Day 1: Introduction to Django & Setup
Goal: Install Django and understand project structure.**
Key Concepts:
- What is Django? (MVT architecture vs. MVC).
- Difference between a **project** and an **app**.
- How to create a project and app using CLI commands.
Tasks:
- Install Python and Django.
- Create a project (`myproject`) and an app (`todo`).
- Explore the files: `settings.py`, `urls.py`, `models.py`, `views.py`.
- Run the development server.
Resources:
- [Django's Official "Writing your first app"
Guide](https://docs.djangoproject.com/en/4.2/intro/tutorial01/).

Day 2: Models & Databases
Goal: Learn Django ORM and create database tables.**
Key Concepts:
- Models as Python classes (representing database tables).
- Common fields: `CharField`, `BooleanField`, `DateTimeField`.
- Migrations: `makemigrations` and `migrate`.
Tasks:
- Define a `Task` model with `title`, `completed`, and `created_at` fields.
- Create migrations and apply them.
- Use Django's admin panel to add sample tasks.
Exercise: Add a `priority` field to the `Task` model and update the database.

Day 3: Views & URL Routing
Goal: Create views and map URLs.**
Key Concepts:
- Function-based views (FBVs).
- URL patterns and the `urls.py` file.
- Passing data from views to templates.
Tasks:
- Write a view to fetch all tasks from the database.
- Map the view to a URL (e.g., `localhost:8000/tasks/`).
- Test the page in the browser.
Exercise: Create a view to display details of a single task.

Day 4: Templates & HTML
Goal: Render dynamic data using Django templates.**

Key Concepts:
- Template syntax: `{% for %}`, `{% if %}`, `{{ variables }}`.
- Organizing templates in the `templates/` directory.
Tasks:
- Build an HTML template to display all tasks.
- Use loops to iterate over tasks and conditionals to show completion status.
Exercise: Add a base template (e.g., `base.html`) for consistent styling.

Day 5: Forms & User Input
Goal: Handle user input with Django forms.**
Key Concepts:
- Creating forms using `forms.py`.
- Form validation and error handling.
- Redirecting after form submission.
Tasks:
- Build a form to add new tasks.
- Create a view to process the form data.
- Add a "Create Task" button to the template.
Exercise: Add a form to edit existing tasks.

Day 6: CRUD Operations
Goal: Implement full CRUD (Create, Read, Update, Delete).**
Key Concepts:
- Linking views and templates for editing/deleting tasks.
- Using `POST` requests for deletions.
Tasks:
- Add "Edit" and "Delete" buttons for each task.
- Create views to handle updates and deletions.
Exercise: Add a confirmation step before deleting a task.

Day 7: Static Files & Styling
Goal: Style the app with CSS and JavaScript.**
Key Concepts:
- Organizing static files (`static/` directory).
- Loading static files in templates (`{% load static %}`).
Tasks:
- Add CSS to style the task list and forms.
- Include Bootstrap for responsive design.
Exercise: Add a JavaScript feature (e.g., toggle task status without reloading).

Day 8: Django Admin Customization
Goal: Enhance the admin interface.**
Key Concepts:
- Customizing the admin panel using `admin.py`.
- Adding filters, search bars, and bulk actions.
Tasks:
- Display `title`, `completed`, and `created_at` in the admin list view.
- Add a filter for completed tasks.
Exercise: Create a custom admin action to mark tasks as "high priority".

Day 9: Class-Based Views (CBVs)
Goal: Simplify code with generic class-based views.
Key Concepts:
- `ListView`, `CreateView`, `UpdateView`, `DeleteView`.
- Advantages of CBVs over FBVs.
Tasks:
- Rewrite the task list view using `ListView`.
- Replace the "Create Task" view with `CreateView`.
Exercise: Add pagination to the task list.

Day 10: Deployment Basics
Goal: Prepare the app for deployment.
Key Concepts:
- Deployment platforms (PythonAnywhere, Heroku).
- Environment variables (hiding `SECRET_KEY`).
- Version control with Git.
Tasks:
- Push the project to GitHub.
- Deploy the Todo App to PythonAnywhere (free tier).
Exercise: Debug common deployment issues (e.g., static files not loading).

Daily Practice Routine (3 Hours)
- **1 Hour**: Study concepts (docs, tutorials, or videos).
- **1.5 Hours**: Hands-on coding (follow tasks above).
- **30 Minutes**: Debug errors, review progress, and document learnings.

Key Resources
1. **Django Official Documentation** (always prioritize this!).
2. **Book**: [*Django for Beginners* by William S. Vincent](https://djangoforbeginners.com/).

3. **Video Course**: [Django Tutorial by Corey Schafer (YouTube)](https://youtu.be/UmljXZIypDc).