Level 3: Advanced

Task 6: Create a program for interactive web scraping.

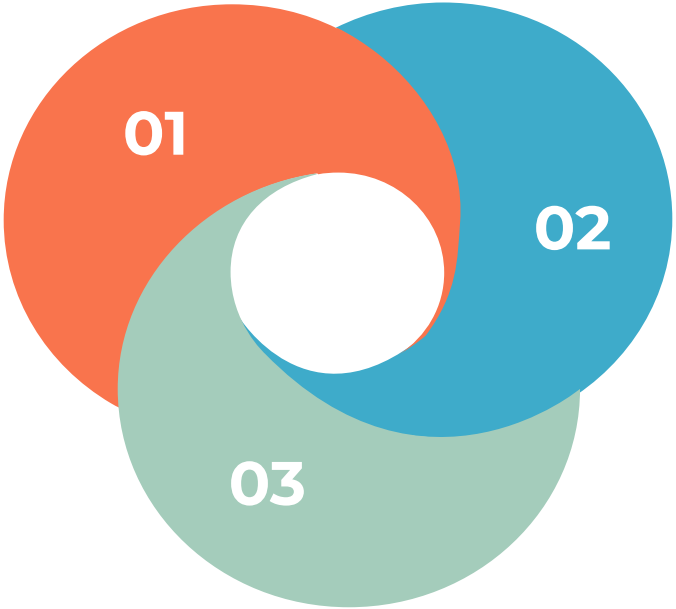
By Shradha Pujari

Creating a Program for Interactive Web Scraping

This presentation covers the process of creating a program for interactive web scraping. It discusses the objective, steps, and testing of the program.

Objective

Fetch data from a website



Present data in a user-friendly way

Use a simple web scraping library

Step 1: Select a Website

- Identify the data to be scraped

Step 2: Utilize a Web Scraping Library

- Fetch the data from the chosen website

Step 3: Design a User-Friendly Presentation Format

01 Organize the data in a visually pleasing way

02 Create a format that is easy to understand

Step 4: Test the Program

01 Ensure it works as expected

02 Test the program with different websites

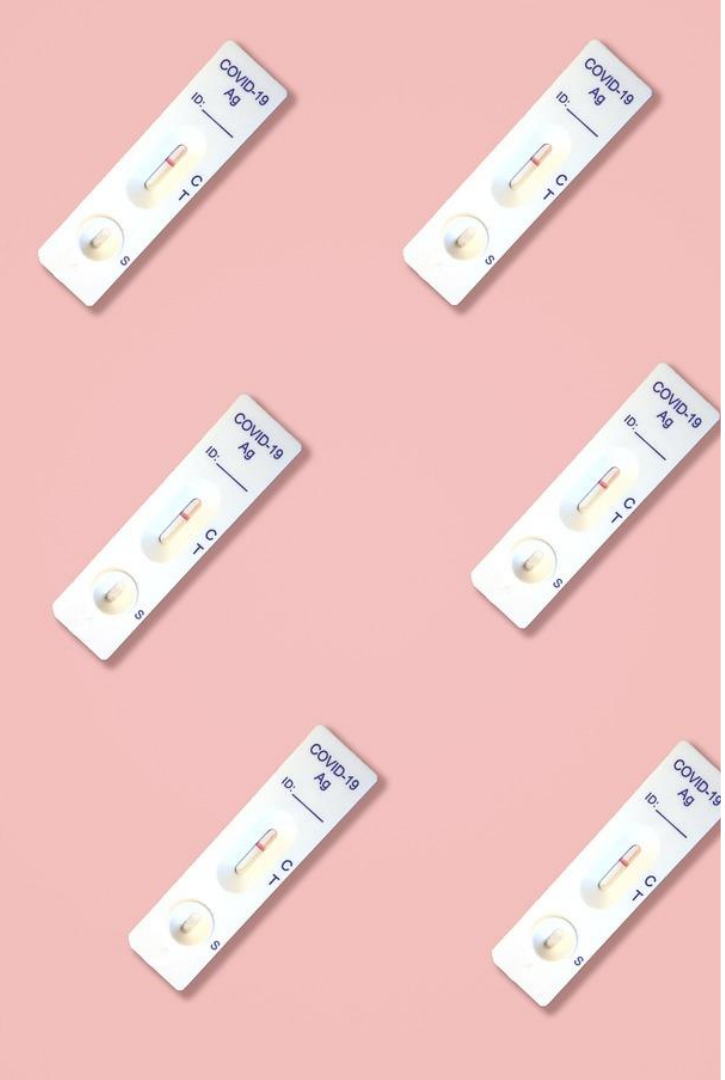
Explanation:

