Overview:

Chapter 1: brief explanation of web APIs and HTTP protocol
Chapter 2: Review the difference between traditional Django and Django Rest Framework
Chapter 3-4: Build a Todo API and connect to a React front-end
Chapter 5-9: Build Production-ready blog API which includes full CRUD Functionality

Chapter 1: Web APIs

- Hypertext Transfer Protocol (HTTP) was the first standard universal way to share documents over the internet
- Uniform Resource Locator (URL) is the address on the internet, when searched your browser sends a request to to the server

- HTTP Verbs
    - Verbs are a list of approved actions known as HTTP verbs
        - CRUD ---------- HTTP Verbs
        - Create POST
        - Read GET
        - Update PUT
        - Delete DELETE

- Endpoints
    - List of available actions that expose data typically in JSON format
- Statelessness
    - Important point to make about HTTP is that it is a stateless protocol, meaning each request/response pair is completely independent of the previous. No stored memory of past interactions know as a state in computer science
- REST
    - REpresentational State Transfer (REST) is an architecture that approaches building APIs on top of the web, thus on top of the HTTP protocol.
    - Every RESTful API
        - Is stateless
        - Support common HTTP verbs
        - Return data in either JSON or XML format

Chapter 2: Library Website and API

- Django REST Framework works alongside the Django Web Framework to create web APIs — cannot create API with only Django Rest Framework
- Django creates websites containing websites

- Django REST Framework creates web APIs which are a collection of URL endpoints containing available HTTP verbs that return JSON
- Traditional Django website consists of a single project and one (or more) apps representing discrete functionality
- After creating a virtual environment, create the first project containing the main files of the django backend.
    - The first folder created will contain the following files:
        - __init__.py - is a python way to treat a directory as a package
        - asgi.py - stands for Asynchronous Server Gateway Interface and is a new option in django 3.0+
        - Settings.py - contains all the configuration for our project
        - Urls.py - controls the top-level URL routes
        - Wsgi.py - stands for Web Server Gateway Interface and helps Django serve the eventual web pages
        - Manage.py - executes various Django commands such as running the local web server
- Run migrate to sync the database with Django's default settings and start up local Django web Server
    - Note: If you plan to change away from default user and create a custom user it is important to change this before the first migration
- Next set is to start adding apps
    - When a new app is created, each directory will have the following files:
        - *admin.py* : is a configuration file for the built-in Django Admin app
        - *apps.py*: is a configuration File for the app itself
        - migrations/directory : stores migrations files for the database changes
        - *models.py* : where we define our database models
        - *tests.py* : for app-specific test
        - *views.py* : where we handle the request /response logic for our web app
        - Typically developers will also create an *urls.py* file within each app for routing
- React Rest Framework
    - Pip3.10 install djangorestframework
    - Add the framework to settings.py
- Serializers
    - A serializer translates data into a format that is easy to consume over the internet typically JSON and displayed as an endpoint
    - Transforms the data from models into json
    - Exposes fields that want to have access of the api
- Views
    - Views in traditional django are used to customize what data is sent to the templates
    - In Django REST Framework views do the same thing but for our serialized data
- CORS
    - Needed to connect front end to backend