webscraping_python

February 3, 2025

1 Webscraping med Python og BeautifulSoup - Af Henrik Sterner (henrik.sterner@gmail.com)

Webscraping betyder groft sagt at hente/skrabe data fra internettet. Det kan være alt fra at hente en enkelt side, til at hente data fra mange sider og gemme dem i en database. Man kan lave webscraping i mange forskellige programmeringssprog inklusiv Python. I Python er der flere biblioteker, der kan bruges til webscraping, men et af de mest populære er BeautifulSoup. BeautifulSoup er et bibliotek, der kan bruges til at hente data fra HTML og XML filer.

For at kunne bruge BeautifulSoup skal det installereres. Dette gøres ved at skrive følgende i terminalen:

```
pip install beautifulsoup4
```

Når BeautifulSoup er installeret, kan det importeres i Python ved at skrive:

```
from bs4 import BeautifulSoup
```

1.1 Advarsel: Læs altid en hjemmesides "Terms of Service" før du webscraper

Det er vigtigt at være opmærksom på, at det ikke er alle hjemmesider, der tillader webscraping. Derfor er det vigtigt at læse en hjemmesides "Terms of Service" før du webscraper. Hvis en hjemmeside ikke tillader webscraping, kan det resultere i at hjemmesiden blokerer din IP-adresse, så du ikke længere kan tilgå hjemmesiden.

Du kan finde en hjemmesides "Terms of Service" ved at scrolle ned til bunden af hjemmesiden og finde et link der hedder noget i retning af "Terms of Service", "Terms and Conditions" eller "Terms of Use". Hvis du ikke kan finde en "Terms of Service" på hjemmesiden, kan du prøve at kontakte hjemmesidens ejer og spørge om det er tilladt at webscrape.

Typisk kan du finde information om webscraping i en hjemmesides "Terms of Service" under afsnit som "Robots.txt", "Robots Meta Tag" eller "Robots Exclusion Standard". Hvis en hjemmeside ikke tillader webscraping, kan du kontakte hjemmesidens ejer og spørge om det er tilladt at webscrape.

1.2 Læsning af HTML

For at kunne webscrape data fra en hjemmeside, skal man først hente HTML koden fra siden. Dette kan gøres på flere måder. Manuelt eller automatisk. Manuelt kan det gøres ved at højreklikke på siden og vælge "Vis kildekode". Automatisk kan det gøres ved at brug af beautifulsoup og request biblioteket. Request biblioteket bruges til at hente data fra internettet. For at kunne bruge request biblioteket skal det installeres. Dette gøres ved at skrive følgende i terminalen:

```
pip install requests
```

Nu er vi klar til at hente HTML koden fra en hjemmeside. Dette gøres ved at skrive følgende kode:

[ ]:
```python
# Scrape hjemmesiden https://henriksterner.github.io/IntroPython/

import requests
from bs4 import BeautifulSoup

url = "https://henriksterner.github.io/IntroPython/"
response = requests.get(url)
soup = BeautifulSoup(response.text, "html.parser")

print(soup.prettify())
```

Vi bemærker, at vi bruger `requests.get()` til at hente HTML koden fra siden. Herefter bruger vi `BeautifulSoup()` til at parse HTML koden. Vi kan nu bruge BeautifulSoup til at finde elementer i HTML koden. soup.prettify() bruges til at formatere HTML koden, så den er nemmere at læse.

I næste afsnit vil vi se, hvordan vi kan finde elementer og info i HTML koden.

1.3 Analyse af HTML-koden

Når vi har hentet HTML koden fra en hjemmeside, kan vi bruge BeautifulSoup til at finde elementer og info i HTML koden. Dette gøres ved at bruge forskellige metoder i BeautifulSoup.

