# Exercise 2.4: Django Views and Templates

## Learning Goals

- Summarize the process of creating views, templates, and URLs
- Explain how the “V” and “T” parts of MVT architecture work
- Create a frontend page for your web application

## Reflection Questions

1. **Do some research on Django views. In your own words, use an example to explain how Django views work.**
   Django view functions, or views, are Python functions. Each view is represented as a Python function, or as a method of a Python class that takes a web request and returns a web response, like an HTLM web page or JSON data. The view itself contains the logic that is necessary to return that response.
   Views are usually put in a file called views.py located in the project or application directory.
   Every view is usually associated with a template that defines the output’s structure. Django selects the view based on the URL coming from the web application in the browser.
   For instance, a movie app probably has a view which, given a specific movie, returns the page (HTML) to render multiple data about that movie, such as the title, genre, directors, authors, and synopsis.

2. **Imagine you’re working on a Django web development project, and you anticipate that you’ll have to reuse lots of code in various parts of the project. In this scenario, will you use Django function-based views or class-based views, and why?**
   In this scenario I would use class-based views. CBVs are based on Python classes, and provide a way to create views that are easy to reuse and extend. CBVs are a better option because you can avoid code duplication (using built-in views).

3. **Read Django’s documentation on the Django template language and make some notes on its basics.**
   Django template language (DTL) is Django’s own template system.

   A Django template is a text document or a Python string marked-up using the Django template language. Some constructs are recognized and interpreted by the template engine. The main ones are variables and tags.

   A template is rendered with a context. Rendering replaces variables with their values, which are looked up in the context, and executes tags. Everything else is output as is.

The syntax of the Django template language involves four constructs:

- Variables - A variable outputs a value from the context, which is a dict-like object mapping keys to values.
- Tags - Tags provide arbitrary logic in the rendering process.
- Filters - Filters transform the values of variables and tag arguments.
- Comments.