Exercise 2.1

Suppose you're a web developer in a company and need to decide if you'll use vanilla (plain) Python for a project, or a framework like Django instead. What are the advantages and drawbacks of each?

Choosing between vanilla Python and a web framework like Django depends on various factors, including project requirements, development team skills, timeline, scalability needs, and long-term maintenance considerations. Here's a breakdown of the advantages and drawbacks of each:

Vanilla Python - Advantages:

- Lightweight: Vanilla Python is lightweight and doesn't have the overhead of a framework, making it faster in some cases.
- Flexibility: You have complete control over the structure and architecture of your application, allowing for more customized solutions.
- Learning: For smaller projects or for developers who are new to web development, starting with vanilla Python can provide a better understanding of core concepts before diving into a framework.

Vanilla Python - Drawbacks:

- Lack of features: Vanilla Python doesn't come with built-in features for web development like URL routing, form validation, or database ORM, requiring developers to implement these functionalities from scratch.
- More time-consuming: Developing web applications without a framework can be more time-consuming, especially for complex projects, as you need to handle many low-level details manually.
- Maintenance overhead: Without the structure provided by a framework, maintaining and scaling the application may become more challenging as the project grows.

Django - Advantages:

- Rapid development: Django provides a high-level framework with built-in features such as ORM, authentication, admin panel, and URL routing, allowing for rapid development of web applications.
- Scalability: Django's built-in scalability features, such as support for caching, database sharding, and asynchronous tasks, make it suitable for handling large-scale applications.
- Security: Django comes with built-in security features such as protection against common web vulnerabilities like SQL injection, CSRF, and clickjacking.

Django - Drawbacks:

- Learning curve: Django has a learning curve, especially for developers new to web development or those accustomed to working with vanilla Python. It may take time to understand and utilize its full potential.
- Overhead: Django has more overhead compared to vanilla Python due to its bundled features and abstractions, which may not be necessary for smaller or simpler projects.
- Less flexibility: While Django provides a structured approach to web development, it may limit flexibility compared to vanilla Python, especially for unconventional project requirements.

In your own words, what is the most significant advantage of Model View Template (MVT) architecture over Model View Controller (MVC) architecture?

The most significant advantage of Model View Template (MVT) architecture over Model View Controller (MVC) is its simplicity. MVT's clear separation of concerns into models, views, and templates makes it easier to understand, maintain, and collaborate on projects compared to MVC.

Now that you've had an introduction to the Django framework, write down three goals you have for yourself and your learning process during this Achievement. You can reflect on the following questions if it helps:

- *What do you want to learn about Django?*
  - I want to learn how it compares to its Javascript counterparts, such as React and Angular.
- *What do you want to get out of this Achievement?*
  - I want to become proficient at using Django for developing a web app.
- *Where or what do you see yourself working on after you complete this Achievement?*
  - I see myself working on similar, but more complicated web apps compared to the recipe app.