Over the course of the summer, I worked over a few coding projects. Here are the details for all of them.

**read_anenometer_wc.py**

After creating a serial connection to the anemometer, this file allows you to run it with the command line arguments of downstream distance, spanwise distance, and duty cycle, in that order, and automatically creates a csv file containing the raw data with those parameters in the name. Allows for easy organization of the anemometer files.

**do_calc_plot.py**

This code sorts through all csv files generated by **read_anenometer_wc.py** and generates a summary table of each test sample's downstream distance, spanwise distance, duty cycle, mean, standard deviation, and turbulence intensity, and saves it all as a csv. It then constructs graphs at each downstream location that was tested and plots the mean velocity as a function of spanwise distance for each duty cycle, using the standard deviations as error bars. It generates these graphs at all different downstream distances used for the testing purposes.

**do_calc_plot_new.py**

This file reads raw data uploaded from a google sheet me and my partner used to record wind characterization data. It sorts through all the sheets inside the Google Sheet file, locates the ones in the format of "span, downstream, duty cycle" eg "1, 2ft, 10%" and parses those numbers as well as reading the actual file to calculate mean, standard deviation and turbulence intensity. After sorting through the entire sheet, it outputs a csv containing a table of each test's span, downstream, duty cycle, mean, standard deviation, and turbulence intensity. Furthermore, it constructs graphs at each downstream location that was tested and plots the mean velocity as a function of spanwise distance for each duty cycle, using the standard deviations as error bars. It generates these graphs at all different downstream distances used for the testing purposes.

**Because my partner and I had already started recording our data with google sheets, we refrained from using **do_calc_plot.py** and **read_anenometer_wc.py** and instead used **do_calc_plot_new.** I think using the former is better though as it does record all of the data for you (less clicking buttons) instead of the user having to do it manually. Both of these graphs do end up creating the same formatted summary table csv that is necessary for the next script.

**getgraphs.py**

This script reads from a summary table of test samples with the columns of downstream distance, spanwise distance, duty cycle, mean, standard deviation, and turbulence intensity, generating velocity and turbulence field graphs with axes of downstream and spanwise

distance, for different duty cycles, as well as corresponding scatter plots of velocity and turbulence intensity as a function of spanwise distance.

**sphereplots.py**

This code reads into a test matrix excel sheet me and my partner made. For the file, there are two sheets with different testing scenarios (one for cones, one for circles). This file reads into the columns of our velocity and drag force columns and generates a graph of Drag Coefficient vs. Reynolds number.

We also created a test procedure for testing our tethered spheres.