Web Scrapping Lab 2

Coding & Explanation

Extract information from a given website Write the scraped data into a csv file.

Extract information from the given website You will extract the data from the below website:

#this url contains the data I need to scrape

url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM DA0321EN-SkillsNetwork/labs/datasets/Programming_Languages.html"

The data I need to scrape is the name of the programming language and the average annual salary. It is a good idea to open the URL in your web browser and study the web page's contents before I start to scrape.

#Import the required libraries

To start, I'll need to import the required libraries for web scraping and working with CSV files. Here are the imports I'll need:

```
import requests

from bs4 import BeautifulSoup

import pandas as pd
```

Explanation:

- **requests**: This library allows you to send HTTP requests to fetch the web page content.
- **BeautifulSoup**: This library is used for parsing HTML and extracting data.
- **pandas**: This library is used for data manipulation and writing data to CSV files.

Next, I should use these libraries to fetch, parse, and extract the required information from the given website.

# Download the webpage at the url

```
# URL of the webpage to download

url = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/IBM-DA0321EN-SkillsNetwork/labs/datasets/Programming_Languages.html"

# Send a GET request to the URL

response = requests.get(url)

# Check if the request was successful
```

```
if response.status_code == 200:
  # Get the content of the webpage
  page_content = response.content
  print("Webpage downloaded successfully.")
else:
  print(f"Failed to download webpage. Status code: {response.status_code}")
```

explanation:

1. **Import requests library**: This allows you to make HTTP requests.
2. **Define the URL**: Set the URL of the webpage to download.
3. **Send a GET request**: Use **requests.get(url)** to fetch the webpage.
4. **Check the response**: Ensure the request was successful by checking the status code.
5. **Get the content**: If successful, retrieve the content of the webpage.

This script will download the webpage's content and store it in the **page_content** variable. You can then proceed to parse and extract the required data using **BeautifulSoup**.

# Create a soup object

To create a **BeautifulSoup** object, I first need to parse the HTML content of the downloaded webpage. Here's how can do it:

```
from bs4 import BeautifulSoup
# Assuming 'page_content' contains the HTML content of the webpage
soup = BeautifulSoup(page_content, 'html.parser')
# Print the title of the webpage to confirm the soup object is created successfully
print(soup.title)
```

Explanation:

1. **Import BeautifulSoup**: Make sure to import `BeautifulSoup` from the `bs4` library.
2. **Create the soup object**: Pass the HTML content (`page_content`) and the parser (`html.parser`) to `BeautifulSoup` to create the soup object.
3. **Verify the soup object**: Print the title of the webpage to ensure the soup object has been created successfully.

Now I has a `BeautifulSoup` object that I can use to extract the required information from the webpage.

# Scrape the `Language name` and `annual average salary`.

To Scraping the Language name and annual average salary from the webpage. Here's the complete code to extract this information:

```
# Find the table containing the data

table = soup.find("table")


# Initialize lists to hold the scraped data

languages = []

salaries = []


# Iterate over each row in the table

for row in table.find_all("tr")[1:]:  # Skip the header row

    cols = row.find_all("td")

    language = cols[0].get_text(strip=True)

    salary = cols[1].get_text(strip=True)


    # Append to lists

    languages.append(language)

    salaries.append(salary)


# Print the scraped data

for language, salary in zip(languages, salaries):

    print(f"Language: {language}, Annual Average Salary: {salary}")
```

Explanation:

1. **Download the Webpage**: We use the **requests** library to fetch the content of the webpage.
2. **Create a Soup Object**: We parse the HTML content using **BeautifulSoup**.

3. **Scrape the Data**: We locate the table on the webpage, iterate over its rows, and extract the **Language name** and **annual average salary**.

This script will output the names of programming languages along with their respective average annual salaries.

# Save the scrapped data into a file named popular-languages.csv

To save the scraped data into a CSV file named popular-languages.csv,

use the **pandas** library. Here's the complete code including the previous steps:

```
# Create a DataFrame from the scraped data

data = {

  "Language": languages,

  "Annual Average Salary": salaries

}

df = pd.DataFrame(data)


# Save the DataFrame to a CSV file

df.to_csv('popular-languages.csv', index=False)

print("Data saved to popular-languages.csv successfully.")
```

Explanation:

1. **Import Libraries**: Make sure to import `requests`, `BeautifulSoup` from `bs4`, and `pandas`.
2. **Download the Webpage**: Fetch the webpage content using `requests`.
3. **Create a Soup Object**: Parse the HTML content using `BeautifulSoup`.
4. **Scrape the Data**: Extract the programming language names and average annual salaries from the table.
5. **Save to CSV**: Use `pandas` to create a DataFrame from the scraped data and save it to a CSV file named `popular-languages.csv`.

This code will scrape the programming language names and their average annual salaries from the specified webpage and save them into a file named `popular-languages.csv`.