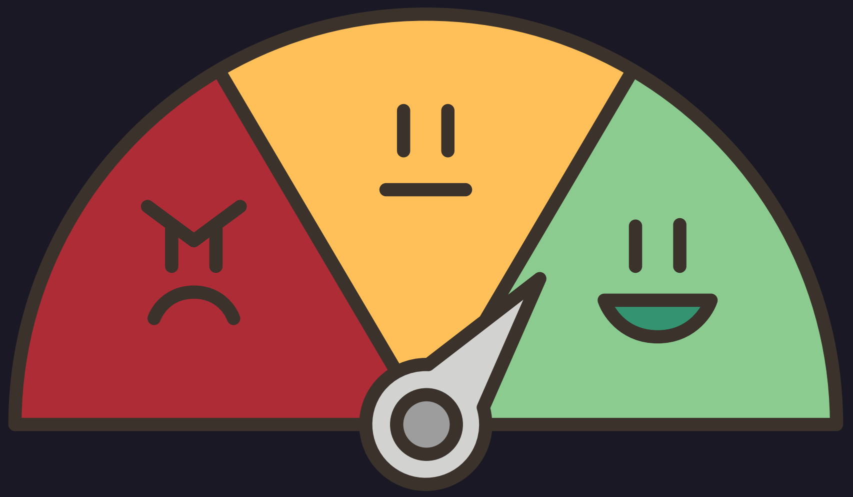
FastAPI

Fast API in Python is same like Express in JS

Instructor-Hitesh Choudhary
Presenter/Learner-Rabindra Mishra

Fast API has Become an Industry Standard



Important for Data Science Students to learn Fast API for Python based web development/ML model consumption is mostly done on Fast API

Fast API requires uvicorn (ASGI Web Server),pydantic(Data Validation),typing (type hints).

pydantic is required with Fast API because Python is dynamically typed.

Documentation:=> https://fastapi.tiangolo.com/

pip freeze> requirements.txt

🔍 What it does:

- **pip freeze:** Lists all installed Python packages and their exact versions in your current environment.
- **>:** Redirects the output of the command to a file.
- **requirements.txt:** The file that will store the list of packages.

Eg:

```
Django==4.2.5
numpy==1.24.3
pandas==2.0.1
```

Pydantic

Usually we Create 2 Models one for incoming Data and one for exporting the Data

Concept of **Decorators** are heavily used in FAST API

- Modifying how a function behaves, without changing the original function's code.
- That new function wraps the original one.
- You then use @decorator_name to apply it easily.

Basic Methods in FAST API

@app.get("/"):
This decorator indicates that the function below it will handle HTTP GET requests

@app.post("/"):
Similar to get, but handles HTTP POST requests
Used when you want to create/submit new data

