Session 5: Accessing Feature Attributes Using Data Access Cursors

In this session we will work with arcpy.da.InsertCursor, arcpy.da.SearchCursor, and arcp.da.UpdateCursor. These data access tools allow you to create new rows (and features), to search rows, and to update rows.

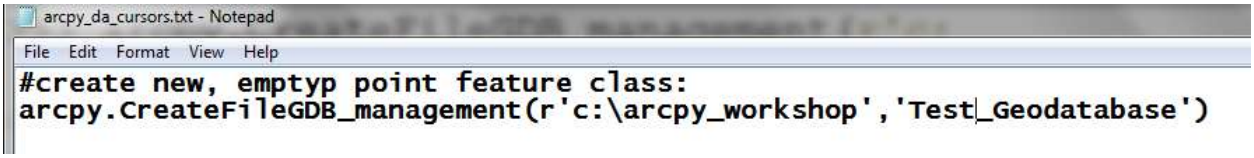
Data Access InsertCursor

We will start with arcpy.da.InsertCursor() to create new point, line, and polygon feature classes.

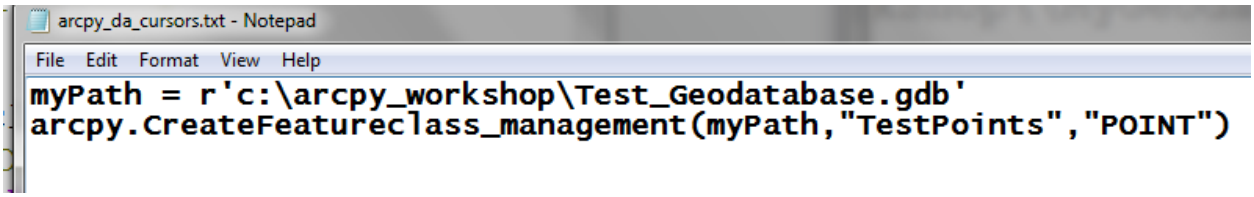
This is typically a three step process.

1) Compute an empty point, line or polygon feature class.
2) Optionally, add new field(s) to your empty feature class.
3) Create the new feature shapes and attributes using arcpy.da.InsertCursor()

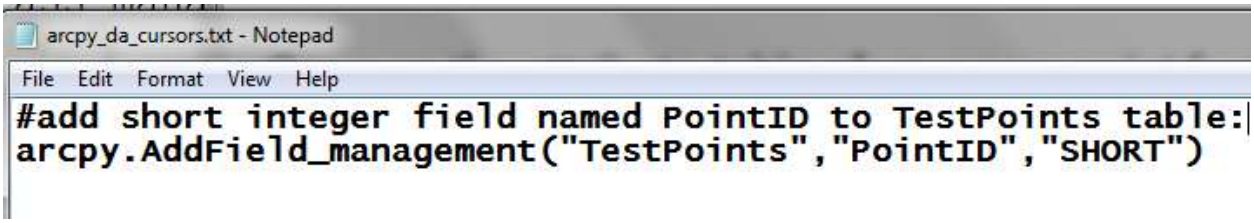
Start by creating a file geodatabase and creating a new point feature class in your geodatabase.



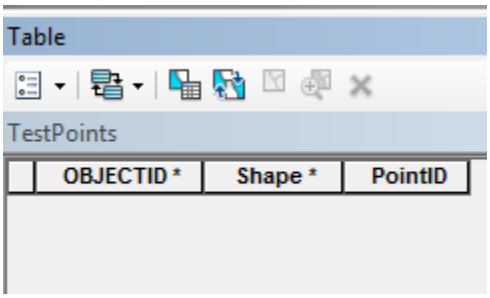
Next, create a new point feature class in your geodatabase container:



Next, add a short integer field named PointID:



Open up the attribute table of your new point feature class in Arcmap:



We have no point records in the table. The next step is to populate this table with records or features. The process is analogous to using the Editor to create new features.

```
Python
>>> editRows = arcpy.da.InsertCursor
 ("TestPoints","*")
```

Think of this as opening a feature class for editing.

```
Python
>>> print editRows.fields
(u'OBJECTID', u'Shape', u'PointID')
>>>
```

We need to set the values for the three fields of ObjectID, Shape, PointID.

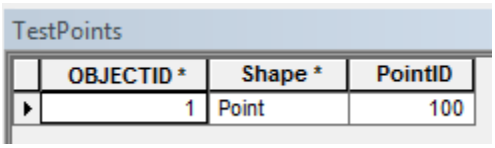
```
>>> OID = 0
>>> ptShape = arcpy.Point(0,0)
>>> ptID = 100
```

The create a Python list of your new field values and insert these values as a new row in your feature attribute table.

```
>>> newRecord = [OID,ptShape,ptID]
>>> editRows.insertRow(newRecord)
1L
>>>
```

Finally, delete your editRows object to save edits and unlock (end editing).

