# 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 views are Python classes or functions. They receive requests, perform business logic when necessary and send responses to URLS. For example, if a user is on a web application that uses Django and presses on a button to get to a different page on the site, a url is sent. This URL is then received by Django views, some logic runs and the URL that the user requested is retrieved. Finally, the URL is send back to the user.

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 think I would use class-based views. I would use class-based views because I will be able to reuse code. This will enhance the performance of the application whilst also saving time.

3. Read Django’s documentation on the Django template language

and make some notes on its basics.

Template
- text file
- contains variables

Variables
- when template encounters variable, variable is evaluated and replaced with result
- consist of letters, numbers and underscores (has to start with a letter)
- No spaces or punctuation in variable names

Tags - {% tag %}
- some tags create text , some control through performing loops/logic, some load external information

Template Inheritance
-  allows you to build a base that contains all the common elements of your site
- block tag defines blocks that child templates can fill in
- extends tag extends another template