1) Why is Django so popular among web developers?

Django is popular because it is a high-level, open-source Python framework that provides many built-in components (database ORM, authentication, templating, admin interface, routing, security, etc.), allowing developers to avoid boilerplate and focus on what is unique to their application. Because Django supports rapid development and clean design, many teams appreciate its productivity and maintainability benefits.

2) Five large companies that use Django

Company	What they do (product/service)	What they use Django for
Instagram	A social media platform for photo/video sharing and social networking	Instagram's backend was built using Django — handling massive user traffic, user accounts, posts, comments, likes, etc.
Mozilla	Web-browser publishers (among other products) also run documentation/help sites and support resources (e.g., MDN Web Docs)	Mozilla uses Django for some of its services, including documentation and support websites.
Pinterest	An online service for discovering and saving ideas, images — a visual social/interest network	Pinterest has used Django for its backend to support data storage, user interactions, content serving, etc.
Disqus	A comment-hosting/discussion system used by many websites	Disqus uses Django for its backend to manage user accounts, comments, threading, moderation, and serve high-volume traffic.
NASA	The U.S. civil space and aeronautics research agency runs many public data portals and project websites.	NASA uses Django to manage several data-heavy portals and websites, leveraging its ability to handle complex content and data and to scale.

3) For each scenario: would you use Django (and why / why not

You need to develop a web application with multiple users.
Yes — Django is a good fit because it offers built-in support for user authentication, session management, and a robust ORM. This makes it relatively straightforward to build a multi-user web application without reinventing standard functionality.

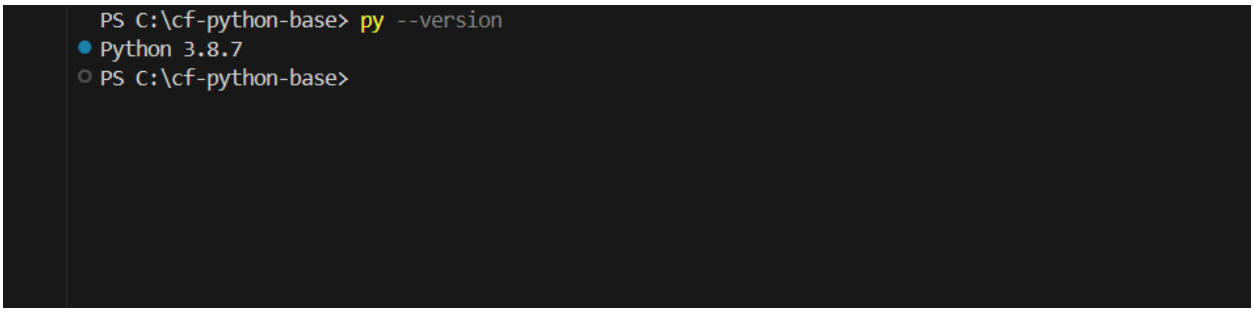
You need fast deployment and the ability to make changes as you proceed.
Yes — one of Django's biggest strengths is rapid development: it comes with many ready-to-use components and lets you go from concept to working application quickly. That flexibility makes it ideal when you expect to iterate and refine features as you go.

You need to build a very basic application, which doesn't require any database access or file operations.
Probably not — if the app is straightforward (static content or minimal dynamic behavior) and doesn't need a database or other server-side resources, Django might be overkill. A lighter-weight framework (or even a plain static site) could be more appropriate.

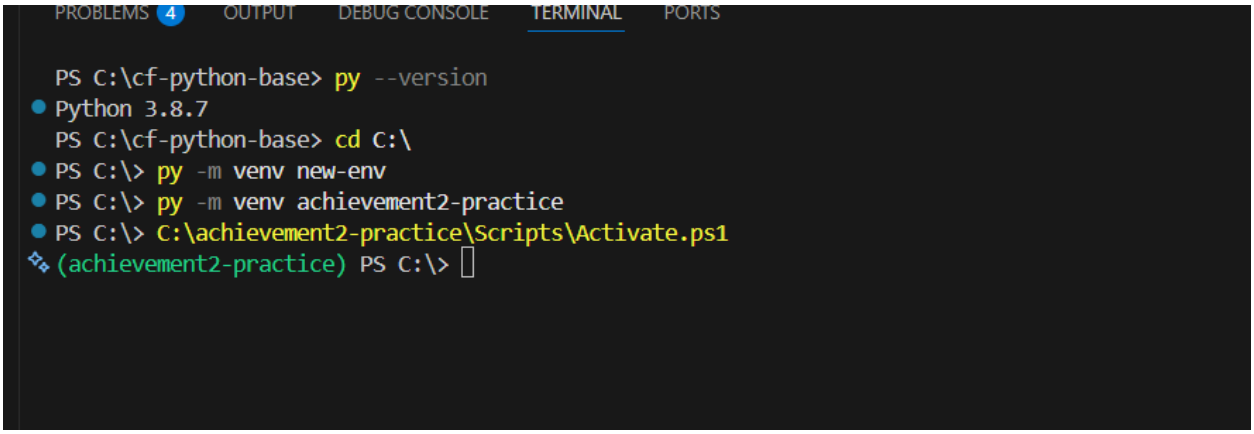
You want to build an application from scratch and want a lot of control over how it works.
Maybe Django gives you a lot of built-in structure, which helps and speeds up development, but that structure could feel limiting if you need very custom or low-level control. If you need maximal flexibility or custom architecture, a more minimal framework might give more freedom.

You're about to start working on a big project and are afraid of getting stuck and needing additional support.
Yes — Django has a large community, rich documentation, and many existing third-party packages and extensions. That support ecosystem can be reassuring when working on large, complex projects with many moving parts.

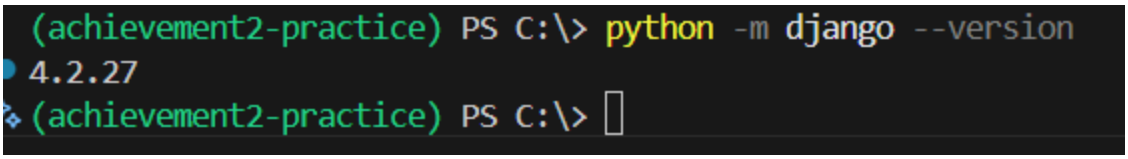
4) Download and install Python (if you haven't done so already)



5)



6)


(achievement2-practice) PS C:\> python -m django --version
4.2.27
(achievement2-practice) PS C:\>