**Ministerul Educației și Cercetării al Republicii Moldova**

**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare, Informatică şi Microelectronică**

**Departamentul Informatică şi Ingineria Sistemelor**

**RAPORT**

Lucrare de laborator nr.5

la cursul „Programarea Declarativă”

**Tema 5:** „Colectarea automată a datelor.”

A efectuat : **st. gr. TI-214 Buza Cătălin**

A verificat:  **M.Rusu**

**Chișinău 2023**

**Exerciții:**

**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.

Codul are scopul de a accesa o pagină Wikipedia despre Microsoft, extrage informații și descărca imagini de pe pagină. Iată o explicație simplificată a fiecărei părți a codului:

1. Importarea bibliotecilor:

- `os`: Folosit pentru operații de sistem, cum ar fi manipularea directoarelor.

- `requests`: Utilizat pentru a face cereri HTTP către pagina Wikipedia.

- `BeautifulSoup`: Folosit pentru a analiza conținutul HTML al paginii.

- `urljoin, urlparse`: Folosite pentru a manipula și forma URL-uri.

- `random`: Folosit pentru a selecta imagini aleatoare din lista de imagini de pe pagină.

- `shutil`: Folosit pentru a efectua operații pe sistem de operare, cum ar fi ștergerea și crearea de directoare.

2. Funcția `download\_image`:

- Descarcă o imagine de la un URL dat și o salvează într-un director specific.

- Returnează calea către imaginea descărcată sau `None` în caz de eroare.

3. Funcția `capture\_wikipedia\_data`:

- Accesează pagina Wikipedia specificată și extrage informații relevante.

- Captură titlul paginii, titlurile secțiunilor și descarcă un număr specific de imagini.

- Creează un director numit "Images" și salvează imaginile descărcate în acest director.

- Returnează titlul paginii și titlurile secțiunilor.

4. Secțiunea `\_\_main\_\_`:

- Definește URL-ul paginii Wikipedia despre Microsoft.

- Apelul funcției `capture\_wikipedia\_data` pentru a obține informațiile dorite și a descărca imagini.

- Afișează titlul paginii și titlurile secțiunilor.

import os

import requests

from bs4 import BeautifulSoup

from urllib.parse import urljoin, urlparse

import random

import shutil

def download\_image(url, folder):

    response = requests.get(url, stream=True)

    if response.status\_code == 200:

        # Extrage numele imaginii din URL

        image\_name = os.path.join(folder, os.path.basename(urlparse(url).path))

        with open(image\_name, 'wb') as file:

            for chunk in response.iter\_content(chunk\_size=128):

                file.write(chunk)

        return image\_name

    else:

        return None

def capture\_wikipedia\_data(url, max\_images=5):

    response = requests.get(url)

    if response.status\_code == 200:

        soup = BeautifulSoup(response.text, 'html.parser')

        # Capturare titlu paginii

        page\_title = soup.title.string.strip()

        # Capturare titluri secțiuni

        section\_titles = [section.get\_text().strip() for section in soup.find\_all('span', {'class': 'mw-headline'})]

        # Găsește toate imaginile de pe pagină

        all\_images = soup.find\_all('img')

        # Alege max\_images imagini aleatoare

        selected\_images = random.sample(all\_images, min(len(all\_images), max\_images))

        folder = 'Images'

        # Curățare director existent

        if os.path.exists(folder):

            shutil.rmtree(folder)

        # Recreare director

        os.makedirs(folder, exist\_ok=True)

        for i, image in enumerate(all\_images):

            image\_url = urljoin(url, image['src'])

            image\_path = download\_image(image\_url, folder)

            if image\_path:

                print(f'Imagine {i+1} descărcată la: {image\_path}')

        return page\_title, section\_titles

    else:

        print(f'Eroare la accesarea paginii: {response.status\_code}')

if \_\_name\_\_ == '\_\_main\_\_':

    wikipedia\_url = 'https://ro.wikipedia.org/wiki/Microsoft'

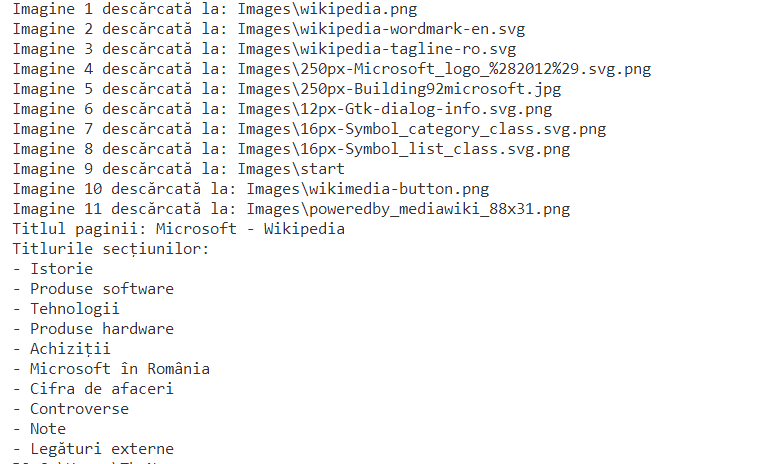
    page\_title, section\_titles = capture\_wikipedia\_data(wikipedia\_url, max\_images=5)

    print(f'Titlul paginii: {page\_title}')

    print('Titlurile secțiunilor:')

    for section\_title in section\_titles:

        print(f'- {section\_title}')



1. 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.

