NR 426 – Programming for GIS I
Lab 4 – Working with tables

Warner College of Natural Resources | Colorado State University | Instructor: Elizabeth Tulanowski

Rev. Feb. 2026

Learning objectives:

1. Access values in a table with the arcpy cursor methods
2. Use the selection tools in Python
3. Use Python in the Calculate Field tool

What to submit:

- For 4A, submit the code (as a .py file) from the self-directed activities.
- For 4B, submit the demo and the self-directed activity (these can be separate files)

Lab 4A: Accessing tables with cursors

In-class hands-on activity (Don't need to submit):

- Brief review of Search and Update Cursors
- Using SQL in Python
- Working with CSVs – read-only in arcpy, but not with other libraries like Pandas
- Putting it together: DMS to DD demo

4A Self-directed activities:

- Refer to the slides, sample code, ArcGIS Pro online help, and the internet for help.
- Your code should include well-named variables, header code, good comments, messaging, and be formatted for readability. Your grade will be both functionality of the script as well as style and good coding practices.

Choose your own Lab 4A adventure! Use one of these two datasets, depending on your own interests. Located in Lab4Data folder.

- ***CO Stream Gages:*** Point layer for USGS Stream Gages in Colorado. Contains fields for Station Name, Status, County, Elevation, Drainage area, and start and end years (among a few other fields) (Source)

OR

- ***LarimerTrails_rank***, a Larimer County Trail popularity ranking (Source) – Line layer for trails in Larimer County with fields for Name, Surface type, Popularity, uses, Difficulty, elevation, and length
- OR use ***your own data*** as long as it has some text and number fields to work with. Complete comparable tasks on your own data.

- Choose your dataset and examine it in ArcGIS to familiarize yourself with the attributes
- Create a new script file that performs the following tasks using arcpy and other relevant modules, for your chosen layer:

CO Stream Gages	**Larimer Trail Rankings**
1. Returning Station Names a. Pick one county (avoid the Eastern Plains, not enough data out there) b. Create a cursor to report the Station Names of each gage in that county and the number of gages in that county c. What type of cursor? *Refer to the slides, help docs, and the videos for some sample code to get you started.* d. Optionally, add a counter to only print out the first 5 or 10 records e. Update your county variable to try it on a different county 2. High-elevation gages a. Create another cursor with the appropriate where clause to return and print only the names and elevations of gages above 11,000 feet 3. Highest value in a field a. Report the name of the station with the largest drainage area, and its drainage area value b. Alternatively, do highest elevation, or report on the top 5, or do this within one county for faster running 4. Update Nulls a. Create a copy of the stream gage layer (manually in Pro is fine) -Call it COStreamGage_original. This will be your backup copy in case the data gets messed up in the next step b. Create the appropriate cursor and *update* all records with a STATUS of Null to be Unknown. Ideally, do this with a where clause in the cursor. An if statement could be used but would be slow. c. Be sure to include code to save/commit your changes	1. Returning trail names a. Pick one surface type (ie, track, snow, gravel, concrete. Dirt has too many, avoid that one) b. Create a cursor to report the trail names of each trail of that surface type and the number of trails c. What type of cursor? *Refer to the slides, help docs, and the videos for some sample code to get you started.* d. Optionally, add a counter to only print out the first 5 or 10 records e. Update your surface variable to try it on a different surface type 2. High-ranking trails a. Create another cursor with the appropriate where clause to return and print only the names and popularity values of trails above 1.7 3. Highest value in a field a. Report the name of the trail with the highest max_elevat, and its elevation value b. Alternatively, do lowest min_elevat, or report on the top 5, or do this for just one surface type for faster running 4. Update Nulls a. Create a copy of the trails layer (manually in Pro is fine) -Call it LarimerTrail_original. This will be your backup copy in case the data gets messed up in the next step b. Create the appropriate cursor and *update* all records with a surface value of Null to be Unknown. Ideally, do this with a where clause in the cursor. An if statement could be used but would be slow. c. Be sure to include code to save/commit your changes

5. ***Optional Challenges:***

a. Create a new layer with only stream gages above 11,000 feet, or Trails with a popularity more than 1.7. Two approaches are possible, choose one:

- Use a tool that allows you to select the high-value records and output to a new feature class. This method creates a second output, leaving the original alone. (Safer? But then you have two versions of the same thing. Be sure to name things clearly.)
- <not recommended> Create another cursor for those records with high values, and *delete* those rows. This method alters the original. (Dangerous?? Maybe.)

