arcpy Cursors

Cursors: Accessing and Editing Feature Attributes and Geometries

Overview

1. What is a cursor?
2. Types of Cursors
3. arcpy.da Cursors
4. Cursors Example
5. Cursors and Geometries

What is a cursor?

What is a cursor?

- A cursor indicates position
  - Like a cursor in a text editor
- Databases use cursors move through rows in a table
- The cursor stores the position in the table
- We can use cursors to retrieve, edit, and insert data into a table

Types of Cursors

Types of Cursors

- Three types of cursors:
    - Search
    - Update
    - Insert

Types of Cursors

Search Cursor

- Used to retrieve data
- Can not manipulate data: read-only

Types of Cursors

Update Cursor

- Retrieves data
- Allows overwriting of data: read/write
- Cannot create new records
- Works like Field Calculator, which happens to be a toolbox tool…
    - DO NOT use Field Calc in python: cursors are more reliable, easier, and much faster

Types of Cursors

Insert Cursor

- Does not retrieve data; works only on new rows
- Used to insert new records into a table: write-only

arcpy.da Cursors

arcpy.da Cursors

- arcpy supplies `SearchCursor()` , `UpdateCursor()` , and `InsertCursor()`
- also `arcpy.da.SearchCursor()` , `arcpy.da.UpdateCursor()` , and `arcpy.da.InsertCursor()`
    - `da` is the Data Access module:
        - `da` cursors are faster
        - Should be used unless a good reason not to

arcpy.da Cursors

- Use tokens to access special fields:
    - `SHAPE@X`: feature's x-coord (centroid?)
    - `SHAPE@Y`: feature's y-coord (centroid?)
    - `SHAPE@XY`: tuple of feature's centroid in x, y
    - `SHAPE@WKT`: well-known text (WKT) string of geometry
    - `SHAPE@AREA`: a feature's area
    - `SHAPE@LENGTH`: a feature's length
    - `OID@`: the value of the ObjectID field

Cursors Example

Cursors Example

Copy fields from one feature class to a new feature class:

```
gdb = r"C:\\Data\water.gdb"
infc1 = os.path.join("mains"
infc2 = os.path.join("main_analysis")

# fields to copy (already exist in both FCs)
fields = ["MAIN_ID", "INSTALL_DATE", "LAST_MAINTAINED"]

# blank dictionary to hold data by ID
data = {}

# create search cursor to get data from mains FC
with arcpy.da.SearchCursor(infc1, fields) as cursor:
    for row in cursor:
        data[row[0]] = row[1:]

# need a cursor to update existing rows in analysis fc
with arcpy.da.UpdateCursor(infc2, fields) as cursor:
        for row in cursor:
            # check if MAIN_ID is in data
            if row[0] in data:
                # if MAIN_ID, update row data
                row[1:] = data[row[0]]
                # commit update to dataset
                cursor.updateRow(row)
```

Cursors Example

Getting Feature Length and area:

```
>>> fc = "C:\\410Labs\Lab2\data\study_area.shp"

# create cursor on object id and feature length fields
>>> with arcpy.da.SearchCursor(fc3, ["OID@", "SHAPE@LENGTH"]) as cursor:
...     # iterate through rows in cursor
...     for row in cursor:
...         # print row[0] (id) and row[1] (feature length)
...         print "id: {}, length: {}".format(row[0], row[1])
...

id: 0, length: 478225.030663
id: 1, length: 346926.448917
id: 2, length: 143068.622396
id: 3, length: 464544.109197

# create cursor on object id and feature area fields
>>> with arcpy.da.SearchCursor(fc3, ["OID@", "SHAPE@AREA"]) as cursor:
...     for row in cursor:
...         print "id: {}, area: {}".format(row[0], row[1])
...

id: 0, area: 12002631497.3
id: 1, area: 5481557045.35
id: 2, area: 736088697.396
id: 3, area: 8260715181.54
```

Cursors and Geometries

Cursors and Geometries

- Cursors are able to access feature geometries in addition to attributes
    - `SHAPE@`: A feature as a geometry object

```
>>> import arcpy

>>> fc = r"C:\\410Labs\Lab2\data\bldg5.shp"

# create a search cursor returning just the geometries
>>> with arcpy.da.SearchCursor(fc, ["SHAPE@"]) as cursor:
...     # iterate through rows in cursor
...     for row in cursor:
...         # row[0] is first (and only) field, the geometry
...         print row[0]
...

<geoprocessing describe geometry object object at 0x12583620>
<geoprocessing describe geometry object object at 0x125835C0>
<geoprocessing describe geometry object object at 0x12583620>
<geoprocessing describe geometry object object at 0x125835C0>
# and so on...
```