Click to watch the Lesson 4A, Part 1video

Working with tables

Accessing and updating values in a table

The arcpy Cursor methods

Making selections

Readings: PSA book Chapter 8

NR426

Programming for GIS I

Lesson 4 (a, b)

Instructor:

Elizabeth Tulanowski | ESS Department

What does Lesson 4 cover?

Two class sessions: Lessons 4a, 4b

The arcpy Cursor methods

SQL statements in Python

Selecting features using arcpy

Using Python expressions in the Calculate Field tool

Lesson 4A

Accessing values in tables with cursors

Search cursor

Update cursor

Refer to Chapter 8 in Python Scripting for ArcGIS Pro and and the ArcGIS help

PSA Book: Ch. 8.1 – 8.2

Getting inside tables with Cursors

- A cursor is a *data access object* -used to read or write values in a table
  - An in-memory copy of the records in a table
  - ArcGIS Help topic on the cursor object
- Basic workflow:
  1. Create the cursor object
     - Think of the cursor object as a collection of all the rows that you grab from a table. (It could be every row, or just some, if you use a query)
  2. Loop through each row in the cursor (table) – Typically will use a for loop
  3. For a given row, do something with it or to it:

     Read a value, make it a variable for use in another operation, report it back to user

     Write a value into the table, ie, copied from another table, result of a calculation, a combination of values from other fields, etc..

Type of Cursors

- **Search** Cursor — Search Cursor Help Doc
  - Used to gain read-only access to the data inside a table.
- **Insert** Cursor — Insert Cursor Help Doc
  - Used to insert new rows into a table.
- **Update** Cursor — Update Cursor Help Doc
  - Used to gain read-write access to a table and update values in fields for existing rows

Learn more in this Esri tech session video, minutes 0:00 – 22:00

0:00 – 5:49 - Slides on cursors
5:50 – 21:00 – Demo on cursors
21:00 – 22:00 - Slides, best practices, advanced tips
22:00 - 27:00 - Geometry, accessing the Shape field.
27:00 - 42:00 - List methods, Walk (a way to dig into subfolders)
43:00 - 54:00 - Open source libraries including NumPy (for rasters) and pandas (data analysis)

Two ways to create cursors

- Old way ("Classic cursors"): arcpy.SearchCursor (don't use this way!)

```
cur = arcpy.SearchCursor("table", {optional where clause})
```

- New and recommended way:

```
cur = arcpy.da.SearchCursor("table", field_names, {where clause})

with arcpy.da.SearchCursor("table", field_names, {where clause}) as cur:
    for loop is indented
```

- Difference: New way has faster performance, returns a tuple. Formatting of rows for output is very different

Take home message: The help has documentation for both. Make sure you find the right one! You want the one with arcpy.**da**.___Cursor()

Search Cursor

- Used to gain read-only access to the data inside a table.
  - Read values from a table, make as a variable, format and print them to the console window, or use them in other calculations
- You loop through a table row by row and for a given row, you access the value from any field
- Basic syntax:

```
cur = arcpy.da.SearchCursor("table", [fields], {where clause})
```

Search Cursor Help Doc

Search cursors

- Add a *where clause* (query) to a cursor to limit the rows using SQL syntax:

```
import arcpy
fc = "c:/data/roads.shp"
field0 = "StreetName"
field1 = "NumLanes"
with arcpy.da.SearchCursor(fc, [field0,field1], "\"StreetName\" = 'Main'") as cur:
      for row in cur:
          print row[0] + row[1]
```

Fields you want access to get passed in as a list

Access values for the current row and field using the row variable and the index number from the field list

Search cursor example

```
env.Workspace = r"C:\Data\Redlands.gdb"
with arcpy.da.SearchCursor ("Hospitals", ["Address", "Name"]) as cur:
```

	Shape*	ObjectID*	Address	NAME	CITY
	Point	1	1710 Barton Rd.	Loma Linda University Medicine Center	Redlands
	Point	2	1620 Laurel Ave.	Inland Surgery Center	Redlands
	Point	3	245 Terracina Blvd.	PrimaCare Medical Group	Redlands
	Point	4	350 Terracina Blvd.	Redlands Community Hospital	Redlands
	Point	5	2 W. Fern Ave.	Beaver Medical Group	Redlands

	Point	1	1710 Barton Rd.	Loma Linda University Medicine Center	Redlands

```
for row in cursor:
    print row[0] + row[1]    # Prints value from Address field
```

None

A

Insert cursor

- Used to insert new rows into a table
  - Table must already exist, and have fields created in it.
  - Requires **write** access to the table
- Specify values for the new row when creating it, otherwise it will be a new empty row
- Generic syntax for an insert cursor

```
cursor = arcpy.da.InsertCursor("table", fields)
```

Insert Cursor Help Doc

Insert cursor code sample

```
cursor = arcpy.da.InsertCursor(fc,[“RowID”, “Distance”])

# Create 25 new rows. Set the initial row ID and distance values
for x in xrange(1, 26):
      cursor.insertRow([x, 100])

# Delete cursor and row objects to remove locks on the data
del row
del cursor
```

Update Cursor

- Used to gain read-write access to a table and update values in fields for existing rows
- Can be used instead of the Calculate Field tool, and is sometimes a better choice
- Requires write access to the table
- Generic syntax for an update cursor

```
cursor = arcpy.da.UpdateCursor("table",[fields], {query})
```

