

## DEVOPS PRACTICAL 2

### Building a REST API with Flask or FastAPI

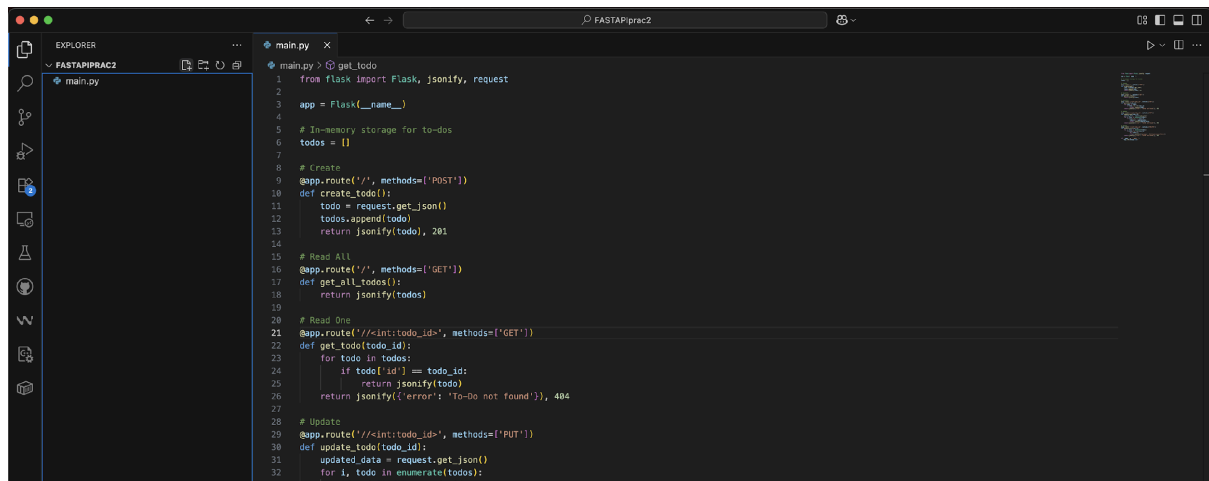
**A] Create a simple REST API using Flask or FastAPI with CRUD (Create, Read, Update and Delete) operations for a "To-Do List" application.**

=>

- 1] Create a new file.
- 2] Install Flask from your terminal.

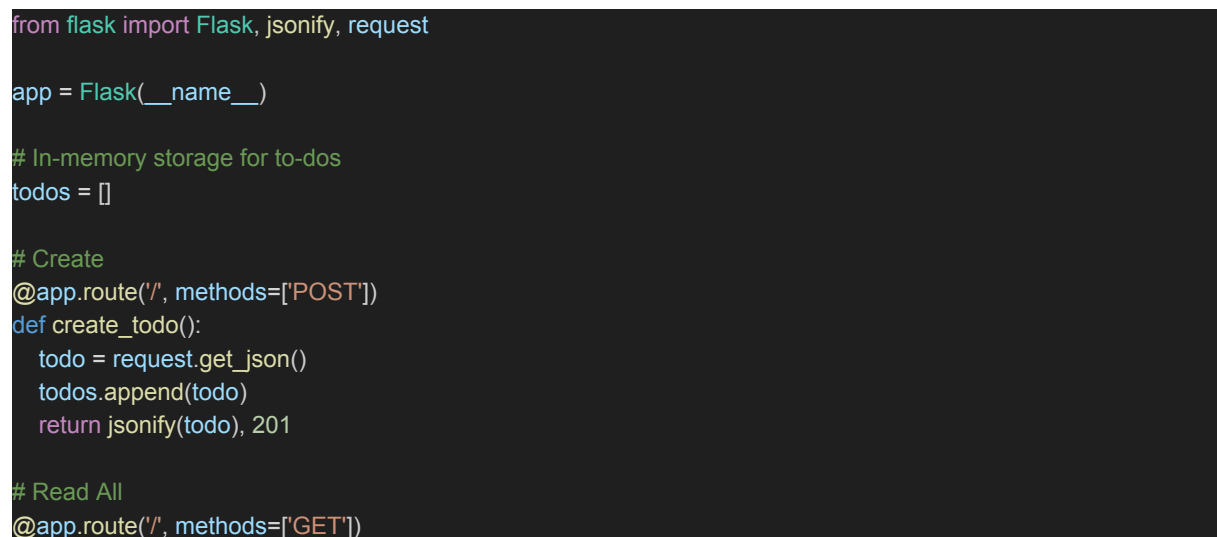
CODE:

