Django

1

Django – Level 1

Getting Started with Django!

Django

- We’ve finally reached the moment we’ve been waiting for - Django!
- Before we dive into the technical details of Django, let’s learn a little more about it and it's interesting background!

Django

- Django is a free and open source web framework.
- It is used by many sites, including Pinterest, PBS, Instagram, BitBucket, Washington Times, Mozilla, and more!

Django

- Django was created in 2003 when the web developers at the Lawrence Journal-World newspaper started using Python for their development.
- The fact that is originated at a newspaper is important!

Django

- Because the original developers were surrounded by writers, good written documentation is a key part of Django!
- This means you have excellent references to check on the official Django docs!

Django

- Django has its own excellent basic tutorial where you are walked through creating a basic polling web app.
- The reason it is a poll also extends back to its newspaper roots!

Django

- When encountering Django tutorials you will often read that you should create a virtual environment or an “venv”
- Let’s talk about what this is and how to use it!

Django

- A virtual environment allows you to have a virtual installation of Python and packages on your computer.
- So why would you ever want or need this?

Django

- Packages change and get updated often!
- There are changes that break backwards compatibility.
- So what do you do if you want to test out new features but not break your web app?

Django

- You create a virtual environment that contains the newer version of the package.
- Luckily, Anaconda makes this really easy for us!
- A virtual environment handler is included!

Django

- To use a virtual environment with conda we use these commands:
  - `conda create --name myEnv django`
- Here we created an environment called “myEnv” with the latest version of Django.

Django

- You can then activate the environment:
  - `conda activate myEnv`
- Now, anything installed with pip or conda when this environment is activated, will only be installed for this environment.

Django

- You can then deactivate the environment
  - `conda deactivate`
- Its encouraged to use virtual environments for your projects to keep them self-contained and not run into issues when packages update!

2

Django – Project

Creating our first django project!

Django

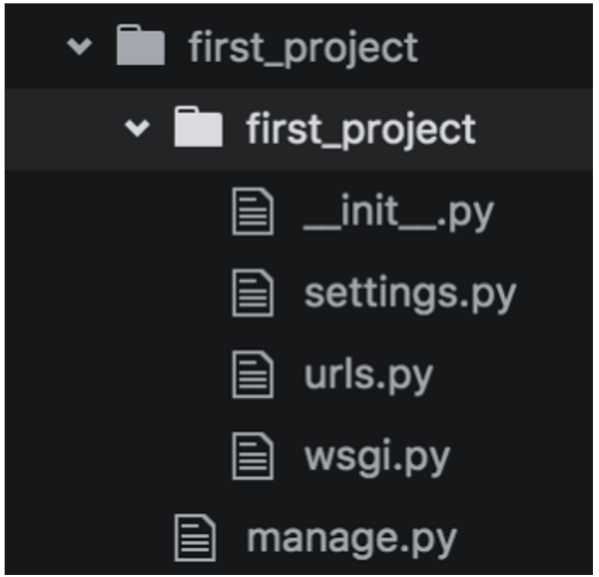
- You can install Django with
  - `conda install django`
- Or for normal python distributions:
  - `pip install django`

Django

- When you install Django, it actually also installed a command line tool called:
  - `django-admin`
- Let’s create our first project. Type:
  - `django-admin startproject first_project`