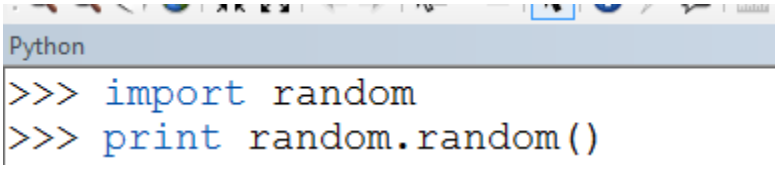
```
>>> del editRows
>>>
```



Note that the ObjectID is really set by ArcGIS since it is used by the software for record-keeping purposes. You specify the Shape and PointID values…

Creating Random Point Features

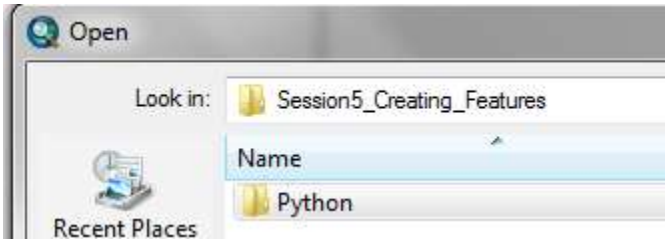
You can use the Python random module to create randomly located points. Try the following in the Python window:

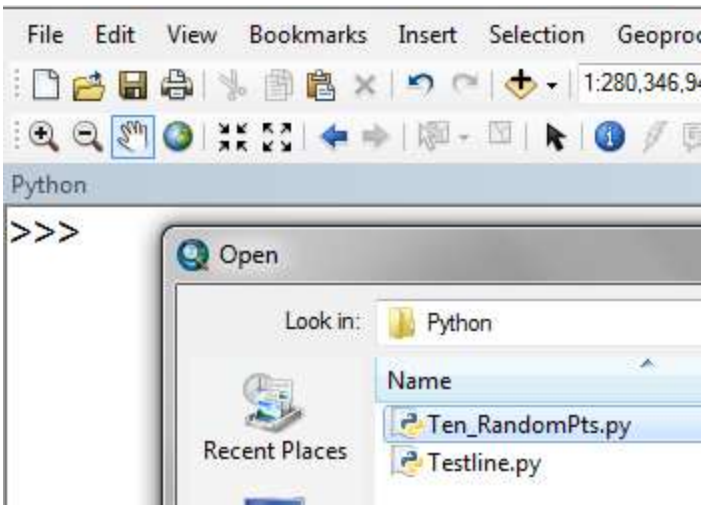


The random.random function will return a random number ranging from 0.0 to 1.0.

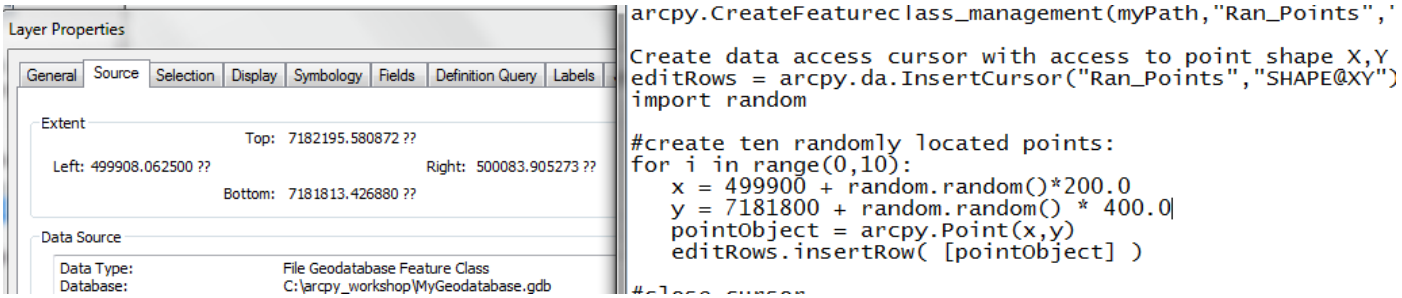
Right mouse click in your Arcmap Python window, select Load…

Load and Run the script ***Ten_RandomPts.py*** in your Arcmap python window.





This Python script creates ten random points ranging in X from 500,000 +/- 100 meters and in Y from 7,181,000 +/- 200 meters.



