Universitatea Tehnică a Moldovei

Facultatea Calculatoare Informatică şi Microelectronică

Departamentul Ingineria Software și Automatică

**RAPORT**

Lucrarea de laborator nr. 5

# La disciplina „Programarea Declarativa”

Tema: “”

A efectuat: st. gr. SI-211 S. Chirita

A verificat: V. Rusu

Chișinău – 2023

Lucrarea de laborator nr. 5

#### Sarcina de lucru

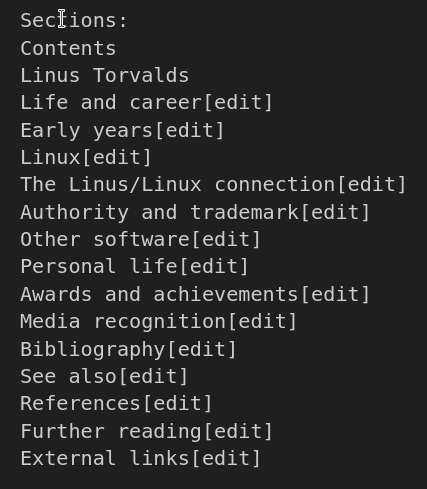
* Alegeți un subiect ce vă interesează pe *wikipedia.org* și îndepliniți următoarele sarcini:
* capturați titlul paginii;

|  |
| --- |
| #Sarcina 1  import requests  from bs4 import BeautifulSoup  def wiki\_title(url):  res = requests.get(url)  soup = BeautifulSoup(res.text, "html.parser")  title = soup.find('title').text  print(f"Title:\n{title}\n")  wiki\_title("https://en.wikipedia.org/wiki/Linus\_Torvalds") |



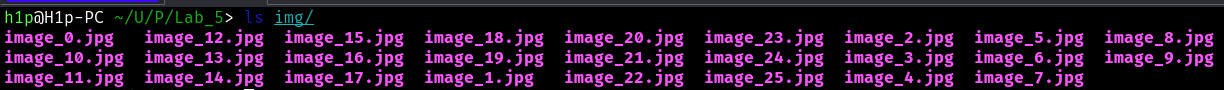
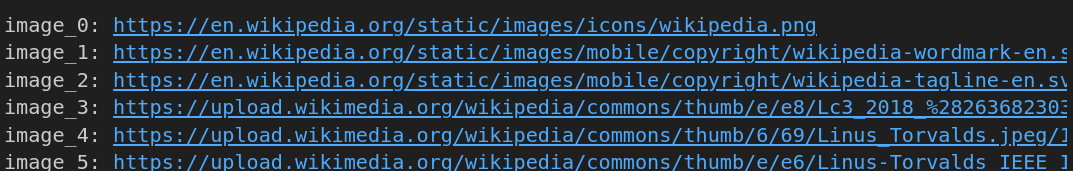
* capturați toate titlurile secțiunilor;

|  |
| --- |
| #Sarcina 2  import requests  from bs4 import BeautifulSoup  def wiki\_sections(url):  res = requests.get(url)  if res.status\_code != 200:  print(f"Error: Unable to retrieve content from {url}")  return  soup = BeautifulSoup(res.text, 'html.parser')  header\_tags = ['h1', 'h2', 'h3', 'h4', 'h5', 'h6']  sections = soup.find\_all(header\_tags)  print("Sections:")  for section in sections:  print(section.text.strip())  wiki\_sections("https://en.wikipedia.org/wiki/Linus\_Torvalds") |



* obțineți minim o imagine de pe acel site.

|  |
| --- |
| #Sarcina 3  import os  import requests  from urllib.parse import urljoin  from bs4 import BeautifulSoup  def wiki\_photo(url):  res = requests.get(url)  if res.status\_code != 200:  print(f"Error: Unable to retrieve content from {url}")  return  soup = BeautifulSoup(res.text, 'html.parser')  images = soup.find\_all('img')  save\_dir = 'img'  if not os.path.exists(save\_dir):  os.makedirs(save\_dir)  counter = 0  for image in images:  name = f"image\_{counter}:"  link = image['src']  absolute\_url = urljoin(url, link)  print(f"{name} {absolute\_url}")  file\_name = f'image\_{counter}.jpg'  file\_path = os.path.join(save\_dir, file\_name)  with open(file\_path, 'wb') as f:  im = requests.get(absolute\_url)  f.write(im.content)  counter += 1  wiki\_photo("https://en.wikipedia.org/wiki/Linus\_Torvalds") |