Update Cursor Help Doc

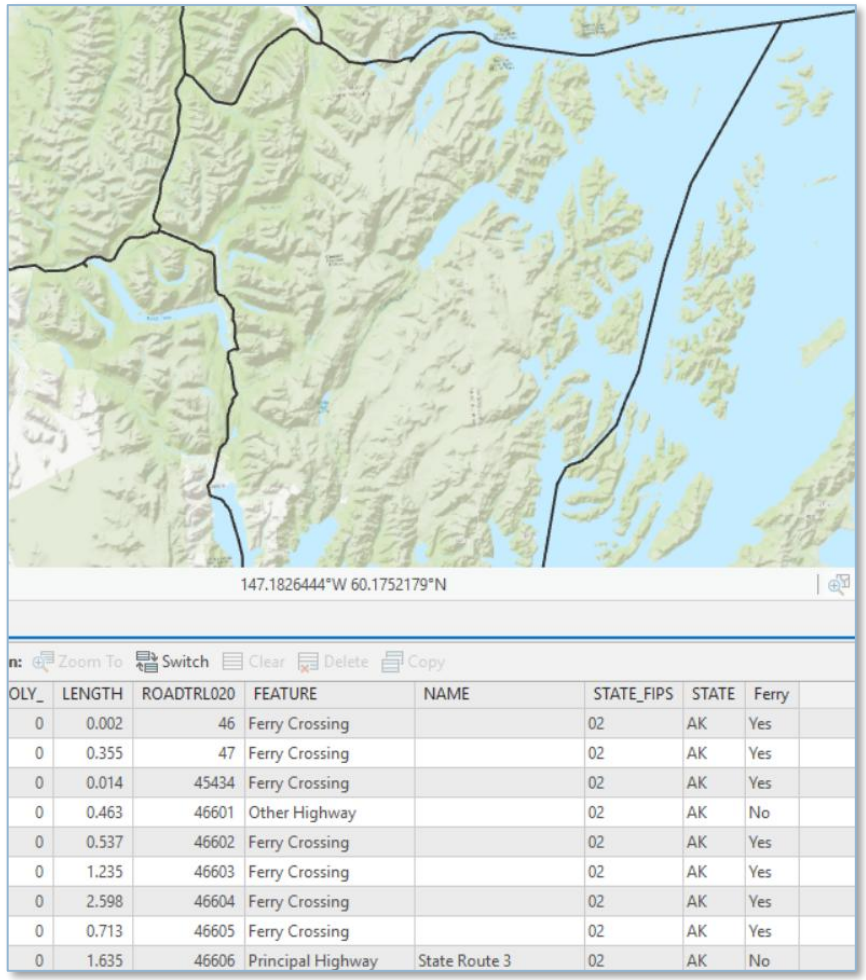
Code sample for an update cursor

```
import arcpy

fc = "c:/database/schools.shp"
field0 = “NAME”
field1 = “FACILITY”
field2 = “FullName”
FldList = [field0, field1, field2]

with arcpy.da.UpdateCursor(fc, FldList) as cur:
  for row in cur:
    # field2 will become equal to field0 concatenated to field1, with title case
    row[2] = row[0].title() + “ “ + row[1].title()
    # This next line commits the change, it’s like hitting Save
    cur.updateRow(row)
```

Update Cursor example using if

See more short examples in the PSA book, Ch. 8.2, and the help docs online

```
with arcpy.da.UpdateCursor(fc, ["FEATURE", “FERRY"]) as cur:
    for row in cur:
        if row[0] == "Ferry Crossing":
            row[1] = "YES"
        else:
            row[1]= "NO"
        cur.updateRow(row)
```



OLY_	LENGTH	ROADTRL020	FEATURE	NAME	STATE_FIPS	STATE	Ferry
0	0.002	46	Ferry Crossing		02	AK	Yes
0	0.355	47	Ferry Crossing		02	AK	Yes
0	0.014	45434	Ferry Crossing		02	AK	Yes
0	0.463	46601	Other Highway		02	AK	No
0	0.537	46602	Ferry Crossing		02	AK	Yes
0	1.235	46603	Ferry Crossing		02	AK	Yes
0	2.598	46604	Ferry Crossing		02	AK	Yes
0	0.713	46605	Ferry Crossing		02	AK	Yes
0	1.635	46606	Principal Highway	State Route 3	02	AK	No

DEMO:

Examine data and attributes in Pro

Use Lesson 4 Update Cursor ferry DEMO.py

Demo: arcpy Cursors

Watch this **Esri tech session video**

5:50 – 21:00 – really great demo with a few extra tips and tricks

Watch the NR426 Demo videos

Click to watch the Lesson 4A Demo on SEARCH cursors

Accessing values from a table with a Search Cursor

Click to watch the Lesson 4A Demo on UPDATE cursors

Walking through the Akroads.shp Ferry route updates from the previous slide

Click to watch the Lesson 4A, Part 2: Data locks, del, and SQL video

Lesson 4A (continued)

Cursors

Data locks

del

Filtering with SQL

Refer to Chapter 8 in Python Scripting for ArcGIS Pro and and the ArcGIS help

PSA Book: Ch. 8.1 – 8.2

Data locks

- Python can lock data, just like the ArcGIS application
  - If you're running a script that requires WRITE access to data, be sure to have ArcGIS closed
  - If you need WRITE access to data that was just created in Python, close Python
    - For example, to delete in Catalog, because it didn't turn out right
- The error message you receive is quite useful:

```
Cannot get exclusive schema lock…
```

Take-home message: If you get lock errors, make sure the data's not in use somewhere else.

