Internship Project 3: Building a Simple Web Scraper

Task 3: Simple Data Manipulation with Pandas

Description:

The intern will create a basic web scraper using Python libraries like BeautifulSoup and Pandas to extract and manipulate data from a static web page.

Responsibilities:

1. Understand Web Scraping Principles:

- Follow tutorials to grasp the fundamentals of web scraping, including ethical considerations and data extra

2. Develop a Python Script:

- Create a simple Python script to scrape data from a static web page. Utilize libraries such as:
  - requests: For sending HTTP requests to fetch web pages.
  - BeautifulSoup: For parsing HTML content and extracting relevant information.

3. Extract Relevant Information:

- Extract various types of data, including:
  - Text: Paragraphs, headings, etc.
  - Links: Hyperlinks to other pages.
  - Images: URLs of images present on the web page.

4. Data Manipulation with Pandas:

- Clean and manipulate the extracted data using the Pandas library.
- Store the manipulated data in structured formats such as CSV, JSON, or Excel.

5. Download and Save Images:

- Download images from extracted URLs and save them locally in a specified folder for future use.

Example Code:

Here's an example code snippet demonstrating the functionality of the web scraper:

```
import os
import requests
from bs4 import BeautifulSoup
import pandas as pd


# Function to download images from extracted URLs
def download_images(image_urls, folder_name='images'):
    if not os.path.exists(folder_name):
        os.makedirs(folder_name)

    for i, url in enumerate(image_urls):
        try:
            image_content = requests.get(url).content
            image_file = os.path.join(folder_name, f'image_{i + 1}.jpg')
            with open(image_file, 'wb') as img_file:
                img_file.write(image_content)
        except Exception as e:
            print(f"Error downloading {url}: {e}")


# Function to extract text, links, and images from the parsed HTML
def extract_data(soup):
```

```
    if not soup:
        return None

    # Extract text
    texts = [p.text for p in soup.find_all('p')]

    # Extract links
    links = [a['href'] for a in soup.find_all('a', href=True)]

    # Extract image URLs
    images = [img['src'] for img in soup.find_all('img', src=True)]
    images = [img if img.startswith('http') else f'https://example.com{img}' for img in images]

    # Download extracted images
    download_images(images)

    return {'Text': texts, 'Links': links, 'Images': images}

# Main function to run the web scraper
def run_scraper(url):
    try:
        # Send HTTP request
        response = requests.get(url)
        soup = BeautifulSoup(response.content, 'html.parser')
```

```
        # Extract data
        data = extract_data(soup)

        # Convert data to DataFrame and save as CSV
        df = pd.DataFrame(data)
        df.to_csv('scraped_data.csv', index=False)

    except Exception as e:
        print(f'Error: {e}')

# Example URL to scrape
url = 'https://example.com'
run_scraper(url)
```

Enhancements:

1. Code Efficiency: The code structure is modular, allowing easy updates and maintenance.
2. Comprehensive Data Extraction: The scraper extracts text, links, and images, providing a holistic approa
3. Image Handling: Images are downloaded and stored locally, ensuring easy access for later use.

Conclusion:

This project showcases the ability to scrape data from web pages and manipulate it efficiently using Python libraries. It serves as a practical application of web scraping principles and data handling techniques, equipping the intern with valuable skills for data analysis.