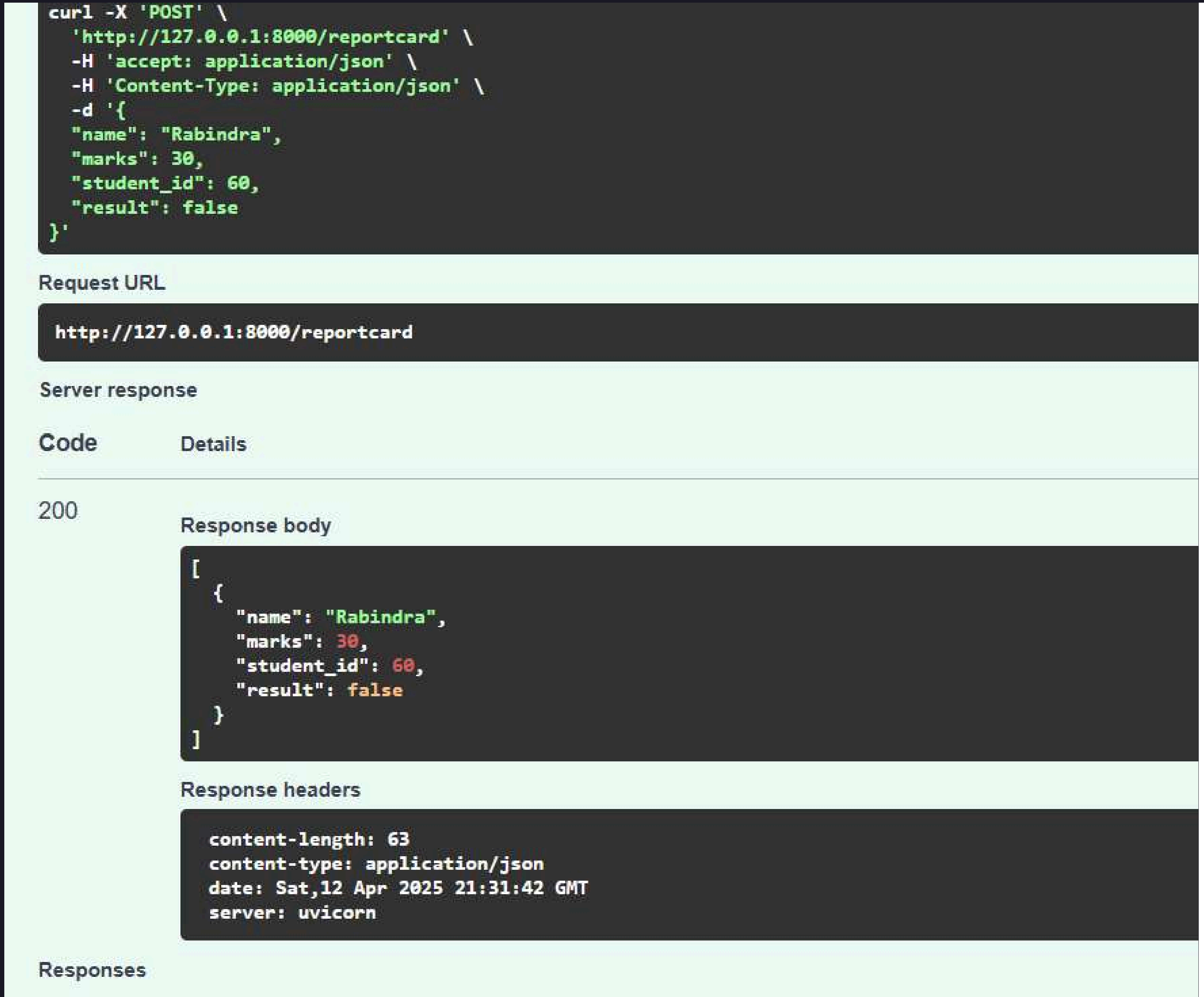
@app.put("/")
Used to handle HTTP PUT requests, which are typically used to update existing resources.
Unlike post it updates existing data usually requires unique id.

@app.delete("/")

How to run File

- File name here is main.py
- Command uvicorn main:app
- can use --reload flag to allow automatic reloading after changes
- Inside main file run app this is what main:app means

Fast API Screenshot



```
curl -X 'POST' \
  'http://127.0.0.1:8000/reportcard' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
  "name": "Rabindra",
  "marks": 30,
  "student_id": 60,
  "result": false
}'
```

