Working with APIs in Python

APIs (Application Programming Interfaces) allow you to interact with web services programmatically. In Python, the **`requests`** library is commonly used to send HTTP requests to APIs and process the responses.

---

1. What is an API?

An API provides a way for applications to communicate with each other. APIs typically return data in formats like **JSON** or **XML**.

---

2. Installing Required Libraries

Ensure you have the `requests` library installed:

bash
Copy code
```
pip install requests
```

---

3. Basic Steps to Work with an API

1. **Find the API endpoint**: URL to access the API.
2. **Understand the request method**: Common methods are:
    - `GET`: Retrieve data.
    - `POST`: Send data.
    - `PUT`: Update data.
    - `DELETE`: Remove data.
3. **Send a request to the API**.
4. **Parse the response data**.

---

4. Example: Working with a Public API

API Endpoint:

We'll use the JSONPlaceholder API, a free service for testing and prototyping.

Making a GET Request
python
Copy code

```python
import requests

# API endpoint
url = "https://jsonplaceholder.typicode.com/posts"

# Send a GET request
response = requests.get(url)

# Check the response status
if response.status_code == 200:
    print("Request Successful!")
else:
    print(f"Failed with status code: {response.status_code}")

# Parse the JSON response
data = response.json()
print(data[:5])  # Print first 5 posts
```

---

5. Making a POST Request

python
Copy code

```python
# API endpoint
url = "https://jsonplaceholder.typicode.com/posts"

# Data to send
payload = {
    "title": "foo",
    "body": "bar",
    "userId": 1
}

# Send a POST request
response = requests.post(url, json=payload)

# Check the response
print(response.status_code)  # Status code (201 for successful POST)
print(response.json())  # Response JSON
```

---

6. Using Query Parameters

Some APIs allow filtering data using query parameters.

python
Copy code
```
# API endpoint with query parameters
url = "https://jsonplaceholder.typicode.com/comments"
params = {'postId': 1}

# Send a GET request with query parameters
response = requests.get(url, params=params)

# Parse the response
data = response.json()
print(data[:5])  # First 5 comments for postId=1
```

---

7. Working with Headers

APIs often require headers for authentication or content type.

python
Copy code
```
url = "https://jsonplaceholder.typicode.com/posts"
headers = {
    "Content-Type": "application/json",
    "Authorization": "Bearer YOUR_API_TOKEN"  # Replace with your
token
}

response = requests.get(url, headers=headers)
print(response.status_code)
print(response.json())
```

---

8. Handling Errors

Handle potential issues like timeouts or invalid responses.

python
Copy code
```
try:
```

```
    response =
requests.get("https://jsonplaceholder.typicode.com/posts",
timeout=5)
    response.raise_for_status()  # Raise HTTPError for bad responses
    print(response.json())
except requests.exceptions.HTTPError as errh:
    print("HTTP Error:", errh)
except requests.exceptions.ConnectionError as errc:
    print("Error Connecting:", errc)
except requests.exceptions.Timeout as errt:
    print("Timeout Error:", errt)
except requests.exceptions.RequestException as err:
    print("OOps: Something Else", err)
```

---

9. Authentication

Some APIs require authentication via API keys, OAuth, or tokens.

Example: API Key Authentication
python
Copy code

```
url = "https://api.example.com/data"
api_key = "YOUR_API_KEY"

headers = {
    "Authorization": f"Bearer {api_key}"
}

response = requests.get(url, headers=headers)
print(response.status_code)
print(response.json())
```

---

10. Pagination

APIs often paginate large datasets. Handle pagination using loops.

python
Copy code

```
url = "https://jsonplaceholder.typicode.com/posts"
all_data = []
```

```
page = 1

while True:
    params = {"_page": page, "_limit": 10}  # Limit 10 results per
page
    response = requests.get(url, params=params)
    data = response.json()
    if not data:  # Stop if no more data
        break
    all_data.extend(data)
    page += 1

print(f"Total Posts Fetched: {len(all_data)}")
```

---

11. Saving API Data

Save the fetched data into a file (JSON, CSV, etc.).

Save as JSON

python
Copy code

```
import json

# Save data to a JSON file
with open('data.json', 'w') as file:
    json.dump(all_data, file, indent=4)
```

Save as CSV

python
Copy code

```
import csv

# Save data to a CSV file
with open('data.csv', mode='w', newline='', encoding='utf-8') as
file:
    writer = csv.DictWriter(file, fieldnames=['userId', 'id',
'title', 'body'])
    writer.writeheader()
    writer.writerows(all_data)
```

---

12. Advanced Topics

Rate Limiting: Respect the API's rate limits to avoid being blocked.
python
Copy code

```
import time
time.sleep(1)  # Wait 1 second between requests
```

-
- **Working with APIs requiring OAuth**: Use libraries like `oauthlib` or `requests-oauthlib`.
- **GraphQL APIs**: Use libraries like `gql` for GraphQL endpoints.