Review and ask questions as soon as possible.

The primary focus of this project is reading and parsing data and creating an attributed feature class from the input data. You can create a script tool, as you did in the last project, but this is not a requirement. This project draws from skills that you learned in Chapters 4-8, with a special emphasis on skills from Chapters 7 and 8. You will want to review working with dictionaries, too (Chapter 4, Section 6.8)

Points will be awarded based on the quality of the deliverables:
up to 2 for your pseudocode/workflow
up to 1 for your script tool and folder structure (only **required** parameter is workspace; other parameters are optional)
up to 4 for script (includes comments)
up to 3 for write-up (includes approach and lessons learned)

Project 4: Parsing rhinoceros sightings

In this project, you're working for a wildlife conservation group that is tracking rhinos in the African savannah. Your field workers' software resources and GIS expertise are limited, but you have managed to obtain an Excel spreadsheet showing the positions of several rhinos over time [1]. Each record in the spreadsheet shows the latitude/longitude coordinate of a rhino along with the rhino's name (these rhinos are well known to your field workers).

You want to write a script that will turn the readings in the spreadsheet into a vector dataset that you can place on a map. This will be a polyline dataset showing the tracks the rhinos followed over the time the data was collected.

Please carefully read all the following instructions before beginning the project. You are not required to use functions in this project but you can gain over & above points by breaking out repetitive code into functions.

Deliverables

This project has the following deliverables:

1. Your plan of attack for this programming problem, written in pseudocode in any text editor. This should consist only of short, focused steps describing what you are going to do to solve the problem. This is a separate deliverable from your customary project writeup.
2. Your script tool and standard folder with subfolders for documents, scripts, and data, a toolbox and a map document.
3. A Python script that reads the data from the spreadsheet and creates, from scratch, a polyline shapefile with *n* polylines, *n* being the number of rhinos in the spreadsheet. Each polyline should represent a rhino's track chronologically from the beginning of the spreadsheet to the end of the spreadsheet. Each polyline should also have a text attribute containing the rhino's name. The shapefile should use the WGS 1984 geographic coordinate system.

4. A short writeup (~300 words) explaining what you learned during this project and which requirements you met, or failed to meet. Also describe any "over and above" efforts here so that the instructor can look for them.

Successful delivery of the above requirements is sufficient to earn 90% on the project. The remaining 10% is reserved for efforts that go "over and above" the minimum requirements. This could include (but is not limited to) a batch file that could be used to automate the script, creation of the feature class in a file geodatabase instead of a shapefile, or the breaking out of repetitive code into functions and/or modules.

Challenges

You may already see several immediate challenges in this task:

- The data is in a format (XLSX) that you cannot easily parse. The first step you must do is manually open the file in Excel and save it as a comma-delimited format that you can easily read with a script. Choose the option **CSV (comma-delimited) (*.csv)**.

  If you are so inclined, you can attempt to download and use a Python library that works directly with XLSX files. Be aware that you will have less comprehensive "technical support" from your fellow students if you use this route. Your instructor has recently had good success with the Python module ***xlrd*** (Excel read).

- The rhinos in the spreadsheet appear in no guaranteed order, and not all the rhinos appear at the beginning of the spreadsheet. As you parse each line, you must determine which rhino the reading belongs to and update that rhino's polyline track accordingly. **You are not allowed to sort the Rhino column in Excel before you export to the CSV file. Your script must be "smart" enough to work with an unsorted spreadsheet in the order that the records appear.**

- You do not immediately know how many rhinos are in the file or even what their names are. Although you could visually comb the spreadsheet for this information and hard-code each rhino's name, your script is required to handle all the rhino names programmatically. The idea is that you should be able to run this script on a different file, possibly containing more rhinos, without having to make many manual adjustments.

- You must find and run ArcGIS geoprocessing tools that will create an empty polyline shapefile with a text field for storing the rhino's name. You must also assign the WGS 1984 geographic coordinate system as the spatial reference for this shapefile.

