web_scraping

May 12, 2025

1 Web Scraping Exercise

Web Scraping allows you to gather large volumes of data from diverse and real-time online sources. This data can be crucial for enriching your datasets, filling in gaps, and providing current information that enhances the quality and relevance of your analysis. Web scraping enables you to collect data that might not be readily available through traditional APIs or databases, offering a competitive edge by incorporating unique and comprehensive insights. Moreover, it automates the data collection process, saving time and resources while ensuring a scalable approach to continuously updating and maintaining your datasets.

Ethical web scraping involves respecting website terms of service, avoiding overloading servers, and ensuring that the collected data is used responsibly and in compliance with privacy laws and regulations.

Use Python, `requests`, `BeautifulSoup` and/or `pandas` to scrape web data:

1.1 Import Libraries

[24]:
```
import requests
from bs4 import BeautifulSoup
import pandas as pd
import time
```

1.2 Define the Target URL

[25]:
```
# ein FOKUS auf AMD-CPU auf der Seite geizhals
url = 'https://geizhals.at/?cat=cpuamdam4'
```

1.3 Send a Request to the Website

Do not forget to check the response status code

[27]:
```
#Headers verwendet --> bessere Chancen um auf die Website draufzugreifen, damit␣
 ↪sagen wir basically den Server das wir eine ganz normale Chrome Browser sind
headers = {
    'User-Agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64)'
}
```

```
response = requests.get(url, headers=headers)
print("Status Code:", response.status_code)
```

```
Status Code: 200
```

1.4 Parse the HTML Content

Use a library to access the HTMl content

[32]:
```
soup = BeautifulSoup(response.text, 'html.parser')
print(response.text[:1000])
```

```
<!DOCTYPE HTML><html lang="de"><head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title>Prozessoren (CPUs) AMD Preisvergleich Geizhals Österreich</title>
<meta property='og:image' content='https://gzhls.at/b/favicons/gh/gh-
ogimage.png?ogircp'><meta property="og:title" content="Prozessoren (CPUs) AMD
Preisvergleich Geizhals Österreich">
<meta property="og:site_name" content="Geizhals.at">
<meta property="og:description" content="Preisvergleich für Prozessoren (CPUs)
AMD  Bewertungen  Produktinfos  Auswahl und Filtern der Produkte nach den
besten Eigenschaften und dem billigsten Preis">
<meta name="theme-color" media="(prefers-color-scheme: light)"
content="#005599">
<meta name="theme-color" media="(prefers-color-scheme: dark)" content="#161b22">

<!-- geizhals.at=gh_at 2605863;97 --><link rel="canonical"
href="https://geizhals.at/?cat=cpuamdam4"><meta property="og:url"
content="https://geizhals.at/?cat=cpuamdam4&amp;hloc=at&amp;nocookie=1">
<meta name="viewpo
```

1.5 Identify the Data to be Scraped

Write a couple of sentence on the data you want to scrape

TODO: I want to scrape a list of AMD desktop CPUs from Geizhals.at.

For each CPU, I want to extract: - The product name - The current listed price (in EUR)

These data points will allow me to later analyze the price development of CPUs over time.

1.6 Extract Data

Find specific elements and extract text or attributes from elements (handle pagination if necessary)

