Exercise 2.1

2.

Django is popular among web developers because it offers fast processing, and allows for rapid development as provides authentification systems, ORM, URL routing, forms, session management, … all out of the box. It also handles security, offering several features from user authentification and password hashing to protection for SQL injections or cross-site requests.

It makes scaling up easier thanks to loosely coupled architecture and the fact it has a strict rules can make it harder but also takes a lot of things off the developer with a lot of precoding.

And of course, it has a strong community which is a major plus for support.

3.

- Instagram (social media platform to share pictures and video content) : They use Django on the backend to handle large-sclae traffic, user authentification, feeds, API for mobile/web apps).

- Mozilla (creator of the Firefox browser as well as various web services) : Used for parts of their web services like the support site or add-ons site in order to manage content and handle a large user base.

- Dropbox (cloud storage and file sharing services) : Used in their web-app layer to handle file sharing, and the communication between the user interface and backend.

- NASA (US space and aeronautic program) : Used for their web and intranet services as well as scientific web apps.

- The Washington Post (newspaper company) : Used for content-heavy parts online, like publishing, article management and high volume traffic.

4. Scenarios in which whether to use Django or not :

- *You need to develop a web application with multiple users :*

Absolutely, especially if you need user accounts and features like moderator/admin permissions, Django offers all the user management tools that would make it easier as any scale.

- *You need fast deployment and the ability to make changes as you proceed.*

Absolutely, Django allows for fast deployment thanks to all the built-in tools and ORM.

- *You need to build a very basic application, which doesn't require any database access or file operations.*

In that case, no. One of the main interest of Django is making DB communication and various features easy thanks to precoding, if a DB is not required and if it is a small project, it loses interest as Django is still heavy, so it wouldn't be needed for a basic app that doesn't require heavy coding, Flask would work better here.

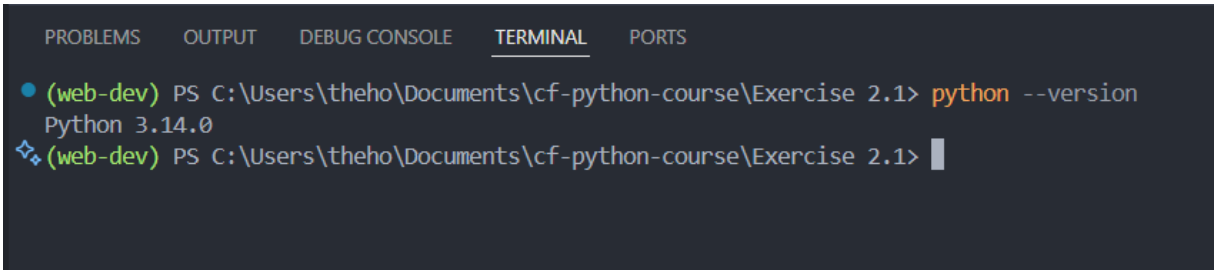
- *You want to build an application from scratch and want a lot of control over how it works.*

The precoding would help in the build, but the rigid structure of Django wouldn't give much control because of hard rules, so in that case, no.

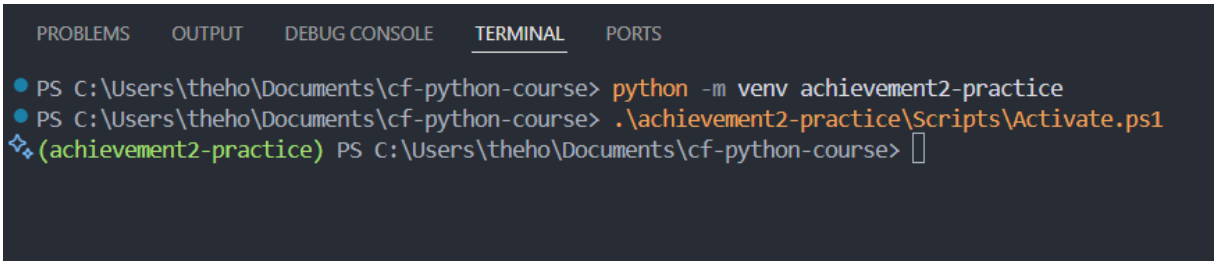
- *You're about to start working on a big project and are afraid of getting stuck and needing additional support.*

Django would be a good choice here, as it's great for scalability and support because of a strong community. On the other hand, it depends of the project.

5. Python version :



7. Created and activated virtual environment :



8. Django version :