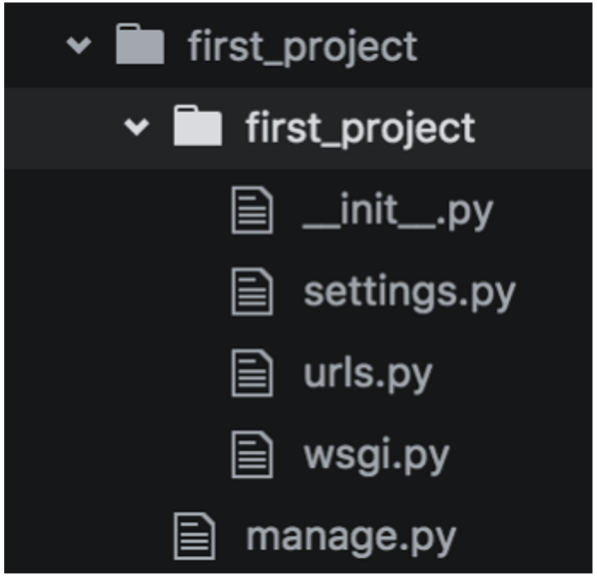
Django

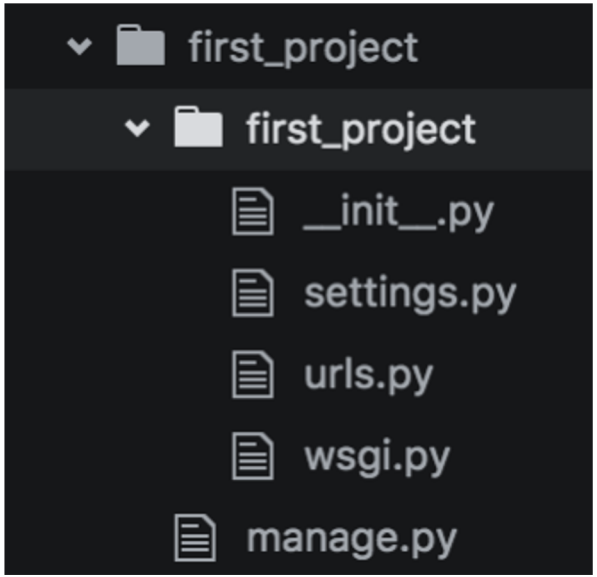
- You will then get something that looks like this:



Django

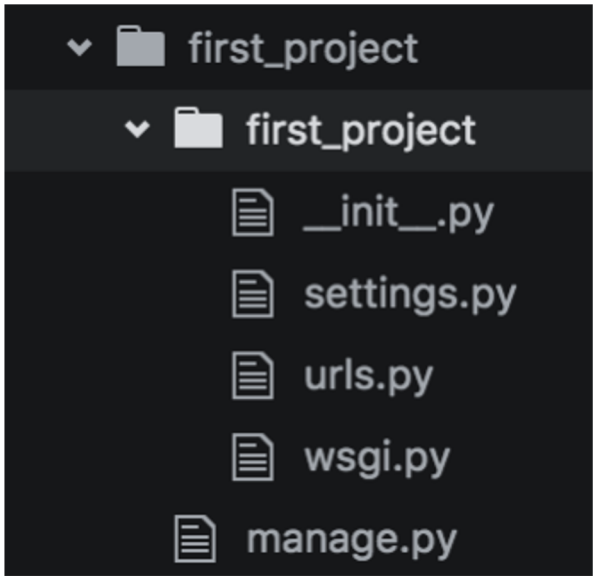
- Let’s explain what is going on here!



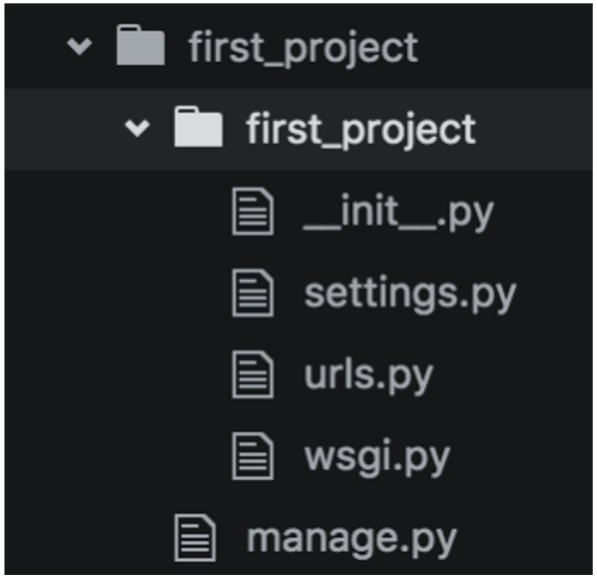


- __init__.py
  - This is a blank Python script that due to its special name let's Python know that this directory can be treated as a package