1.3.1 Find metoden

Find metoden bruges til at finde det første element, der matcher det angivne tag. Hvis vi f.eks. vil finde det første `<p>` tag i HTML koden, kan vi skrive følgende kode:

```
soup.find('p')
```

Herunder er et eksempel på, hvordan vi kan finde titlen på hjemmesiden ved at finde det første `<title>` tag:

[2]:
```python
print(soup.title)
print(soup.title.string)
print(soup.find('title'))
print(soup.find('title').string)
```

```
<title>Introduktion til Programmering i Python ‘ | IntroPython</title>
Introduktion til Programmering i Python ‘ | IntroPython
<title>Introduktion til Programmering i Python ‘ | IntroPython</title>
Introduktion til Programmering i Python ‘ | IntroPython
```

Vi bemærker, at vi har flere måder at finde titlen på hjemmesiden, men den mest almindelige er at bruge `soup.title.string`.

1.3.2 Find_all metoden

Find_all metoden bruges til at finde alle elementer, der matcher det angivne tag. Hvis vi f.eks. vil finde alle `<p>` tags i HTML koden, kan vi skrive følgende kode:

`soup.find_all('p')`

I det følgende vil vi prøve at trække alle elementer i tabellen ud. Vi kan bruge `find_all()` metoden til at finde alle `<tr>` tags i HTML koden. Herefter kan vi bruge en for-løkke til at finde alle `<td>` tags i hvert `<tr>` tag. Vi kan nu udskrive indholdet af hvert `<td>` tag ved at bruge `td.text`:

[ ]:
```
# Find alle tr tags
rows = soup.find_all('tr')
for row in rows:
    # Find alle td tags i hver tr tag
    cells = row.find_all('td')
    for cell in cells:
        print(cell.text)
```

1.3.3 Find_parent metoden

Find_parent metoden bruges til at finde det første parent element, der matcher det angivne tag. Hvis vi f.eks. vil finde det første parent element til et `<p>` tag, kan vi skrive følgende kode:

`soup.find('p').find_parent()`

I det følgende vil vi prøve at finde parent elementet til et `<td>` tag. Vi kan bruge `find_all()` metoden til at finde alle `<td>` tags i HTML koden. Herefter kan vi bruge `td.find_parent()` til at finde parent elementet til hvert `<td>` tag. Vi kan nu udskrive indholdet af parent elementet ved at bruge `parent.text`:

[ ]:
```
alltds = soup.find_all('td')
for td in alltds:
    print(td.find_previous('th').text, td.text)
```

1.4 Andre metoder i BeautifulSoup

Der er mange andre metoder i BeautifulSoup, der kan bruges til at finde elementer og info i HTML koden. Nogle af de mest almindelige metoder er:

- `find_next()`: Bruges til at finde det næste element, der matcher det angivne tag.
- `find_previous()`: Bruges til at finde det forrige element, der matcher det angivne tag.
- `find_all_next()`: Bruges til at finde alle næste elementer, der matcher det angivne tag.
- `find_all_previous()`: Bruges til at finde alle forrige elementer, der matcher det angivne tag.

Det overlades til læseren at prøve disse metoder i praksis.

1.5 Opgaver

Start med at finde en hjemmeside, som du vil webscrape. Læs hjemmesidens "Terms of Service" for at sikre, at det er tilladt at webscrape. Hvis du ikke kan finde en så kig herunder for at finde

en hjemmeside, som du kan webscrape.

1. Hent HTML koden fra hjemmesiden ved at bruge request biblioteket.
2. Brug BeautifulSoup til at parse HTML koden.
3. Brug BeautifulSoup til at finde elementer og info i HTML koden.
4. Find titlen på hjemmesiden.
5. Find alle tabeller på hjemmesiden og hent herefter alle elementer i tabellen ud
6. Gem dataene i en komma-separeret fil (CSV fil).
7. Find andre elementer og info på hjemmesiden og gem dem i en CSV fil.
8. Skriv en funktion der finder alle links på en hjemmeside og gemmer dem i en CSV fil.
9. Skriv en funktion der finder alle billeder på en hjemmeside og gemmer dem i en CSV fil.
10. Skriv en funktion der finder alle e-mail adresser på en hjemmeside og gemmer dem i en CSV fil.
11. Skriv en funktion, der finder alle tags med en bestemt værdi og gemmer dem i en CSV fil.
12. Skriv en funktion, der finder alle tags med en bestemt klasse og gemmer dem i en CSV fil.
13. Skriv en funktion, der finder alle tags med en bestemt id og gemmer dem i en CSV fil.
14. Skriv en funktion, der finder alle tags med en bestemt attribut og gemmer dem i en CSV fil.
15. Skriv en funktion, der finder alle tags med en bestemt attributværdi og gemmer dem i en CSV fil.

