## FastAPI – Recommended for modern AI agents

**Pros:**

- **Asynchronous support** out of the box (great for concurrent AI tasks, I/O-bound ops).
- Built-in **data validation** with Pydantic (great for AI input/output schemas).
- **Auto-generated OpenAPI docs** (useful for agent API introspection and integration).
- **High performance** (based on Starlette and uvicorn, close to Node.js and Go).
- Great for **serving ML models**, especially with batch prediction and streaming.

**Use FastAPI if**:

- You want **performance + async support**.
- Your AI agent uses **APIs heavily**, needs **webhooks**, or exposes an endpoint.
- You want modern tooling and developer experience.

# Step-by-Step Plan to Learn FastAPI

### Step 1:-

## Create a Virtual Environment (Recommended)

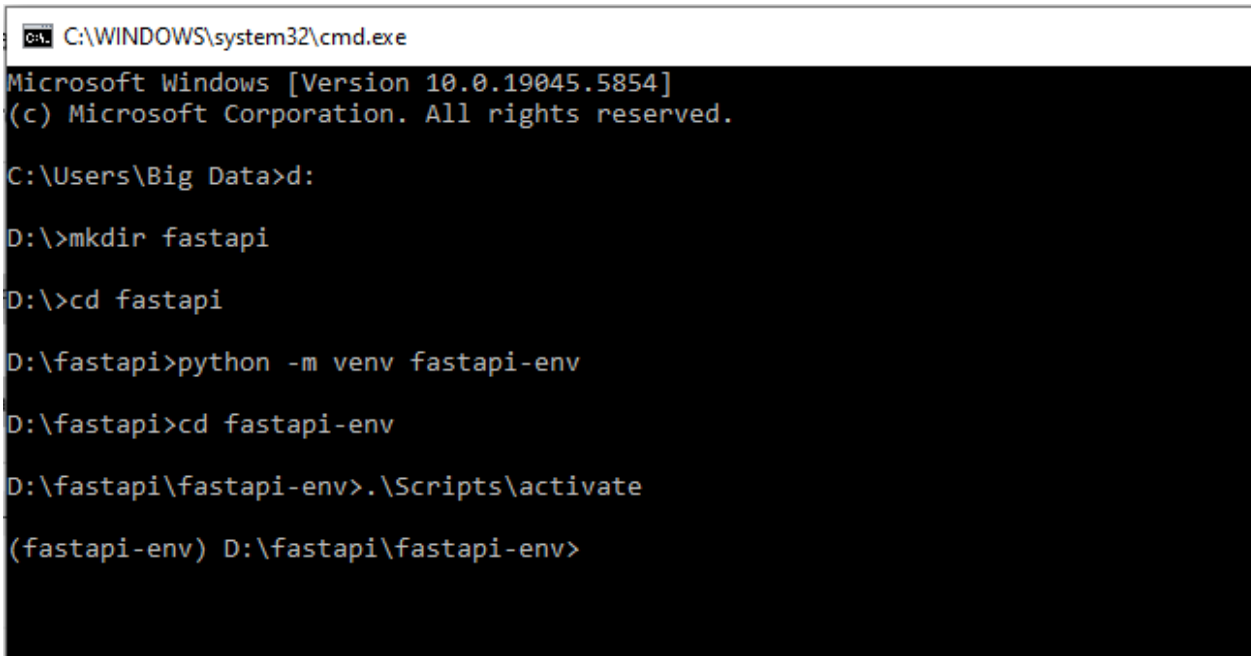
This isolates your FastAPI project dependencies.

Open **Command Prompt** or **PowerShell**, then run:

```
python -m venv fastapi-env
cd fastapi-env
.\Scripts\activate
```

You'll see `(fastapi-env)` at the start of your prompt, meaning it's activated.

Below are example we have created a folder fastapi and inside fastapi we have created virtual environment




## 3. Install FastAPI and Uvicorn

FastAPI is just the framework; you need a server like **Uvicorn** to run it.

```
pip install fastapi uvicorn
```

## 4. Verify Installation

Run a simple FastAPI app.

Create a file called `main.py`:

```python
from fastapi import FastAPI

app = FastAPI()

@app.get("/")
def read_root():
    return {"message": "Hello FastAPI"}
```

Then run:

```
uvicorn main:app –reload
```

```
Visit http://127.0.0.1:8000 in your browser.
```

Here's a basic **FastAPI example** that serves a **static HTML page** to display

```
fastapi/
│
├── main.py
├── static/
│   ├── style.css
│   └── script.js
└── templates/
    └── index.html
```

## Install Jinja2:-

Run this in your terminal(command prompt) (inside your virtual environment, if you're using one):

```
pip install jinja2
```

### templates/base.html file code:-

```
<!DOCTYPE html>
<html>
<head>
    <title>{% block title %}FastAPI Navigation{% endblock %}</title>
    <link rel="stylesheet" href="/static/style.css">
</head>
<body>
    <nav>
        <a href="/">Home</a>
        <a href="/about">About</a>
    </nav>
    <hr>
    <div class="content">
        {% block content %}{% endblock %}
    </div>
</body>
</html>
```

## templates/index.html file code:-

```
<!DOCTYPE html>
<html>
<head>
  <title>Visual Component</title>
  <!-- Link to external CSS -->
  <link rel="stylesheet" href="/static/style.css">
</head>
<body>
<nav>
        <a href="/">Home</a>
        <a href="/about">About</a>
    </nav>
  <h1>Hello from FastAPI with JS and CSS</h1>
  <button onclick="disp()">click to display</button>

  <div id="output">Waiting for JS...</div>

  <!-- Link to external JavaScript -->
  <script src="/static/script.js"></script>
</body>
</html>
```

## templates/about.html file code:-

```
{% extends "base.html" %}

{% block title %}About{% endblock %}

{% block content %}
<h1>About This App</h1>
<p>This is a sample app to demonstrate FastAPI with Jinja2 and navigation.</p>
{% endblock %}
```

## static/script.js file code:-

```
function disp(){
  document.getElementById('output').innerText = "JS is working!";
}
```

## static/style.css file code:-

```css
body {
  background-color: #f0f0f0;
  font-family: Arial, sans-serif;
}

h1 {
  color: #007BFF;
}

#output {
  padding: 10px;
  color: green;
}
```

**`main.py`**

Make sure you're **mounting the `static/` folder** correctly:

```python
from fastapi import FastAPI, Request
from fastapi.responses import HTMLResponse
from fastapi.staticfiles import StaticFiles
from fastapi.templating import Jinja2Templates

app = FastAPI()

app.mount("/static", StaticFiles(directory="static"), name="static")
templates = Jinja2Templates(directory="templates")

@app.get("/", response_class=HTMLResponse)
def home(request: Request):
    return templates.TemplateResponse("index.html", {"request": request})

@app.get("/about", response_class=HTMLResponse)
def read_about(request: Request):
    return templates.TemplateResponse("about.html", {"request": request})
```

**Run the app:**

```
uvicorn main:app –reload
```


After click on about