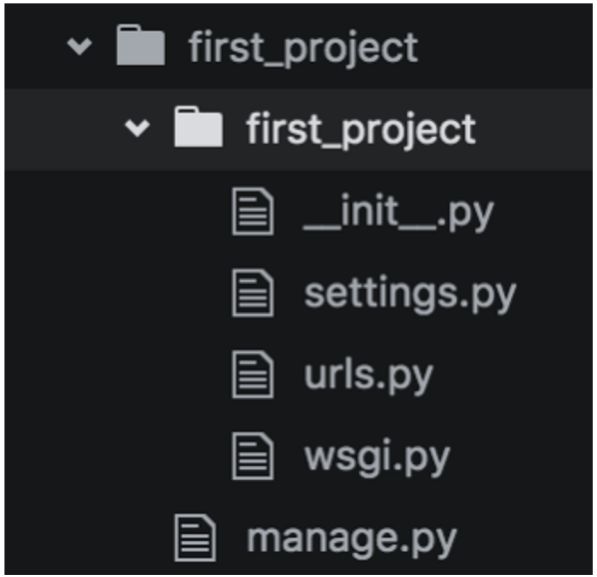
Django



- settings.py
  - This is where you will store all your project settings

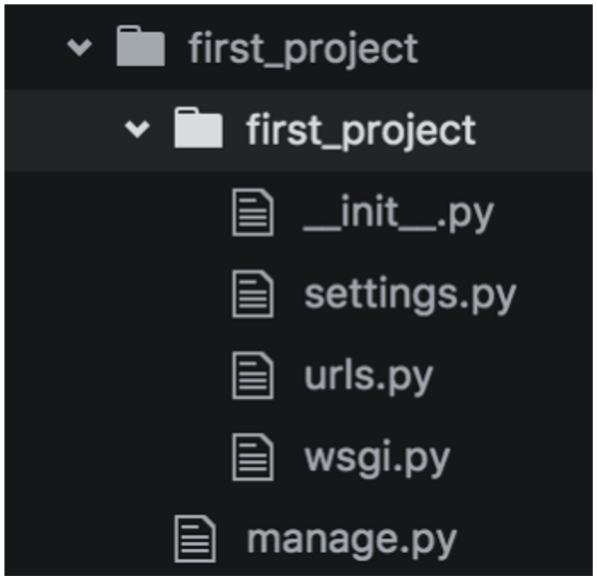


- urls.py
  - This is a Python script that will store all the URL patterns for your project. Basically the different pages of your web application.



- wsgi.py
  - This is a Python script that acts as the Web Server Gateway Interface. It will later on help us deploy our web app to production

Django



- manage.py
  - This is a Python script that we will use a lot. It will be associates with many commands as we build our web app!

Django

- Let’s use manage.py now:
  - `python manage.py runserver`
- You will see a bunch of stuff but at the bottom you will see something like:

```
Django version 1.10.5, using settings
'first_project.settings'
Starting development server at
http://127.0.0.1:8000/
```

Django

- Copy and paste that url into your browser
  - http://127.0.0.1:8000/
- You should now see your very first web page being locally hosted on your computer.
- Congratulations!

Django

- You should have also noticed a warning about migrations.
- This has to do with databases and how to connect them to Django
- What is a Migration?

Django

- A migration allows you to move databases from one design to another, this is also reversible.
- So you can “migrate” your database
- We will touch back on this later, for now you can ignore this warning.

Django

- That was the basics of getting started with Django!
- Up next we will continue by creating a very simple Hello World Django Application!

3

Django – Project

Creating our first django application!

Django

- So far we have been able to use runserver to test our installation of Django.
- Now let’s move on to creating our first Django Application.
- We’ll learn about views and how to use them.

