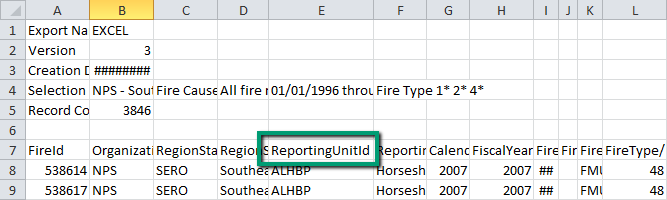
**Wildfire Ignitions Project**

A sample comma separated value (csv) file, containing geographic points which represent wildfire ignitions, is provided. The csv contains attributes on fire ignitions, that is, dates, acreages, locations, and so forth. In its current state, the csv file can’t be automatically uploaded into ArcGIS and visualized as a set of wildfire points. A GIS professional has been repeatedly performing the process described in this report, by hand and has requested a tool to perform these steps automatically to streamline the workflow.

This application should be able to modify this file (or files like it) and create corresponding shapefiles and reports. These csv files don’t fulfill the requirements for ArcGIS table formats. The field names are not in the first row and they contain spaces and special characters. The application needs to remove some header rows, modify some of the field names (see ‘ArcGIS Table Field Names’ note below), remove some unneeded columns, and create the following outputs:

* An edited csv file that is importable to ArcGIS, saved in the output directory.
* One shapefile corresponding to the entire csv file (‘allFires.shp’), plus a separate shapefile for each individual park unit (ReportingUnitId). The resulting individual shapefiles need to be named with the following convention: { ReportingUnitId}\_YYYYMMDD. The date reflects the date in which the shapefile was made. Newly created shapefiles should be added to an ArcMap Document (which should have a basemap for context).
* An HTML report for the input file. The HTML report should at a minimum, show the name of the input file, the total number of Reporting UnitId’s found, the total number of fire ignitions, and an image of the resulting map (the arcpy.mapping module allows you to export an image). As an additional challenge, you can also try to add one row for each ReportingUnitId, giving the name, the number of fire ignitions for that ReportingUnitId.

The resulting csv, shapefiles, and HTML files need to be saved in a new folder, designated by the end user. Within ArcMap a button needs to be created. This button will launch a GUI interface (created with a script tool), prompting the user for several pieces of input information. Allow the user to select:

* The csv or a directory containing multiple csv files for conversion.
* The output location of the new folder where the shapefiles will be saved.
* A spatial reference for the new shapefiles. To give them a default spatial reference of NAD 83, use this path to the projection file as the default spatial reference:   
  Coordinate Systems/Geographic Coordinate Systems/North America/NAD 1983.prj
* A latitude and longitude fields. Use LatitudeNAD83 and LongitudeNAD83 as the default. Points with no coordinate specified (for example, “LongitudeNAD83" and “LatitudeNAD83” are empty strings), should be placed at (0,0).
* A field name row number. Use 7 as the default.
* A last ‘column of interest’ number. Use 72 as default. When 72 is used, it means you’ll strip off any columns from 73 on up (or from BU to the right).

The best way to approach this is to open the csv file, read the contents, modify the contents and create an edited csv that can be imported into ArcGIS. This edited csv should be importable to ArcGIS. Then a single shapefile with all the data can be created. Then the individual unit shapefiles can be created by doing selections based on the park unit as specified above. Use the file open/read/close functions you learn in class, not the csv module.

Complete the csv processing and shapefile generation for the ‘project core Python code’ submission. You don’t have to have a button and GUI yet at that point, but your code should be handling the user input via the system arguments by then.

As you start the project, you will not yet know how to automatically add data to the map, how to write an HTML file, or how to make a button and a graphical user interface to get input from the user. But you will already know how to do the geoprocessing involved in this project. Start by creating a smaller csv that you ‘fix’ by hand. For your test file, keep only a few rows , enough so that you have more than one entry for a few different ReportingUnitId’s. Get your code to create the main shapefile for this test file first. Next, get your code to create the individual ReportingUnitId shapefiles. Next, you’ll learn how to read and write text files in class. The csv is a text file (with commas separating the column values). So the next step is to redo the csv file fixing that you did by hand with your script and put this together with the geoprocessing to create the shapefiles. Next, make your script flexible by using arguments instead of hard-coding the input variable, such as the field name row number and so forth. Finally, you’ll need to create the button, GUI, and output HTML.

**ArcGIS Table Field Names**

-- Make the field names are in the first row.

-- Follow these practices for field naming:

* Field names must start with a letter.
* Field names must contain only letters, numbers, and underscores. (replace any blanks or special characters with underscores).
* Field names must not exceed 64 characters.

\*Project request from Justin Shedd.\*