* 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;**

Pentru a putea parcurge fiecare pagină vom avea nevoie de un URL care se va modifica conform normelor de căutare astfel putem folosi acest URL al catalogului

[**http://books.toscrape.com/catalogue/page-{page\_nr}.html**](http://books.toscrape.com/catalogue/page-{page_nr}.html)

* **parsați fiecare pagină din catalog;**

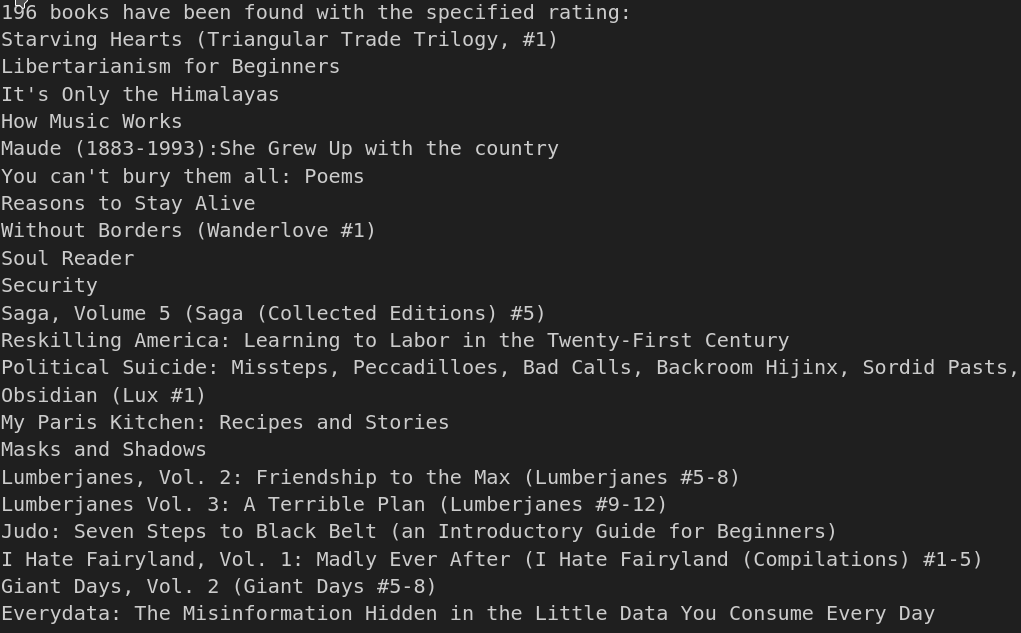
Astfel după ce am descoperit structura URL care ne va permite sa iterăm paginile vom putea folosi o variabila care se va incrementa după fiecare ciclu astfel atât timp cât request code-ul va fi 200 noi vom putea cerceta această pagină și selecta cărțile

* **găsiți ce etichetă/clasă reprezintă evaluarea cu stele;**

Clasa ce reprezintă valoarea cu stele este “star-rating nr\_of\_stars”

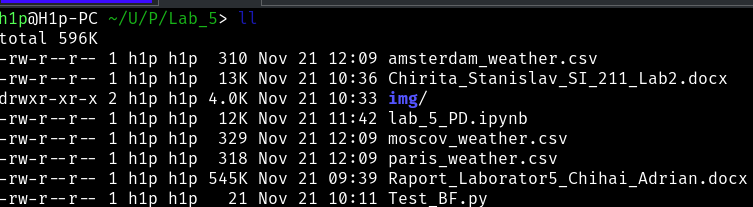
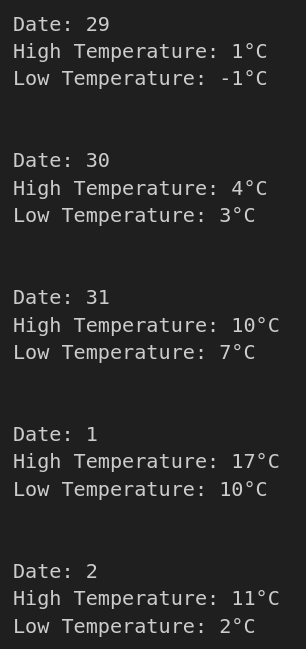
* + - filtrați cu if evaluarea cu stele;
    - stocați rezultatele într-o listă.

