**Flask API Multiple Routes**

Python flask api code examples with multiple routing

Here’s an example of a Python Flask API with multiple routes. This example demonstrates how to create a simple Flask application with different endpoints for handling various HTTP methods and paths.  
  
### Flask API with Multiple Routes  
  
```python  
from flask import Flask, jsonify, request  
  
app = Flask(\_\_name\_\_)  
  
# Sample data to simulate a database  
users = [  
 {"id": 1, "name": "Alice", "age": 25},  
 {"id": 2, "name": "Bob", "age": 30},  
 {"id": 3, "name": "Charlie", "age": 35}  
]  
  
# Home route  
@app.route('/')  
def home():  
 return "Welcome to the Flask API!"  
  
# Route to get all users  
@app.route('/users', methods=['GET'])  
def get\_users():  
 return jsonify(users)  
  
# Route to get a single user by ID  
@app.route('/users/<int:user\_id>', methods=['GET'])  
def get\_user(user\_id):  
 user = next((user for user in users if user["id"] == user\_id), None)  
 if user:  
 return jsonify(user)  
 else:  
 return jsonify({"error": "User not found"}), 404  
  
# Route to create a new user  
@app.route('/users', methods=['POST'])  
def create\_user():  
 new\_user = request.get\_json()  
 if "name" in new\_user and "age" in new\_user:  
 new\_user["id"] = len(users) + 1  
 users.append(new\_user)  
 return jsonify(new\_user), 201  
 else:  
 return jsonify({"error": "Invalid data"}), 400  
  
# Route to update an existing user  
@app.route('/users/<int:user\_id>', methods=['PUT'])  
def update\_user(user\_id):  
 user = next((user for user in users if user["id"] == user\_id), None)  
 if user:  
 updated\_data = request.get\_json()  
 user.update(updated\_data)  
 return jsonify(user)  
 else:  
 return jsonify({"error": "User not found"}), 404  
  
# Route to delete a user  
@app.route('/users/<int:user\_id>', methods=['DELETE'])  
def delete\_user(user\_id):  
 global users  
 users = [user for user in users if user["id"] != user\_id]  
 return jsonify({"message": "User deleted"}), 200  
  
# Route with query parameters  
@app.route('/search', methods=['GET'])  
def search\_users():  
 name\_query = request.args.get('name')  
 if name\_query:  
 filtered\_users = [user for user in users if name\_query.lower() in user["name"].lower()]  
 return jsonify(filtered\_users)  
 else:  
 return jsonify({"error": "No query parameter provided"}), 400  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 app.run(debug=True)  
```  
  
### Explanation of Routes:  
1. \*\*`/`\*\*: A simple home route that returns a welcome message.  
2. \*\*`/users` (GET)\*\*: Returns a list of all users.  
3. \*\*`/users/<int:user\_id>` (GET)\*\*: Returns a specific user by their ID.  
4. \*\*`/users` (POST)\*\*: Creates a new user. Expects JSON data in the request body.  
5. \*\*`/users/<int:user\_id>` (PUT)\*\*: Updates an existing user by their ID. Expects JSON data in the request body.  
6. \*\*`/users/<int:user\_id>` (DELETE)\*\*: Deletes a user by their ID.  
7. \*\*`/search` (GET)\*\*: Searches for users by name using query parameters (e.g., `/search?name=Alice`).  
  
### How to Run:  
1. Save the code in a file, e.g., `app.py`.  
2. Install Flask if you haven’t already:  
 ```bash  
 pip install flask  
 ```  
3. Run the application:  
 ```bash  
 python app.py  
 ```  
4. Open your browser or use a tool like Postman to test the endpoints at `http://127.0.0.1:5000`.  
  
### Example Requests:  
- \*\*Get all users\*\*: `GET /users`  
- \*\*Get a user by ID\*\*: `GET /users/1`  
- \*\*Create a user\*\*: `POST /users` with JSON body:  
 ```json  
 {  
 "name": "David",  
 "age": 40  
 }  
 ```  
- \*\*Update a user\*\*: `PUT /users/1` with JSON body:  
 ```json  
 {  
 "age": 26  
 }  
 ```  
- \*\*Delete a user\*\*: `DELETE /users/1`  
- \*\*Search users\*\*: `GET /search?name=Alice`