More in PSA Book Ch. 8.2 – Pages 249 and 251

The del statement

- Python has a del statement which can delete the reference to objects created in the script
- It's good practice to use these to remove potential locks on the data and to clean up memory, especially for Update and Insert cursors.

```
del cursor
del row
```

Filtering rows retrieved using SQL

- Search and Update Cursors can apply a query
  - Limits what rows get returned
  - Uses standard SQL syntax

```
import arcpy
fc = "C:/Data/study.gdb/roads"
with arcpy.da.SearchCursor(fc, ["STREET_NAME", "CLASSCODE"],
                           '"CLASSCODE" = 1') as cursor:
    for row in cursor:
        print(row[0])
```

```
import arcpy
fc = "C:/Data/study.gdb/roads""
fields = "STREET_NAME"
sql = (None, "ORDER BY STREET_NAME")
with arcpy.da.SearchCursor(fc, fields, sql_clause=sql) as cursor:
    for row in cursor:
        print(row[0])
```

GDB only: Use the SQL Order By function to sort alphabetically rather than by ID

  - Use DISTINCT to get unique values, ORDER BY to sort

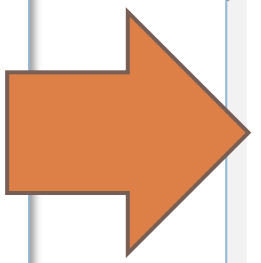
```
sql = ("DISTINCT", "ORDER BY STREET_NAME")
```

GDB Only: Use the SQL Distinct function with Order By to remove duplicates!!

More in Lesson 4B and PSA Book Ch. 8.3

Using SQL with cursors, Distinct and Order By

```python
# NR426 Lesson 4 Demo Code: Using SQL statement in Search and Update Cursor
# Purpose:  Create a Search Cursor that returns the unique values in a field
# Refer to Python Scripting for ArcGIS Pro, Ch. 8.3 for more detail.

# Created by Elizabeth Tulanowski, Colorado State University
# Sept. 2020

import arcpy, sys

arcpy.env.workspace = r"C:\Users\MapGirl\OneDrive - Colostate\NR426-427\426 Files\Lesson 4

sql = ("DISTINCT", "ORDER BY SPECIES")
#sql = ("DISTINCT", None)

print ("Creating the search cursor...")
with arcpy.da.SearchCursor("RoadkillSightings", ["SPECIES"],sql_clause=sql) as cur:
    # Create a counter
    numspecies = 0

    print ("Trying to print list of unique species names:")
    for row in cur:    # For each row in the table that we asked for...
        print (row[0])
        numspecies+=1

print ("\nThere are {0} unique species in the database".format(numspecies))
```



```
Lesson 4 sample SQL OrderBy and Distinct
"C:\Program Files\ArcGIS\Pro\bin\Python\envs\arcg
Creating the search cursor...
Trying to print list of unique species names:
American beaver
Big brown bat
Black bear
Common muskrat
Common raccoon
Deer mouse
Eastern chipmunk
Eastern cottontail
Eastern gray squirrel
Gray fox
Long-tailed weasel
Masked shrew
Meadow jumping mouse
Meadow vole
Norway rat
Pine vole
Porcupine
Red bat
Red fox
Red squirrel
Short-tailed shrew
Short-tailed weasel
Smoky shrew
Southern redback vole
Striped skunk
Virginia opossum
White-footed mouse
White-tailed deer
Woodchuck
Woodland jumping mouse

There are 30 unique species in the database

Process finished with exit code 0
```

What did you just learn?

- What are the three types of cursors?
  - What purpose does each one serve?
- What arcpy “submodule” do we need to call when creating a cursor?
- How do we pass in the fields we want access to within a cursor?
- Explain how data locks occur and how they can be removed.
- How can we filter the rows that get retrieved when using a cursor?
  - What is the argument called in the syntax, and what type of expression/language is used?
- Which cursors require write access to the table?
  - What table formats allow write access?

Lesson 4A – Working with Cursors

In-class Activity:

- Brief review of Search and Update Cursors
- Using SQL
- Working with CSVs
- Examine *Sum Roads in Buffer* script, DD to DMS demo

Lab 4A:

- Write the code to return values, get max values, from an attribute field
- Write the code to update values in a field
- *Optional challenges* – try them out!

Click here to watch the Lesson 4B, Part 1 – Selections and SQL- VIDEO

Lesson 4B – Working with Selections

Selecting data with Python

- Using SQL
- Make Feature Layer
- Selecting by attribute and location

Using Calculate Field with Python expressions

Selecting data with Python

- Why do we make selections?

  What types of operations can we do with a selection?

- Various ways to query data, we'll focus on 4 tools:
  - Make Feature Layer and Make Table View
  - Select Layer by Attribute and Select Layer by Location,

- *Feature class is to Feature Layer as Table is to Table View*
  - Feature class is the data on the hard drive, a layer is in memory
  - Table is the data on the hard drive, a table view is the visualization of the table in memory

Selection tools

- Select Layer by Attribute and Location require an in-memory feature layer or table view as an input.
  - Use Make Feature Layer or Table View to create these in-memory layers.
  - Make feature layer can also include a simple query expression, sometimes eliminating a line of code.