Hints

- Before you start writing code, write a plan of attack describing the logic your script will use to accomplish this task. Break up the original task into small, focused chunks. You can write this in Word or even Notepad. Your objective is not to write fancy prose, but rather short, terse statements of what your code will do: in other words, pseudocode. Here's an example of some pseudocode that might appear in your file:

  ```
  . . .
  Read the next line.
  ```

```
Split the line.
Determine the rhino referenced in this line.
Determine if the dictionary has a key for the rhino.
If no key exists, create a new array object.
Create a new point object.
Assign the X reading to the X coordinate of the point. Assign the Y
reading to the Y coordinate of the point.
Add the point to the array.
Add the array to the dictionary using the rhino name as the key.
. . .
```

If you do a good job writing your pseudocode, you'll find that each line translates into about one line of code. Writing your script then becomes a matter of translating from English to code. You may also find it helpful to sketch out a diagram of the workflow and logistical branches in your script.

- You will have a much easier time with this assignment if you first create the array objects representing each rhino track, then use insert cursors to add the arrays once they are completed. Not only is this easier to code, it's better for performance to open the insert cursor only once near the end of the script.

- A Python dictionary is an excellent structure for storing a rhino name coupled with the rhino's array of observed locations. A dictionary is similar to a list, but it stores items in key-value pairs. For example, a key could be a string representing the rhino name, and that key's corresponding value could be an array object containing all the points where the rhino was observed. You can retrieve any value based on its key, and you can also check whether a key exists using a simple **if key in dictionary:** check.

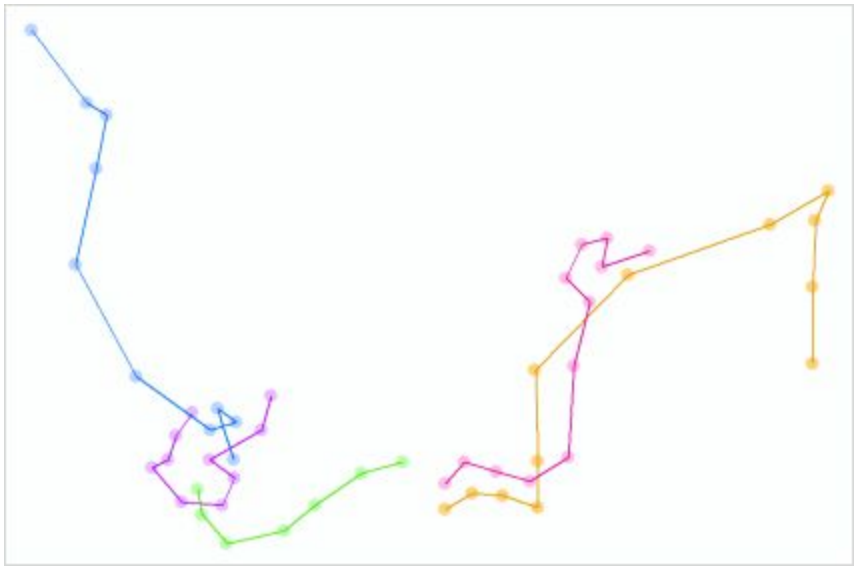
  We have not worked with dictionaries much in this course, but your Zandbergen text has an excellent section about them and there are abundant Python dictionary examples on the Internet.

  You can alternatively use lists to keep track of the information, but this will probably take more code. If you find yourself getting confused or writing a lot of code with lists, you may try to switch to dictionaries. Your instructor has a lot of experience in using dictionaries for parsing data.

- To create your shapefile programmatically, use the CreateFeatureClass tool. The ArcGIS Desktop Help has several examples of how to use this tool. If you can't figure this part out, I suggest you create the feature class manually and work on writing the rest of the script. You can then return to this part at the end if you have time.

- In order to get the shapefile in WGS 1984, you'll need to create a spatial reference object that you can assign to the shapefile at the time you create it. One way to accomplish this is using the SpatialReference.CreateFromFile() method and pointing at the appropriate .prj file in C:\Program Files\ArcGIS\Coordinate Systems\. Be warned that if you do not correctly apply the spatial reference, your polyline precision could be diluted.

If you do things right, your polylines should look like this (points are included only for reference):



Published on *GEOG 485: GIS Programming and Automation* (https://www.e-education.psu.edu/geog485)