A **framework** is a set of tools, libraries, and conventions that help developers build applications more efficiently. It provides a structured environment, making it easier to handle common tasks like data handling, user authentication, and more. Frameworks often come with pre-written code and reusable components to streamline development and enforce best practices.

For example, a framework for web development would typically provide components for handling HTTP requests, routing, session management, and database interactions.

**Django** is a popular, high-level web development framework written in Python. It follows the **Model-View-Template (MVT)** architectural pattern and focuses on "batteries-included," meaning it comes with many built-in tools for common web development tasks. This includes:

* An **ORM** (Object-Relational Mapping) system to interact with databases using Python code instead of raw SQL
* **Admin interface** to manage application data through a web interface
* **Form handling** for validating and processing user input
* **User authentication** and authorization
* **URL routing** to map URLs to views
* **Templating engine** for rendering dynamic HTML pages

Django emphasizes **rapid development** and **clean, pragmatic design**. It’s often chosen for projects where scalability, security, and ease of use are priorities. With Django, developers can build applications faster while following best practices and leveraging reusable components.

Key Features:

**1. Rapid Development**

* Django is designed to help developers build applications quickly without needing to reinvent the wheel. Its "batteries-included" philosophy means it comes with a lot of built-in functionality, allowing you to focus more on your application's unique aspects.

**2. Batteries-Included**

* Django includes a wide range of built-in libraries and tools to handle common tasks, such as form validation, user authentication, URL routing, and more, which helps save time and reduce the need for external dependencies.

**3. Secure by Design**

* Django emphasizes security and includes built-in protections against common web vulnerabilities, such as:
  + SQL injection
  + Cross-site scripting (XSS)
  + Cross-site request forgery (CSRF)
  + Clickjacking
* Additionally, Django manages passwords securely and enforces best practices for storing and handling sensitive data.

**4. Scalable**

* Django is suitable for both small and large-scale applications. Its architecture supports the handling of high volumes of traffic, making it a preferred choice for scalable applications.

**5. Object-Relational Mapping (ORM)**

* Django's ORM allows developers to interact with databases (like PostgreSQL, MySQL, and SQLite) using Python code instead of SQL queries. This simplifies database interactions and makes the code more readable and easier to maintain.

**6. Admin Interface**

* Django automatically generates an admin interface for managing application data. This feature is especially useful for quickly managing content, users, and other data without writing custom code for an admin panel.

**7. Versatile Templating System**

* Django’s templating engine allows for the creation of dynamic HTML with Python-like syntax. This feature makes it easy to create complex templates with loops, conditions, and filters to render content dynamically.

**8. Built-in User Authentication**

* Django has a built-in authentication system that provides user registration, login, logout, and password management functionality, as well as more advanced features like permissions and groups.

**9. URL Routing**

* Django uses a clean and straightforward URL dispatcher to map URLs to views, making it easy to define clear, human-readable URLs. This enhances SEO and makes the application more user-friendly.

**10. Testing Framework**

* Django includes tools for creating and running tests, which makes it easier to ensure that the application is functioning as expected. This helps with maintaining code quality and reliability.