- **Proper query syntax**
  - Requires specific formatting for quotation marks Refer to: Queries in Python

```
"Fieldname  = ‘value’ "
"Fieldname  = number "
```

SQL Examples

- Formatting depends on data type
- Wildcards make expressions more flexible
- Use Help or the Selection tools in ArcGIS as a guide

Learn more in the ArcGIS SQL ref. and PSA Book Ch. 8.3

Wildcard for:	Shapefiles and File GDBs	Personal GDBs (.mdb)
Basic query	" County = 'Larimer' " ' Shape_Area > 100000' ' "County" = \'Larimer\'' *Likely won't need this method*	' [Shape_Area] > 100000 ' " [County] = 'Larimer' "
Single character	_ (underscore) "ZoneCode Like 'RES-_'" Would find RES-1, RES-2, or RES-3	? "[ZoneCode] Like 'RES-?'" Would find RES-1, RES-2, or RES-3
Any number of characters	% "StreetName Like 'M%'" Anything starting with M	* "[StreetName] Like 'M*'" Anything starting with M

FYI: Personal GDBs are no longer supported at ArcGIS Pro

Selection statements

- Looking for (or not for) null data
  - `"Field" IS NULL` or `NOT "Field" IS NULL`
- Select for multiple values with `in`
- Using and, or, String and Data functions:
  - Complex SQL statements can refine your queries.
  - Look for features that contain certain characters, or where one field divided by another has a certain value.
- Use the SQL help or the Desktop help to learn more about them.
  - Play with them in ArcGIS before trying it in Python

SQL expressions using a variable

- Variable must be outside of the string, and concatenated into the statement
- Values require single quotation marks, these must be concatenated into statement

```
arcpy.env.workspace = r"C:\Lesson 4 - Working with tables\Data\RoadkillData.gdb"
County = "Ulster"   #The user can supply the county name to query for
CountySQL = County.capitalize()
# A hard-coded where clause might look like this:
wherex = "COUNTY_G = 'Orange'"
# Using a variable makes it tricky
where = "COUNTY_G = '"+CountySQL+"'"
```

Queries with variables:

```
where = "COUNTY_G = '"+CountySQL+"'"
where = f"COUNTY_G = '{CountySQL}'"
```

```
#Not convinced? Just print it out before running any more code to see that it's formatted right
print (where)
```

Step 1: Make the feature layer

- Use the Make Feature Layer / Table View tool:
  - Example:

```
arcpy.MakeFeatureLayer_management(feat_class, feat_layer, query)
```

  - Input is a feature class on the hard drive.
  - Output gets created *in memory*, not a tangible file on the hard drive. The layer is deleted from memory when Python is closed.
- Option to query the input data, similar to a definition query on a layer.

- You *can* get away with not doing this step, but it returns a result object, which needs to be treated differently: details on the next slide

```
#arcpy.SelectLayerByAttribute_management(flayer, "New_Selection", qry)
print ("Making the selection....")
burgsel = arcpy.SelectLayerByAttribute_management("Burglary", "New_Selection", qry)
print(type(burgsel))
#Get and report the number of selected features
cnt = arcpy.GetCount_management(burgsel)
print("The number of selected records is: " + str(cnt))

print ("Try a buffer")
arcpy.analysis.Buffer(burgsel, burgsel+"_buf", "500 f
print ("Buffer worked")
```

Can't just concatenate `burgsel` variable to other text in Buffer syntax, because it's a result object

```
Making the selection....
<class 'arcpy.arcobjects.arcobjects.Result'>
The number of selected records is: 6001
Try a buffer
Error: unsupported operand type(s) for +: 'Result' and 'str'
```

```
print ("Try a buffer")
arcpy.analysis.Buffer(burgsel, str(burgsel)+"_buf3", "500 feet", "
print ("Buffer worked")
```

Have to use str() or just pass in a new string for the output name

```
print ("Try a buffer")
arcpy.analysis.Buffer(burgsel, "burgselbuf", "500 feet", "", "", "ALL")
print ("Buffer worked")
```

Step 2: Run the selection tool

- Select Layer by Attribute
  - Example:

```
arcpy.SelectLayerByAttribute_management(feat_layer, "NEW_SELECTION", query)
```

  - Input is the in-memory layer just created with the Make Feature Layer tool
- Selection type: New, Add, Remove, Subset, Switch, Clear
  - See the book or Help for exact syntax and definitions
- Output is the same in-memory layer, now with a selection on it.

