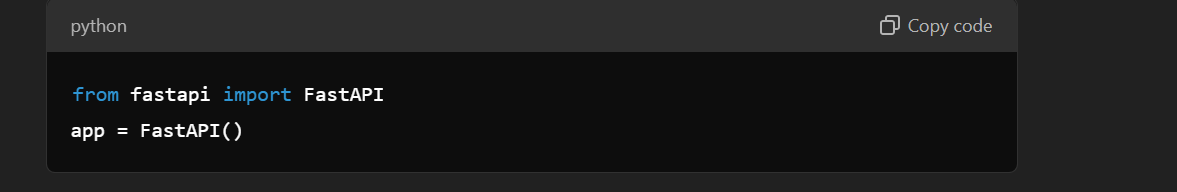
FastAPI is a modern, high-performance web framework for building APIs with Python. It’s designed to make it easy to develop APIs quickly and with minimal code, while also achieving excellent speed and efficiency. FastAPI is particularly popular for building RESTful APIs and web applications that require real-time data handling, like machine learning models or data-intensive apps.

You're absolutely right! In FastAPI, we use \*\*decorators\*\* to define a \*\*URL (or "path")\*\* that the API will respond to, and then we write a \*\*function\*\* to handle requests sent to that URL. Let's break it down into simple steps to understand how it works.

### Step-by-Step Breakdown

1. \*\*Creating a FastAPI App\*\*

- First, you start by creating an instance of the `FastAPI` class, which is the main app where all your routes (URLs) will live.



2. \*\*Using a Decorator to Define a Route (URL)\*\*

- A \*\*decorator\*\* in FastAPI, such as `@app.get("/some-path")`, is used to tell FastAPI which URL (path) should trigger a specific function.

- Here’s an example decorator with `@app.get("/hello")`, meaning this function will respond when a user visits `/hello`.

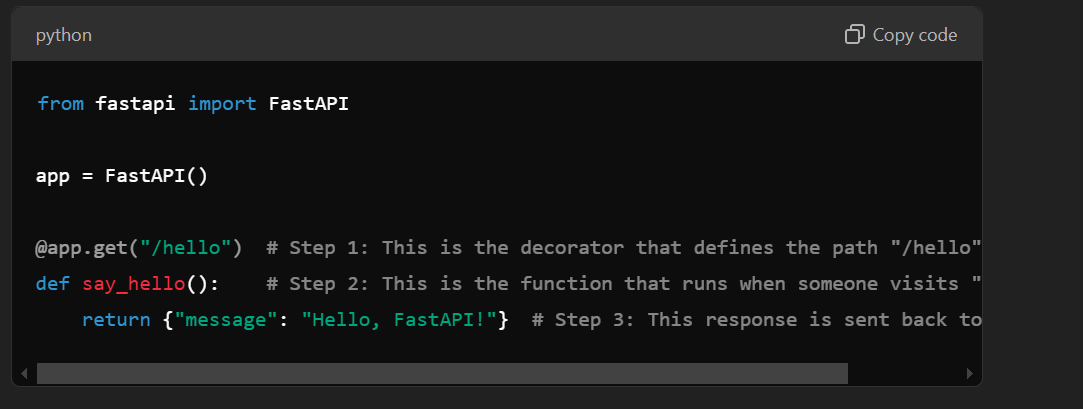
3. \*\*Writing the Function that Handles the Request\*\*

- After defining the URL path with the decorator, you write the \*\*function\*\* that FastAPI will run when someone accesses that path.

- The function’s \*\*return value\*\* is the \*\*response\*\* that FastAPI sends back to the user.

