Web Scraping Workshop: Student Worksheet

No API Key Required!

Pre-Demo Setup

```python
# Install our new tool
!pip install beautifulsoup4 vaderSentiment wordcloud

# Import everything we need
import requests
from bs4 import BeautifulSoup
import pandas as pd
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
from wordcloud import WordCloud
import matplotlib.pyplot as plt
```

Part 1: Understanding the HTML Structure

Your First Look

Visit http://quotes.toscrape.com in your browser

- Right-click on any quote → "Inspect Element"
- Notice the pattern? Each quote is in a `<div class="quote">`

Finding Patterns (Fill in the blanks)

```html
<div class="_____">
  <span class="_____">Quote text here</span>
  <small class="_____">Author name</small>
</div>
```

Part 2: Basic Scraping

Step 1: Get the webpage

```python
url = "http://quotes.toscrape.com/"
response = requests.get(url)
print(f"Status code: {response.status_code}")  # Should be 200
```

Step 2: Create the soup

```python
soup = BeautifulSoup(response.text, 'html.parser')
```

Step 3: Find all quotes

```python
quote_blocks = soup.find_all('div', class_='quote')
print(f"Found {len(quote_blocks)} quotes")
```

Your turn: How many quotes did you find? _____

Part 3: Extracting Data

Loop Through and Extract

```python
quotes_data = []
for block in quote_blocks:
    text = block.find('span', class_='text').text
    author = block.find('small', class_='author').text

    quotes_data.append({
        'author': author,
        'text': text
    })

# Convert to DataFrame
df = pd.DataFrame(quotes_data)
```

Check your work:

```python
print(df.shape)  # Should show (10, 2)
print(df.head(2))  # Shows first 2 quotes
```

Part 4: Sentiment Analysis

Add Sentiment Scores

```python
analyzer = SentimentIntensityAnalyzer()
df['sentiment'] = df['text'].apply(
    lambda x: analyzer.polarity_scores(x)['compound']
)
```

Find Extremes

```python
# Most positive quote
most_positive = df.loc[df['sentiment'].idxmax()]
print(f"Most positive ({most_positive['sentiment']:.2f}):")
print(f"  {most_positive['text']}")

# Most negative quote
most_negative = df.loc[df['sentiment'].idxmin()]
print(f"Most negative ({most_negative['sentiment']:.2f}):")
print(f"  {most_negative['text']}")
```

Record your findings:

- Most positive author: _____
- Most negative author: _____
- Average sentiment: _____

Part 5: Try These Modifications

Mod 1: Scrape Tags Too

Each quote has tags. Can you extract them?

```python
```

```
# Hint: Look for <a class="tag">
tags = block.find_all('a', class_='tag')
tag_list = [tag.text for tag in tags]
```

Mod 2: Multi-Page Scraping

```python
all_quotes = []
for page in range(1, 4):  # Pages 1-3
    url = f"http://quotes.toscrape.com/page/{page}/"
    # Your scraping code here

print(f"Total quotes from 3 pages: {len(all_quotes)}")
```

Mod 3: Author Analysis

```python
# Which author appears most?
author_counts = df['author'].value_counts()
print("Most quoted authors:")
print(author_counts.head(3))
```

Part 6: Real-World Practice Sites

Safe Practice Sites

1. **Books.toscrape.com** - Book catalog
   - Challenge: Scrape titles and prices
   - Bonus: Calculate average book price
2. **Scrapethissite.com** - Various datasets
   - Challenge: Scrape country data
   - Bonus: Find countries by population
3. **Webscraper.io/test-sites** - E-commerce practice
   - Challenge: Product names and prices
   - Bonus: Find most expensive item

Debugging Checklist

If nothing works:

- [ ] Check URL is correct (no typos)
- [ ] Print `response.status_code` (should be 200)
- [ ] Print `response.text[:500]` to see raw HTML
- [ ] Verify class names match exactly (case-sensitive!)

If data looks weird:

- [ ] Use `.strip()` to remove extra spaces
- [ ] Check for special characters
- [ ] Print one item first before looping all

Mini Challenges

Challenge 1: Sentiment by Author

Which author has the most positive average sentiment?

```python
```

```python
author_sentiment = df.groupby('author')['sentiment'].mean()
# Your code to find the answer
```

Challenge 2: Word Frequency

What's the most common word (excluding common words)?

```python
from collections import Counter
all_words = ' '.join(df['text']).lower().split()
# Remove quotes and punctuation first!
```

Challenge 3: Quote Length Analysis

Do longer quotes have different sentiment?

```python
df['length'] = df['text'].str.len()
# Check correlation between length and sentiment
```

Your Project Ideas

Beginner

- [ ] Scrape your favorite blog's headlines
- [ ] Extract prices from an online store
- [ ] Collect motivational quotes

Intermediate

- [ ] Track product prices over time
- [ ] Compare headlines from different news sites
- [ ] Analyze restaurant menu prices

Advanced

- [ ] Build a job market analyzer
- [ ] Create a review sentiment tracker
- [ ] Monitor competitor pricing

Important Reminders

Always Check First

- robots.txt (add to any domain URL)
- Don't scrape personal data
- Add delays between requests
- Identify yourself in headers

Good Practice

```python
headers = {
    'User-Agent': 'Student Learning Project'
}
response = requests.get(url, headers=headers)
```

Quick Reference

BeautifulSoup Methods

- `soup.find()` - First matching element

- `soup.find_all()` - All matching elements
- `.text` - Extract text content
- `.get('attribute')` - Get attribute value
- `.parent` - Go up one level
- `.children` - Go down one level

Common Selectors

- By tag: `soup.find_all('div')`
- By class: `soup.find_all('div', class_='quote')`
- By id: `soup.find('div', id='main')`
- By attribute: `soup.find_all('a', href=True)`

Today's Key Learning

You just learned the skill behind:

- Google (web crawling)
- Trivago (price comparison)
- Indeed (job aggregation)
- Any "finds the best deals" website

Remember: With great scraping power comes great responsibility!