Step 2: Or select by location

- Select Layer by Location
  - Example:

```
arcpy.SelectLayerByLocation_management(poly_layer, "INTERSECT",
feat_layer, search_distance, "NEW_SELECTION")
```

- More complex -involves 2 input layers and a method for how they interact (intersect, contain, within a distance of, etc…)
- Select features from the INPUT feature layer that have a certain SPATIAL RELATIONSHIP with the SELECT features.
  - Refer to the help topic on the tool for info
  - Experiment with the tool in ArcGIS before diving into Python

Additional useful tasks

- Making a selection permanent:
  - In ArcMap, you right-click on a layer and choose Data > Export Data to save a selected set as its own new feature class
  - In Python, you run the CopyFeatures tool from the toolbox.

```
arcpy.CopyFeatures_management(poly_layer, out_poly_fc)
```

- Counting the number of records:
  - In ArcMap, you look at the attribute table to see how many records are selected.
  - In Python, you run the GetCount tool from the toolbox.

```
result = arcpy.GetCount_management (feat_layer)
```

Demo: Selections

Click here to watch the Lesson 4B Selections Demo Video

- Creating selections in Python:
  1. Make feature layer
  2. Select by attribute
  3. Select by location
  4. Modify the SQL expressions

```
import arcpy


#Make the workspace a folder to look at SHPs
arcpy.env.workspace = r"C:\Student\ProgrammingPro\Databases"
myshp = "schools.shp"

#Create some SQL expressions to select features with VERIFIED va
#FYI: The value in a query is case sensitive
qry = "VERIFIED IN ('y', 'Y')"

# Now select multiple values with OR, or use IN:

# 1. Create the feature layer in memory for the schools.shp:
flayer = arcpy.MakeFeatureLayer_management(myshp, "Sel_layer")

# 2. Select features by attribute using variables for the layer
arcpy.SelectLayerByAttribute_management(flayer, "NEW_SELECTION",

# 3. Report back to the user how many features got selected:
cnt = arcpy.GetCount_management(flayer)
print ("The layer is: "+str(flayer[0]))
print("The number of selected records is: " + str(cnt))

# 4. Use the layers to select selected schools that intersect pa
arcpy.management.SelectLayerByLocation(flayer, "INTERSECT", "coa

# 5. Report back to the user how many features are within the pa
cnt = arcpy.GetCount_management(flayer)
print ("The layer is: "+str(flayer[0]))
print("The final number of selected schools is: " + str(cnt))
```

Demo and Data from Programming ArcGIS Pro with Python pages 176- 188

Click here to watch the Lesson 4B, Part 2 video – Using Python in Calculate Field

Lesson 4B, Part 2

Demo / In-class activity

- Manually make a new field and run the collapse status .cal file on the Roadkill layer
- Manually make a new field and make sure all the scientific names are in title case
  - Which string method to use? How should it be formatted?

Lab 4B

Self Directed Activity:

Work with selections on Roadkill Data

Work with Calculate Field and attributes (also on Roadkill Data)

Updating/Modifying values inside tables

Calculate Field tool

- Bulk updates to a table
- Run from ArcGIS or in Python script as:

```
arcpy.CalculateField (arguments)
```

- Use Python code directly in the Calculate Field tool

PSA Book Ch. 8.6 (pages 262-273)

Cursors

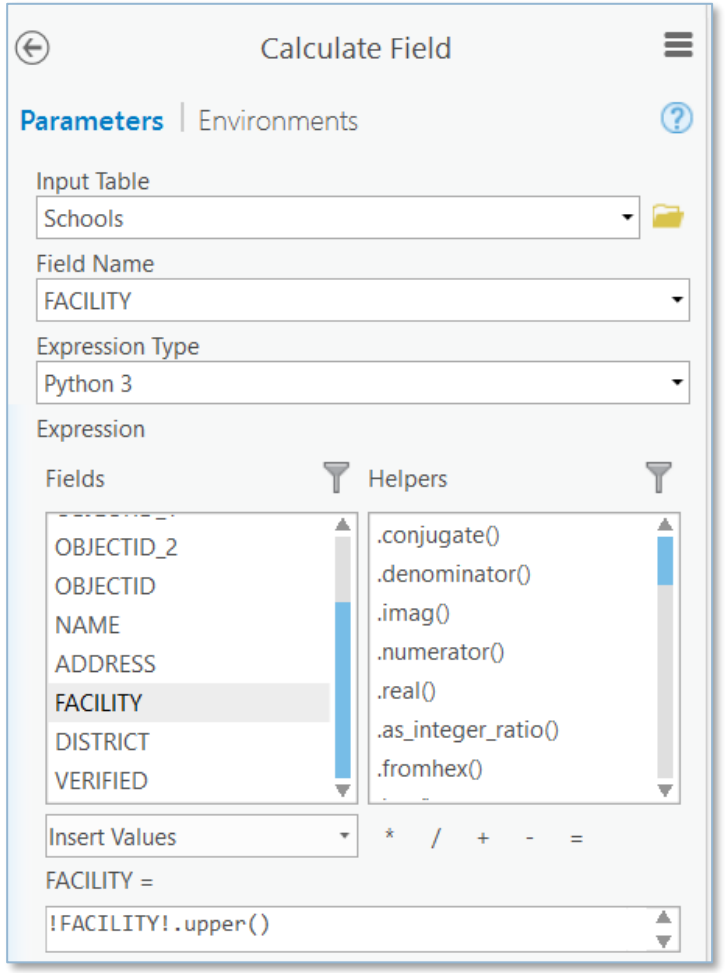
- Retrieving values from a table, updating values in a table, creating new features with coordinates
- Run as:

```
arcpy.da.SearchCursor(args)
arcpy.da.UpdateCursor(args)
arcpy.da.InsertCursor(args)
```

Lesson 4A and PSA Book Ch. 8.2

Using Python in Field Calculator

- Capabilities
  - Field calculator can call any Python method, perform if statements, looping, string manipulations and more
- Formatting/ syntax
  - Requires basic Python constructs like indentation, case sensitivity, colons
- Be sure expression type is set to PYTHON 3