|  |
| --- |
| #Sarcina 4  import requests  from bs4 import BeautifulSoup  PRODUCT\_POD\_CLASS = 'product\_pod'  STAR\_RATING\_CLASS = 'star-rating Two'  def check\_rating(book\_item):  rating\_element = book\_item.find('p', class\_=STAR\_RATING\_CLASS)  return rating\_element is not None  def find\_rating(default\_url):  books = []  page\_nr = 1  while True:  url = f"{default\_url}/catalogue/page-{page\_nr}.html"  res = requests.get(url)  if res.status\_code == 200:  soup = BeautifulSoup(res.text, 'html.parser')  book\_items = soup.find\_all('article', class\_=PRODUCT\_POD\_CLASS)  for book\_item in book\_items:  title\_element = book\_item.find('h3')  if title\_element:  title = title\_element.a['title']  if check\_rating(book\_item):  books.append(title)  page\_nr += 1  else:  print(f"Can't go to page {page\_nr}\n")  break  return books  url = "http://books.toscrape.com"  books = find\_rating(url)  print(f"{len(books)} books have been found with the specified rating:")  for book in books:  print(book) |

****

* 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.

|  |
| --- |
| #Sarcina 5  import requests  from bs4 import BeautifulSoup  import csv  def fahrenheit\_to\_celsius(fahrenheit):  if '--' in fahrenheit:   1. return None   if '°' in fahrenheit:  fahrenheit = fahrenheit.replace('°', '')  return round((float(fahrenheit) - 32) \* 5 / 9)  def convert\_and\_write\_csv(csv\_writer, date, high\_temp\_str, low\_temp\_str):  high\_temp\_c = fahrenheit\_to\_celsius(high\_temp\_str)  low\_temp\_c = fahrenheit\_to\_celsius(low\_temp\_str)  csv\_writer.writerow({'D': date, 'H Temp': high\_temp\_c, 'L Temp': low\_temp\_c})  print(f"Date: {date}")  print(f"High Temperature: {high\_temp\_c}°C")  print(f"Low Temperature: {low\_temp\_c}°C")  print("\n")  def get\_weather(url, csv\_name):  response = requests.get(url)  if response.status\_code == 200:  soup = BeautifulSoup(response.text, 'html.parser')  buttons = soup.find\_all('button', class\_='Button--default--2gfm1')  with open(f'{csv\_name}.csv', 'w', newline='') as csvfile:  fieldnames = ['D', 'H Temp', 'L Temp']  csv\_writer = csv.DictWriter(csvfile, fieldnames=fieldnames)  csv\_writer.writeheader()  for button in buttons:  date\_element = button.find('span', class\_='CalendarDateCell--date--JO3Db')  if date\_element is None:  continue  date = date\_element.text.strip()  high\_temp\_element = button.find('div', class\_='CalendarDateCell--tempHigh--3k9Yr')  low\_temp\_element = button.find('div', class\_='CalendarDateCell--tempLow--2WL7c')  if high\_temp\_element is None or low\_temp\_element is None:  continue  high\_temp\_str = high\_temp\_element.text.strip()  low\_temp\_str = low\_temp\_element.text.strip()  convert\_and\_write\_csv(csv\_writer, date, high\_temp\_str, low\_temp\_str)  print(f"Weather data saved to {csv\_name}.csv")  else:  print(f"Failed to retrieve the page. Status code: {response.status\_code}")  url\_moscov = "https://weather.com/weather/monthly/l/aeea22202ca7af67cd001c86700f5486514081107a161ba8f78177867024ae43"  get\_weather(url\_moscov, 'moscov\_weather')  url\_paris = "https://weather.com/weather/monthly/l/d2a540efb4e9604b3c1d01b7851a1d9d2ab4c7b3ba428e5799936ac54404c035"  get\_weather(url\_paris, 'paris\_weather')  url\_amsterdam = "https://weather.com/weather/monthly/l/968d2f1a5509a2f71fca25929b7d83139ac5134f61611a9c6637c90354cd6da8"  get\_weather(url\_amsterdam, 'amsterdam\_weather') |



**Concluzie: P**rin explorarea diferitelor sarcini de web scraping pe Wikipedia și pe un site special conceput pentru testare, am reușit să extragem informații variate. Am capturat titlul paginii, titlurile secțiunilor și imagini de pe Wikipedia pentru un subiect de interes. Pe site-ul de test, am obținut titlurile cărților care au o evaluare de 2 stele și le-am salvat într-o listă Python.

De asemenea, am analizat și structura URL-ului pentru a parcurge fiecare pagină a cărților și am efectuat cereri către acesta, parcurs fiecare pagină din catalog, identificat clasa care reprezintă evaluarea cu stele și filtrat acele cărți cu evaluare de 2 stele, stocând rezultatele într-o listă.