Django

- Let’s get some terminology straight:
  - A Django Project is a collection of applications and configurations that when combined together will make up the full web application (your complete website running with Django)

Django

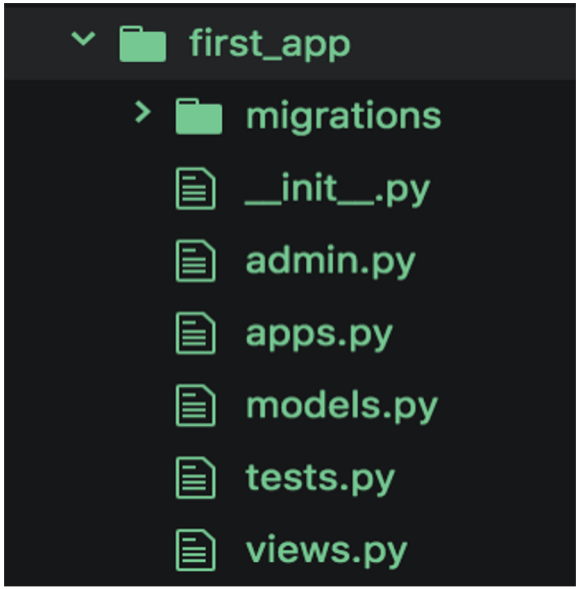
- Let’s get some terminology straight:
  - A Django Application is created to perform a particular functionality for your entire web application. For example you could have a registration app, a polling app, comments app, etc.

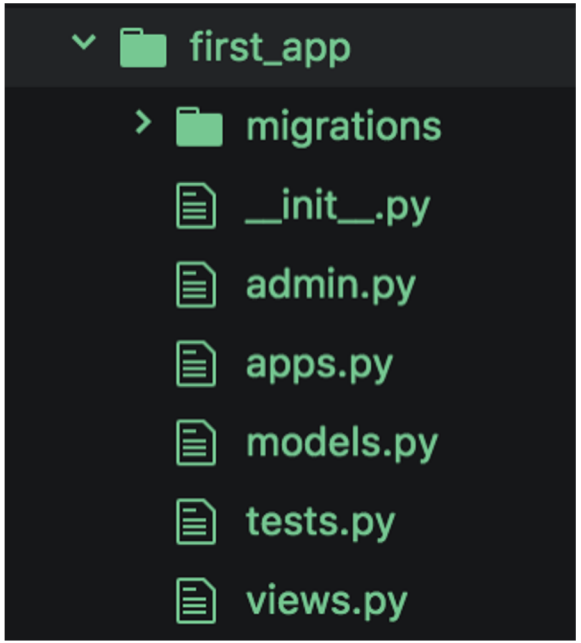
Django

- These Django Apps can then be plugged into other Django Projects, so you can reuse them! (Or use other people's apps)
- Let's create a simple application with:
  - `python manage.py startapp first_app`

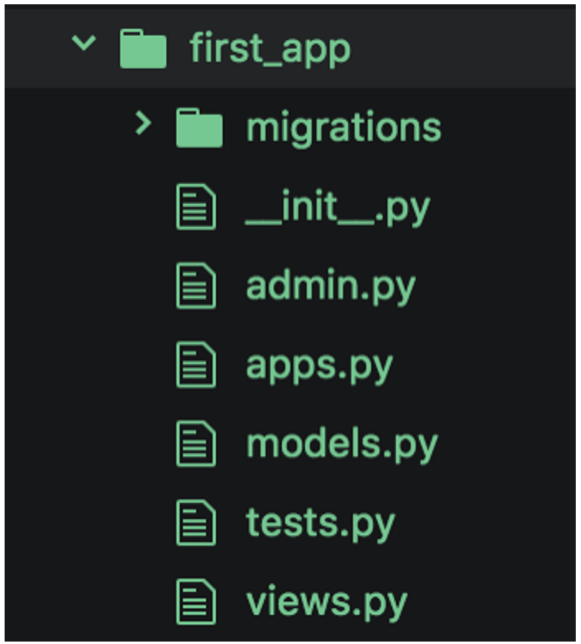
Django

- Let’s quickly discuss all of these files!

