code: {weatherstack\_response.status\_code}")  
  
# Write all data to a common CSV file  
with open('weather\_data\_combined.csv', 'w', newline='') as csvfile:  
 fieldnames = ['City', 'Temperature (Celsius)', 'Weather Description']  
 writer = csv.DictWriter(csvfile, fieldnames=fieldnames)  
 writer.writeheader()  
  
 # Write data for all APIs  
 for data in openweathermap\_data + weatherbit\_data + weatherstack\_data:  
 writer.writerow(data)  
  
print("Weather data has been saved in weather\_data\_combined.csv.")

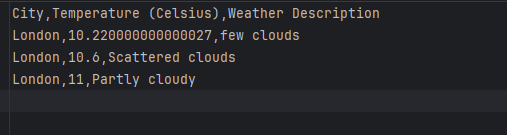


Figura 5. Rezultatul exercitiului 3

**Concluzie**

În această lucrare de laborator, am dobândit competențe în realizarea cererilor API în Python pentru obținerea diverselor informații, cum ar fi condițiile meteorologice. Am dezvoltat abilități în manipularea datelor JSON și CSV, implementând soluții eficiente pentru extragerea, combinarea și stocarea datelor într-un format util.