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**

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

```
Titlul paginii: World War II

Titlurile sectiunilor:
- Contents
- Start and end dates
- History
- Background
- Aftermath of World War I
- Germany
- European treaties
- Asia
- Pre-war events
- Italian invasion of Ethiopia (1935)
- Spanish Civil War (1936-1939)
- Japanese invasion of China (1937)
- Soviet-Japanese border conflicts
- European occupations and agreements
```

Figura 2.1. Rezultatul exercitiul 2

```
URL-urile imaginilor:
- https://en.wikipedia.org/static/images/icons/wikipedia.png
- https://en.wikipedia.org/static/images/mobile/copyright/wikipedia-wordmark-en.svg
- https://en.wikipedia.org/static/images/mobile/copyright/wikipedia-tagline-en.svg
- https://upload.wikimedia.org/wikipedia/en/thumb/9/94/Symbol_support_vote.svg/19px-Symbol_support_vote.svg.png
- https://upload.wikimedia.org/wikipedia/en/thumb/1/1b/Semi-protection-shackle.svg/20px-Semi-protection-shackle.svg.png
- https://upload.wikimedia.org/wikipedia/commons/thumb/1/10/Bundesarchiv_Bild_101I-646-5188-17%2C_Flugzeuge_Junkers_Ju_87.jpg/169px-Bundesar
- https://upload.wikimedia.org/wikipedia/commons/thumb/6/6c/Matilda_tanks_on_the_move_outside_the_perimeter_of_Tobruk%2C_Libya%2C_18_Novembe
- https://upload.wikimedia.org/wikipedia/commons/thumb/e/e0/Nagasakibomb.jpg/103px-Nagasakibomb.jpg
- https://upload.wikimedia.org/wikipedia/commons/thumb/a/ae/Bundesarchiv_Bild_183-R76619%2C_Russland%2C_Kesselschlacht_Stalingrad.jpg/181px-
- https://upload.wikimedia.org/wikipedia/commons/thumb/a/a0/Raising_a_flag_over_the_Reichstag_600x778.png/145px-Raising_a_flag_over_the_Reic
- https://upload.wikimedia.org/wikipedia/commons/thumb/c/c5/USS_Pennsylvania_moving_into_Lingayen_Gulf.jpg/139px-USS_Pennsylvania_moving_int
- https://upload.wikimedia.org/wikipedia/commons/thumb/f/f3/Flag_of_the_USSR_%281936-1955%29.svg/22px-Flag_of_the_USSR_%281936-1955%29.svg.p
- https://upload.wikimedia.org/wikipedia/commons/thumb/f/f5/Flag_of_the_United_States_%281912-1959%29.svg/22px-Flag_of_the_United_States_%28
- https://upload.wikimedia.org/wikipedia/en/thumb/a/ae/Flag_of_the_United_Kingdom.svg/22px-Flag_of_the_United_Kingdom.svg.png
- https://upload.wikimedia.org/wikipedia/commons/thumb/7/72/Flag_of_the_Republic_of_China.svg/22px-Flag_of_the_Republic_of_China.svg.png
```

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

```
Rezultatul final este: [
    "Starving Hearts (Triangular Trade Trilogy, #1)",
    "Libertarianism for Beginners",
    "It's Only the Himalayas",
    "How Music Works",
    "Maude (1883-1993):She Grew Up with the country",
    "You can't bury them all: Poems",
    "Reasons to Stay Alive",
    "Without Borders (Wanderlove #1)",
    "Soul Reader",
    "Security",
    "Saga, Volume 5 (Saga (Collected Editions) #5)",
    "Reskilling America: Learning to Labor in the Twenty-First Century",
```

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.openweather-
map.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': openweather-
map_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']['descrip-
tion']
        })
    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_descrip-
tions'][0]
        })
    else:
        print(f"Failed to fetch Weatherstack data for {city}. HTTP sta-
tus 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 Descrip-
tion']
    writer = csv.DictWriter(csvfile, fieldnames=fieldnames)
    writer.writeheader()

    # Write data for all APIs
    for data in openweathermap_data + weatherbit_data + weath-
erstack_data:
        writer.writerow(data)

print("Weather data has been saved in weather_data_combined.csv.")
```

```
City,Temperature (Celsius),Weather Description
London,10.220000000000027,few clouds
London,10.6,Scattered clouds
London,11,Partly cloudy
```

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.