**1. What is Flask?**

**Flask is a lightweight, micro web framework for Python that allows developers to build web applications quickly with minimal setup. It is flexible and easy to extend but does not include built-in features like authentication or admin panels.**

**2. Differences Between Flask and Django**

| **Feature** | **Flask** | **Django** |
| --- | --- | --- |
| **Framework Type** | **Micro-framework** | **Full-stack framework** |
| **Built-in Features** | **Minimal (needs extensions)** | **Batteries included (ORM, auth)** |
| **Project Complexity** | **Simple, lightweight** | **Larger, feature-rich** |
| **Flexibility** | **Highly flexible** | **More structured and opinionated** |
| **Learning Curve** | **Easier for beginners** | **Steeper due to features** |

**3. Pros and Cons of Flask**

**Pros:**

* **Simple and lightweight**
* **Highly flexible and modular**
* **Easier learning curve for beginners**

**Cons:**

* **Lacks built-in components (more setup needed)**
* **Not ideal for very large, complex apps out-of-the-box**
* **Fewer conventions can lead to inconsistent code**

**4. Best Projects for Flask**

**Flask suits small to medium projects or APIs, such as:**

* **Prototypes or MVPs**
* **RESTful APIs**
* **Simple web apps like blogs, portfolios, or dashboards**

***Example:* A simple blog where you want to quickly deploy without heavy dependencies.**

**5. Project Setup Complexity Comparison**

* **Flask: Minimal setup; you write routing and configurations manually.**
* **Django: Comes with project structure, ORM, admin panel out of the box; more initial setup but speeds development on bigger projects.**

**6. Role of WSGI in Flask**

**WSGI (Web Server Gateway Interface) is a standard interface between web servers and Python web applications. Flask apps run on WSGI servers (like Gunicorn) to handle HTTP requests and responses.**

**7. Companies Using Flask**

* **Netflix**
* **Airbnb**
* **Reddit**
* **Lyft**
* **Uber**

**These companies use Flask for microservices or specific parts of their platforms due to Flask's simplicity and flexibility.**

**8. Mind Map (Description) Comparing Flask and Django Architecture**

* **Flask: Minimal core → Extensions for DB, Auth → Custom app structure**
* **Django: Monolithic framework → Built-in ORM, Templates, Admin, Auth → Defined project/app structure**

**9. Why Beginners Prefer Flask**

* **Simple to understand and start coding**
* **Minimal setup allows focus on learning Python and web basics**
* **Less overwhelming than full-stack frameworks**

**10. Sample Use-Case: TODO App**

**Flask is better for a TODO app because:**

* **The app is simple and small-scale, so minimal setup is needed.**
* **Flask lets you build REST APIs or web forms quickly without the overhead of Django’s full stack.**
* **Flexibility to add only necessary features, keeping the app lightweight.**

**37. Difference between python app.py and flask run**

| **flask run** | **python app.py** |
| --- | --- |
| Uses environment variables like FLASK\_APP, FLASK\_ENV | No need for FLASK\_APP |
| Recommended for development | Explicitly starts app.run() |
| Supports automatic reloading | Must manually set debug=True |
| Runs from CLI context | Directly runs code in file |

**40. Note on common issues**

* **Port conflicts**:
  + Default port 5000 might be in use → Use --port to change.
* **Missing env vars**:
  + Forgot to set FLASK\_APP → Get error: "Error: Could not locate Flask application."
* **Firewall issues**:
  + Access from another device blocked if firewall restricts port.
* **.env not loaded**:
  + Ensure python-dotenv is installed (comes by default with Flask 2.x).
* **Debugger security risk**:
  + Never run debug mode in production → exposes internal variables.