# ✅ FastAPI Interview Questions and Answers (Entry-Level)

## 1. What is FastAPI, and why is it used?

FastAPI is a modern, fast web framework for building APIs using Python. It is used to create RESTful APIs quickly and efficiently.

### Key Points:

* ✔ Built on Starlette and Pydantic.
* ✔ Fast due to asynchronous support.
* ✔ Automatic OpenAPI documentation generation.

## 2. How do you install FastAPI and run a FastAPI application?

1. Install FastAPI and Uvicorn using pip:  
  
 pip install fastapi uvicorn  
  
2. Run a FastAPI app:  
  
 uvicorn main:app --reload

### Key Points:

* ✔ fastapi is the framework.
* ✔ uvicorn is the ASGI server to run the app.

## 3. What is Uvicorn, and why is it required?

Uvicorn is a high-performance ASGI server used to run FastAPI applications.

### Key Points:

* ✔ ASGI stands for Asynchronous Server Gateway Interface.
* ✔ Needed to serve FastAPI apps.

## 4. What is the structure of a basic FastAPI application?

from fastapi import FastAPI  
  
app = FastAPI()  
  
@app.get("/")  
async def home():  
 return {"message": "Hello, FastAPI"}

### Key Points:

* ✔ Use FastAPI() to create the app.
* ✔ Define routes using decorators (@app.get, @app.post).

## 5. How do you define a GET and POST endpoint in FastAPI?

# GET Request (Fetch data):  
@app.get("/items")  
def get\_items():  
 return {"item": "Sample Item"}  
  
# POST Request (Send data):  
@app.post("/items")  
def create\_item(item: dict):  
 return {"item": item}

### Key Points:

* ✔ @app.get() for reading data.
* ✔ @app.post() for sending data.

## 6. What are path parameters in FastAPI?

Path parameters are dynamic values passed through the URL.  
  
@app.get("/items/{item\_id}")  
def get\_item(item\_id: int):  
 return {"item\_id": item\_id}

### Key Points:

* ✔ Use curly braces {} to define parameters.
* ✔ FastAPI automatically converts the type.

## 7. What are query parameters in FastAPI?

Query parameters are key-value pairs sent in the URL after ?.  
  
@app.get("/items/")  
def get\_items(q: str = None):  
 return {"query": q}

### Key Points:

* ✔ Used for optional or filtering data.
* ✔ Default value is set using =.

## 8. What is Pydantic and why is it important in FastAPI?

Pydantic is a Python library used in FastAPI for data validation and serialization.  
  
from pydantic import BaseModel  
  
class Item(BaseModel):  
 name: str  
 price: float

### Key Points:

* ✔ Ensures data validation.
* ✔ Use BaseModel to define data schemas.

## 9. How do you validate data in FastAPI?

Use Pydantic models to validate incoming data.  
  
@app.post("/items/")  
async def create\_item(item: Item):  
 return item

### Key Points:

* ✔ FastAPI automatically checks type errors.
* ✔ Uses Pydantic for form validation.

## 10. What is the difference between Query, Path, and Body in FastAPI?

- Query: For optional parameters in the URL.  
- Path: For required parameters in the URL path.  
- Body: For sending request body (usually JSON).  
  
from fastapi import Query, Path, Body

### Key Points:

* ✔ Query for URL query parameters.
* ✔ Path for required path parameters.
* ✔ Body for JSON or request body.