First we create an empty point feature class in the geodatabase, and open the feature class with the arcpy.da.InsertCursor(), accessing the shape x,y properties:

myPath = r'c:\arcpy_workshop\Test_Geodatabase.gdb'

#create new point feature class:

arcpy.CreateFeatureclass_management(myPath,"Ran_Points","POINT")

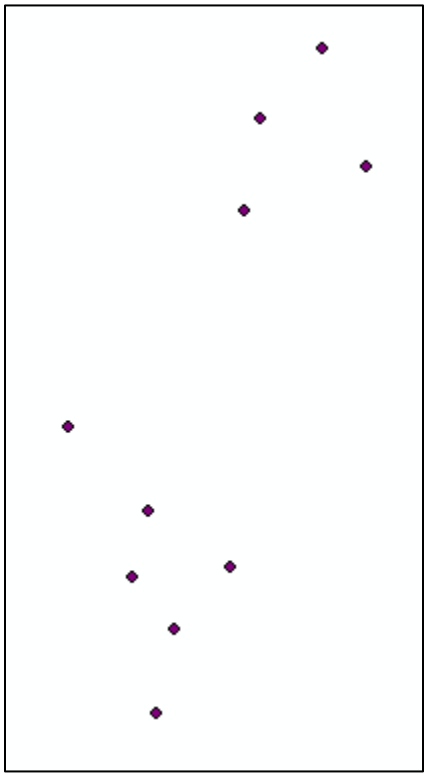
editRows = arcpy.da.InsertCursor("Ran_Points","SHAPE@XY")

Next we import the Python random module, and create points 0 to 10 in a loop. The x-coordinate will be 499900 + (random value ranging from 0.0 to 200.0)

The y-coordinate will be 7181800 + (random value ranging from 0.0 to 400.0)

```
import random
#create ten randomly located points:
for i in range(0,10):
  x = 499900 + random.random()*200.0
  y = 7181800 + random.random() * 400.0
  pointObject = arcpy.Point(x,y)
  editRows.insertRow( [pointObject] )
```

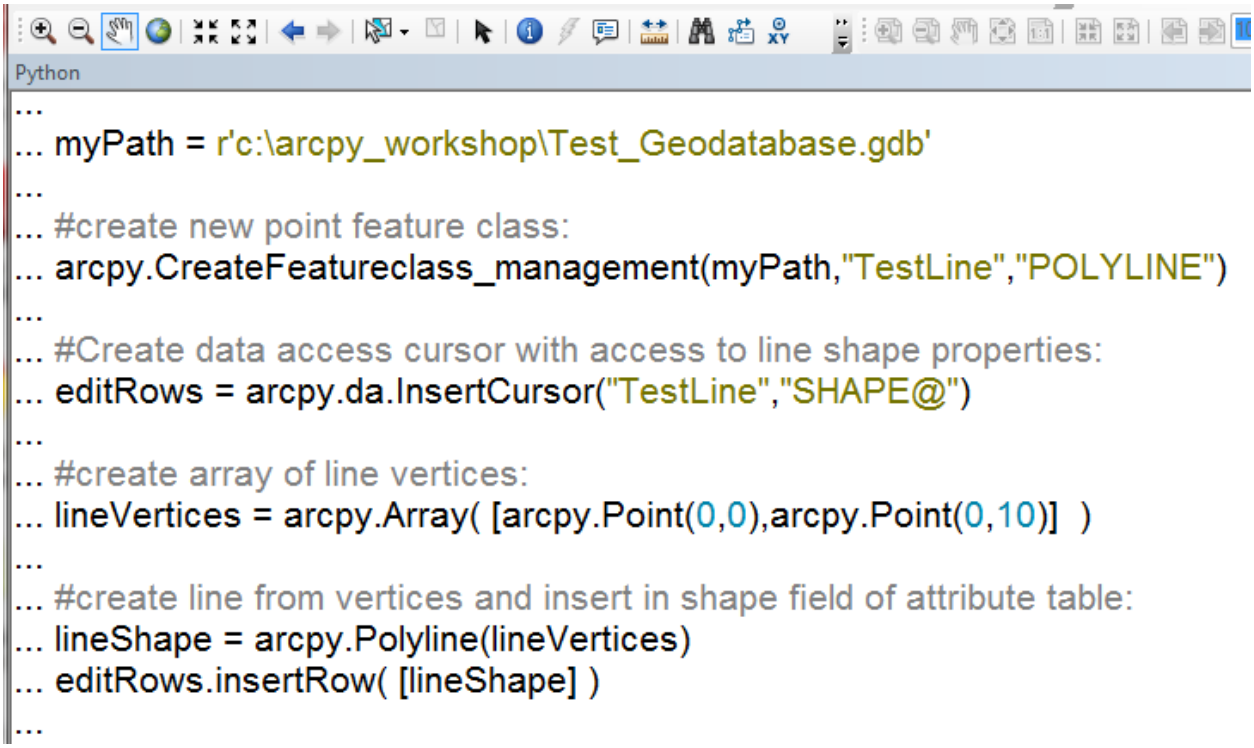
Notice that the point cloud is more variable in the Y-direction since we gave it a potential range of 400 in our Python script, compared to a potential range of 200 in the X-direction.



