Exercise 2.1: Getting Started with Django

Learning Goals

- Explain MVT architecture and compare it with MVC
- Summarize Django's benefits and drawbacks
- Install and get started with Django

Reflection Questions

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?

   Vanilla Python is ideal if you want complete control over how requests are handled and data is stored. It is lightweight and doesn't come with any extra dependencies, it's ideal for non-web apps, automation scripts, data processing or machine learning prototypes, you're free to design your own architecture and use only the libraries you need.

   The drawbacks are that you have to handle things like routing, authentication, database connections, session management, security, and everything that slows down development. Without the structure of a framework, things can become messy quickly.
   Ideal for:
   - small scripts or utilities
   - Prototyping or experimenting without needing a full-blown framework

   Django comes with all the built-in features you need for rapid development, like ready-made tools for routing, models, forms, authentication, admin dashboards, and more. It is modular and structured, making it easier when the codebase grows. Because of the structure and conventions, it is better for teams to collaborate smoothly.
   However, Django is opinionated, which can feel restrictive, and it would be overkill for small apps that don't need all the features. It is more difficult to learn for beginners, and if you require something that is highly customisable, like non-standard architectures, Django might get in the way.
   Ideal for:
   - full-featured web applications
   - Speed of development, security and scalability.
   - Working in a team environment.

2. 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, in my opinion, is that MVT simplifies the developers' workload by eliminating the need for a separate controller layer. For example, with MVC, if you wish to retrieve an object of data from the database, you have to write a controller method. With MTV, you just write a view() function that queries the model and returns a template, so you skip the extra controller step, which ultimately speeds up development.

3. 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?

- What do you want to get out of this Achievement?
- Where or what do you see yourself working on after you complete this Achievement?

Three goals I have would be:

1. Understand the Model-View-Template workflow. Practice writing views that return responses.
2. Get familiar with Object Relational Mapping with C.R.U.D. and how to define models
3. Work with Templates and Forms, understanding template syntax and how to handle basic user input.

After the Achievement, I would like to be proficient enough to start building my own small Python/ Django apps to become comfortable enough in a work environment. I would also like to understand authentication and deployment.