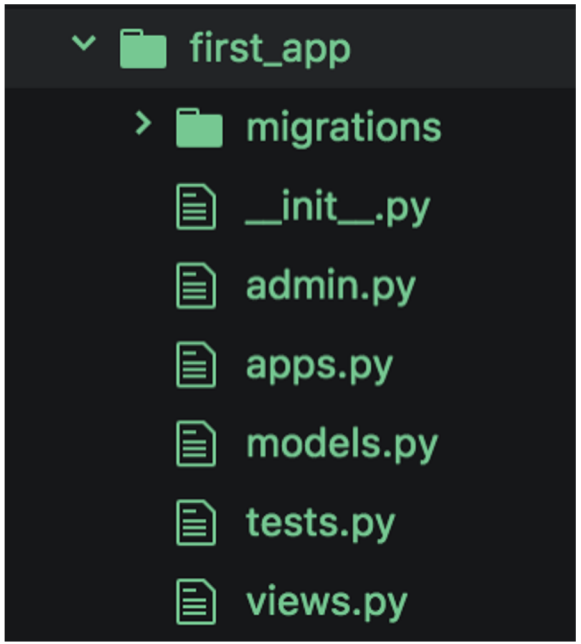




- __init__.py
  - This is a blank Python script that due to its special name let's Python know that this directory can be treated as a package

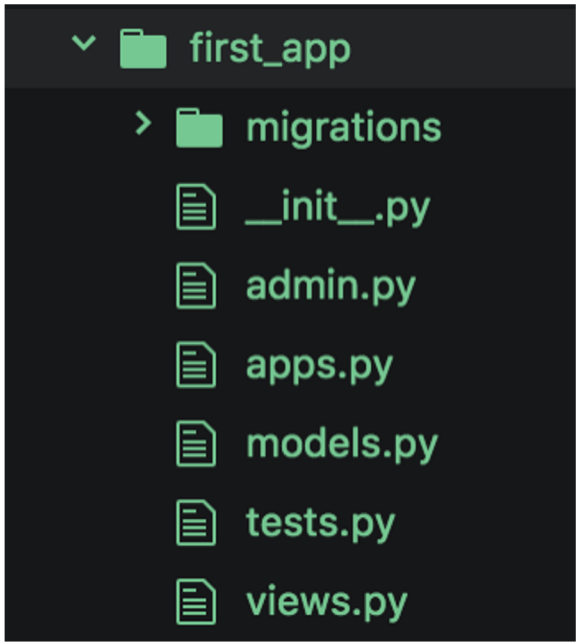


- admin.py
  - You can register your models here which Django will then use them with Django's admin interface.



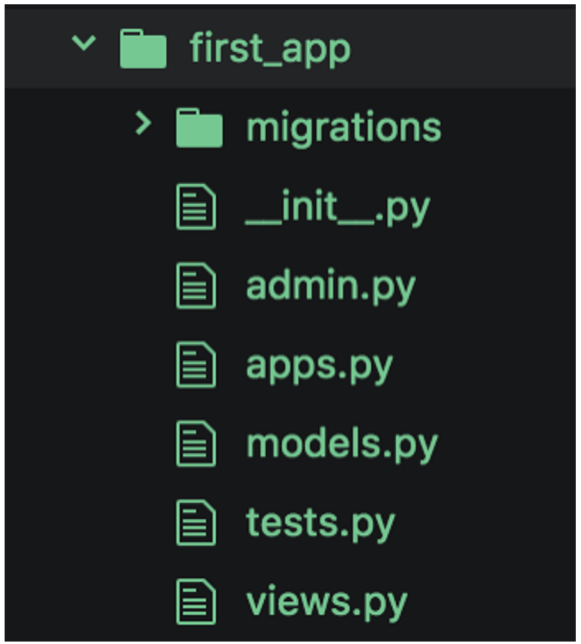
- apps.py
  - Here you can place application specific configurations