pip install flask (if you are on windows or linux).

A screenshot of a code editor window titled 'FASTAPIprac2'. The left sidebar shows a file explorer with 'main.py' selected. The main editor area displays the following Python code:

```
1 from flask import Flask, jsonify, request
2
3 app = Flask(__name__)
4
5 # In-memory storage for to-dos
6 todos = []
7
8 # Create
9 @app.route('/', methods=['POST'])
10 def create_todo():
11     todo = request.get_json()
12     todos.append(todo)
13     return jsonify(todo), 201
14
15 # Read All
16 @app.route('/', methods=['GET'])
17 def get_all_todos():
18     return jsonify(todos)
19
20 # Read One
21 @app.route('/<int:todo_id>', methods=['GET'])
22 def get_todo(todo_id):
23     for todo in todos:
24         if todo['id'] == todo_id:
25             return jsonify(todo)
26     return jsonify({'error': 'To-Do not found'}), 404
27
28 # Update
29 @app.route('/<int:todo_id>', methods=['PUT'])
30 def update_todo(todo_id):
31     updated_data = request.get_json()
32     for i, todo in enumerate(todos):
33         if todo['id'] == todo_id:
```

3] CODE:

A close-up screenshot of the Python code from the previous block, showing the first 33 lines. The code is color-coded: keywords like 'from', 'def', and 'return' are in blue; function names and variables like 'Flask', 'jsonify', 'request', 'app', 'todos', 'todo', and 'updated\_data' are in green; and string literals are in red. The code implements a REST API for a To-Do List with endpoints for creating, reading all, reading one, and updating a todo item.

```
from flask import Flask, jsonify, request

app = Flask(__name__)

# In-memory storage for to-dos
todos = []

# Create
@app.route('/', methods=['POST'])
def create_todo():
    todo = request.get_json()
    todos.append(todo)
    return jsonify(todo), 201

# Read All
@app.route('/', methods=['GET'])
```

```

def get_all_todos():
    return jsonify(todos)

# Read One
@app.route('/<int:todo_id>', methods=['GET'])
def get_todo(todo_id):
    for todo in todos:
        if todo['id'] == todo_id:
            return jsonify(todo)
    return jsonify({'error': 'To-Do not found'}), 404

# Update
@app.route('/<int:todo_id>', methods=['PUT'])
def update_todo(todo_id):
    updated_data = request.get_json()
    for i, todo in enumerate(todos):
        if todo['id'] == todo_id:
            todos[i] = updated_data
            return jsonify(updated_data)
    return jsonify({'error': 'To-Do not found'}), 404

# Delete
@app.route('/<int:todo_id>', methods=['DELETE'])
def delete_todo(todo_id):
    for i, todo in enumerate(todos):
        if todo['id'] == todo_id:
            del todos[i]
            return jsonify({'message': 'Deleted successfully'})
    return jsonify({'error': 'To-Do not found'}), 404

if __name__ == '__main__':
    app.run(debug=True)

```

4] Save the code.

