1. Do some research on Django views. In your own words, use an example to explain how Django views work.

Django views work by taking in user requests, and then returning the appropriate template, which is loaded as the content the user sees in the browser. Based on the URL entered by the user in the browser, Django looks up which view should handle the request. The view contains a function or class that takes data about the users request via a request object, and performs logic defined in the view's function/class methods.

For example, in my recipe project, currently if I enter the URL for the locally hosted version of my project without an additional path specified, the home page will be returned, as the view informs Django to load the home.html template file, but if /admin was included as the path the admin site would be displayed.

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 a class-based view. Despite being somewhat harder to read, class-based views allow for easier reuse of code across multiple views, eliminating the need to re-write similar code for various parts of the project, which would likely be the case if function-based views wee used.

3. Read Django's documentation on the Django template language and make some notes on its basics.

On first glance, the language employed in Django's template has some similarities with JSX, in that it allows for the insertion of dynamic content into what would otherwise be static markup. Variables can be defined within double sets of curly braces: {{ variable }}, and should follow similar conventions to Python variables (alphanumeric + underscores, cannot start with an underscore, cannot just be a number). Variable values can be modified by used of filters, which are included in the braces and applied with a pipe (|). Logic can be included in a template file via the use of tags, which are marked with this syntax: {% tag %}.

There are also certain similarities with Angular in how logic from other files, such as the view and other templates which can extend the functionality of a base template.