Féidearthachtaí as Cuimse
Infinite Possibilities

Programming for Analytics

Lecture 5: Working with External Data and APIs

Bojan Božić
School of Computer Science
TU Dublin, Grangegorman

bojan.bozic@tudublin.ie

OLLSCOIL TEICNEOLAÍOCHTA BHAILE ÁTHA CLIATH
TU DUBLIN
TECHNOLOGICAL UNIVERSITY DUBLIN

Overview

- What is an API?
- Requests
- Using JSON Data
- Parameters
- Error Handling
- Save to File

What is an API?

- API = Application Programmer Interface
- APIs allow programs to communicate over the Internet
- REST APIs are the most common (stateless, use HTTP methods)

Types and API Requests

- GET – retrieve data
- POST – send data
- PUT update existing data
- DELETE – remove data

Activity: Some API Examples

- Examples: OpenWeatherMap, Spotify, Agify, GitHub API
- Look up these websites and try to find information about their APIs and how to access them.

Using the `requests` Library

- Install: `pip install requests`
- Basic use: `requests.get(url)`
- Common properties: `.status_code`, `.text`, `.json()`
- Documentation: https://docs.python-requests.org/en/latest/index.html

Example

```
import requests
url = 
“https://api.agify.io?name=
Anomander”
response = requests.get(url)
```

Activity: GET from GitHub API

- URL: https://api.github.com
- Print status code and body content
- Hint: `status_code` and `content` are variables of the response object

Using JSON Data from APIs

- Most APIs return JSON data
- Use `.json()` to convert to Python dictionary/list
- Access fields with dict-style access

Example

```
response.json()
```

```
{'count': 2, 'name': 'Anomander',
'age': 58}
```

Activity: Extract JSON Field

- API: https://catfact.ninja/fact
- Extract and print the ‘fact’ from the response

API Parameters

- Use `params` to pass query string arguments
- Example: `params={‘name’: ‘Kalam’}`
- Requests builds correct URL from it

Example

```
response = requests.get(url,
params={'name': 'Kalam’})
```

or

```
params = {'name': 'Kalam’}
response = requests.get(url,
params)
```

Activity: Use API with Query Params

- API: https://api.agify.io
- Fetch age prediction based on input name
- HINT: use `input()` to collect name

API Error Handling

- Check `status_code`
- Use `try/except` for request or JSON decode errors
- Avoid crashing your script on network failures

Example

```
if response.status_code == 200:
   …
try:
   response = requests.get(..)
except requests.exceptions.Timeout:
   …
```

Activity: Handle Bad URL

- Try to fetch from a bad URL
- Catch error and print user-friendly message

Headers and Authentication

- Some APIs require API keys via headers
- Example:

```
headers={'Authorization':
'Bearer <token>'}
```

- Don’t hardcode keys; keep them secure

Example

```
url = "https://api.spotify.com/v1/artists/1Xyo4u8uXC1ZmMpatF05PJ"

# Normally you'd obtain this token through OAuth (never hardcode real ones!)
access_token = "YOUR_SPOTIFY_ACCESS_TOKEN"

# Create headers with the token
headers = { "Authorization": f"Bearer {access_token}"}
response = requests.get(url, headers=headers)
```

Activity: Use Public API (No Auth)

- API: https://api.nationalize.io?name=Ben
- Extract and print top 2 country predictions

Working with Nested JSON

- JSON may contain nested dicts/lists
- Use loops and conditionals to navigate structure
- Use `pprint` or `json.dumps(…, indent=2)` for readability

Example

```
response = requests.get(url).json()
for item in response:
  print(item)
```

or (doesn’t work in jupyter)

```
pprint(response)
```

or (also doesn’t really work in jupyter)

```
json.dumps(response, indent=2)
```

Activity: Inspect Nested JSON

- Use the API of Ice and Fire: https://anapioficeandfire.com/api/characters with a name parameter.
- Extract a deeply nested value safely.

Saving API Results to Files

- Use json.dump(data, file) to save JSON
- Log requests or cache results
- Name files based on timestamps or input parameters

Example

```
with open(‘file.json’) as f:
    json.dump(response.json(), f)
```

Activity: Save API Result to File

- Fetch from Agify or Cat Fact API
- Save result to JSON file named after query
- Chain Agify and Nationalize results
- Print custom message based on predictions

OLLSCOIL TEICNEOLAÍOCHTA BHAILE ÁTHA CLIATH
DUBLIN
TECHNOLOGICAL UNIVERSITY DUBLIN

Rate Limits and Ethics

- APIs have usage limits – avoid abuse
- Read API documentation and Terms of Service
- Add delay with `time.sleep()` between requests

Activity: Delay Between Requests

- Use loop to send 3 requests with 1 sec delay
- Use `time.sleep(1)`

Function-based API Calls

- Encapsulate API calls in functions
- E.g.

```
def get_age(name):
   return agify response
```

- Easier to reuse and test

Activity: Wrap API in Function

- Write `get_nationality(name)` function
- Use in a loop for 3 different names

Final Practical Challenge

- Ask user for a list of names
- Fetch age and nationality prediction for each
- Store result in structured JSON file
- Handle errors and include timestamp in filename

Next Week

- You’ve made your first real API integration
- Next: Data analysis using pandas and real datasets

Questions?

OLLSCOIL TEICNEOLAÍOCHTA BHAILE ÁTHA CLIATH
DUBLIN
TECHNOLOGICAL UNIVERSITY DUBLIN