Talking Points

* Data source (Kaggle) from Washington post and the Guardian, two csvs one from 2015, other from 2015-2017 (labelled as 2015-present), geoJson
* Quick run through of web dashboard
* 2015 dataset from the Guardian had latitude/longitude data which was used for the markers map
* 2015-2017 data from the Washington Post was used for our choropleth map
* GeoJson was used to outline each state in choropleth
* Cleaning the data and extracting the necessary columns and making pie charts
* Read in the csv and load into pandas dataframe for both sets of data
* Extracted columns needed for both dataframes and why
* Added full state names to the pandas dataframes
* Replaced race from the format in the csv/datasets
* Pie chart creation – 2 approaches
* Create Database connection and make tables from pandas dataframes
* Used SQLAlchemy to create a SQLite database in jupyter notebook
* Once pandas dataframes were cleaned and loaded with correct data, .to\_sql injected the table into the database
* Flask stuff
* Used flask-sqlalchemy to connect to the SQLite database
* Used the db.Model method to create classes for each dataset
* Created 4 routes, 1 for the intro, 3 others for visualizations
* Two dynamic REST endpoints were created for each dataset, with each endpoint changing based on the specified race
* Use of jinja templates
* Used layout.html to make a layout which can be extended onto other html pages
* url\_for used to access files
* web scraper value was passed as a variable onto each page which can be accessed through jinja
* Animation (three.js) library
* Webscraping from Washington post
* Used beautiful soup to collect the number of killings in the Washington post header
* Javascript visualizations (choropleth, markers)
* GeoJson structure, had to get states and inject number of killings into each state properties
* Show flask endpoint for one race
* The createChoropleth function grabs the race data from the flask endpoint and counts the number of killings per state
* This number is injected into the GeoJson which is the value used to color each state
* Dropdown menu, how the different maps show change based on selection
* Flask Endpoint which shows race, latitude, longitude
* Explain layers and how each marker is segregated, L.control.layers to add dropdown menu
* Created markers from these latitudes and longitudes
* Pop-up was added to show state and race on click
* Used L.icon to create different markers based on race
* Uploaded to Heroku
* Created new virtual env and pip installed needed packages
* Pip freeze returns a list of requirements which can be copy pasted into a requirements.txt file
* Runtime.txt to specify which version of python to install, and gunicorn to run the app