Programmieren mit Python

Teil 11: APIs

Dr. Aaron Kunert
aaron.kunert@salemkolleg.de

13. Dezember 2021

Web APIs

Wie Python im Netz surfen kann

Definition: API

Eine *API* (Application Programming Interface) ist im Allgemeinen eine klar definierte Schnittstelle zwischen Programmen.

Im Kontext des Internets geht es dabei meist um Webanwendungen, die statt einer für menschen lesbaren Seite, ein maschinenlesbares JSON ausliefern. Auf diese Weise ist ein Datenaustausch zwischen Deinem Programm und einer Webanwendung in beide Richtungen möglich.

Eine Web-Api besteht meist aus einer (oder mehreren) Webadressen und einer Anleitung, wie sie zu benutzen ist.

Beispiele für APIs

- **Instagram Api**: Erhalte Infos, wer Dir folgt und welche Reaktionen Du auf Deine Bilder erhältst.
- **GoogleMaps Api**: Sende Koordinaten an Google-Maps und erhalte die Adresse.
- **REST Countries**: Erhalte Daten von Ländern dieser Erde.
- **Stock Market Apis**: Erhalte live Daten zu Börsenkursen.

Anwendungsbeispiele

- **Instagram**: Schreibe Dir einen Python-Scheduler, wann welche Bilder von Dir veröffentlicht werden sollen.
- **REST Countries**: Nutze die Daten, um ein Quiz zu schreiben.
- **Stock Market Apis**: Trading Apps, SMS-Warnung bei einbrechenden Preisen.

Wie macht man einen API-Call?

Im Folgenden gehen wir durch die Schritte, die man ausführen muss, um einen lesenden API-Call auszuführen. Wir verwenden dafür die **REST Countries Api**. Die Dokumentation (*Docs*) der Api findet sich unter `https://restcountries.com/`.

Schritt 1: Finde die richtige URL (d.h. Webadresse)

Zunächst benötigt man eine URL (den sogennante *Endpoint*). Diese kann völlig statisch sein, oftmals aber sind Teile der URL dynamisch bzw. hängen von Deiner genauen Anfrage ab.

Beispiel:

`https://restcountries.com/v3.1/all`

oder

`https://restcountries.com/v3.1/name/france`

Schritt 2: Schicke einen Request an den Endpoint

Verwende folgenden Code, um einen Request zu schicken:

```
import requests
...

response = requests.get("https://restcountries.com/v3.1/name/france")
data = response.json()
# print(data)
```

Schritt 3: Erweitere den Request ggf. um *Query-Parameter*

```
import requests
...

payload = {"fields": ["capital","population","continents"]}
response = requests.get("https://restcountries.com/v3.1/name/france", params=payload)
data = response.json()
# print(data)
```

Schritt 4: Transformiere, filtere und speichere die Daten je nach Bedarf

```
import requests
import json
...

payload = {"fields": ["capital","population","continents"]}
response = requests.get("https://restcountries.com/v3.1/name/france", params=payload)
data = response.json()

with open("france.json","w") as my_file:
  json.dump(data, my_file)
```

Übung

Daten eines Landes extrahieren

Stelle einen Request zu Italien an die Countries-Api. Bereite die zurückgegebenen Daten so auf, dass folgendes auf der Konsole erscheint:

```
Hauptstadt: Rome
Bevölkerung: 59554023
Kontinent: Europe
```

Lösung

Daten eines Landes extrahieren

```python
import requests

response = requests.get("https://restcountries.com/v3.1/name/italy")
data = response.json()

data = data[0]
capital = data["capital"][0]
population = data["population"]
continent = data["continents"][0]

print(f"Hauptstadt: {capital}")
print(f"Bevölkerung: {population}")
print(f"Kontinent: {continent}")
```

Übung

Hauptstadt-Orakel

Schreibe eine Funktion, die den (englischsprachigen) Namen eines Landes erwartet und den Namen der Hauptstadt zurückgibt.

Lösung

Hauptstadt-Orakel

```python
import requests

def get_capital(country):
    response = requests.get("https://restcountries.com/v3.1/name/" + country)
    data = response.json()
    data = data[0]
    capital = data["capital"][0]
    return capital
```

Übung

Datenaufbereitung

Erstelle eine Liste aller Länder. Jedes Land soll dabei als Dictionary der folgenden Form abgespeichert werden:

```
{
  "name": "Italy",
  "capital": "Rome",
  "continent": "Europe",
  "population": 59554023
}
```

Speichere diese Liste als JSON in der Datei `countries.json`.