Your point cloud will look differently since the points are randomly located…

Creating PolyLine Features

To create polylines, you create a list of vertices. Load the following Python script named ***TestLine.py*** into your Arcmap Python window and execute the following lines:



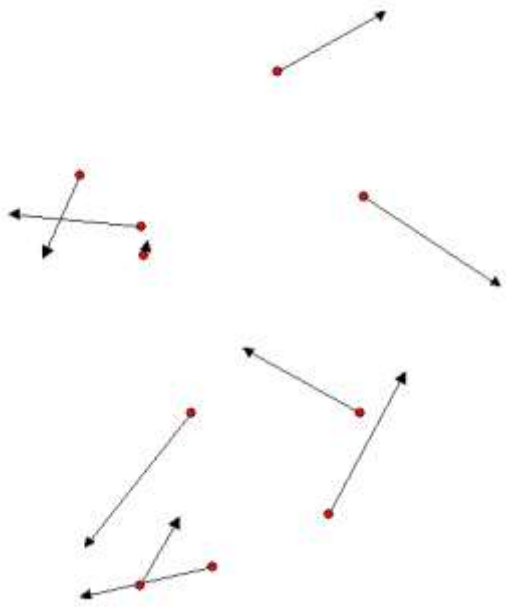
After executing your script, look at the layer properties extent and the line attribute table…is the extent from 0,0 to 0,10?

Is the line length 10.0?

Data Access Search Cursor

The data access ***insert cursor*** allows you to create ***new*** features or rows and insert new field values. The data access ***search cursor*** allows you to access (but not change) ***existing*** features or rows and field values.

We want a random vector at each point location.

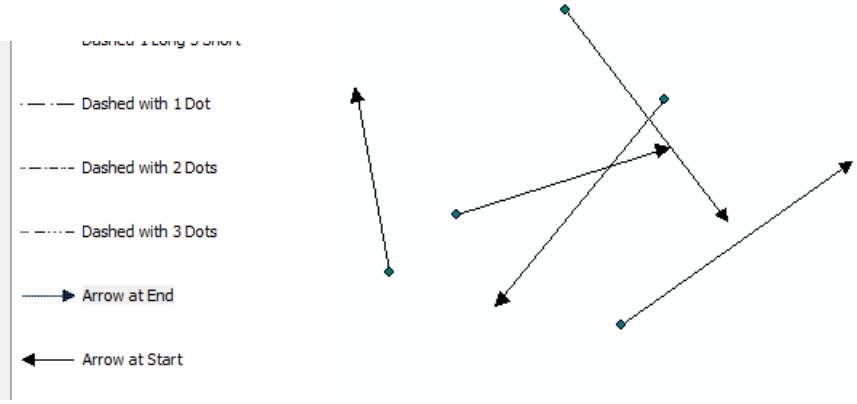


You can do this by accessing the existing point X,Y values and then randomly adding or subtracting and random value from the X and Y value to define the end of each new line. Load and run the Python script ***RandomVectors.py*** to create a random vector ranging in length from 0 to 141 at each point location, by randomly adding -100 to + 100 from each point x,y to create and line endpoint.

```
Python
... arcpy.CreateFeatureclass_management(myPath,"RanVectors","POLYLINE")
...
... #Create data access cursor with access to line shape properties:
... editRows = arcpy.da.InsertCursor("RanVectors","SHAPE@")
...
... #Create search cursor to read point shape field:
... ptRows = arcpy.da.SearchCursor("Ran_Points","Shape@XY")
...
... import random
... #loop through each row in point attribute table:
... for row in ptRows:
...     lineVertices = arcpy.Array( ) #array to hold first and last vertex
...     pointObj = row[0] #first field which is Shape with X,Y properties
...     x = pointObj[0]
...     y = pointObj[1]
...     lineVertices.add( arcpy.Point(x,y) )#location at beginning of line
...     endX = (x - 100.0) + random.random() * 200.0
...     endY = (y - 100.0) + random.random() * 200.0
...     lineVertices.add( arcpy.Point(endX,endY) )
...     lineShape = arcpy.Polyline(lineVertices)
...     editRows.insertRow([lineShape]) #save square shape in polyline table
...
... #all done, so quit editing and unlock tables:
```