Request URL

```
http://127.0.0.1:8000/reportcard
```

Server response

Code Details

200

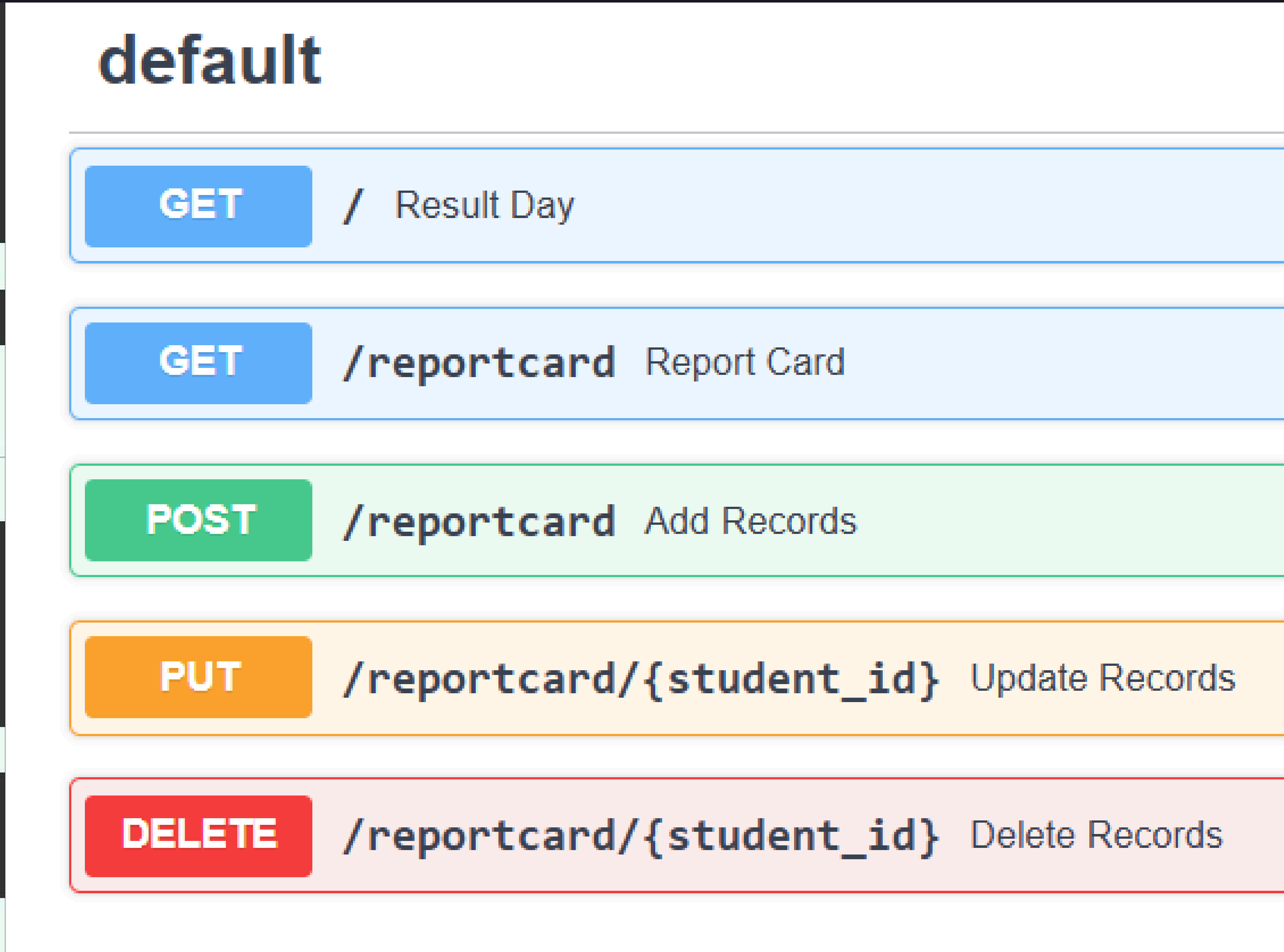
Response body

```
[
  {
    "name": "Rabindra",
    "marks": 30,
    "student_id": 60,
    "result": false
  }
]
```

Response headers

```
content-length: 63
content-type: application/json
date: Sat,12 Apr 2025 21:31:42 GMT
server: uvicorn
```

