What is flask?

**Flask** is a lightweight and versatile web framework for Python, renowned for its simplicity and flexibility. It's an ideal choice for creating APIs due to its minimalistic design and ease of use. With Flask, developers can quickly build powerful web applications and APIs with minimal boilerplate code.

# Creating a simple **api** in **python** using **flask**

1. **Setting up your environment**

* Ensure you have Python installed on your system. We recommend using Python 3 and above.
* Use pip, the Python package manager, to install Flask: pip install Flask.

1. **Creating your flask app**

* Create a new Python file for your Flask application, e.g., app.py.
* Import Flask and create an instance of the Flask class:

| from flask import Flask app = Flask(\_\_name\_\_) |
| --- |

1. **Define your endpoints**

* Define route functions to handle different endpoints. Routes are defined using the @app.route decorator.

| @app.route('/') def home():  return '<h1>Welcome to our app</h1>' |
| --- |

1. **Define your api endpoints**

* Now that we have learned how to define endpoints, let’s start defining endpoints for our api. A quick thing to note is that usually in API the data that you get in response is in json format, which contains key value pairs just like a dictionary in python.

| from flask import jsonify  @app.route('/api/json') def api\_json():  data = {'message': 'Hello, World!'}  return jsonify(data) |
| --- |

* We’ve used flask’s jsonify to convert our data into json format.

1. **Handling HTTP methods**

* We can also set which http methods are allowed for which endpoints. By default, Flask routes handle GET requests.

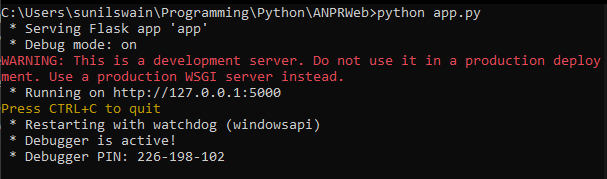
| from flask import jsonify, request  @app.route('/api/data', methods=['GET', 'POST']) def api\_data():  if request.method == 'GET':  data = {'message': 'Used GET method'}    elif request.method == 'POST':  data = {'message': 'Used POST method'}  return jsonify(data) |
| --- |

1. **Running your application**

* Add the following code at the end of your Python file to run the Flask application.

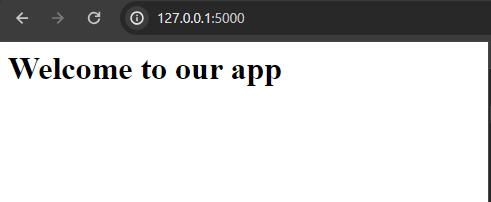
| if \_\_name\_\_ == '\_\_main\_\_':  app.run(debug=True) |
| --- |

* Once your python file is ready, open cmd and run python app.py, this will start your flask app to listen for incoming http requests.

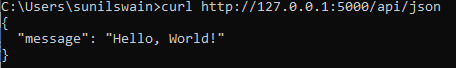
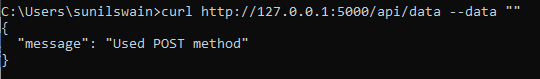


1. **Testing your application**

* Now that your app is up and running, you can grab the url from cmd. In my case it's running on <http://127.0.0.1:5000>. You can access your API endpoints using a web browser, command-line tools like cURL, or API testing tools like Postman.
* Using Web browser



* Using cURL

1. 
2.  We’ve added empty data to make a POST request using cURL, as you can see it returns a response that we have specified for only post requests.

## Conclusion

In conclusion, Flask offers a lightweight and straightforward approach to building web applications and APIs in Python. Its simplicity and extensive documentation enable rapid development, making it a preferred choice for projects of various sizes and complexities