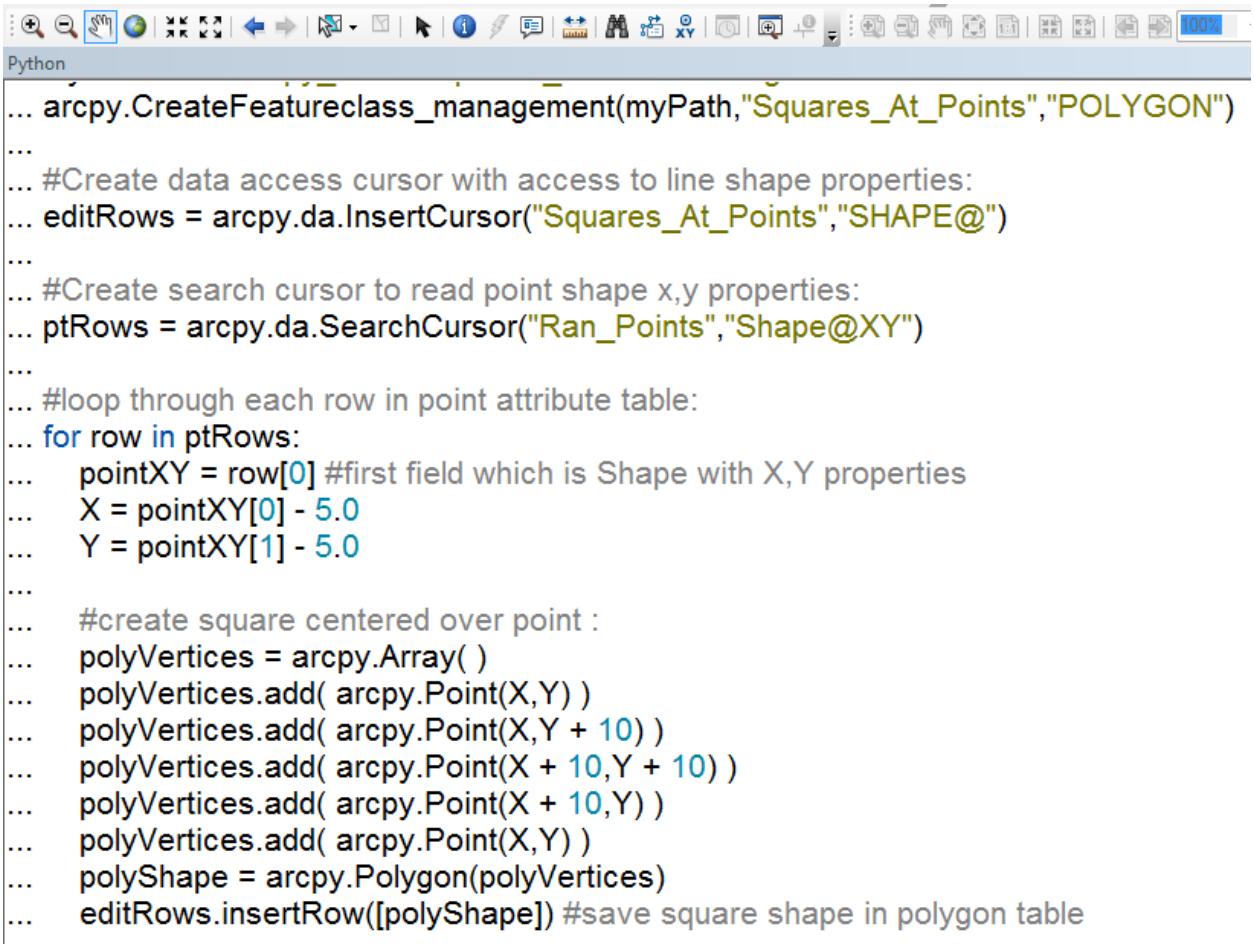
Note that we use the da.SearchCursor() to get each X,Y shape properties from the shape field of Ran_Points row, then create an array starting at the point X,Y location and then randomly creating the ending X,Y values for each vector.

Change your line symbology to arrow at end:

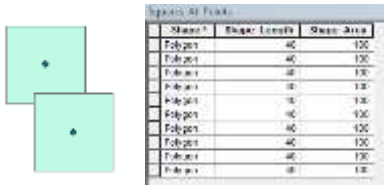


Creating Square Buffers

To create polygons, you create a container of vertices (an Array) and then use the vertices to create a polygon shape. In the next example, we will create a square buffer around each point. In your Arcmap window, load and execute the Python script ***SquareBuffers.py***.