Hint: Du kan i læse og skrive til en CSV fil ved at bruge `csv` biblioteket. For at kunne bruge `csv` biblioteket skal det importeres i Python ved at skrive:

```
import csv
```

Herefter kan du bruge `csv.reader()` til at læse fra en CSV fil og `csv.writer()` til at skrive til en CSV fil.

```
with open('data.csv', 'w', newline='') as file:
    writer = csv.writer(file)
    writer.writerow(['Name', 'Age', 'City'])
    writer.writerow(['John', '25', 'New York'])
    writer.writerow(['Jane', '30', 'Los Angeles'])
```

Du kan finde mere information om `csv` biblioteket i Python dokumentationen: https://docs.python.org/3/library/csv.html

1.6 Bilag

Herunder en html-side som du kan bruge til opgaven hvis du ikke har en hjemmeside at webscrape:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Unge Menneskers Rygevaner og Andre Statistikker</title>
    <style>
        table {
            width: 100%;
            border-collapse: collapse;
```

```
            margin-bottom: 20px;
        }
        th, td {
            border: 1px solid #ddd;
            padding: 8px;
        }
        th {
            background-color: #f2f2f2;
        }
        h1, h2 {
            text-align: center;
        }
    </style>
</head>
<body>
    <h1>Unge Menneskers Rygevaner og Andre Statistikker</h1>

    <h2>Rygevaner blandt unge</h2>
    <table>
        <thead>
            <tr>
                <th>Alder</th>
                <th>Procent der ryger dagligt</th>
                <th>Procent der ryger lejlighedsvis</th>
                <th>Procent der aldrig har røget</th>
            </tr>
        </thead>
        <tbody>
            <tr>
                <td>15-17</td>
                <td>10%</td>
                <td>15%</td>
                <td>75%</td>
            </tr>
            <tr>
                <td>18-20</td>
                <td>20%</td>
                <td>25%</td>
                <td>55%</td>
            </tr>
            <tr>
                <td>21-23</td>
                <td>25%</td>
                <td>20%</td>
                <td>55%</td>
            </tr>
        </tbody>
    </table>
```

```
<h2>Andre sundhedsrelaterede vaner</h2>
<table>
    <thead>
        <tr>
            <th>Alder</th>
            <th>Procent der dyrker motion regelmæssigt</th>
            <th>Procent der spiser sundt</th>
            <th>Procent der drikker alkohol ugentligt</th>
        </tr>
    </thead>
    <tbody>
        <tr>
            <td>15-17</td>
            <td>60%</td>
            <td>70%</td>
            <td>30%</td>
        </tr>
        <tr>
            <td>18-20</td>
            <td>50%</td>
            <td>65%</td>
            <td>40%</td>
        </tr>
        <tr>
            <td>21-23</td>
            <td>55%</td>
            <td>60%</td>
            <td>50%</td>
        </tr>
    </tbody>
</table>

<h2>Uddannelsesniveau</h2>
<table>
    <thead>
        <tr>
            <th>Alder</th>
            <th>Procent med gymnasial uddannelse</th>
            <th>Procent med erhvervsuddannelse</th>
            <th>Procent med videregående uddannelse</th>
        </tr>
    </thead>
    <tbody>
        <tr>
            <td>15-17</td>
            <td>80%</td>
            <td>10%</td>
```

```
                <td>10%</td>
            </tr>
            <tr>
                <td>18-20</td>
                <td>70%</td>
                <td>20%</td>
                <td>10%</td>
            </tr>
            <tr>
                <td>21-23</td>
                <td>60%</td>
                <td>25%</td>
                <td>15%</td>
            </tr>
        </tbody>
    </table>
</body>
</html>
```