Django



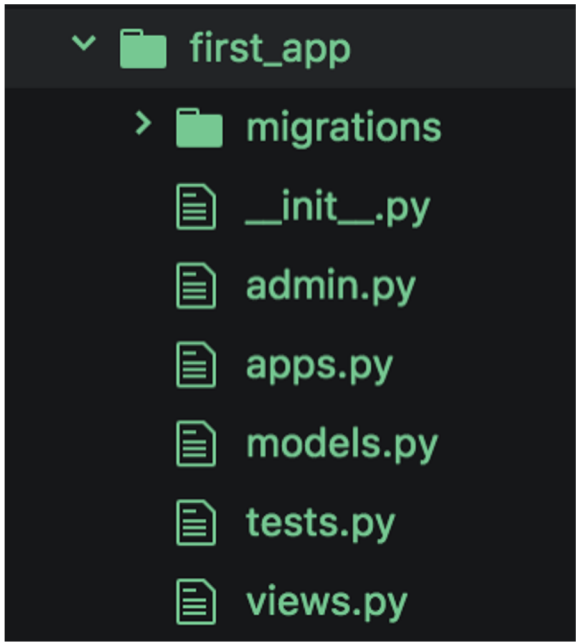
- models.py
  - Here you store the application’s data models

Django

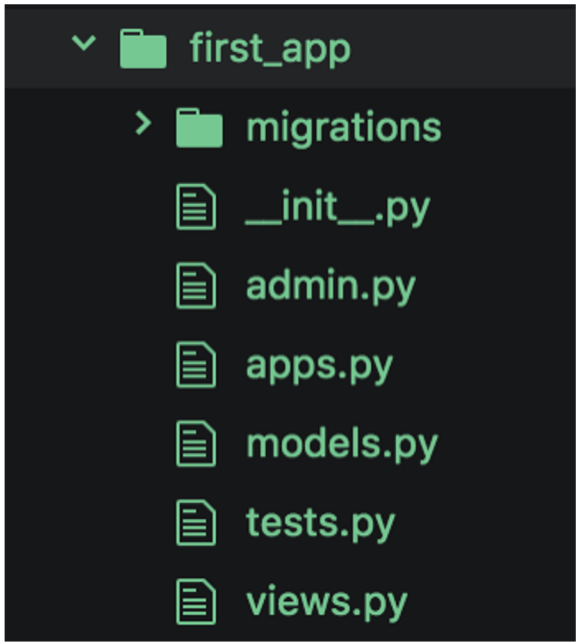


- tests.py
  - Here you can store test functions to test your code

Django



- views.py
  - This is where you have functions that handle requests and return responses



- Migrations folder
  - This directory stores database specific information as it relates to the models

Django

- Now let’s learn the process of creating a view and mapping it to a URL!

4

Django – Challenge!

Time to put your skills to the test!

Django

- We’ve learned enough now that before we continue to learn about URL mappings, we should challenge you to make sure you can test your new skills!

Django

- Complete the following tasks:
  - Create a New Django Project: “ProTwo”
  - Create a New Django App: “AppTwo”
  - Create an Index View that returns:
    - My Second App
  - Link this view to the urls.py file

Django

- Best of luck, you already have all the knowledge needed to complete this!

5

Django – Mapping URLS

Let's quickly cover some more URL mappings!

Django

- As we continue on through the course we are going to be dealing with mapping URLs quite a bit!
- There are several ways of doing this, let’s briefly touch upon another way!

