Django TODO App

Step 1: Project Setup

Django Resource

- Create a virtual environment, activate it, install Django, and start a new project called `todoapp`.
- Create an app named `todolist`
- Register the app!
- Create a view that returns a simple text to the `/todo` endpoint URL
- Register the URL routes using both the app-level as well as the project-level `urls.py` file

Step 2: Models

Django Resource

- Create a `TodoList` class that contains a `title` and a `completed` attribute. `completed` will be a count of items associated with this list that have been completed
- Create a `TodoItem` class that contains a reference to the TodoList that it is part of, as well as a `name`, a `description`, a `date_created` and a `due_date` attribute
- Add a custom function to your `TodoItem` class that allows querying how many days are left before a TodoItem is due
- Add descriptive `__str__()` methods to create a readable string representation of your objects
- Complete the necessary migrations to create the tables and make your db functional
- Register both models in your admin panel
- Validate the db's functionality by adding two TodoLists and a couple of TodoItems each through the admin panel

Step 3: Views and Templates

Django Resource

Todolists overview page

- Edit the view that points to the `/todo` endpoint so that it lists all available TodoLists
- Create a `/template` folder for your app (make sure you *closely* follow the suggestions in regards to that on the Django tutorial!)
- Create a template for the previously mentioned function, so that it lists all available TodoLists as **clickable** items in a bullet-point list
  - The links should redirect to a new URL that is specific to the clicked TodoList (hint: use the TodoList object's ID)
- Use Django's `render()` shortcut to pass the necessary information from the db to your template

Todolist details page

- Create another function in `views.py` that takes as an input a `TodoList`s ID and fetches all associated `TodoItems` from the database
- Create a template that displays all TodoItems of that TodoList as checkboxes
- Remember to register the URLs to allow for correct URL routing

Step 4: HTML Forms

Django Resource

- Update the template that handles your TodoList's details (displaying the TodoItems as checkboxes) so that it allows users to click a button and submit a POST request containing the checked boxes
- Create a new function in `views.py` that handles the logic when a user submits a POST request. It should:
  - count how many items have been completed

    - update that number inside the db at `TodoList`s `completed` attribute
    - compare the `completed` attribute to the length of the items associated to the TodoList
    - redirect to a new URL with the endpoint `/notice` (you'll write that one next)

- Finally, create a new view function and an associated template that routes to `/notice` and renders a sentence stating (depending on what the user checked when they submitted) either:
    - that all items in the list were completed, or
    - that x out of y items were completed