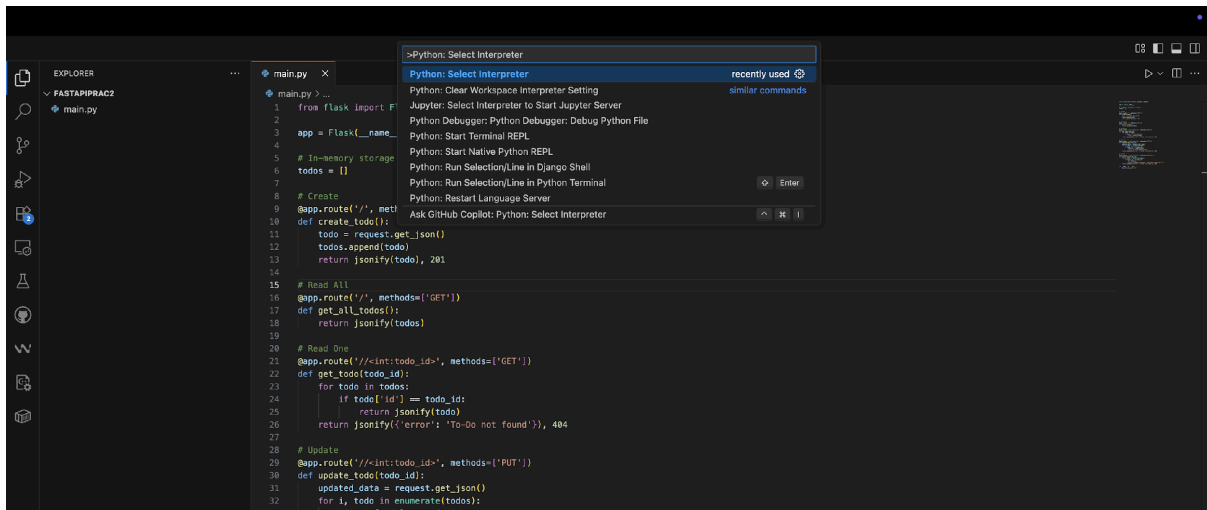
{

*NOTE: Make sure you are using a correct python interpreter.*

*Steps on how to choose a Python Interpreter:*

*I] Press ctrl+shift+P' if you are on windows or linux.*

*II] Type Python: Select Interpreter.*



III] Choose global version.

}

## B] Test the API using Postman or curl.

{

NOTE: you will have 2 terminals running

On one terminal: write `python Pract2` to execute (if you are on windows or linux)

