HW5 numpy CS103 Fall 2020 Johnstone due: 11.18.20

Materials (what I give you)

- hw5_20fa103.pdf (this document)
- hw5_20fa103.py
- hw5_concise.py (this version is equivalent to hw5_20fa103.py except stripped of docstrings; this is given to you to more easily see the structure of the set of functions; you can submit this version if you prefer)
- tri_4x4.txt (an example triangle data matrix)
- soon: an advanced version of the homework for an A (building on this basic version)

Deliverables (what you submit on Canvas)

- hw5_20fa103.py (either basic or advanced versions)

structure

This homework practices what you have learned in the last few homeworks, such as iteration and arithmetic computation, and adds in a treatment in terms of numpy. Note that there is no recursion in this homework.

constraints

- only 4 modules are necessary, and only 4 are allowed: numpy/turtle/random/math
- 80-character lines
- use the NumPy scalar type np.float64 throughout (this is the default, which makes it easy)
- a point in this homework will always be a NumPy array with shape (2,), not a tuple or list of length 2 (as we used before)
- only the driver function is called directly in a test call; all the other functions are called indirectly by other functions
- no teams on this homework: we will return to standard solo homework for this last homework
- leave all the assertions in the code

functions to implement

Implement all of the given functions. Do not add other functions. Every function should be called by some function (that is, you should use every function). I have specified where to add code by 'ADD CODE HERE'. The assertions should be left in the code: they help guide you in the right direction. The helper functions I have given you are hints to proper implementation. Although the docstrings are often quite long, the code is usually quite short.

- randomPt

- randomTri
- vtx
- angle
- randomRobustTri
- buildTriData
- writeData
- writeTriData
- readTriData
- drawTri
- drawManyTri
- driver

the data matrix

A data matrix is a matrix (2-dimensional np.ndarray of shape (m,n)) where each row of the matrix represents a piece of data. If the dataset is a set of points in 3-space (a point cloud), the piece of data would be a point in 3-space, and each row would contain 3 scalars. If the dataset is a set of triangles in 2-space, the piece of data would be a triangle in 2-space (say defined by three points), and each row would contain 6 scalars (x1 y1 x2 y2 x3 y3). I have given you an example of a triangle data matrix in tri_4x4.txt, prefaced by its shape. You could also imagine a data matrix of lines, of circles, and so on. It may interest you that data matrices are a fundamental way to store and compute with data in machine learning.