Codul folosește bibliotecile `requests` și `BeautifulSoup` pentru a extrage titlurile cărților cu o evaluare de 2 stele de pe un site web.

1. Importarea bibliotecilor:

import requests

from bs4 import BeautifulSoup

Aceste două biblioteci sunt utilizate pentru a face cereri HTTP către site și pentru a analiza conținutul HTML al paginilor web.

2. Definirea funcției `get\_titles\_with\_two\_stars`:

def get\_titles\_with\_two\_stars(url):

Această funcție primește un URL și returnează o listă de titluri ale cărților cu o evaluare de 2 stele de pe pagina respectivă și de pe paginile următoare.

3. Inițializarea listei de titluri:

titles = []

Aceasta este lista în care vor fi stocate titlurile cărților cu evaluarea de 2 stele.

4. Buclele de extragere a datelor din pagini multiple:

while True:

Această buclă rulează până când nu mai există pagini următoare.

5. Trimite o cerere către URL:

response = requests.get(url)

Se face o cerere HTTP la URL-ul specificat pentru a obține conținutul paginii.

6. Utilizarea BeautifulSoup pentru analizarea HTML-ului:

soup = BeautifulSoup(response.text, 'html.parser')

BeautifulSoup este utilizată pentru a analiza conținutul HTML al paginii și a-l face ușor de parcurs și căutat.

7. Identificarea elementelor relevante:

books = soup.find\_all('article', class\_='product\_pod')

Se identifică toate elementele de tip `article` cu clasa `product\_pod`, care conțin informații despre cărți.

8. Parcurgerea cărților și verificarea rating-ului:

for book in books:

rating = book.find('p', class\_='star-rating')['class'][1]

if rating.lower() == 'two':

title = book.h3.a['title']

titles.append(title)

Se parcurg cărțile, se extrage rating-ul și, dacă este 2 stele, se extrage titlul și se adaugă în lista `titles`.

9. Verificarea existenței unei pagini următoare:

next\_page = soup.find('li', class\_='next')

if next\_page:

url = url.rsplit('/', 1)[0] + '/' + next\_page.a['href']

else:

break

Se verifică dacă există o pagină următoare și, în caz afirmativ, se actualizează URL-ul pentru a accesa pagina următoare.

10. Returnarea listei de titluri:

return titles

11. Definirea URL-ului de start și apelul funcției:

start\_url = 'http://books.toscrape.com/catalogue/page-1.html'

result = get\_titles\_with\_two\_stars(start\_url)

12. Afișarea rezultatului:

print(result)

Afișează lista de titluri a cărților cu evaluarea de 2 stele.

import requests

from bs4 import BeautifulSoup

# Function to get the titles of books with a 2-star rating

def get\_titles\_with\_two\_stars(url):

    titles = []

    while True:

        # Send a request to the URL

        response = requests.get(url)

        soup = BeautifulSoup(response.text, 'html.parser')

        # Find the relevant elements on the page

        books = soup.find\_all('article', class\_='product\_pod')

        # Iterate through each book and check its rating

        for book in books:

            rating = book.find('p', class\_='star-rating')['class'][1]

            if rating.lower() == 'two':

                title = book.h3.a['title']

                titles.append(title)

        # Check if there's a next page

        next\_page = soup.find('li', class\_='next')

        if next\_page:

            url = url.rsplit('/', 1)[0] + '/' + next\_page.a['href']

        else:

            break

    return titles

# URL of the starting page

start\_url = 'http://books.toscrape.com/catalogue/page-1.html'

# Get the titles with two stars

result = get\_titles\_with\_two\_stars(start\_url)

# Print the result

print(result)



1. Faceți cereri către minim 3 site-uri pentru a obține informația dorită (la alegere: date meteo, curs valutar, preț produse, rating hotele, restaurante etc.). Ca rezultat trebuie să aveți 3 fișiere .csv cu date similare, de ex. date meteo pe luna octombrie de la 3 site-uri de prognoză meteo, cursul valutar pentru o lună de la 3 bănci diferite, ratingul la o listă de hotele de la 3 site-uri diferite etc.

Notă: asigurați-vă că site-urile permit web scraping.

1. Importarea bibliotecilor:

- `requests`: Folosit pentru a face cereri HTTP către pagini web.

- `BeautifulSoup`: Utilizat pentru a extrage informații din conținutul HTML al paginilor web.

- `csv`: Folosit pentru a scrie date în fișiere CSV.

2. Listă de URL-uri pentru bănci:

- Se definește o listă (`urls`) care conține URL-urile a trei bănci din Moldova, de la care dorim să extragem cursul valutar.

