Truss Analysis with Method of Joints

`Truss` Class:

Truss class is used for analyzing a 2D truss system with external loads, using the method of joints. It consists of `__init__`, `PlotGeometry()`, `Analysis()` and `__repr__()` methods. Input arguments required to initiate the class are joints and beams files. An optional plot file name can be the third input if the user wants to plot the geometry of the truss system.

`__init__(joints_file,beams_file) :`

Gets the file names as input and stores into local variables. Then initiates the `ImportFiles()` method.

`ImportFiles():`

Saves the data in joints and beams file in seperate variables. Finds total beam and node numbers, and fixed joints.

`PlotGeometry(plot_file):`

Plots the truss geometry and saves it to the current directory with the name as given input plot_file.

`Analysis():`

Executes the static analysis of the truss system with method of joints, using sparse matrix $T_{n\times n}$ and vector $F_{n\times 1}$ where n is the number of unknown forces in each node in 2D plane. ($n = 2\times$ # of joints). This is held in these steps:

- Initiates T and F
- In a for loop for joints, each joints coordinates are stored
- In a nested for loop to this loop, each beam is iterated for each joint and checked if it is connected to the current joint or not. If it is connected, the other joint of beam is found and its coordinates are stored.
- Distances between the current joint and beam's other joint in x and y direction are calculated, and beam length (hip) is found.
- $cos()$ and $sin()$ of the angle between beam and the joint are calculated and stored in T matrix, which are the coefficients of beam forces in summation of forces in x and y directions.
- At the end of joint loop, joint is checked whether it is a fixed one or not, if it is fixed, the coefficient for reaction force, which is 1, is stored in T matrix in the corresponding index.
- After the joint loop, external forces are stored into F vector. F is multiplied by -1 since the original equation is $TM + F = 0$ and we are solving $TM = -F$
- $TM = -F$ is solved with `scipy.sparse.linalg.spsolve()`, where T consists of coefficients of unknown forces, M, being the unknowns which are sorted as $[F_1, F_2, ..., F_n, R_1, R_2, ...R_n]$ where $F_1, F_2...F_n$ are the beam forces and $R_1, R_2...R_n$ are the support reactions.
- It is checked if the number of equations are equal to the number of unknowns, if not, system is overdetermined, and a `RuntimeError` is raised.
- It is checked if singularity warning was raised by `spsolve()`, if yes, which leads to an unstable system, a `RuntimeError` is raised.

`__repr__():`

Overloads `print()` for the class. Initiating `print(truss)` displays the resulting force in each beam in two columns. First column is the beam number and second is the resulting force in that beam.