Assignment 8 :

CODE:

```
import bs4
import requests
import pandas as pd
import time
import random


class BookScraperBot:
    def __init__(self, base_url='https://books.toscrape.com'):
        self.base_url = base_url
        self.catalogue_url = f"{base_url}/catalogue/"
        self.books_df = pd.DataFrame()


    def get_soup(self, url):
        """Get BeautifulSoup object from URL with error handling"""
        try:
            headers = {
                'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/91.0.4472.124 Safari/537.36'
            }
            response = requests.get(url, headers=headers)
            response.raise_for_status()  # Raise exception for 4XX/5XX responses
            return bs4.BeautifulSoup(response.text, 'lxml')
        except requests.exceptions.RequestException as e:
            print(f"Error fetching {url}: {e}")
```

```
        return None

def extract_book_details(self, book_url):
    """Extract detailed information for a single book"""
    soup = self.get_soup(book_url)
    if not soup:
        return None

    # Extract all product information from table
    product_info = {}
    info_table = soup.select('table.table-striped tr')
    for row in info_table:
        header = row.select_one('th').text.strip()
        value = row.select_one('td').text.strip()
        product_info[header] = value

    # Get title, price, and rating
    title = soup.select_one('div.product_main h1').text.strip()
    product_info['Title'] = title

    # Get price
    price = soup.select_one('p.price_color').text.strip()
    product_info['Price'] = price.replace('£', '')

    # Get availability
    availability = soup.select_one('p.availability').text.strip()
```

```
product_info['Availability'] = availability

# Get rating
rating_element = soup.select_one('p.star-rating')
if rating_element:
    rating_class = rating_element.get('class')
    if len(rating_class) > 1:
        product_info['Rating'] = rating_class[1]
    else:
        product_info['Rating'] = 'Not Rated'
else:
    product_info['Rating'] = 'Not Rated'

# Get category
breadcrumb = soup.select('ul.breadcrumb li')
if len(breadcrumb) >= 3:
    category = breadcrumb[2].text.strip()
    product_info['Category'] = category
else:
    product_info['Category'] = 'Unknown'

# Get description
desc_element = soup.select_one('div#product_description + p')
if desc_element:
    product_info['Description'] = desc_element.text.strip()
else:
```

```
            product_info['Description'] = 'No description available'

        return product_info

    def scrape_book_listings(self, page_num):
        """Scrape book listings from a specific page"""
        url = f"{self.base_url}/catalogue/page-{page_num}.html"
        soup = self.get_soup(url)
        if not soup:
            return []

        book_links = []
        for article in soup.select('article.product_pod'):
            link_element = article.select_one('h3 a')
            if link_element:
                href = link_element.get('href')
                if href:
                    # Handle relative URLs
                    if href.startswith('../'):
                        href = href.replace('../', '')
                    full_url = f"{self.catalogue_url}{href}"
                    book_links.append(full_url)

        return book_links

    def scrape_all_books(self, max_pages=50, delay=1):
```

```
"""Scrape all books from the website"""
all_books = []

for page in range(1, max_pages + 1):
    print(f"Scraping page {page} of {max_pages}...")

    # Get all book links from this page
    book_links = self.scrape_book_listings(page)

    # Check if we got any books (if not, we might have reached the end)
    if not book_links:
        print(f"No books found on page {page}. Stopping.")
        break

    # Process each book
    for i, link in enumerate(book_links):
        print(f"  Processing book {i+1}/{len(book_links)} on page {page}...")
        book_details = self.extract_book_details(link)
        if book_details:
            all_books.append(book_details)

        # Add a small delay to avoid overwhelming the server
        time.sleep(delay)

    # Random delay between pages
    time.sleep(delay + random.uniform(0.5, 1.5))
```

```
        # Create DataFrame from all books
        self.books_df = pd.DataFrame(all_books)
        return self.books_df

    def save_to_csv(self, filename='books_data.csv'):
        """Save scraped data to CSV file"""
        if not self.books_df.empty:
            self.books_df.to_csv(filename, index=False)
            print(f"Data saved to {filename}")
        else:
            print("No data to save. Please run scrape_all_books() first.")

    def save_to_excel(self, filename='books_data.xlsx'):
        """Save scraped data to Excel file"""
        if not self.books_df.empty:
            self.books_df.to_excel(filename, index=False)
            print(f"Data saved to {filename}")
        else:
            print("No data to save. Please run scrape_all_books() first.")

# Usage example
if __name__ == "__main__":
    # Create the bot
    bot = BookScraperBot()
```

```
# Scrape books - adjust max_pages and delay as needed
# Lower max_pages for testing (e.g., 2-3 pages)
# Higher delay (e.g., 1-2 seconds) to be respectful to the server
books_df = bot.scrape_all_books(max_pages=3, delay=1)


# Print summary
print("\nScraping complete!")
print(f"Total books scraped: {len(books_df)}")
print("\nSample of scraped data:")
print(books_df.head())
# Save data
bot.save_to_csv()
```

Output:

```
  Processing book 13/20 on page 3...
  Processing book 14/20 on page 3...
  Processing book 15/20 on page 3...
  Processing book 16/20 on page 3...
  Processing book 17/20 on page 3...
  Processing book 18/20 on page 3...
  Processing book 19/20 on page 3...
  Processing book 20/20 on page 3...

Scraping complete!
Total books scraped: 60
```

```
Sample of scraped data:
          UPC Product Type Price (excl. tax) Price (incl. tax)  ...  Price Rating         Category                                        Description
0  a897fe39b1053632      Books         Â£51.77         Â£51.77  ...  Â51.77  Three          Poetry  It's hard to imagine a world without A Light i...
1  90fa61229261140a      Books         Â£53.74         Â£53.74  ...  Â53.74    One Historical Fiction  "Erotic and absorbing...Written with starling ...
2  6957f44c3847a760      Books         Â£50.10         Â£50.10  ...  Â50.10    One         Fiction  Dans une France assez proche de la nÂ´tre, un ...
3  e00eb4fd7b871a48      Books         Â£47.82         Â£47.82  ...  Â47.82   Four         Mystery  WICKED above her hipbone, GIRL across her hear...
4  4165285e1663650f      Books         Â£54.23         Â£54.23  ...  Â54.23   Five         History  From a renowned historian comes a groundbreaki...

[5 rows x 12 columns]
Data saved to books_data.csv
```