Django

- We previously showed a very direct mapping from the views.py to the urls.py
- Now we want to show the ability of using the include() function from django.urls

Django

- The include() function allows us to look for a match with regular expressions and link back to our application’s own urls.py file.
- We will have to manually add in this urls.py file

Django

- So we would add the following to the project's urls.py
  - `from django.urls import include`
  - `urlpatterns = [ ... url('first_app/',include('first_app.urls')), ...]`

- This would allow us to look for any url that has the pattern:
  - www.domainname.com/first_app/...
- If we match that pattern, the include() function basically tells Django to go look at the urls.py file inside of first_app folder

Django

- This might seem like a lot of work for a simple mapping, but later on we will want to try to keep our project's urls.py clean and modular
- So we set the reference to the app, instead of listing them all in the main urls

Django

- Let’s quickly walk through an example of all of this to show how it works!

6

Django – Templates

Let's learn how to use Templates!

Django

- Templates are a key part to understanding how Django really works and interacts with your website.
- Later on we will learn about how to connect templates with models so you can display data created dynamically.

Django

- For now, let’s focus on the basics of templates and template tags.
- The template will contain the static parts of an html page (parts that are always the same)

Django

- Then there are template tags, which have their own special syntax.
- This syntax allows you to inject dynamic content that your Django App’s views will produce, effecting the final HTML

Django

- To get started with templates you first need to create a templates directory and then a subdirectory for each specific app's templates.
- It goes inside of your top level directory:
  - first_project/templates/first_app

Django

- The next step is to let Django know of the templates by editing the DIR key inside of the TEMPLATES dictionary in the settings.py file.
- However, there is an issue we have to deal with before we do this!

Django

- We want our Django Project to be easily transferrable from one computer to another, but the DIR key will require a “hard-coded” path
- How do we resolve this?

Django

- We can use Python's os module to dynamically generate the correct file path strings, regardless of computer!
- Import os and try out the following:
  - `print(__file__)`
  - `print(os.path.dirname(__file__)`

Django

- We will use this os module to feed the path to the DIR key inside of the TEMPLATES dictionary.
- Once we've done that we can create an html file called index.html inside of the templates/first_app directory

Django

- Inside this HTML file we will insert template tags (a.k.a Django Template Variable).
- These template variables will allow us to inject content into the HTML directly from Django!

Django

- This is now starting to reveal the power of why we would use a Web Framework
- Django will be able to inject content into the HTML
- Which means we can later on use Python code to inject content from a database!

Django

- In order to achieve this, we will use the render() function and place it into our original index() function inside of our views.py file.
- Let’s now code through everything we just discussed!

7

Django – Templates Challenge!

Test your knowledge of Templates!

Django

- Templates is a big leap forward for us, so it is a good time to quickly practice using them!
- We will use your older ProTwo project (recreate it if you no longer have it)
- Complete the following tasks...

Django

- Create a templates directory and connect it to the settings.py file
- Create a new view called help and use url mapping to render it for any page with the extension /help
- Add template tags to return “Help Page”

Django

- Best of luck

8

Django – Static Files

Learn how to insert static media files.

Django

- So far we've used templates to insert simple text.
- But we don't always just want text, what about other types of media, for example, returning a User's Photo?
- Let's discuss static media files!

Django

- To do this, we will create a new directory inside of the project called static (just like we did for templates)
- Then we will add this directory path to the project's settings.py file
- We will also add a STATIC_URL variable

Django

- Once we've done that we need a place to store our static image files
- We create a directory inside of static called images
- Place a favorite .jpg file inside this images directory (or just download one)

Django

- To test that this all worked you can go to:
  - 127.0.0.1:8000/static/images/pict.jpg
- That will confirm that the paths are set up and connected properly.
- But what we really want to do is set up a template tag for this!

Django

- To do this inside an html file, we add in a few specific tags, at the top:
  - `{% load staticfiles %}`
- Then we want to insert the image with an HTML <img src= > style tag using:
  - `<img src="{%static "images/pic.jpg" %}" />`

Django

- Notice how this template tag is a little different in that it uses
  - `{% %}`
- instead of
  - `{{ }}`

Django

- We will discuss and show these differences more clearly in future lectures, but for now consider {{ }} as being used for simple text injection, and we can use {% %} for more complex injections and logic

Django

- Now let’s code through an example of serving up a static image!
- Afterwards we can dive into models and databases!