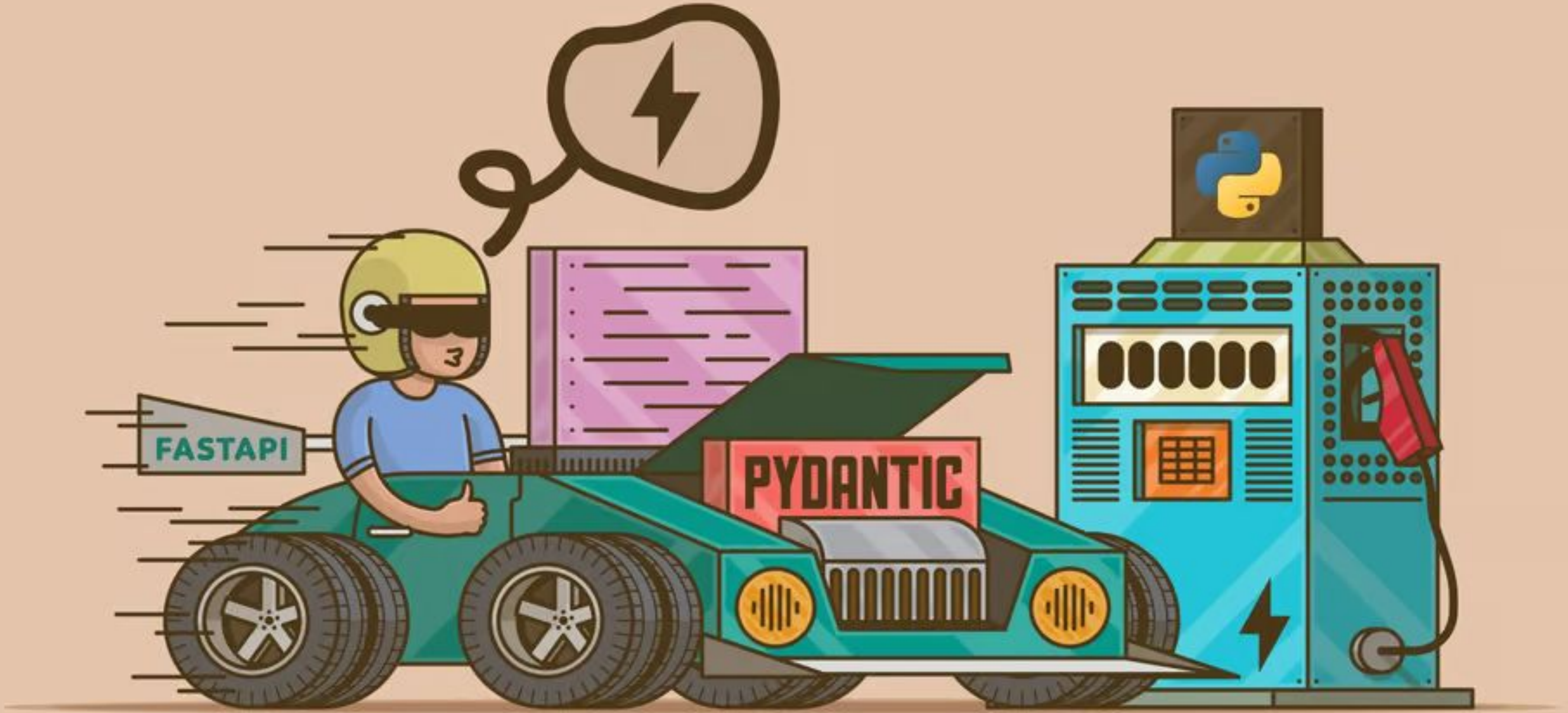


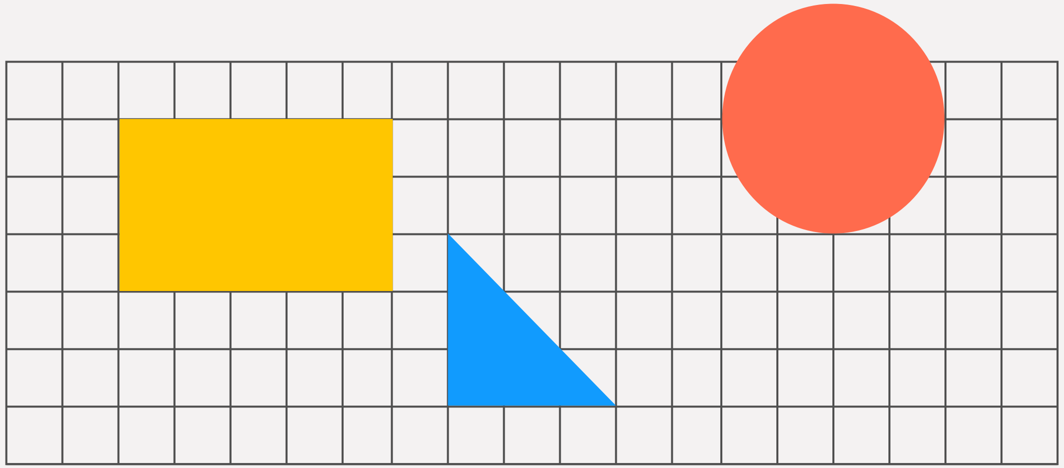
Get Faster with FastAPI

Build Python Web APIs

Img src

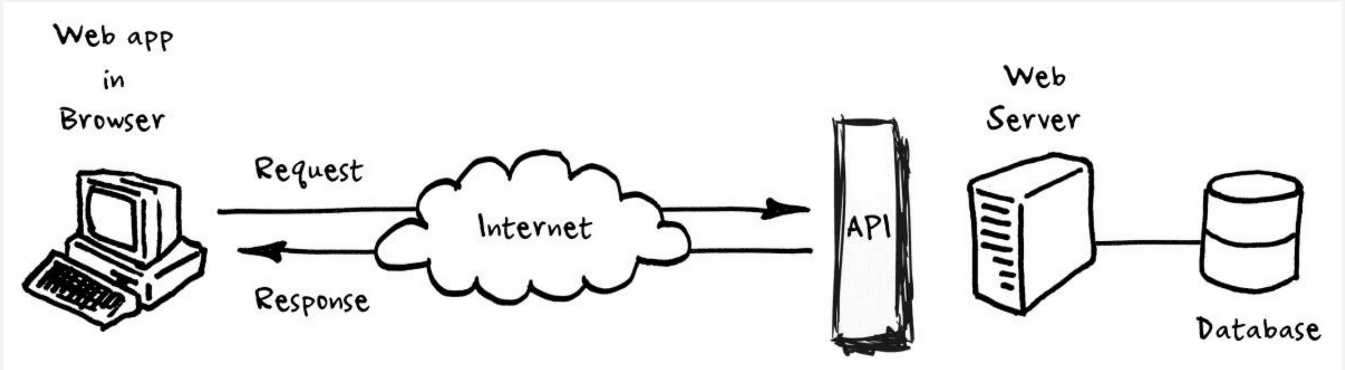
Content

- API Refresher
- What is FastAPI?
- Why FastAPI?
- Get Started with FastAPI
- Road to Glory



API Refresher

API stands for **Application Programming Interface**. An API is a software intermediary that allows *two applications to talk to each other.*



Img src

an API takes a request from an application and sends it to a server. The server then processes the request and sends the data back to the application. The application then interprets the data and presents it to the user.

What is FastAPI?

Automatic Validation

Serialization

Robust Error handling

FastAPI is a modern, high-performance web framework for building APIs with Python based on standard type hints.