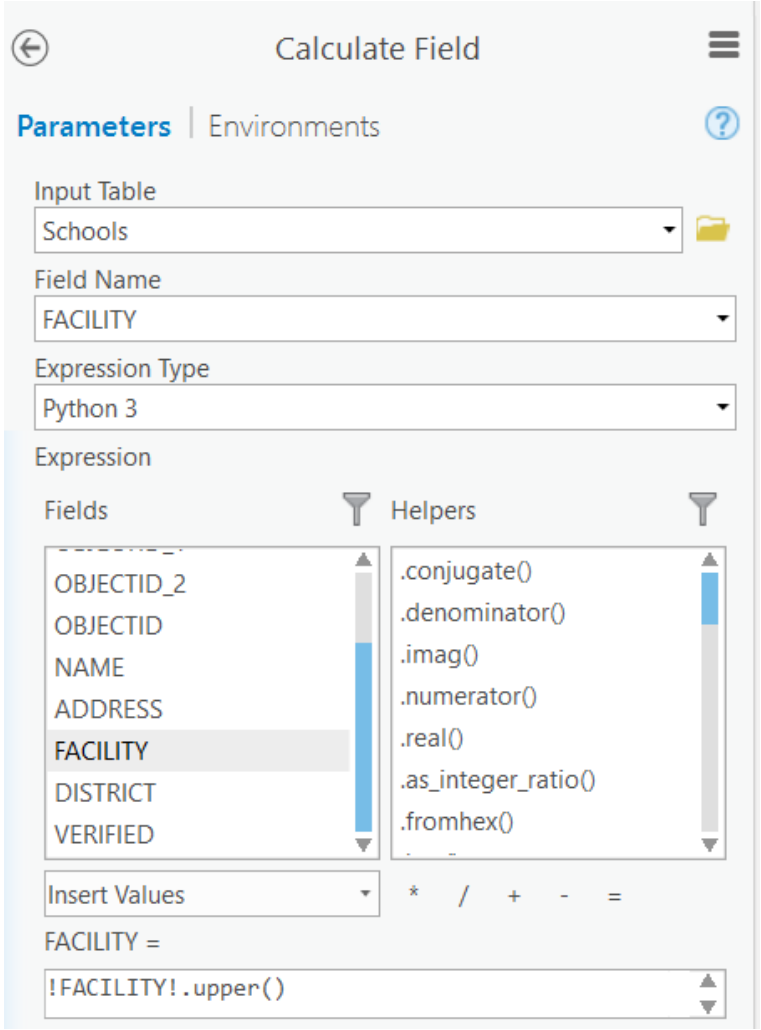


Examples of string manipulation

- .capitalize() → Capitalizes the first letter of the whole string
- .title() → Capitalizes the first letter of each word
- .lower() → Forces everything to lower case
- .upper() → Forces everything to upper case
- .split(delimiter) → Returns a list of each piece of original string
- List[x] → Use indexing to access items in the list, and apply string functions to them

