# 

# 

# Section 2: Building a Simple API with Flask

### **Section 2: Building a Simple API with Flask**

* **Introduction to Flask:**
  + What is Flask?
  + Creating a basic Flask app
  + Routing and request handling
  + JSON responses
* **Example:**
  + Create a simple API endpoint to return a list of users.

### **Introduction to Flask:**

**What is Flask?**

Flask is a lightweight Python web framework that allows you to build web applications quickly and easily. It's known for its simplicity and flexibility, making it a great choice for beginners and experienced developers alike.

**Creating a basic Flask app**

To create a Flask app, you'll need Python installed on your system. You can download it from<https://www.python.org/>.

**Software to download:**

* **Python:** Essential for running Flask applications.
* **Flask:** The web framework itself.

**Code:**

**from flask import Flask**

**app = Flask(\_\_name\_\_)**

**@app.route('/')**

**def hello\_world():**

**return 'Hello, World!'**

**if \_\_name\_\_ == '\_\_main\_\_':**

**app.run(debug=True)**

**Explanation:**

1. Import the Flask class from the flask module.
2. Create a Flask application instance named app.
3. Define a route '/' using the @app.route decorator. This route will handle requests to the root URL of your application.
4. The hello\_world() function is the view function that will be executed when a request is made to the / route. It returns the string "Hello, World!".
5. The if \_\_name\_\_ == '\_\_main\_\_': block ensures that the app only runs if the script is executed directly, not when imported as a module.
6. The app.run(debug=True) line starts the Flask development server with debug mode enabled. This allows for automatic reloading of the app when changes are made and provides helpful error messages.

**Routing and request handling**

Flask uses decorators to define routes and their corresponding view functions. Routes specify the URL patterns that your app can handle. View functions are Python functions that handle the request and return a response.

**JSON responses**

Flask can easily return JSON responses using the jsonify function from the flask module.**Example: Creating a simple API endpoint to return a list of users**

**from flask import Flask, jsonify**

**app = Flask(\_\_name\_\_)**

**users = [**

**{'id': 1, 'name': 'Alice'},**

**{'id': 2, 'name': 'Bob'},**

**{'id': 3, 'name': 'Charlie'}**

**]**

**@app.route('/users')**

**def get\_users():**

**return jsonify(users)**

**if \_\_name\_\_ == '\_\_main\_\_':**

**app.run(debug=True)**

**Explanation:**

1. Import the jsonify function from the flask module.
2. Create a list of users.
3. Define a route /users to handle requests for the list of users.
4. The get\_users() function returns the users list as a JSON response using jsonify.

**Running the app:**

1. Save the code as a Python file (e.g., app.py).
2. Open a terminal or command prompt and navigate to the directory where the file is saved.
3. Run the following command: python app.py

This will start the Flask development server, and you can access the API endpoint at http://localhost:5000/users. The response will be a JSON list of users.