GIS 5578
Wrap-Up 1

Choose <u>one</u> (1) of the following tasks and write a Python program that accomplishes it. Use multiple python files if necessary. Choose the task that will demonstrate your knowledge of programming. Be ambitious, but please do not torture yourself. There are no scripts attached but mock data are provided for Task C. Having said that, lecture notes, demos, and labs are sufficient to finish any of the projects below. We have done various elements of this work in class and you should be able to fall back on your previous homework or in-class exercises.

Your code **<u>must work</u>** and is expected to be **<u>efficient</u>, <u>compact</u>** but **<u>readable</u>,** and **<u>properly commented</u>**.

<u>Task A</u>

Write a script that takes coordinates of multiple points from the user and calculates the total Euclidean distance between subsequent pairs of points (i.e., line segments). The use story is as follows:

1. The user specifies the number of points they want to input (e.g., 3).
2. Sequentially, for each point, the user enters its coordinates, for example, (x_1, y_1), (x_2, y_2), (x_3, y_3) or x_1, y_1, x_2, y_2, x_3, y_3.
3. Your code calculates the distance from point P_k to the <u>consecutive</u> point P_{k+1} (e.g., from point P_1 to point P_2, from P_2 to P_3; see figure 1).
4. The total distance (Total_d) is derived by summing up the lengths of the individual line segments (e.g., Total_d = $d_{P1P2} + d_{P2P3}$).
5. The total distance is displayed to the user in a descriptive manner.
6. Make sure that your code works for any number of points (e.g., 0, 1, 10, 100+).

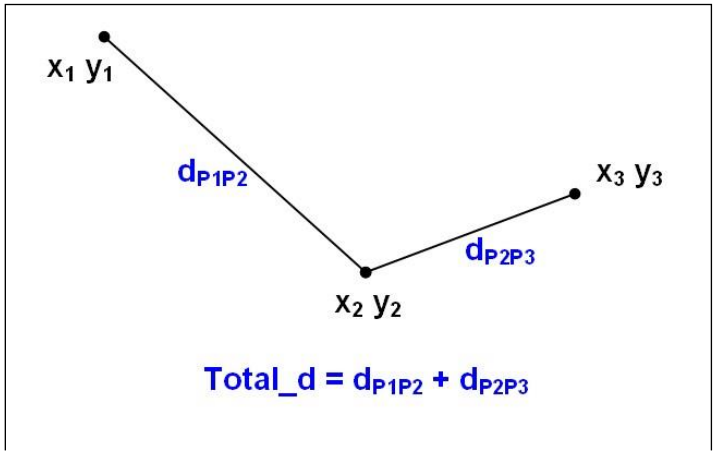

Figure 1 Total distance

Task B

Write a script that generates and saves a random integer ASCII grid based on a user-specified size and range. The use story is as follows:

1. The user specifies the number of columns and rows for the raster to be generated (e.g., 4 cols and 5 rows).
2. Then, the user specifies the range and step of values to pool from when populating the grid values (e.g., from 2 to 8 with step of 2 which results in the following set of integers: 2, 4, 6, 8; see next step).
3. Based on the input, the system generates a 2-dimensional array of a given size and then populates it with values randomly (!) selected from the pool of eligible integers (as described above; see figure 2 for an illustration).
4. The array is saved to a text or comma separated csv file without the header.

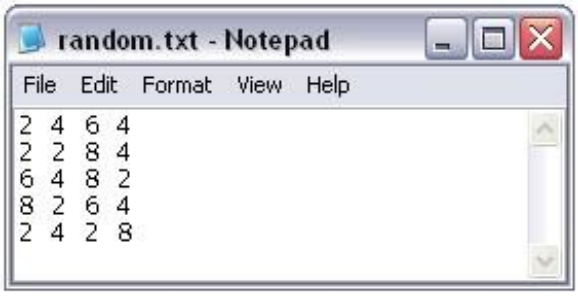

Figure 2 Sample random.txt result

Task C

Write a simple spatial calculator, which takes two ASCII grid files from the user (sample data are attached), performs a user-specified calculation on them, and saves the result to a file in the ASCII grid file format. The use story is as follows:

1. The user specifies the name and path of the input ASCII grid files (with a separate path for each of the input files)
2. The script checks if the two grids are of the same size (same number of columns and rows). If not, then the system displays an error message and either quits or loops until the user specifies a second ASCII file with grid of the same size as the first one).
3. The user is asked about what type of operation to perform on the input grids. Provide the following options: sum, difference, multiplication, and division. For division, make sure that you do not divide by zero (in this case assign a NoData value to the output; see figure 3).
4. The script performs the selected local operation (i.e., cell-by-cell map algebra operation) and saves the result back to an ASCII grid file (excluding the ASCII file header).

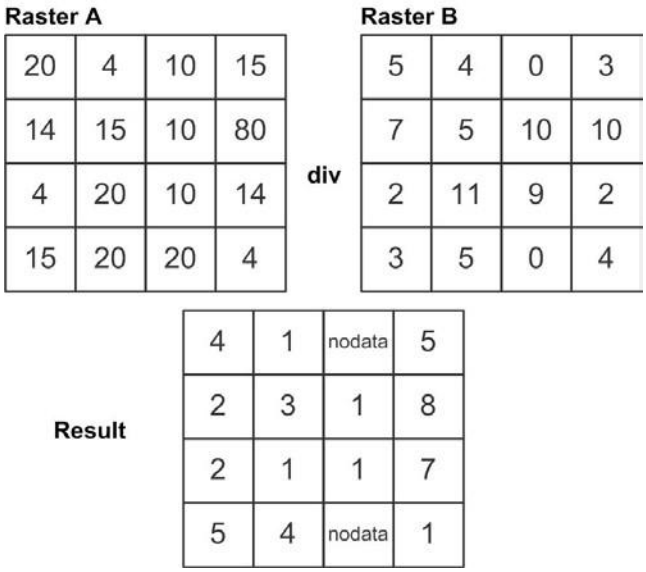

Figure 3 Raster-raster division

Upload the completed, commented, and tested scripts, including all auxiliary files, into GitHub. Please specify your name and the task you chose to work on. The deadline for the submission of the wrap-up assignment is 6:00 P.M. November 5, 2018.