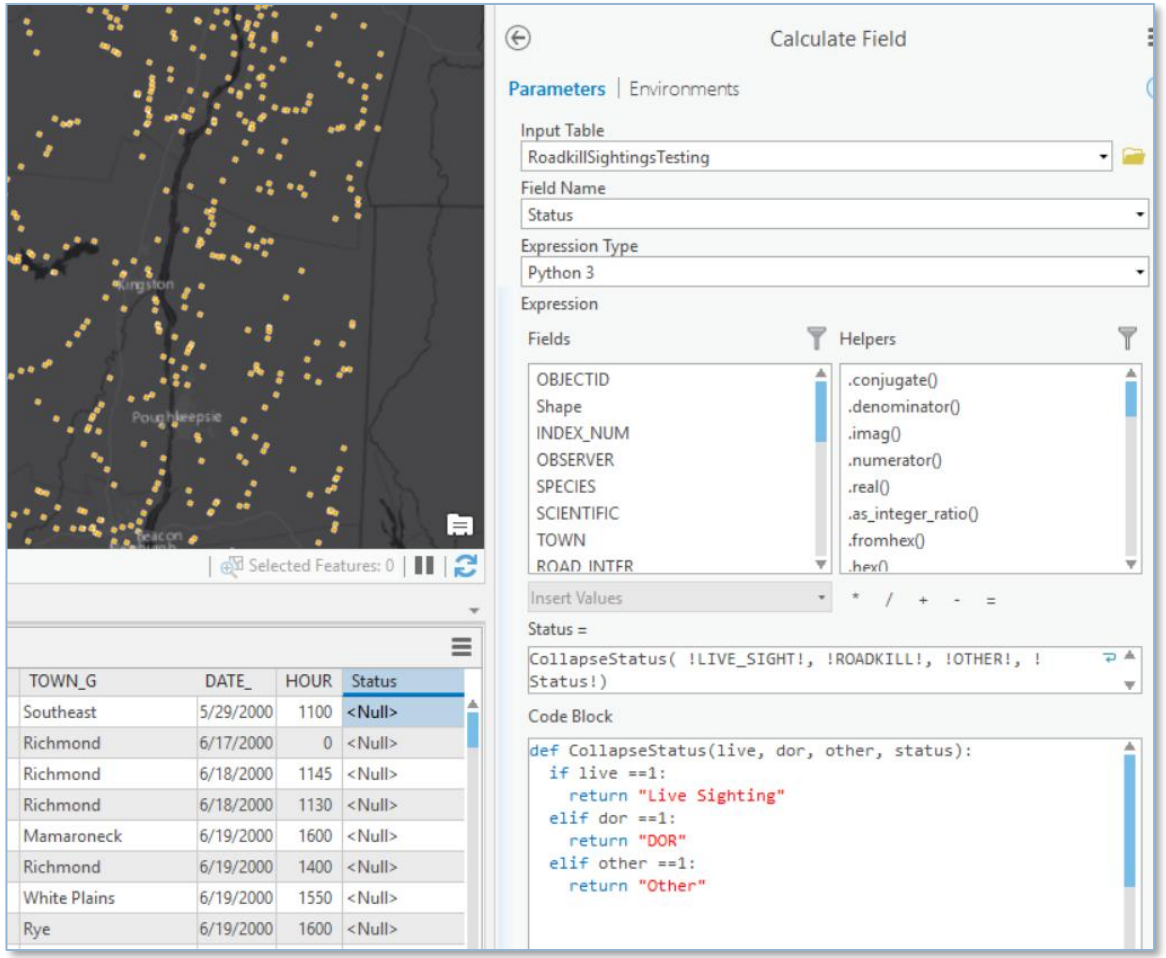
Python syntax in field calculator

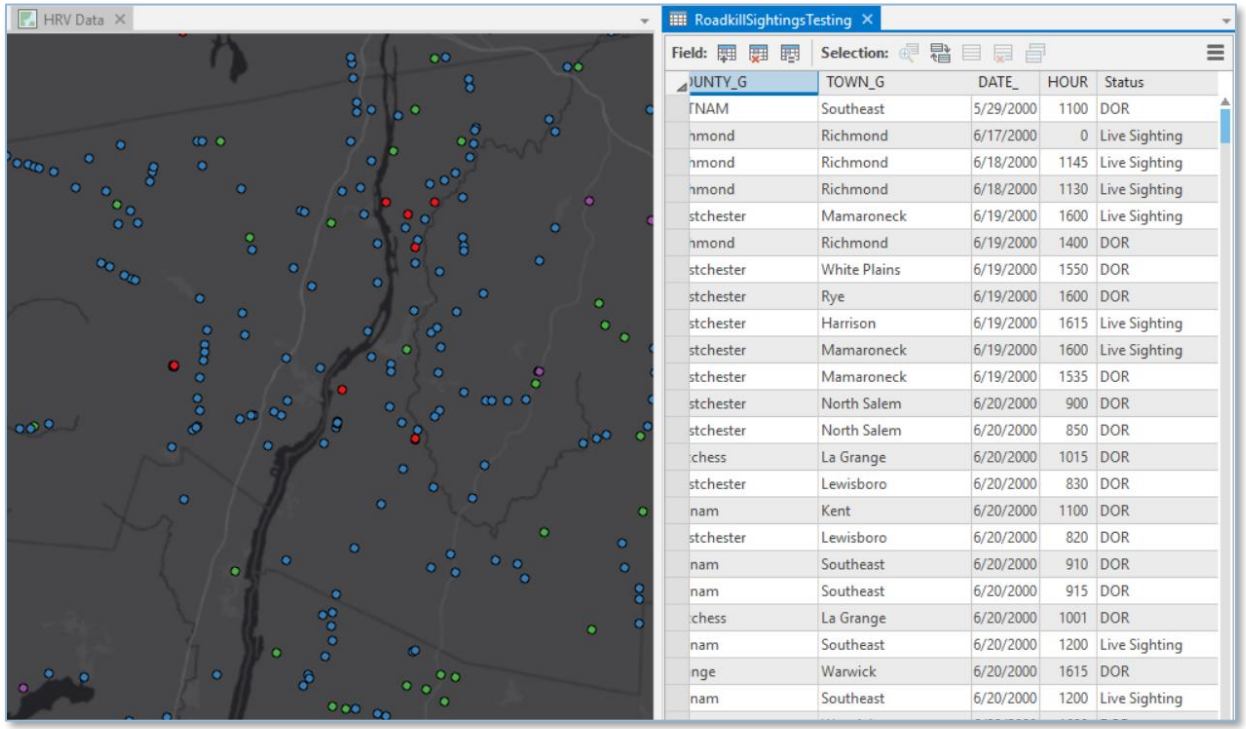
- Object.method()
- Enter into field calculator with field name as the “object”
- Find examples here:
  - PSA Book Ch. 8.6
  - Calculate Field Python Examples