Responses



default

GET / Result Day

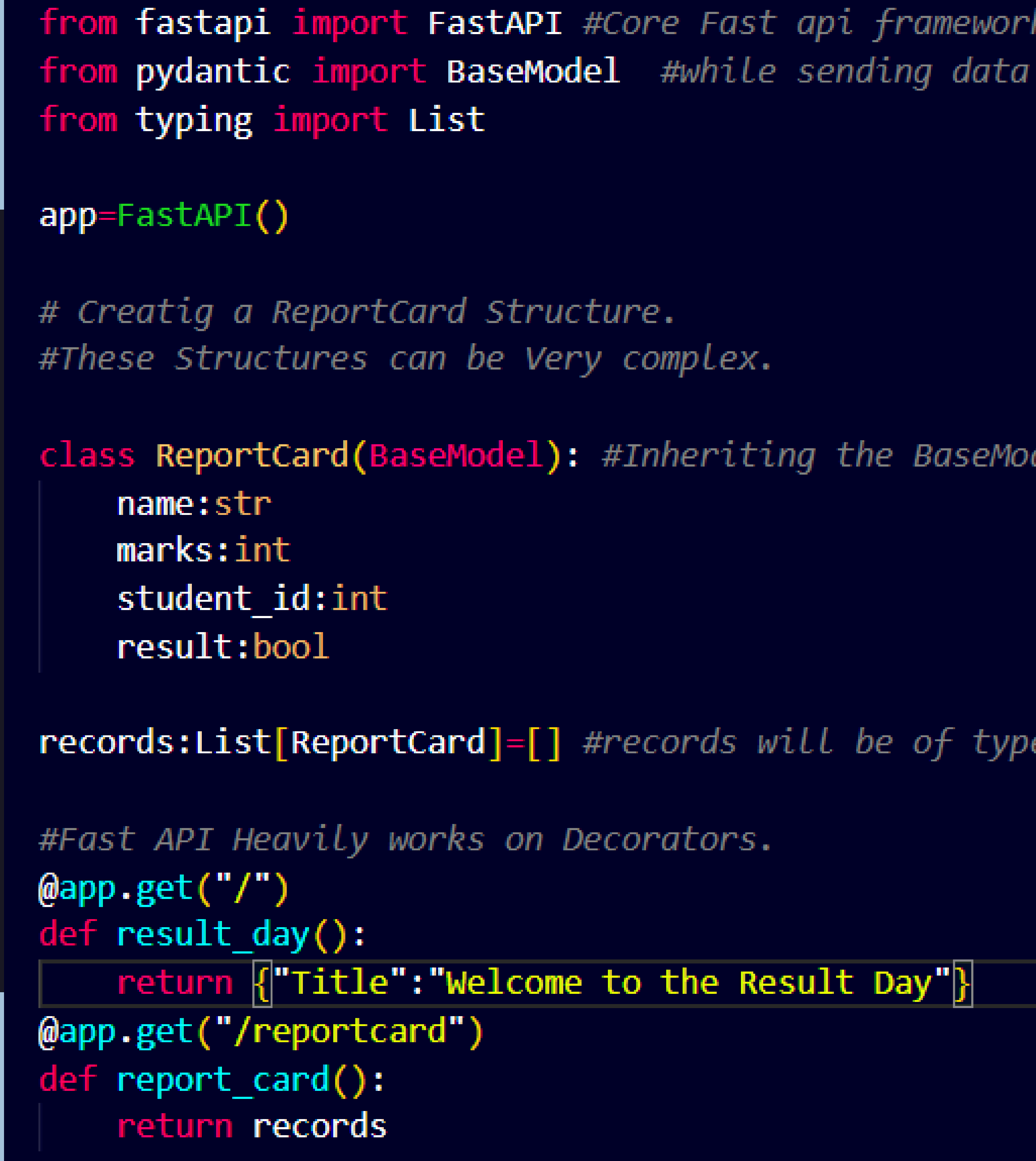
GET /reportcard Report Card

POST /reportcard Add Records

PUT /reportcard/{student_id} Update Records

DELETE /reportcard/{student_id} Delete Records

Fast API Code Snippets



```python
from fastapi import FastAPI #Core Fast api framewor
from pydantic import BaseModel  #while sending data
from typing import List

app=FastAPI()

# Creatig a ReportCard Structure.
#These Structures can be Very complex.

class ReportCard(BaseModel): #Inheriting the BaseMod
    name:str
    marks:int
    student_id:int
    result:bool

records:List[ReportCard]=[] #records will be of typ

#Fast API Heavily works on Decorators.
@app.get("/")
def result_day():
    return {"Title":"Welcome to the Result Day"}
@app.get("/reportcard")
def report_card():
    return records
```

Understanding ASGI v/s WSGI

ASGI stands for Asynchronous Server Gateway Interface. It's the successor to WSGI (Web Server Gateway Interface), which is used by most traditional Python web frameworks (like Django and Flask in older versions).

- WSGI = Synchronous only
- ASGI = Supports both synchronous and asynchronous communication
- ASGI enables handling of modern web features like:
- WebSockets 🔄
- Long-lived connections 🔁
- Background tasks
- High-performance async APIs ⚡

Understanding ASGI v/s WSGI

- When a user opens your website, their browser sends a request (like: "Give me the homepage").
- Your web app (built in Python) has to handle this request and return a response.
- But web browsers and Python apps don’t talk directly. They need a translator: this is where Gateway Interfaces come in.

WSGI (Web Server Gateway Interface) and ASGI (Asynchronous Server Gateway Interface) are standards that describe how that conversation should happen.

WSGI was made when websites were simple (just show pages).
But modern apps do chat, live updates, real-time data, and IoT.

🆚 When to use **ASGI over WSGI**?

Situation	Use WSGI	Use ASGI
Simple blog/portfolio site	✅	
API with low traffic	✅	
Real-time chat app	❌	✅
Live stock market data	❌	✅
Lots of concurrent users	❌	✅
Using Django + Channels or FastAPI	❌	✅