Introduction to APIs in Python

HKBU FIN7830 2025

Connect, retrieve, and send data like a professional developer.

Shengwei You

🌍 What Are APIs?

APIs = **Application Programming Interfaces**
They let programs **communicate** with other systems.

Examples:

- Fetch weather data 🌦️
- Generate text with GPT 🤖
- Send emails via SendGrid 📧

🔗 HTTP and Requests

APIs use **HTTP**, the language of the web.

- **GET** → retrieve data (read)
- **POST** → send data (write)

```
GET https://api.example.com/users
POST https://api.example.com/messages
```

💡 Think of GET as "reading" and POST as "writing".

🌐 API Endpoints

An endpoint is a URL that performs one task.

Example:

```
https://api.openai.com/v1/chat/completions
```

Each endpoint = a specific action or resource.

💡 The Rise of APIs

APIs power most modern platforms:

- 🧠 OpenAI → AI models
- ✉️ SendGrid → Email delivery
- 💸 Stripe → Payments
- 🗺️ Google Maps → Location data

They’re the backbone of modern software.

🔤 JSON Basics

APIs communicate with JSON (JavaScript Object Notation).

```
{
  "name": "Alice",
  "age": 25,
  "isStudent": false
}
```

✅ Simple, lightweight, and universal.

🔧 Why JSON?

- Easy for humans to read
- Easy for programs to parse
- Built-in Python support via json module

Most API data is sent and received as JSON.

🔐 API Keys

APIs require authentication using a key.

Keys:

- Identify who you are
- Control access and usage
- Enable billing or limits

💾 Storing Keys Securely

Never hard-code your keys.

Instead, save them in a `.env` file:

```
OPENAI_API_KEY=sk-abc123...
```

Load it in Python:

```
from dotenv import load_dotenv
import os

load_dotenv()
api_key = os.getenv("OPENAI_API_KEY")
```

📦 Client Libraries

A client library is a package that simplifies API use.

Without it:

```
import requests
requests.post("https://api.example.com", headers={...}, json={...})
```

With it:

```
from openai import OpenAI
client = OpenAI(api_key)
```

Cleaner. Safer. Easier.

🧭 Steps to Use Any API

1. Sign up on the provider's site
2. Read documentation → endpoints
3. Get your API key
4. Add it to `.env`
5. Install libraries with uv

```
uv add openai python-dotenv
```

6. Write Python code → make requests
7. Handle responses (usually JSON)

🤖 Example: OpenAI API

Step 1: Install

```
uv add openai python-dotenv
```

Step 2: Add Your Key

`.env` file:

```
OPENAI_API_KEY=sk-abc123...
```

Step 3: Use in Code

```python
from openai import OpenAI
from dotenv import load_dotenv
import os

load_dotenv()
openai = OpenAI(api_key=os.getenv("OPENAI_API_KEY"))

response = openai.chat.completions.create(
  model="gpt-3.5-turbo",
  messages=[
    {"role": "system", "content": "You are a helpful assistant."},
    {"role": "user", "content": "What's the capital of France?"}
  ]
)

print(response.choices[0].message.content)
```

Output → Paris 🇫🇷

✉️ Example: Send Email with SendGrid

Step 1: Setup

Sign up → get API key → enable "Mail Send".

Save in `.env`:

```
SENDGRID_API_KEY=SG.abc123...
```

Step 2: Install

```
uv add sendgrid python-dotenv
```

Step 3: Send an Email

```
from sendgrid import SendGridAPIClient
from sendgrid.helpers.mail import Mail
from dotenv import load_dotenv
import os

load_dotenv()

message = Mail(
  from_email='you@example.com',
  to_emails='student@example.com',
  subject='Hello from Python!',
  plain_text_content='This is a test email.'
)

try:
  sg = SendGridAPIClient(os.getenv("SENDGRID_API_KEY"))
  response = sg.send(message)
  print(f"Status: {response.status_code}")
except Exception as e:
  print(f"Error: {e}")
```

🧠 Summary

Concept	What It Means
API	Interface to external services
Endpoint	URL for a specific action
JSON	Data format for communication
API Key	Authentication credential
Client Library	Python wrapper for API calls
Steps	Sign up → Get key → Install → Use

💪 Practice Exercises

1. Make a GET request to a public API (e.g., https://api.github.com)

```
import requests
response = requests.get("https://api.github.com")
print(response.json())
```

2. Save and load an API key using `.env`

```
from dotenv import load_dotenv
import os

load_dotenv()
api_key = os.getenv("MY_API_KEY")
print(f"Loaded key: {api_key}")
```

3. Use requests to call an endpoint manually

```
import requests

url = "https://api.example.com/data"
headers = {"Authorization": "Bearer YOUR_KEY"}
response = requests.get(url, headers=headers)
print(response.json())
```

4. Install openai and test a simple chat call

```
uv add openai python-dotenv
```

```
from openai import OpenAI
import os

client = OpenAI(api_key=os.getenv("OPENAI_API_KEY"))
response = client.chat.completions.create(
    model="gpt-3.5-turbo",
    messages=[{"role": "user", "content": "Hello!"}]
)
print(response.choices[0].message.content)
```

5. Try sending a test email with SendGrid

```
from sendgrid import SendGridAPIClient
from sendgrid.helpers.mail import Mail
import os

message = Mail(
    from_email='sender@example.com',
    to_emails='recipient@example.com',
    subject='Test Email',
    plain_text_content='Hello from Python!'
)

sg = SendGridAPIClient(os.getenv("SENDGRID_API_KEY"))
response = sg.send(message)
print(f"Email sent! Status: {response.status_code}")
```

🎓 You’re API-Ready!

Explore. Connect. Build.

Bring your Python projects to life with real data 🌐✨