1. Download packages/requirements, set up environment (note: this is best done within a linux environment, but it is possible on windows using windows subsystem for linux as of the date this document was created)
    a. Download root: https://root.cern/install/
        i. If working with windows, it should be available as a beta release
    b. Create a python environment with the following version dependencies saved in requirements.txt: https://github.com/herolada/tth-mass-reco/blob/main/requirements.txt
        i. The link above may be used to create an anaconda environment, but working from an anaconda environment is obviously not a requirement.
        ii. When I was working on windows, I used linux subsystem for windows and downloaded my packages and python version on the linux side from the command line.
2. Running the code
    a. Take note that these steps must be followed in order. Also, the constants.py file may need to be moved if working with Adam's version of the code into the directory with the file you are running.
    b. Adam's version of the code from his thesis is available publicly on GitHub: https://github.com/herolada/tth-mass-reco
    c. Emily's version building off of Adam's work is available on GitHub: https://github.com/emahannon/tth-mass-reco
    d. Extract data from root files
        i. Run the file **root_data_extraction/root_data_extraction.py**
            1. If you are working with a different/modified set of variables (features), you will need to modify the *used_ntuple_variables* array to **include the variables you are using**. You may also need to modify the *event_to_write* variable in the for loop on line 505. Refer to the comments and variable names in this code block for guidance.
            2. You will also need to **modify the paths to your data** depending where it is stored on your machine. The path will be set at the *data_folders* variable in the main method, which is near the end of the file.
            3. The data should be extracted to the **data** directory. Refer to the last few code blocks at the end of the **root_data_extraction.py** file for specific locations.
    e. Run particle assignment
        i. Run the file **particle_assignment/particle_assignment_training.py** (this is also be runnable as a jupyter notebook .ipynb file of the same name)
            1. You should be able to just run this file with no modifications. If not, the path to the data file may have to be adjusted beginning on line 181.
        ii. Run the file **particle_assignment/particle_assignment_extraction.py** (this is also be runnable as a jupyter notebook .ipynb file of the same name)
            1. This file should also run with no modifications unless your data directory is in a different location than specified in the file.
    f. Run mass reconstruction

Using Trexfitter with Docker:

1. Download docker and log on to Trexfitter environment.
2. Commands to transfer data into docker (from a Windows command line):

   Transfer to docker (replace the 0s with code associated with docker container):

   docker cp 00000000:/filePath fileName

   Transfer from docker:

   docker cp fileName 00000000:/filePath

3. Run Trexfitter commands from the Trexfitter folder. Not from the config folder.

   I used this command initially:

   trex-fitter nd config/ttH_2lSS1tauhad_Nazim.config "Samples=ttH"

To change the number of features in the particle assignment portion of the code (this will be useful for looking at feature importance, etc.)

1. You will only be working with the document particle_assignment_training.ipynb
2. Inside the DataGenerator class in the data_generation function, modify the good_feature_names variable to include *only the features that you want to use*.
   a. Note: you must always include the 'event id' and the 'num permutations' as these will be handled later.
3. Modify the number of features. This has been hard coded and you will have to change it in multiple places. You will not be including the two variables 'event id' and 'num permutations' I your calculation of the number of features. It may be helpful to know that the original number of features I used for the ttH is 61.
   a. On the line X_train = np.empty((num_samples,61), dtype=float) change the second parameter inside the second parenthesis to reflect the number of features.
   b. On the line model = lep_assignment_model(61, initial_bias) change the first parameter to reflect the number of features.
4. Before you can train the model, you need to retrain the scaler first.
   a. Inside the DataGenerator class in the data_generation function, comment out the line X = (X-self.scaler[0])/self.scaler[1] and uncomment the line X=X
   b. Uncomment the scaler training block of code

```
# scaler = StandardScaler()
# scaler.fit(X_train)

# f = open("../scaler_params/particle_assignment_scaler_params.csv", "w")
```

```
# writer = csv.writer(f)
# writer.writerow(scaler.mean_)
# writer.writerow(scaler.scale_)
# f.close()
```

c. Run until you get to the line f.close(). After this line runs, the scaler will be trained.
d. Revert your changes made in step a and b.

5. Run the entire document of code. The feature importance graph should reflect the number of features you have specified. https://www.geeksforgeeks.org/box-plot-in-python-using-matplotlib/