Lösung

Datenaufbereitung

```python
import requests
import json

response = requests.get("https://restcountries.com/v3.1/all")
data = response.json()

countries = []
for country in data:
    if "capital" in country.keys():
        countries.append({
            "name": country["name"]["common"],
            "capital": country["capital"][0],
            "population": country["population"],
            "continent": country["continents"][0]
        })

with open("countries.json", "w") as my_file:
    json.dump(countries,my_file, indent=4)
```

Projekt: Geografie-Quiz

Das *minimal viable product* (MVP)

Anforderungen:

- Es werden Länder aus einem JSON gelesen.
- Zu einem Land aus der Liste wird die Hauptstadt abgefragt.
- Es erscheinen 4 Lösungsmöglichkeiten (Multiple Choice).
- Durch Eingabe einer Zahl zwischen 1 bis 4 kann getippt werden.
- Nach Eingabe erscheint ein kurzes Feedback (Richtig/Falsch).

Beispielausgabe

```
Was ist die Hauptstadt von Frankreich?
(1) Bratislava
(2) Berlin
(3) Paris
(4) Stockholm


Antwort:
```

Beispielausgabe

```
Was ist die Hauptstadt von Frankreich?
(1) Bratislava
(2) Berlin
(3) Paris
(4) Stockholm


Antwort: 3
```

Beispielausgabe

```
Was ist die Hauptstadt von Frankreich?
(1) Bratislava
(2) Berlin
(3) Paris
(4) Stockholm


Antwort: 3


Das war korrekt!
```

Übung

Multiple Choice Frage

Lies die Datei `"countries.json"` ein. Nimm das erste Land aus der Liste (Afghanistan) und stelle die Frage nach der Hauptstadt mit 4 Antwortmöglichkeiten wie oben auf der Konsole dar.

Tip: Aufgabe 4 vom letzten Übungsblatt kann helfen

Lösung

Multiple Choice Frage

```
import random
import json
with open("countries.json") as file:
    countries = json.load(file)

capitals = []
for country in countries:
    capitals.append(country["capital"])

capital = capitals[0]
random.shuffle(capitals)
answer_options = capitals[0:3]
answer_options.append(capital)
random.shuffle(answer_options)
country = countries[0]

print(f"Was ist die Hauptstadt von { country['name'] }?\n")
for index,option in enumerate(answer_options):
    print(f"({ index + 1 }) { option }")
```

Welche Verbesserungen sind dringend nötig?

Schritte nach Priorität:

1. Antwort einlesen und Feedback geben (Antwort war richtig/falsch)
2. Statt Afghanistan soll ein zufälliges Land abgefragt werden
3. Sicherstellen, dass die Antwort-Optionen stets paarweise unterschiedlich sind

Übungen

Antwort einlesen

Frage eine Antwort ab.

Antwort prüfen

Entscheide, ob die Antwort richtig oder falsch ist, und gib das Ergebnis auf der Konsole aus.

Land zufällig auswählen

Sorge dafür, dass das abgefragte Land zufällig ausgewählt wird.

Lösungen

Antwort einlesen

```
...
for index, option in enumerate(answer_options):
  print(f"({index + 1}) {option}")
answer = input("Antwort: ")
```

Antwort prüfen

```
...
answer = input("\nAntwort: ")
index = int(answer) - 1
if answer_options[index] == country["capital"]:
  print("Das war korrekt.")
else:
  print("Das war leider falsch.")
```

Lösungen

Land zufällig auswählen

```
...
with open("countries.json") as my_file:
  countries = json.load(my_file)
random.shuffle(countries)
country = countries[0]
...
```

Möglicher Bug

Um die Lösungsmöglichkeiten zu generieren, werden 3 Haupstädte zufällig ausgewählt und dann die korrekte Hauptstadt hinzugefügt. Theoeretisch ist es möglich, dass die korrekte Hauptstadt schon bei den 3 zufälligen Haupstädten dabei war.

Ist nur ein kleines Problem

Das ist nur ein kosmetisches Problem. Die Funktionalität geht davon nicht kaputt.

Mögliche Lösung

Mittels `.remove()` kann man die korrekte Hauptstadt aus der Liste `capitals` entfernen.

Bessere Lösung

```
...
for current_country in countries:
  if current_country["capital"] != country["capital"]:
    capitals.append(current_country["capital"])
...
```

Warum ist die zweite Lösung besser?

Übung

Feature Request

Es sollen (zunächst) nur Hauptstädte aus Europa abgefragt werden.

Lösung

Feature Request: Nur europäische Hauptstädte

```
...
with open("countries.json") as my_file:
  countries = json.load(my_file)

filtered_countries = []
for country in countries:
  if country["continent"] == "Europa":
    filtered_countries.append(country)

countries = filtered_countries

random.shuffle(countries)
...
```