Summary

In the final rendition of our Python project, we present a web layout and design built on a Django framework. As warned, the Django framework proved finicky to navigate, but we managed to build the site on it. The webscraper was built to pull news relevant to "Black Lives Matter" from the website newsapi.org. The search term can easily be altered whenever. The forloop we used to write the code is stored in 'home.html' surrounded by the {% block body %} tags that Django requires. We then used these in 'base.html' to call the python code for the webscraper into the html for the rest of the webpage. Once in the webpage, you can see the html and css code that was used to construct a modest site with an 'about', 'team', and 'news' section (where the webscraper comes alive!).

Django Framework

The Django framework is built on two directories: *news* and *newsweb*, with news being the baseline directory storing important information about the entire program and *newsweb* storing important directional information as well as the templates and static files.

News Baseline

In *news*, you can see the *settings.py* file, where we import important functions from Django, install apps, including *newsweb* and other important apps such as *django.contrib.staticfiles*, which has the instructions to read static tags. We also define the templates we later build (the html files themselves), set the base directory so that Django knows where to find it, and also define the static root, where all images and css are stored. These settings tell Django where our important files are and how to configure the site.

Also importantly in the *news* directory is the first *urls.py* folder which is where we import the commands for *path* and *include* as well as set the *url.patterns* for the home page and to read the *newsweb.urls.* This file crucially helps Django understand the structure of the app.

Newsweb App

Moving into *newsweb*, there are many elements of the final site. *static* contains all images stored on the site, clearly labeled. *templates* contains the *base.html* and *home.html* files as discussed above. The *urls.py* file reiterates the *url.patterns*, this time naming the index page as *home.html* and adding direction for the static files.

Finally, we reach the most important part of the Django framework (as is relevant for the webscraper), the *views.py* file. We first create the home function with the *request* package. Requests come in from newsapi.org in the *json* format. We use *request.get* to set the parameters for what python requests from newsapi.org: the url to search, the query, dates to search through, a sort by method, and an api key. We then tell it to return to the *home.html* file. And voila! Through a magical combination of 1.2million puzzling files, Django came through for us and compiled a webscraper!