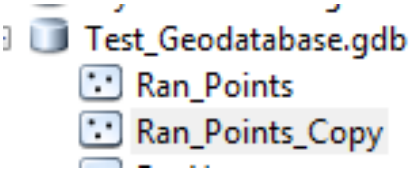
Once again we use a data access Search Cursor to get the shape X,Y coordinates from each point. Then we loop through each point and create a square polygon 10 meters right, left, up, down from the point location.



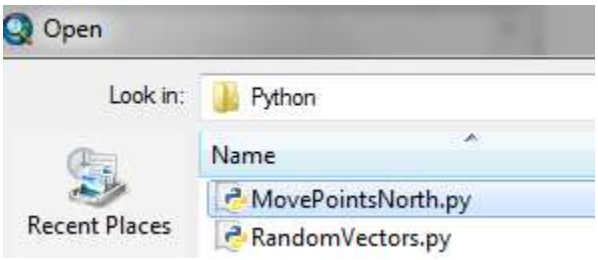
Data Access Update Cursor

The data access ***insert cursor*** allows you to create ***new*** features or rows and insert new field values. The data access ***search cursor*** allows you to access (but not change) ***existing*** features or rows and field values. The data access ***update cursor*** allows you to ***change*** features or rows and change field values.

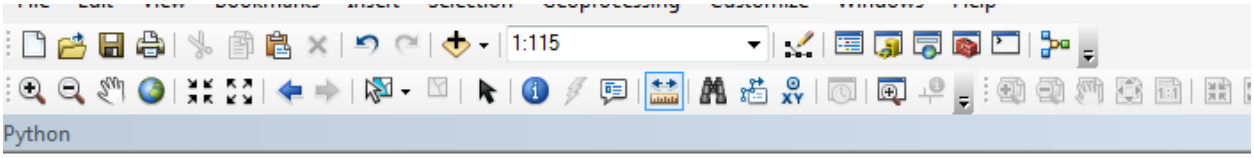
As an example, imagine that you need to move your first, third and ninth random point north by 2 meters. First use the Catalog window to make a copy of your random points feature class.



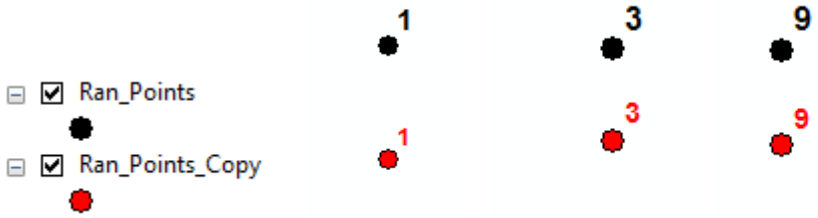
Then load the Python script into your Arcmap Python window:



This script uses an Update Cursor to select points that satisfy an SQL query and then moves those selected points north 2 meters by updating the point shape X,Y properties.



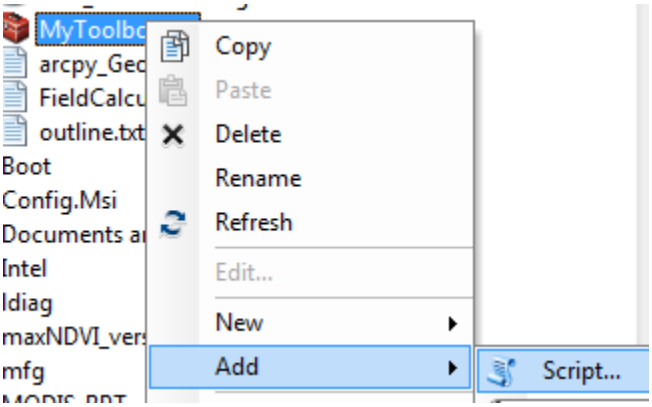

```
>>> #Update cursor to move selected random points north 2 meters
... myPath = r'c:\arcpy_workshop\Test_Geodatabase.gdb'
...
... #Create update cursor for updating selected records
... myQuery = '  "OBJECTID" in (1,3,9)  '
... ptRows = arcpy.da.UpdateCursor("Ran_Points","Shape@XY",myQuery)
...
... #loop through selected (myQuery)row in point attribute table:
... for row in ptRows:
...     pointObj = row[0] #first field which is Shape with X,Y properties
...     x = pointObj[0]
...     y = pointObj[1] + 2.0 #move north 2 meters
...     row[0] = (x,y)
...     ptRows.updateRow(row) #update selected row in point attribute table
...
... #all done, so quit editing and unlock tables:
... del ptRows;del pointObj
```