1. **Libraries:** This program uses requests to fetch the website content and BeautifulSoup to parse the HTML code.
2. **get_user_input():** This function prompts the user for the target website URL and the data they want to scrape (e.g., product names, article titles).
3. **scrape_data():** This function attempts to fetch the website content using requests. It parses the HTML with BeautifulSoup and searches for elements containing the desired data based on user input. Note: You'll need to adjust the element selection logic (class, data-testid) based on the specific website structure.

Explanation:

4. **present_data():** This function presents the scraped data in a user-friendly list format.

5. **main():** This function calls the other functions to get user input, scrape data, and present the results.



Testing the Program

01 Ensure that the program handles exceptions and non-200 status codes gracefully.

02 You can test the program by running it with different website URLs and data to scrape.

Code

```
import webbrowser


def main():
  print("Welcome to the Interactive Web Scraper!")
  website_choice = input("Choose a website (1 - Wikipedia, 2 - Project Gutenberg): ")


  if website_choice == "1":
    search_term = input("Enter your search term for Wikipedia: ")
    wikipedia_url = f"https://en.wikipedia.org/wiki/{search_term}"
    print(f"Opening Wikipedia search results for '{search_term}': {wikipedia_url}")
```

Code

```
webbrowser.open(wikipedia_url)

  elif website_choice == "2":

    print("Project Gutenberg offers a vast collection of free ebooks. Let's explore some genres:")

    print("1. Fiction")

    print("2. Non-Fiction")

    genre_choice = input("Enter your preferred genre (1 or 2): ")

    if genre_choice == "1":

      fiction_url = "https://www.gutenberg.org/ebooks/search/?query=fiction&sort=rank"
```

Code

```
        print(f"Here are some fiction ebooks: {fiction_url}")
        webbrowser.open(fiction_url)
    elif genre_choice == "2":
        nonfiction_url =
"https://www.gutenberg.org/ebooks/search/?query=nonfiction&sort=rank"
        print(f"Here are some non-fiction ebooks: {nonfiction_url}")
        webbrowser.open(nonfiction_url)
    else:
        print("Invalid genre choice. Please select 1 or 2.")
```

Code

```
else:
    print("Invalid choice. Please select 1 or 2.")


if __name__ == "__main__":
  main()
```

Output Example (Choosing Wikipedia):

```
Welcome to the Interactive Web Scraper!

Choose a website (1 - Wikipedia, 2 - Project Gutenberg): 1

Enter your search term for Wikipedia: Artificial Intelligence

Opening Wikipedia search results for 'Artificial Intelligence':
https://en.wikipedia.org/wiki/Artificial_intelligence
```

Output Example (Choosing Project Gutenberg with Genre Choice):

```
Welcome to the Interactive Web Scraper!

Choose a website (1 - Wikipedia, 2 - Project Gutenberg): 2

Project Gutenberg offers a vast collection of free ebooks. Let's explore some genres:

1. Fiction

2. Non-Fiction

Enter your preferred genre (1 or 2): 1

Here are some fiction ebooks:
https://www.gutenberg.org/ebooks/search/?query=fiction&sort=rank
```

Questions

- Any questions?

Thank you for your time and attention 🙂