- High performance, on par NodeJS and Go, thanks to starlette and pydantic
- Allows for significant increases in development speed
- Offers great editor support, with completion everywhere and less time debugging
- Short, robust and straightforward
- Based on the open standards for APIs, OpenAPI and JSON Schema

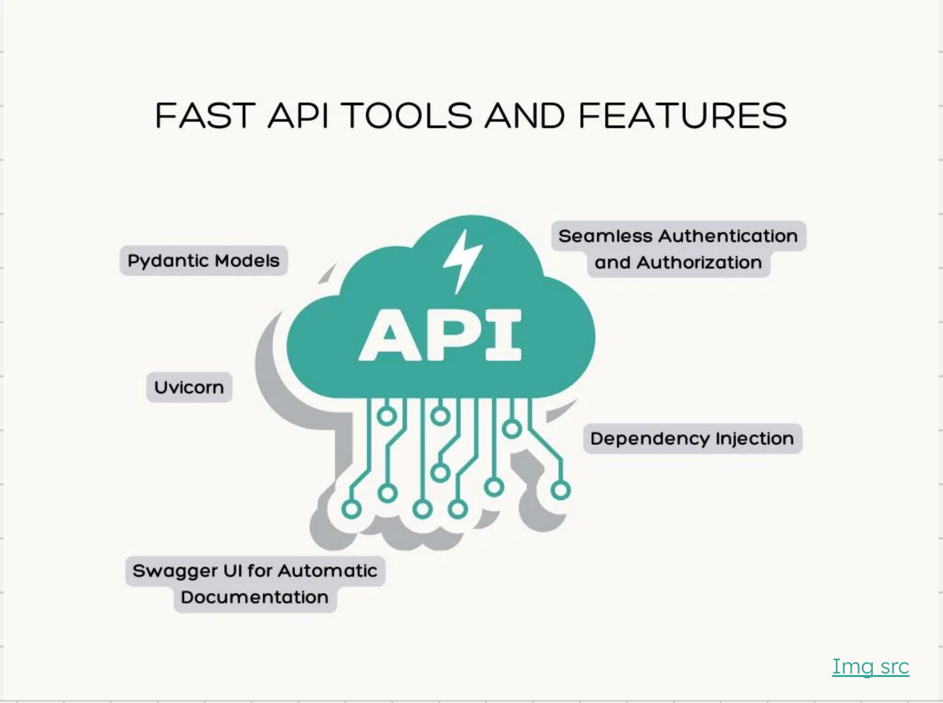
FastAPI is particularly well-suited for building RESTful APIs, microservices, and backend services for real-time applications.

Tips

Why FastAPI?

Core Features:

- Asynchronous Programming
- Type Driven Development
- Data Validation with Pydantic
- Automatic Interactive API Documentation
- Built in Dependency Injection
- Security & Authentication (OAuth2, JWT)

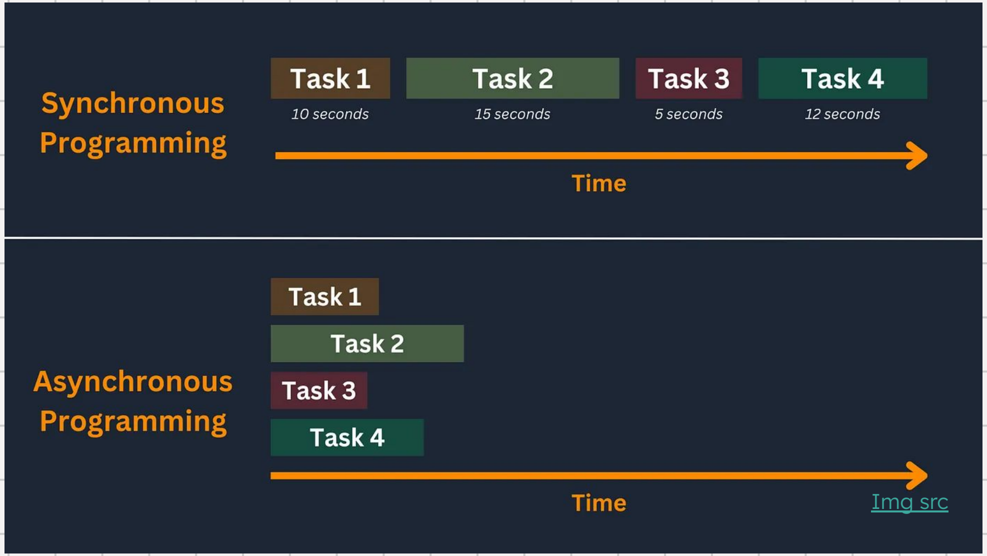

FAST API TOOLS AND FEATURES
Pydantic Models
Seamless Authentication and Authorization
API
Uvicorn
Dependency Injection
Swagger UI for Automatic Documentation

