GIS 211 – Spatial Modeling and Python

Python Project – Overview and Ideas

**The PROJECT**

* Write a script to perform some task, as simple or complex as you are able to.
  + Ideally, the script should actually run. If it doesn’t, you’ll get credit for the parts that do run, and for any pseudo-code/logic that you lay out in the script for the parts that don’t run.
  + It may involve starting with a Model, and exporting to Python to get started.
* Intended to be done in class, so just a small script, not a major undertaking.
* You can get inspiration from, and steal code from, any source possible
  + Other scripts, code samples in the help, online searches, exercises
* 15% of your total grade (in comparison, each test was 10%)
* Some sample data may be found in N:\GIS 211 Data\Project Ideas\ GIS211ProjectData.gdb

Good code should contain the following:

* Descriptive and relevant comments
* Author, date, purpose of script
* Descriptively named variables
* User arguments where appropriate
* Basic error handling, messaging, exists, overwrite, data types
* Well-formatted print statements and messaging
* Pseudo-code in comments to explain logic/reasoning

**IDEAS for a Python project**

**1. Converting data and renaming fields**

* Data to use: Take some of the shapefiles from the Python book dataset, and write a script to bring them into a geodatabase
* You can use the script to create the geodatabase
* Will make use of listing data, string formatting, listing fields, copying data. Can also include checks for projection or other properties
* You’ll need to list the data inside the folder you choose, loop through the list and for each one, name it properly and copy the data into the GDB
* Optionally, you could rename the fields, too

**2. Recreating a class exercise using Python**

* Take the original model from Exercise 1, convert it to Python to get the bulk of the code written for you.
* Modify the code to include bells and whistles like user arguments for data, parameters, query values, looping to accommodate multiple inputs, error checking using Describe, if statements, data exists…

**or**

* Take the watershed creation workflow and work it out as a Python script
* Will require use of the spatial analyst extension
* You could also start this one with a model and export to Python.
* Modify the code to add bells and whistles and other things as in the previous example (above)

**3. Write a script to add the name of the file as an attribute in the table**

I’ve seen this needed in cases where an organization had dozens of layers for an entity, all stored separately. Ie, each county had its own layer but you want to merge them all into one. And each file is named by the county name, but each feature has no other attributes.

* For each layer you would need to:
  + Add a new field
  + Grab the file name as a variable
  + Use a method to put the file name into the new field.
* Then merge all the layers together

**4. Using parcel data – selecting parcels near an event, writing out well-formatted addresses or a form letter to the Shell window or a txt file.**

* + Homeowners within 1500 feet of a fire station (you pick one of the 6) need to be informed that the Fire Station will be conducting emergency testing on a certain date and time, and not to be alarmed (pun intended!!) by the sirens.
  + Data to use: BoulderCoParcelsforLongmont and the LongmontFireStations layers
  + This one would be like combining the Selections exercise with the Cursors exercise
  + Will need to either combine the StreetNo, StreetDir, StreetName, and StreetSuf fields into one, or concatenate them on the fly when you loop through every record in the table
  + Will make use of the Select Layer by Location tool, cursors, loops.
    - Just like in Chapter 5, you’d need to take the data, create layers, make your selection
    - Loop through every record in the selection with a cursor to get the address, add your own text, format it like a letter to the homeowner
  + Will need a second cursor to get the name and address of the Fire Station for the letter, too.
  + I will provide you:
    - Some examples of string formatting to create a nice output
    - An example of writing out to a text file, if you want to try it, or you can use print statements
  + The output letter should look something like this, with < > indicating where variables might be plugged in.

To Homeowner at <address>,

You are located within <Search Distance> feet of <Fire Station #x> at <Fire Station Address>. There will be some emergency testing on this day and time, *(Choose whatever date you want*). Please disregard all the sirens at that time.

Thank you.

You can word it however you want, as long as it’s a nicely formatted multiple-line block of text that contains the house address and the Fire Station name and address.

**5. Using parcel data: Identify which parcels are owned by people who don’t live there.**

* Data to use: BoulderCoParcelsforLongmont and the LongmontBndBuf layer
* First, clip the data to the Longmont city boundary
* Need to combine the StreetNo, StreetDir, StreetName, and StreetSuf fields into one
* Test if the Mailing address field is equal to the new Address field
  + Could use a cursor, Field Calculator, or even an advanced SQL statement
* If the mailing address and actual address don’t match, it means the owner is remote and that parcel should be flagged or selected.
* You could do this all in a script with cursors and if, or a model
  + The clip is done with the Toolbox tool
  + The fields could be concatenated with the Calculate Field tool using Python syntax
  + The address matching could be done on a new empty field with a second Calculate Field tool with a Python codeblock using an if statement.

**6. Creating a complex query using values in a table**

* This idea would contain some non-script-based data management and pre-processing tasks
* Data to use: Coloverts.xls, County95 coverage
  + Since County95 is a coverage, you’ll want to import that into a GDB or a shp. You can do this manually, it doesn’t have to be part of the script.
  + You’ll want to copy/import the xls data into a format ArcGIS can write to – like a geodatabase or DBF. You can also do this manually.
  + You’d want to select out just a few counties (preferably ones that are adjacent to each other) to make the query easier to manage. You can put this in the script, or do it manually.
  + You’ll probably also want to select just a few species from the list of 600. You may want to test the script by hard-coding in one species name until you get the query syntax working
* Will make use of cursors, if, loops, selection tools, string formatting to create the query string
* The finished product will create a query such that if the value for a given county is greater than 3, the county should be added to the query.
* The query is then used to make a selection on the Counties layer