The end result would be the same.

b. Programmatically calculate differences/ranges in values:

   i. In CO Stream Gages, identify the longest-running gage, using the BEGIN_DISCH and END_DISCH fields. Note these are text fields.

   ii. Or in LarimerTrailRanking, identify the trail with the greatest elevation range using min_elevat and max_elevat

c. Extra-challenge, for extra credit: Save the table from ArcGIS Pro as a CSV (from the Pro interface or Geoprocessing pane) and use the Python Pandas library to answer the same questions above

Lab 4B: Using the selection tools in Python

In-class hands-on activity (Submit just step 2 of this):

1. Comparing selections in ArcGIS to Python
   - How to generate Python code for your SQL statements
2. Work as a class to manipulate the Roadkill data table:
   - In Pro, manually make a new field called Status, load, examine, and run the collapse status.cal file on the Roadkill layer
   - Manually make a new field, called SpCodes, and manipulate the scientific names in the Roadkill layer:
     - Put all values in title case
     - Create the abbreviation using the first two letters of the genus and species, in capital letters
       - Ie, Marmota monax → MAMO
     - Use the Python console window to figure out how to set it, then "translate" it into the Calculate Field tool. ***Submit a screenshot of your Calculate Field tool window.***

4B Self-directed activities:

Using the Roadkill data in the Lab 4 Data folder and the selection tools you just learned, write the code to answer these questions with Python: (**Use selection tools** for these, not cursors)

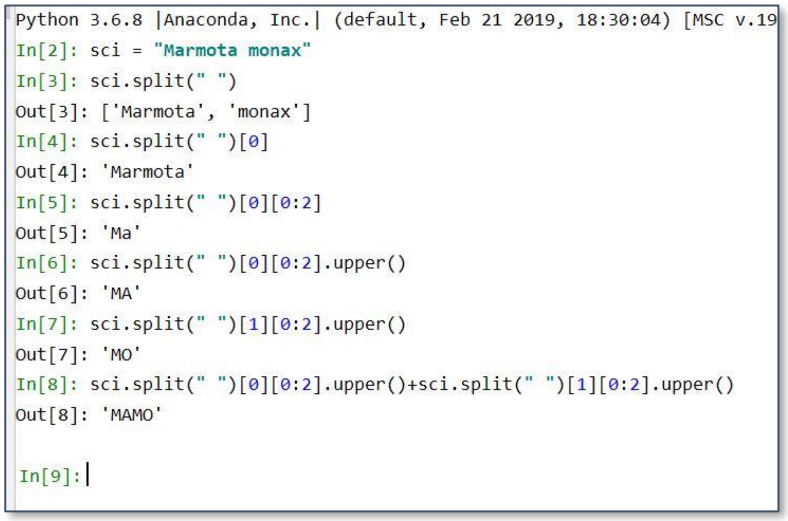
Tips/Requirements:

- ✓ Start simple, work step by step, make use of variables, keep track of your plan with pseudocode and comments.
- ✓ Number/Identify each step using a comment in your code.
- ✓ Report the answers (# of points or location of output data) as a well formatted message to the user.
- ✓ Write clean, readable code with some error handling like we've seen in class and in the book.

1. How many points did EAH collect alone (no partner)?
2. How many total points did EAH collect (including with other people?) How would you go about writing the SQL statement for this? Use a wildcard - don't hard-code names or initials into the query.
3. Run an operation that gets all the points that fall inside of Ulster County and outputs it to its own new feature class. Inform the user where they could find that feature class and report the number of points in Ulster County.
4. Starting with your layer from #3, buffer the *Striped skunk* points that are in Ulster County by .25 miles. Report the number of points in the output, and the location of the layer.
5. For the whole area (not just Ulster Co.): How many points are ***not*** within 200 feet of a road? Run the operation and return the number in a well-formatted sentence.
6. *Optional challenge*: Report the above information to a text file. (Using a text file and .write, or try out the Python *stdout* functions
7. *Challenging optional challenge*: Look up the Mapping Module in PSA Chapter 11 or the help, and write code that will add at least one of the output layers above to map with a title stating the layer name. It is probably easiest to do this in the Python window, but PyCharm works too.

In-class activity SOLUTION (sort of)

Here's the thought process for how to reformat the scientific name into the species code:



```
Python 3.6.8 |Anaconda, Inc.| (default, Feb 21 2019, 18:30:04) [MSC v.19
In[2]: sci = "Marmota monax"
In[3]: sci.split(" ")
Out[3]: ['Marmota', 'monax']
In[4]: sci.split(" ")[0]
Out[4]: 'Marmota'
In[5]: sci.split(" ")[0][0:2]
Out[5]: 'Ma'
In[6]: sci.split(" ")[0][0:2].upper()
Out[6]: 'MA'
In[7]: sci.split(" ")[1][0:2].upper()
Out[7]: 'MO'
In[8]: sci.split(" ")[0][0:2].upper()+sci.split(" ")[1][0:2].upper()
Out[8]: 'MAMO'

In[9]:
```