Img src

Go Async: Waiting is Boring

What is Async in Python?

- Async = Do many tasks at once without blocking
- Uses async/await to handle slow operations efficiently
- Runs on an event loop instead of threads
- Best for I/O-heavy apps (APIs, DB, networks)
- Powers fast web frameworks like FastAPI



Others vs FastAPI

	Django	Flask	FastAPI
Community	Big. 66k GitHub stars, Since 2005	Big. 61k GitHub stars, Since 2010	Big. 50k GitHub stars, Since 2018
Type	Full-stack	Microframework (minimalistic)	Microframework (modern, async-first)
Use Case	Big apps: admin panels, e-commerce, CMS	Small to medium apps, simple APIs	APIs, microservices, async systems
Performance	Isn't the best	Performs better than Django	Fastest web frameworks
Learning Curve	Its massive and complicated	Easy to learn & straightforward in use	Moderate (Async, Typing)
Async Support	Yes, with limited latency	No, needs Asyncio	Provides native async support

	Django	Flask	FastAPI
Data Validation	Manual Validation	Manual Validation	Automatic via Pydantic
Built-in Admin	Very Powerful	Need to build manually	Need to build manually
ORM	Django ORM	Need Extensions	Need Extensions
Architecture	Monolithic	Flexible	Flexible
Interactive Documentation	Manual	Manual	Auto Generated (Swagger, Redoc)
Deployment	Traditional (uWSGI, Gunicorn)	Traditional (uWSGI, Gunicorn)	ASGI (Uvicorn, Hypercorn)
Best For	Full website, fast admin dashboard	Lightweight services, quick prototypes	High Performance APIs, async systems

Get Started

Installation

```
pip install fastapi uvicorn loguru pytest

pip install fastapi[standard]
```

- Will need uvicorn, a lightning fast ASGI, to handle HTTP requests and serve responses
- Standard practice to use logs and write tests. My choice: pytest and loguru!

Have requirements file(s) for your project, always.

Tips

Get Started

First API with FastAPI

```
# main.py
from fastapi import FastAPI

app = FastAPI()

@app.get("/") # endpoint
async def root():
    return {"message": "Hello World"} # service
```

```
fastapi dev main.py

uvicorn main:app --reload
```

Plan ahead the endpoints, edge cases, exceptions handling and testing.

Tips

Road to Glory: What's Next?