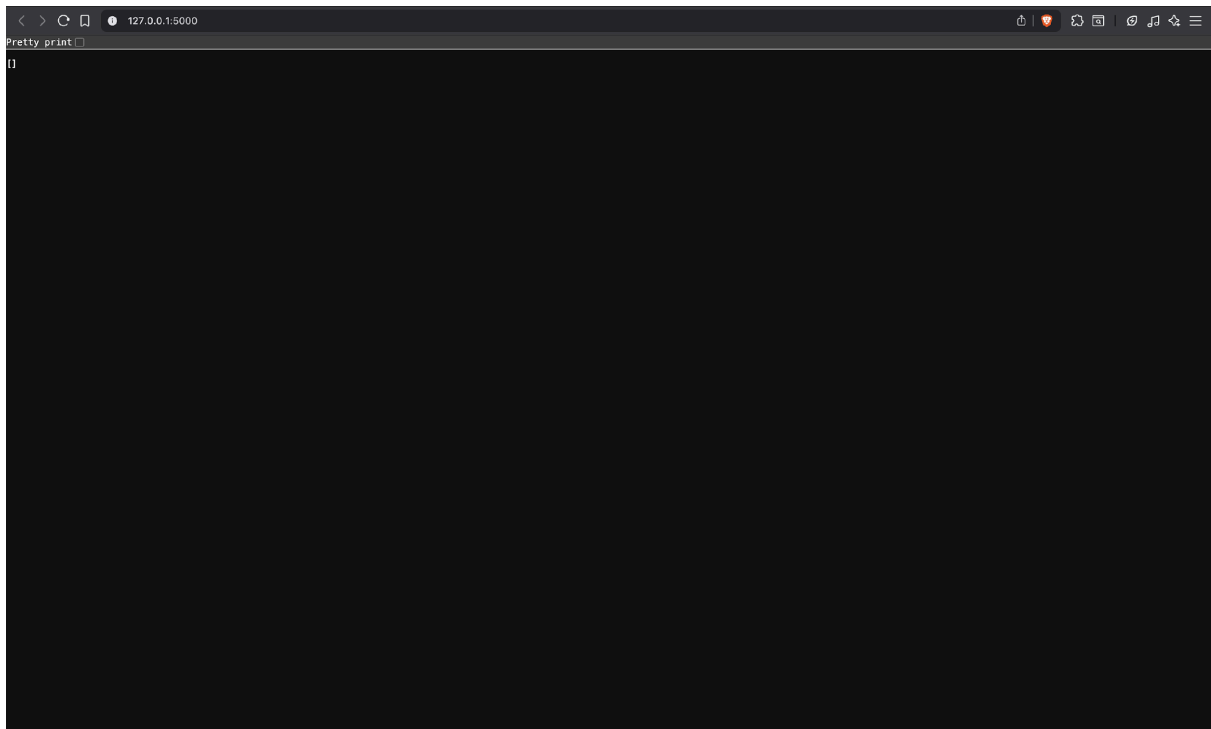
Go to the http address where your server is running by pressing ctrl and clicking on the link:

```

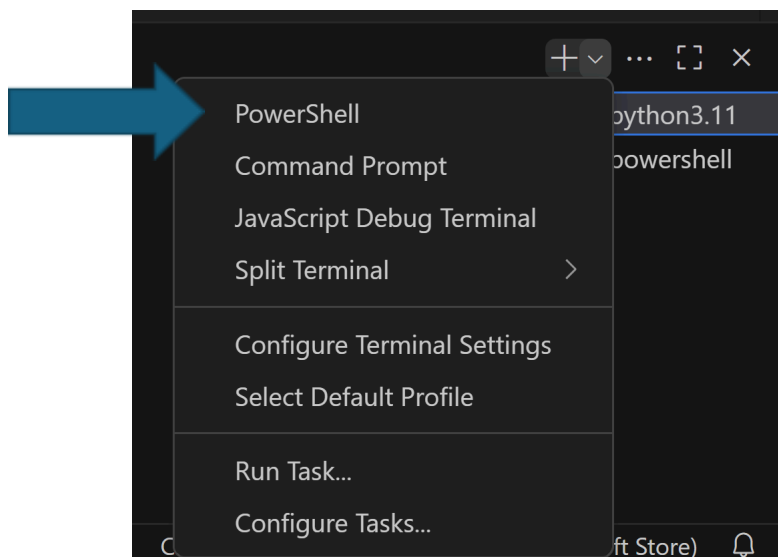
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS D:\Personal_24\MCC\Devops\Pract\Pract2> python Pract2
* Serving Flask app 'Pract2'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 552-893-538

```



*Next, open new bash terminal wherein you will do your CRUD operations for the RESTAPI.*



*Open new powershell.*

}

4] To check with curl:

**Create a To-Do**

In Powershell:

```
curl.exe -X POST http://127.0.0.1:5000 -H "Content-Type: application/json" -d '{"id": 1, "title": "Learn Flask", "completed": false}'
```

### **Get All To-Dos**

In Powershell:

```
curl.exe http://127.0.0.1:5000
```

### **Get One To-Do**

In Powershell:

```
curl.exe http://127.0.0.1:5000/1
```

### **Update To-Do**

In Powershell:

```
curl.exe -X PUT http://127.0.0.1:5000/1 -H "Content-Type: application/json" -d '{"id": 1, "title": "Learn Flask", "completed": true}'
```

### **Delete To-Do**

In Powershell:

```
curl.exe -X DELETE http://127.0.0.1:5000/1
```

>> In terminal 1 press 'ctrl+C' to end the running program.

### **TO DO:**

Check the outcome for every instance above.