4. \*\*Combining the Decorator and Function\*\*

- Let’s put this all together in an example. This example creates a simple API with a single endpoint, `/hello`, that responds with a message:

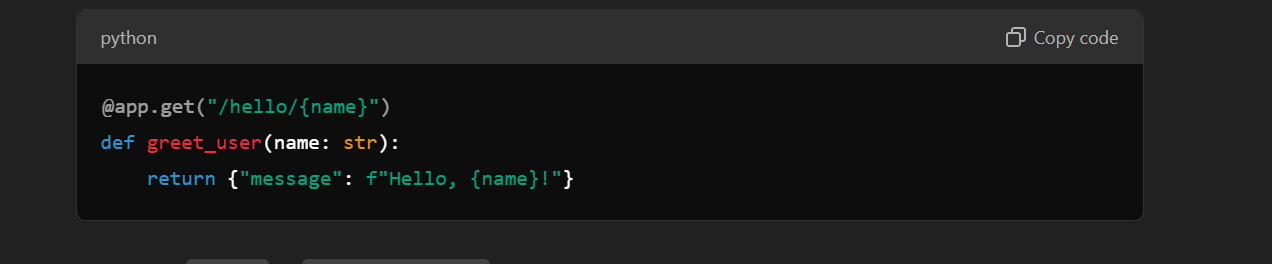


- \*\*Explanation\*\*:

- When someone visits `http://localhost:8000/hello`, FastAPI will run `say\_hello()` and return `{"message": "Hello, FastAPI!"}` as the response.

### Expanding with Path Parameters

You can add \*\*parameters\*\* to the path, which allows you to create more dynamic routes. For example:

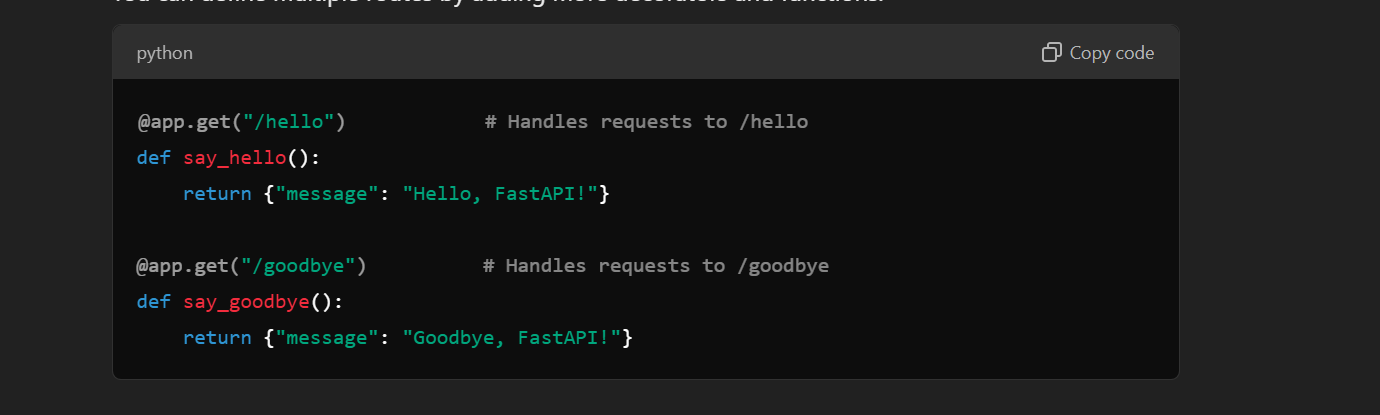


- Here, `{name}` in `"/hello/{name}"` is a \*\*placeholder\*\* for a path parameter.

- If you visit `/hello/Alice`, FastAPI will run `greet\_user(name="Alice")`, and the response will be `{"message": "Hello, Alice!"}`.

### Adding Multiple Endpoints

You can define multiple routes by adding more decorators and functions:



- This code will have two endpoints: `/hello` and `/goodbye`, each with its own response.

### Recap

- \*\*Decorator\*\* (`@app.get("/path")`): Defines the URL path that FastAPI should respond to.

- \*\*Function\*\*: Executes when someone accesses that path and returns a response.

- \*\*Path Parameters\*\* (`/path/{parameter}`): Make routes dynamic by capturing variables in the URL.

Let me know if this clarifies it or if you have specific parts you'd like to dive deeper into!