To create this file I made the following main functions:
1. an __init__ function to create the object. It takes in the input of jointfile and beamfile and creates dictionaries for each respective file. The dictionary that stores the data from jointfile initializes 4 different dictionaries within the self.joints[joint] dictionary (a dictionary within a dictionary). The dictionary that stores the beam data is also a dictionary within a dictionary, wherein the (ja, jb) data sets map directly to the beam number. These dictionaries will be called later on when creating the matrix. The __init__funciton also computes the direction of force given by the beams in the form of (cos, sin). This does so by calculating the difference between x1 and x2, and y1 and y2 coordinates on the joints, then calculating the direction of force by dividing these by the square root of deltax plus delta y. The beam number is also stored here, to be used later when establishing the column index.

2. An EstablishMatrix function, that creates the linear system Ax = b, and returns the sparse matrix A and vector b through initialization through csr_matrix. This will help solve for the unknown x vector. Matrix A is organized as follows : the rows order of A is the equations for [joint1x coordinate, joint1 y coordinate, joint 2 x coordinate, joint 2 y coordinate ……. all the way till the last joint]. The column order is the corresponding Beam1 Beam 2….last beam, and reaction force of joint 1 x coordinate , reaction force of joint 1 y coordinate values. The B vector will just be the external forces (the Fx and Fy values).

This does this by iterating first through the columns of the matrix (the beams) then matching them to the corresponding row value (the joint). Csr_Matrix is used to create the sparse matrix through the arrays row_idx, col_idx and val_data (they store the row index, column index, and value of the data, respectively). The direction of the force is then appended to the matrix through the val_data array. I made two for loops for appending the joint (one for the joints x coordinate and another for the y coordinate). I also made sure to append the column index and row indexes for every for loop iteration. To append the reaction force, I made another for loop that appended 1s or 0s if there a reaction force was present. I also made sure to append the Fx and Fy values for matrix B for every iteration of the beam. I also made sure to check if the number of rows and columns were equal, through raising a RuntimeError if the length of matrixB did not match the maximum column index value.

3. a __repr__ function, which prints the tension force of the beams. This does so by calling the EstablishMatrix function on matrix A and vector B, then solving this matrix to get the values in matrix x. These values are then appended to a list that prints it in the desired format.

4. a PlotGeometry function, which plots the geometry of the truss.

This does to by recording the two ends of a beam, then drawing a line between these two joints to create the figure.