2. Why is Django so popular among web developers?

Django has a lot of perks that make it popular with web developers including the fact that it comes out of the box with a lot of useful features like authentication and security. Django also allows developers to use Python, a popular language in general, to design their applications. Its modularity and structure also make it easier to scale up over time.

3. List five large companies that use Django. Specify what the company's product or service is and what they use Django for.

Instagram - They use Django to process the massive amount of data that flows through the site everyday from all of the user-posted images and videos.

YouTube - Much like Instagram, unimaginable amounts of data are uploaded to and streamed from the site each day and they need a powerful framework like Django to keep it all running smoothly.

Spotify - Yet another streaming service, though this time focused on Audio. Django is excellent for smooth and speedy content delivery, making it a great choice for spotify.

NASA - NASA certainly benefits from Django' speedy content delivery but as a government organization they also make use of Django's security features.

Venmo - Venmo processes tons of interactions between users everyday and needs all of these interactions to remain secure. Django's fast delivery and good security features made it a great choice.

4. For each of the following scenarios, explain if you would use Django:

You need to develop a web application with multiple users.

I would use Django because it comes with a built-in user authentication system for managing user info and handling user logins.

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

I would use Django because its many built-in features would allow me to get my application deployed much faster than if I were building with less robust apps. Django's modularity and scalability also make it easier to make changes as I proceed in building the app.

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

I would not use Django because although it's many built-in features are nice, I wouldn't use most of them in smaller scale applications like this. Furthermore, Django is a full-stack framework and if I do not plan on accessing any database or having file operations then I could use a more front-end focused framework instead.

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

In this case I would not use Django because although Django comes with a lot of useful features, it requires you to do things the Django way and forces you to relinquish some control in order to use those features.

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

I would use Django because it scales easily with growing projects and has a lot of built-in features so I wouldn't have to worry about getting additional support for most features.

5. Download and install Python. Run appropriate command to check python version.

```
[(web-dev) justin $ python --version
Python 3.12.5
```

7. Set up and create a virtual environment and name it achievement2-practice.

```
[(achievement2-practice) justin $ workon achievement2-practice
(achievement2-practice) justin $
```

8. Install Django and verify the installation by checking the version.

```
Successfully installed asgiref-3.8.1 django-5.1.1 sqlparse-0.5.1
[(achievement2-practice) justin $ django-admin --version
5.1.1
```