Python in Calculate Field tool

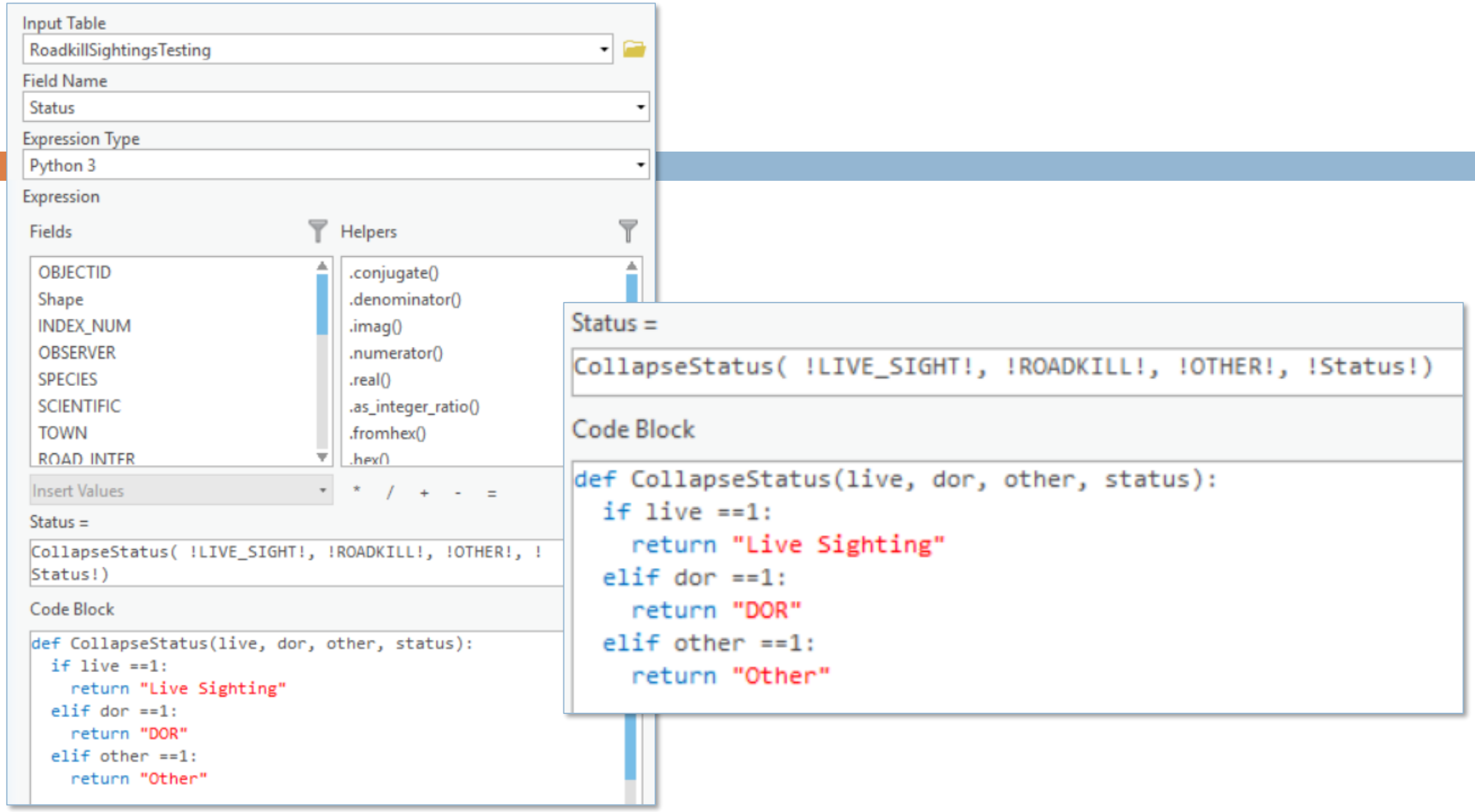
Click here to watch the Field Calculator Demo and last few slides for Lesson 4B





Defining functions

- The **def** statement
  - Python statements can be simple or complex, including a codeblock, and requiring a function to be defined with the def statement.
  - A function is a task with a defined purpose, the code runs and returns a value. That value is what goes into the field for each row in the table. We'll do a demo with this.
- Accessing field names with ! !
  - When using the Python parser in Field Calculator, you can point to or access values from other fields in the table by enclosing the field name in exclamation points. We'll see this in the demo.


Input Table
RoadkillSightingsTesting
Field Name
Status
Expression Type
Python 3
Expression
Fields
Helpers
OBJECTID
Shape
INDEX_NUM
OBSERVER
SPECIES
SCIENTIFIC
TOWN
ROAD_INTER
.conjugate()
.denominator()
.imag()
.numerator()
.real()
.as_integer_ratio()
.fromhex()
.hex()
Insert Values
* / + - =
Status =
CollapseStatus(!LIVE_SIGHT!, !ROADKILL!, !OTHER!, !Status!)
Code Block
def CollapseStatus(live, dor, other, status):
if live ==1:
return "Live Sighting"
elif dor ==1:
return "DOR"
elif other ==1:
return "Other"
Status =
CollapseStatus(!LIVE_SIGHT!, !ROADKILL!, !OTHER!, !Status!)
Code Block
def CollapseStatus(live, dor, other, status):
if live ==1:
return "Live Sighting"
elif dor ==1:
return "DOR"
elif other ==1:
return "Other"

What did you just learn?

- When writing SQL expressions, why are quotation marks an issue? How can you prevent these issues?
- Explain the use of a custom function to perform field calculations with the Calculate Field tool.
- Review the key terms at the end of Chapter 8 in the textbook.
  - Do you understand all (or most) of them?

Lesson 4B

Demo / In-class activity

- Manually make a new field and run the collapse status .cal file on the Roadkill layer
- Manually make a new field and make sure all the scientific names are in title case
  - Which string method to use? How should it be formatted?

Lab 4B

- Write the code to create several selections