3. Funcția `extract\_currency\_exchange\_rate`:

- Această funcție primește un URL și utilizează biblioteca `BeautifulSoup` pentru a analiza conținutul HTML al paginii respective.

- Încearcă să găsească un tabel care conține informații despre cursul valutar și extrage datele necesare.

4. Extragerea informațiilor despre cursul valutar:

- Se caută în pagina HTML a băncii informații despre cursul valutar, cum ar fi numele valutei și cursul de schimb.

- Aceste informații sunt colectate și salvate sub forma unor dicționare într-o listă numită `exchange\_rates`.

5. Salvarea datelor în fișiere CSV separate:

- Se parcurge lista de URL-uri ale băncilor și pentru fiecare bancă, se apelează funcția `extract\_currency\_exchange\_rate` pentru a obține datele.

- Datele sunt apoi scrise într-un fișier CSV separat pentru fiecare bancă, unde fiecare rând reprezintă o valută și cursul său de schimb.

- Fișierele CSV sunt denumite corespunzător pentru a indica de la care bancă provin.

6. Afisarea informațiilor:

- Pe măsură ce datele sunt salvate în fișiere CSV, se afișează un mesaj care indică în ce fișier s-au salvat datele pentru fiecare bancă.

import time

import pandas as pd

from bs4 import BeautifulSoup

import requests

def get\_exchange\_rates(url):

    main\_df = pd.DataFrame(columns=['USD', 'EUR', 'RON'])

    for day in range(1, 32):

        temp\_url = url.format(day)

        response = requests.get(temp\_url)

        if response.status\_code == 200:

            soup = BeautifulSoup(response.text, 'html.parser')

            tbody = soup.find('tbody')

            if tbody:

                exchange\_rates = {}

                rows = tbody.find\_all('tr')

                for row in rows:

                    columns = row.find\_all('td')

                    currency = columns[0].text.strip()

                    exchange\_rates[currency] = columns[3]

                df = pd.DataFrame.from\_dict(exchange\_rates, orient='index', columns=[f'2023-10-{day}'])

                df = df.T

                df = df[['USD', 'EUR', 'RON']]

                main\_df = pd.concat([main\_df,df])

            else:

                print("Table body not found on the page.")

        else:

            print(f"Failed to retrieve the page. Status code: {response.status\_code}")

    main\_df[['USD', 'EUR', 'RON']] = main\_df[['USD', 'EUR', 'RON']].replace(',', '.', regex=True).astype(float).round(2)

    return main\_df

url\_micb = "https://www.curs.md/ro/office/micb/2023-11-{}/csv"

url\_bcr = 'https://www.curs.md/ro/office/bcr/2023-11-{}/csv'

url\_eximbank = 'https://www.curs.md/ro/office/eximbank/2023-11-{}/csv'

micb\_rates\_df = get\_exchange\_rates(url\_micb)

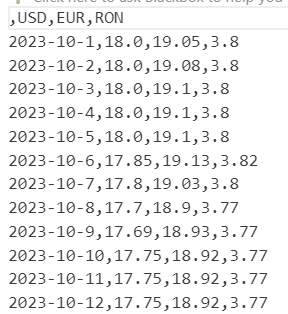
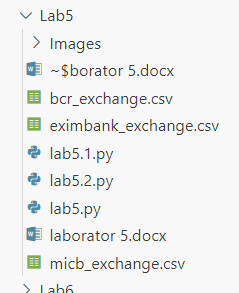
bcr\_rates\_df = get\_exchange\_rates(url\_bcr)

eximbank\_rates\_df = get\_exchange\_rates(url\_eximbank)

micb\_rates\_df.to\_csv('./micb\_exchange.csv')

bcr\_rates\_df.to\_csv('./bcr\_exchange.csv')

eximbank\_rates\_df.to\_csv('./eximbank\_exchange.csv')



**Concluzie**:

În cadrul acestui laborator, am abordat mai multe sarcini legate de manipularea datelor de pe pagini web folosind Python și diverse biblioteci, cum ar fi `requests`, `BeautifulSoup`, și `csv`.

Am început prin a dezvolta o funcție pentru extragerea titlurilor cărților cu evaluare de 2 stele de pe un site de cărți, utilizând tehnici de web scraping. Acest proces a implicat navigarea prin pagini multiple și identificarea elementelor relevante.

Ulterior, am implementat un script care să descarce informații despre cursul valutar de la trei bănci din Moldova și să le salveze în fișiere CSV separate. Această sarcină a necesitat gestionarea cererilor HTTP, analiza conținutului HTML și scrierea datelor în fișiere CSV.

Am realizat un script pentru descărcarea de informații și imagini de pe o pagină Wikipedia despre Microsoft. Acesta a implicat descărcarea imaginilor, extragerea datelor relevante din conținutul HTML al paginii, și organizarea acestora într-un format utilizabil.

În ansamblu, aceste sarcini au permis să aplic concepte esențiale de web scraping, manipulare a datelor și gestionare a fișierelor în mediul Python. Am dezvoltat abilități practice în accesarea și prelucrarea datelor de pe pagini web, extinzând astfel cunoștințele noastre în domeniul programării.