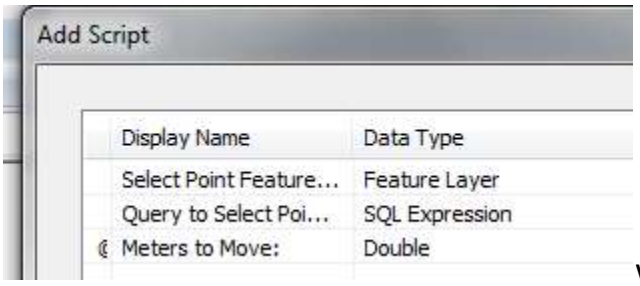


Any of these Python scripts can be modified to be more flexible as Script tools…

```
#Update cursor to move selected random points northDist meters
ptFC = arcpy.GetParameterAsText(0)

#Create update cursor for updating selected records
myQuery = arcpy.GetParameterAsText(1)
northDist = arcpy.GetParameterAsText(2)

ptRows = arcpy.da.UpdateCursor(ptFC,"Shape@XY",myQuery)

#loop through selected (myQuery)row in point attribute table:
for row in ptRows:
    pointObj = row[0] #first field which is Shape with X,Y properties
    x = pointObj[0]
    y = pointObj[1] + float(northDist) #move north 2 meters
    row[0] = (x,y)
    ptRows.updateRow(row) #update selected row in point attribute table

#all done, so quit editing and unlock tables:
del ptRows;del pointObj
```

which would be our next step if we had more time. For example:

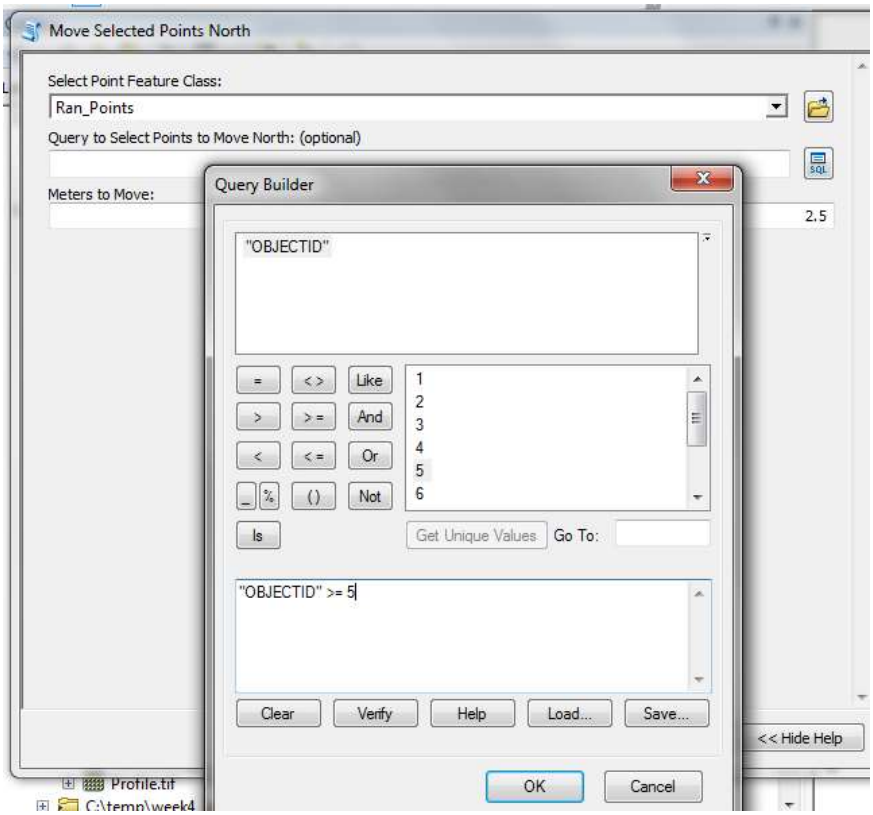




where the SQL expression is obtaining the fields from the first parameter…***Obtained from***

