**Python API Learning Guide**

Python API learning material with code examples

Here’s a guide to learning how to work with APIs in Python, along with code examples to help you get started.  
  
---  
  
## \*\*What is an API?\*\*  
An API (Application Programming Interface) allows two applications to communicate with each other. In Python, APIs are often used to interact with web services, databases, or other software.  
  
---  
  
## \*\*Steps to Learn Python API Development\*\*  
  
### 1. \*\*Understand HTTP Methods\*\*  
APIs often use HTTP methods to perform actions:  
- `GET`: Retrieve data  
- `POST`: Send data  
- `PUT`: Update data  
- `DELETE`: Remove data  
  
### 2. \*\*Install Required Libraries\*\*  
Python provides libraries like `requests` for consuming APIs and `Flask` or `FastAPI` for building APIs.  
  
Install them using pip:  
```bash  
pip install requests flask fastapi uvicorn  
```  
  
---  
  
## \*\*Consuming APIs with Python\*\*  
  
### Example 1: Using the `requests` Library  
The `requests` library is commonly used to interact with APIs.  
  
#### Code Example: Fetching Data from a Public API  
```python  
import requests  
  
# Example: Fetching data from a public API  
url = "https://jsonplaceholder.typicode.com/posts"  
response = requests.get(url)  
  
if response.status\_code == 200:  
 data = response.json() # Parse JSON response  
 for post in data[:5]: # Print the first 5 posts  
 print(f"Title: {post['title']}\nBody: {post['body']}\n")  
else:  
 print(f"Failed to fetch data. Status code: {response.status\_code}")  
```  
  
---  
  
### Example 2: Sending Data to an API  
You can use the `POST` method to send data to an API.  
  
#### Code Example: Sending Data  
```python  
import requests  
  
url = "https://jsonplaceholder.typicode.com/posts"  
data = {  
 "title": "My New Post",  
 "body": "This is the content of the post.",  
 "userId": 1  
}  
  
response = requests.post(url, json=data)  
  
if response.status\_code == 201: # 201 means resource created  
 print("Post created successfully!")  
 print(response.json())  
else:  
 print(f"Failed to create post. Status code: {response.status\_code}")  
```  
  
---  
  
### Example 3: Handling API Authentication  
Some APIs require authentication, such as API keys or tokens.  
  
#### Code Example: Using an API Key  
```python  
import requests  
  
url = "https://api.openweathermap.org/data/2.5/weather"  
params = {  
 "q": "London",  
 "appid": "your\_api\_key\_here" # Replace with your API key  
}  
  
response = requests.get(url, params=params)  
  
if response.status\_code == 200:  
 weather\_data = response.json()  
 print(f"Weather in {weather\_data['name']}: {weather\_data['weather'][0]['description']}")  
else:  
 print(f"Failed to fetch weather data. Status code: {response.status\_code}")  
```  
  
---  
  
## \*\*Building APIs with Python\*\*  
  
### Example 1: Using Flask to Create a Simple API  
Flask is a lightweight framework for building APIs.  
  
#### Code Example: Simple Flask API  
```python  
from flask import Flask, jsonify, request  
  
app = Flask(\_\_name\_\_)  
  
# Sample data  
posts = [  
 {"id": 1, "title": "First Post", "content": "This is the first post."},  
 {"id": 2, "title": "Second Post", "content": "This is the second post."}  
]  
  
# Route to get all posts  
@app.route('/posts', methods=['GET'])  
def get\_posts():  
 return jsonify(posts)  
  
# Route to add a new post  
@app.route('/posts', methods=['POST'])  
def add\_post():  
 new\_post = request.json  
 posts.append(new\_post)  
 return jsonify(new\_post), 201  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app.run(debug=True)  
```  
  
Run the script and access the API at `http://127.0.0.1:5000/posts`.  
  
---  
  
### Example 2: Using FastAPI for Modern API Development  
FastAPI is a modern framework for building APIs with automatic documentation.  
  
#### Code Example: Simple FastAPI Application  
```python  
from fastapi import FastAPI  
from pydantic import BaseModel  
  
app = FastAPI()  
  
# Sample data  
posts = [  
 {"id": 1, "title": "First Post", "content": "This is the first post."},  
 {"id": 2, "title": "Second Post", "content": "This is the second post."}  
]  
  
# Pydantic model for validation  
class Post(BaseModel):  
 id: int  
 title: str  
 content: str  
  
# Route to get all posts  
@app.get("/posts")  
def get\_posts():  
 return posts  
  
# Route to add a new post  
@app.post("/posts")  
def add\_post(post: Post):  
 posts.append(post.dict())  
 return post  
```  
  
Run the app using:  
```bash  
uvicorn filename:app --reload  
```  
  
Access the API at `http://127.0.0.1:8000` and view the documentation at `http://127.0.0.1:8000/docs`.  
  
---  
  
## \*\*Additional Resources\*\*  
- [Requests Documentation](https://docs.python-requests.org/)  
- [Flask Documentation](https://flask.palletsprojects.com/)  
- [FastAPI Documentation](https://fastapi.tiangolo.com/)  
- [Postman](https://www.postman.com/) for testing APIs  
  
---