Django is very popular among developers because of its “batteries-included” philosophy. It offers a comprehensive standard library with a wide range of built-in features for common web development tasks. This accelerates development by reducing the need to build numerous components from scratch.

Instagram - Utilizes Django to handle its large amounts of data and user interactions, ensuring smooth performance despite the platform’s massive audience.

Spotify - Spotify combines Python with Django for backend services, optimizing app functionality.

Youtube - relies on Django for adding new features and implementing upgrades efficiently, minimizing errors and enhancing developer performance.

Disqus - Anetwork-based comment system widely used in the blogging community, Django’s scalability and wide range of ready-to-implement solutions help Disqus cater to its growing user base.

Dropbox - uses Django for synchronization features, sharing options, and large file storage capabilities.

Scenarios:

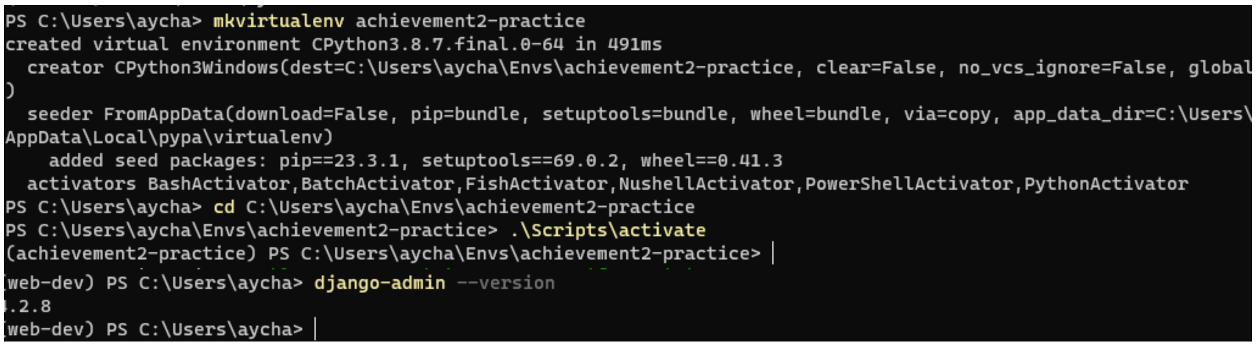
1. Developing a Web Application with Multiple Users:
   - Use Django: Yes.
   - Why: Django is well-suited for applications that require user management and authentication, which are common needs for multi-user web applications. It provides built-in support for user authentication, including user accounts, permissions, and sessions, which simplifies the development process for such features.

2. Fast Deployment and Ability to Make Changes as You Proceed:
   - Use Django: Yes.
   - Why: Django's "batteries-included" approach offers a lot of built-in functionalities, which can speed up the initial development and deployment process. Its design is also conducive to iterative development, allowing for relatively easy modifications and additions as the project evolves.

3. Building a Very Basic Application Without Database Access or File Operations:
   - Use Django: Not recommended.
   - Why: Django is a high-level framework that's generally overkill for very basic applications that don't require database interactions or file operations. For such simple applications, a lightweight framework or even just plain Python scripting might be more efficient.

4. Building an Application From Scratch with a Lot of Control Over How It Works:
   - Use Django: Yes, but with considerations.
   - Why: Django allows for a significant degree of customization and control, especially since it is open-source and you can modify its behavior. However, it also comes with its own conventions and structure (like any framework), which might not offer the same level of low-level control as building an application from scratch without a framework.

5. Starting on a Big Project with Concerns About Getting Stuck and Needing Support:
   - Use Django: Yes.
   - Why: Django has a large and active community, which is a significant advantage when it comes to finding support, resources, and solutions to problems. Its extensive documentation and wealth of third-party packages also make it easier to find help when you're stuck on a large project.



```
PS C:\Users\aycha> mkvirtualenv achievement2-practice
created virtual environment CPython3.8.7.final.0-64 in 491ms
  creator CPython3Windows(dest=C:\Users\aycha\Envs\achievement2-practice, clear=False, no_vcs_ignore=False, global
)
  seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy, app_data_dir=C:\Users\
AppData\Local\pypa\virtualenv)
    added seed packages: pip==23.3.1, setuptools==69.0.2, wheel==0.41.3
  activators BashActivator,BatchActivator,FishActivator,NushellActivator,PowerShellActivator,PythonActivator
PS C:\Users\aycha> cd C:\Users\aycha\Envs\achievement2-practice
PS C:\Users\aycha\Envs\achievement2-practice> .\Scripts\activate
(achievement2-practice) PS C:\Users\aycha\Envs\achievement2-practice> |
web-dev) PS C:\Users\aycha> django-admin --version
.2.8
web-dev) PS C:\Users\aycha> |
```