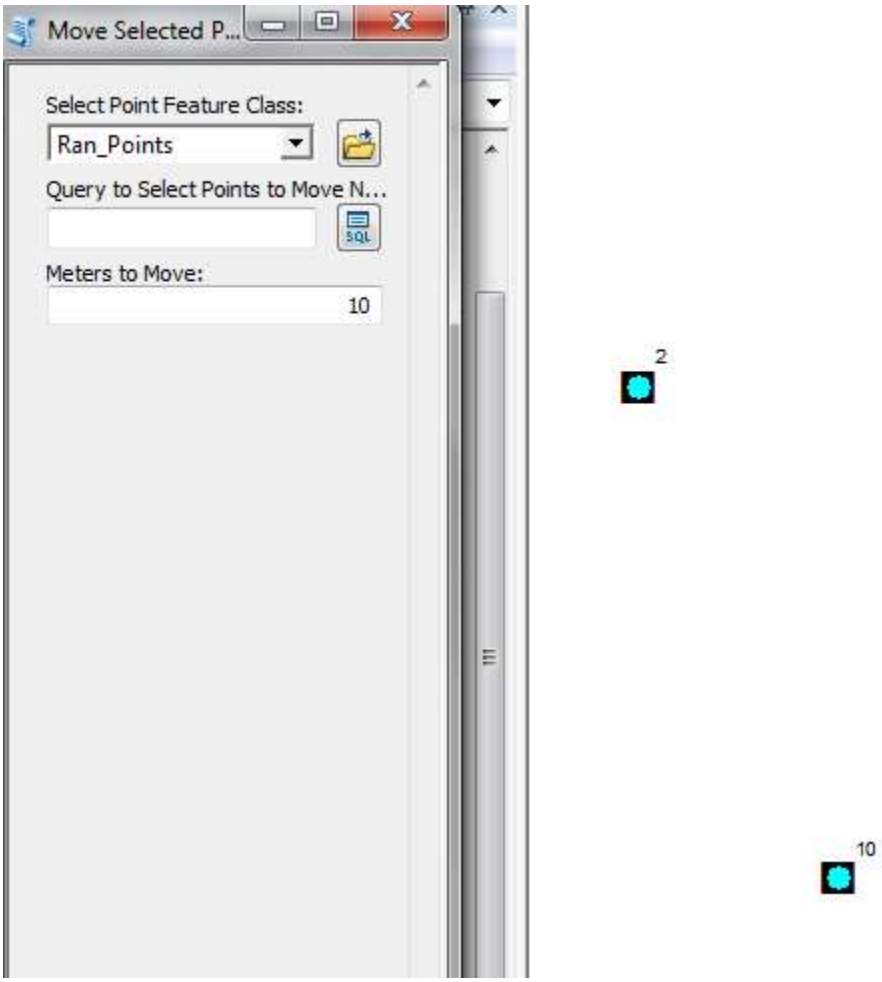
Parameter Properties

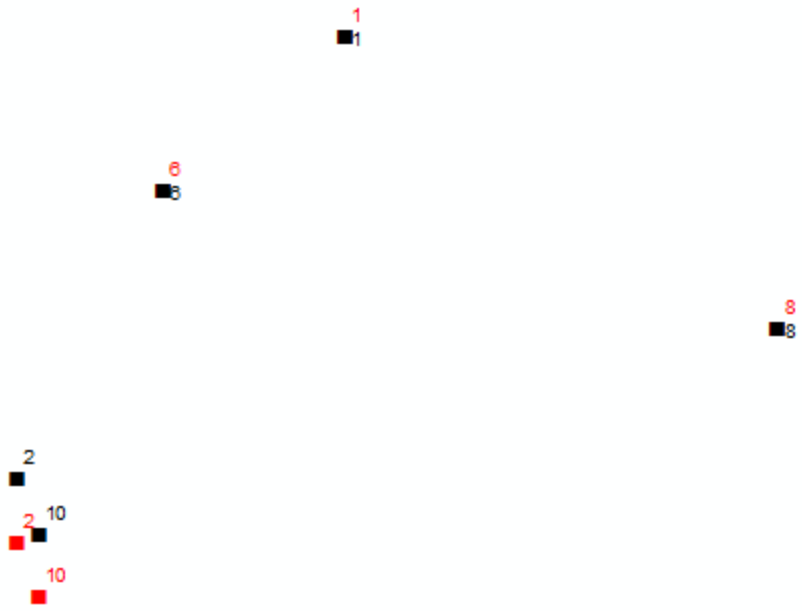
Property	Value
MultiValue	No
Default	
Environment	
Filter	None
Obtained from	Select_Point_Feature_Class_
Symbology	



Without using the optional sql query, you could select the points you want to move north.

For example, here we select points 2 and 10:





So only the selected points were moved 10 meters north in this example.

Thank you for participating in this workshop.

The workshop materials are posted and downloadable at:
http://nrm.salrm.uaf.edu/~dverbyla/workshops/arcpy_workshop/

For some youtube sessions on Python and arcpy see:

http://www.youtube.com/user/dlverbyla/playlists

I will also be teaching a full semester elearning course on arcpy Spring semester 2015…