[20]:
```
cpu_list = []


page = 1  # beginne bei Seite 1
```

```
max_pages = 30 # hört bei Seite 30 auf, weil es auf geizhals zuviele versteckte␣
 ↪Seiten gibt

while page <= max_pages:
    print(f"Scraping Seite {page}...")
    url = url + str(page)

    # productlist__product ist die Zeile in der Produktliste
    products = soup.find_all('div', class_='productlist__product')

    if not products:
        print(f"Keine Produkte auf Seite {page} gefunden - Schleife wird␣
 ↪beendet.")
        break  # keine weiteren Produkte -> beenden


    for product in products:
        name_container = product.find('div', class_='productlist__name')
        price_container = product.find('div', class_='productlist__price')

        if name_container and price_container:

            title_span = name_container.find('span', class_='notrans')
            price_span = price_container.find('span', class_='notrans')

            if title_span and price_span:
                #Hier sind die aktuellen Infos die wir brauchen
                title = title_span.get_text(strip=True)
                price_text = price_span.get_text(strip=True)
                #falls Preis Euro(€) und Punkt(.) hat soll es leer sein und bei␣
 ↪Komma (,) soll es ein Punkt (.) haben damit wir es in ein float umwandeln␣
 ↪können
                price = price_text.replace('€', '').replace('.', '').
 ↪replace(',', '.')

                try:
                    price_float = float(price)
                except ValueError:
                    continue

                cpu_list.append({
                    'Title': title,
                    'Price_EUR': price_float
                })

    page += 1
    time.sleep(1) # Pause, damit man Server nicht allzu stresst
```

```
df = pd.DataFrame(cpu_list)
print(f"\n Insgesamt {len(df)} CPUs gefunden")
df.head()
```

```
Scraping Seite 1…
Scraping Seite 2…
Scraping Seite 3…
Scraping Seite 4…
Scraping Seite 5…
Scraping Seite 6…
Scraping Seite 7…
Scraping Seite 8…
Scraping Seite 9…
Scraping Seite 10…
Scraping Seite 11…
Scraping Seite 12…
Scraping Seite 13…
Scraping Seite 14…
Scraping Seite 15…
Scraping Seite 16…
Scraping Seite 17…
Scraping Seite 18…
Scraping Seite 19…
Scraping Seite 20…
Scraping Seite 21…
Scraping Seite 22…
Scraping Seite 23…
Scraping Seite 24…
Scraping Seite 25…
Scraping Seite 26…
Scraping Seite 27…
Scraping Seite 28…
Scraping Seite 29…
Scraping Seite 30…

 Insgesamt 900 CPUs gefunden
```

[20]:

	Title	Price_EUR
0	AMD Ryzen 7 9800X3D, 8C/16T, 4.70-5.20GHz, box…	494.90
1	AMD Ryzen 9 9950X3D, 16C/32T, 4.30-5.70GHz, bo…	782.90
2	AMD Ryzen 7 7800X3D, 8C/16T, 4.20-5.00GHz, box…	399.00
3	AMD Ryzen 7 5700X3D, 8C/16T, 3.00-4.10GHz, box…	219.88
4	AMD Ryzen 7 9700X, 8C/16T, 3.80-5.50GHz, boxed…	308.84

1.7 Store Data in a Structured Format

Give a brief overview of the data collected (e.g. count, fields, …)

[36]:
```
df = pd.DataFrame(cpu_list)


print("Spalten:", df.columns.tolist())


print("Anzahl der CPUs:", len(df), "\n")


print("Datentypen:")
print(df.dtypes)

df.head()
```

```
Spalten: ['Title', 'Price_EUR']
Anzahl der CPUs: 900

Datentypen:
Title         object
Price_EUR    float64
dtype: object
```

[36]:

	Title	Price_EUR
0	AMD Ryzen 7 9800X3D, 8C/16T, 4.70-5.20GHz, box…	494.90
1	AMD Ryzen 9 9950X3D, 16C/32T, 4.30-5.70GHz, bo…	782.90
2	AMD Ryzen 7 7800X3D, 8C/16T, 4.20-5.00GHz, box…	399.00
3	AMD Ryzen 7 5700X3D, 8C/16T, 3.00-4.10GHz, box…	219.88
4	AMD Ryzen 7 9700X, 8C/16T, 3.80-5.50GHz, boxed…	308.84

1.8 Save the Data

[37]:
```
from datetime import datetime

#Datum wird automatisch in den Filenamen gespeichert
datum = datetime.today().strftime('%Y-%m-%d')
output_filename = f"geizhals_amd_cpus_{datum}.csv"

# Speichern
df.to_csv(output_filename, index=False, encoding='utf-8')
print(f"Datei gespeichert unter: {output_filename}")
```

```
Datei gespeichert unter: geizhals_amd_cpus_2025-05-12.csv
```

[ ]: