***BE 1600***

***Introduction to***

***Programming and Computation***

***Python***

**Assignment 05**

60 points

**Due 07/27/2021 (11:45 A.M.)**

Assignment Objectives:

* To use list of lists (2D Array)
* To modify a nontrivial data mining application
* To practice writing recursive functions

*Solution for this assignment will not be posted on Canvas; however, the solution of any of the assignment problems can be discussed in the class upon request of a student.*

All assignments must be submitted by the Canvas. **No email or hard copy** is accepted. You must follow the following format:

1. For non-programming questions, use a word file to type your answers. Don’t use the text box on the Canvas to answer the questions or to write comments, we will not read it. State your answer clearly.
2. For programming questions, include only the source file of each programming problem.
3. Submit your files to the Canvas. You must submit your files on time; otherwise, you will receive zero.
4. Use “Add Another File” feature on Canvas to upload each additional file; do not upload the files as a compressed folder.
5. You can upload your files as many times as you like. Only the last attempt counts because the last files you uploaded are the only files your instructor will see.
6. There will be several modules on the Canvas. You need to upload your files using the correct module on the Canvas.
7. Name each file: *Assignment (assignment number)* for the word file [e.g. Assignment 02] and *Assignment (assignment number) \_ (Question number)* for each programming question [e.g. Assignment 02\_Q03].
8. To upload your file(s):

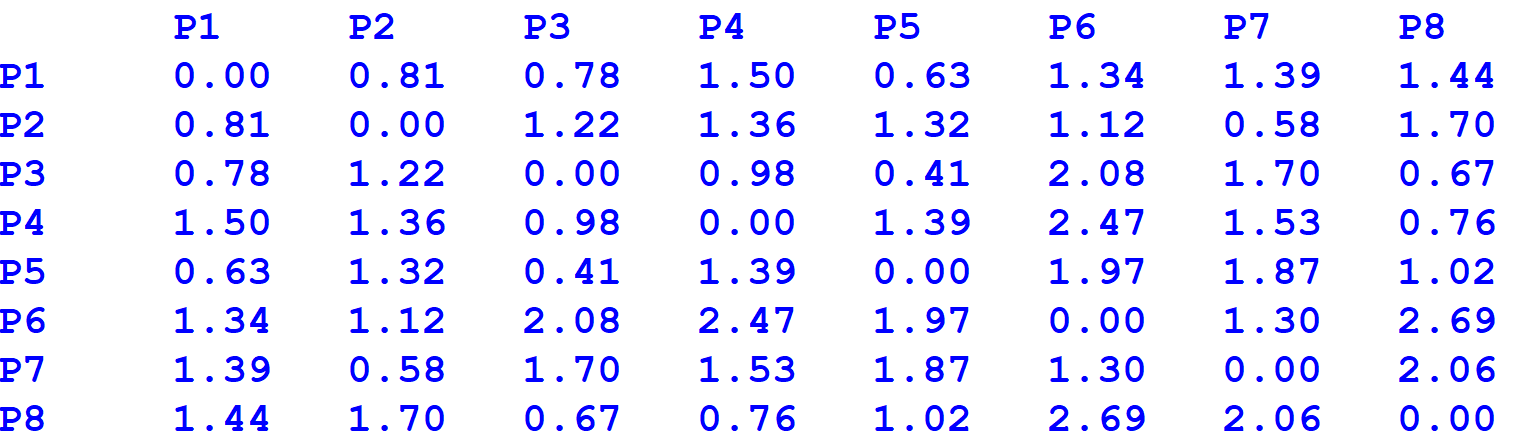
* In Course Navigation, click the ASSIGNMENTS module.
* Click the title of the assignment.
* Click the **Submit** Assignment button.
* Add **File**. ...
* Add Another **File**. ...
* **Submit** Assignment. ...
* View **Submission**.

*It is your responsibility to make sure that each file is uploaded correctly. If you uploaded a wrong file, you receive zero; files will not be accepted after due date even if you have a prove that the file is created before the due date.*

***Make sure you review the Cheating & Plagiarism policy on Canvas***

Use Python Script Window and write a program for Q.1. to Q.7. Submit 7 files .py files to Canvas by the due date. Do not include text files, only .py files.

1. **(10 points)** write a program that read 2D data points from the data.txt file (available on Canvas) and calculate the distance of each point from all other points; use the Euclidean function discussed in Chapter 07. The program should store all distances in a list of lists(two-dimensional array), then, print it as shown below.



1. **(20 points)** Using the earthquake data, cluster the quakes by their depth instead of by latitude and longitude. Use latitude and longitude to plot each earthquake location.

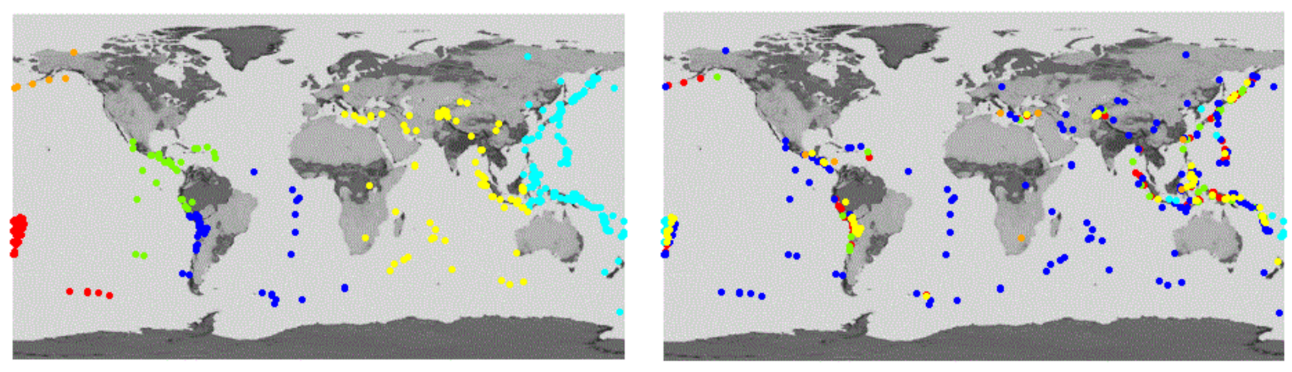
Modified the code discussed in Chapter 07 to solve this problem. The entire code is included in a single file (visualizeClusters.py). You will also need the earthquake data file (earthquakes.csv) and the world map image file (worldmap.gif) to run the program; all files are available on Canvas under assignment’s link.

Submit only the modified file, visualizeClusters.py.

Hints: Depends on your solution, you may need to modify:

1. readEarthquakeFile function: read the depths in addition to locations.
2. createCentroids function: returns a list of depths instead of a list of locations.
3. euclidD function: returns the distance between two depths instead of two locations.
4. createClusters function: recomputes the centroids from depths instead of locations

A sample output for the original code (left) and an output for a possible solution (right) are shown below; your output does not have to be similar since the initial centroids are chosen randomly.



1. **(6 points)**

Design a recursive function that accepts an integer argument, n, and prints the numbers 1

up through n.

*Sample output for n = 5*



1. **(6 points)** Design a recursive function that accepts two arguments into the parameters x and y. The function should return the value of x times y. Remember, multiplication can be performed as repeated addition as follows: 7 \* 4 = 4 + 4 + 4 + 4 + 4 + 4 + 4

(To keep the function simple, assume that x and y will always hold positive nonzero integers.)

*Sample output for x = 3 and y = 5*



1. **(6 points)** Design a function that accepts an integer argument and returns the sum of all the integers from 1 up to the number passed as an argument. For example, if 50 is passed as an argument, the function will return the sum of 1, 2, 3, 4, . . . 50. Use recursion to calculate the sum.

*Sample output for n = 6*



1. **(6 points)** Design a function that uses recursion to raise a number to a power. The function should accept two arguments: the number to be raised and the exponent. Assume that the exponent is a nonnegative integer.

*Sample output for arguments 2 and 4*



1. **(6 points)** You can create a more interesting and realistic looking tree by
2. randomizing the angle the turtle turns. Rather than always using a 30 degree angle, select an angle between 15 and 45 degrees.
3. randomizing how much the branches shrink each time you make a recursive call. Instead of always subtracting 15, try subtracting a random amount between 5 and 25.
4. add color to the tree by making the large branches brown, and the small branches green. Choose a threshold value (or use the existing value 5) for the length of the trunk and set the color accordingly.

Use recursiveTree.py file to implement the above changes. Here is a simple version that creates a "dot" in the base case (for a leaf). Your program output does not have to be the same.

A sample output for the original code (left) and an output for a possible solution (right) are shown below; your output does not have to be similar since the angle and the branch length are chosen randomly.