Topic

Starter Kit

+

Advanced Approaches

Learning Path

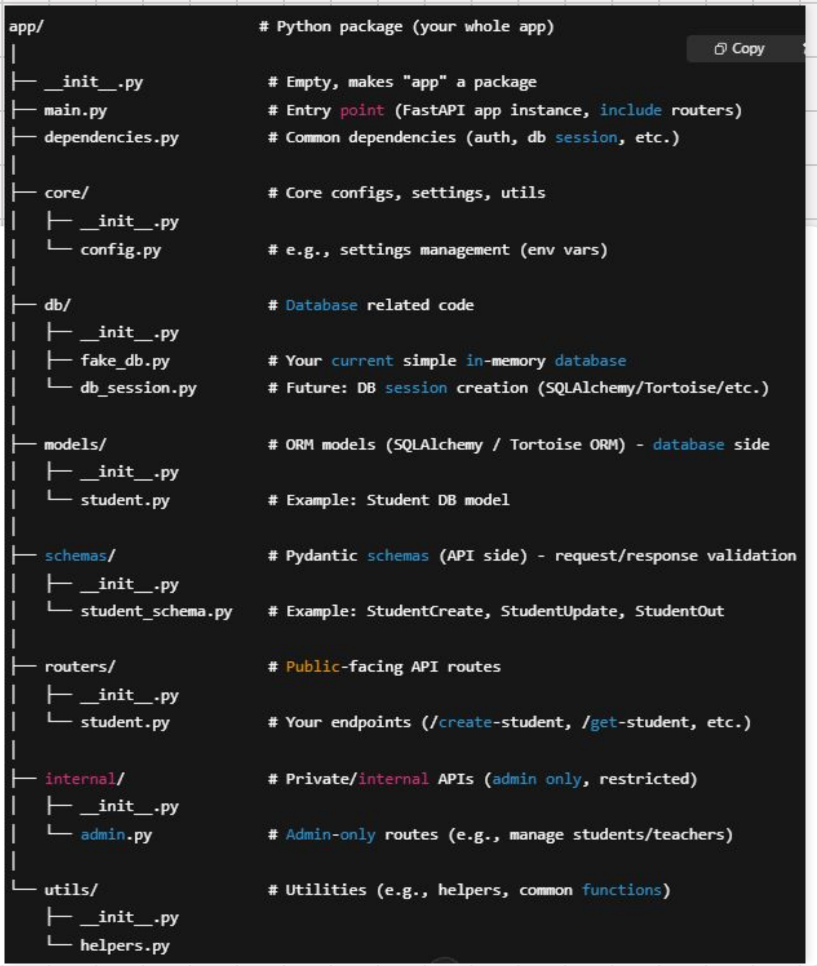
- API Endpoints/Routes
- Auto API Documentation
- Path Parameters
- Query Parameters
- Data Validation
- Request Body
- Response Model
- Status Codes
- CRUD

- Get Modular
- DB Connection
- Dependencies
- Middleware
- Security
- Life span, Events
- Background Tasks
- Custom Error/Exception Handling
- Testing & Github Actions

GOAT of all Resources: FastAPI Official Documentation

App Structure

- Example Code: learn_ fastapi _repo
- Make it modular: Bigger Applications - Multiple Files
- Also don't forget to include requirements, tests and custom exception handling directories!
- You did forget to write comprehensive readme side by side, didn't you? Have a milestone plan!
- Check this out for reference: Hotel Transylvaniya



```
app/                        # Python package (your whole app)
│
├── __init__.py              # Empty, makes "app" a package
├── main.py                  # Entry point (FastAPI app instance, include routers)
├── dependencies.py          # Common dependencies (auth, db session, etc.)
│
├── core/                    # Core configs, settings, utils
│   ├── __init__.py
│   └── config.py            # e.g., settings management (env vars)
│
├── db/                      # Database related code
│   ├── __init__.py
│   ├── fake_db.py           # Your current simple in-memory database
│   └── db_session.py        # Future: DB session creation (SQLAlchemy/Tortoise/etc.)
│
├── models/                  # ORM models (SQLAlchemy / Tortoise ORM) - database side
│   ├── __init__.py
│   └── student.py           # Example: Student DB model
│
├── schemas/                 # Pydantic schemas (API side) - request/response validation
│   ├── __init__.py
│   └── student_schema.py    # Example: StudentCreate, StudentUpdate, StudentOut
│
├── routers/                 # Public-facing API routes
│   ├── __init__.py
│   └── student.py           # Your endpoints (/create-student, /get-student, etc.)
│
├── internal/                # Private/internal APIs (admin only, restricted)
│   ├── __init__.py
│   └── admin.py             # Admin-only routes (e.g., manage students/teachers)
│
└── utils/                   # Utilities (e.g., helpers, common functions)
    ├── __init__.py
    └── helpers.py
```

Search more

- Have comprehensive self note of FastAPI learning
- Search for resources, plenty of works available online
- One day at a time, let's grow.

QnA Time

Thank You!

Made & presented by: @ahnaftanjid@cloudly.io