



Lesson 6: Working with and Creating Geometries

Basic Programming in ArcGIS with Python
Workshop

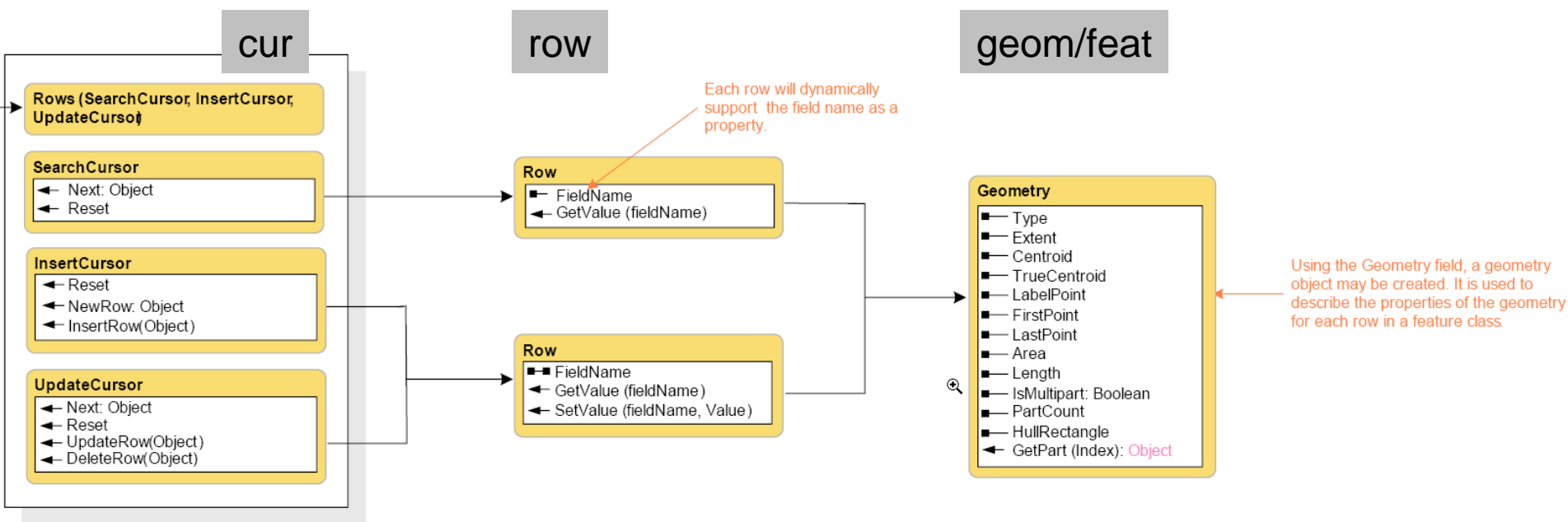
RS/GIS Lab, Utah State University
With Material from ESRI

Learning Objectives:

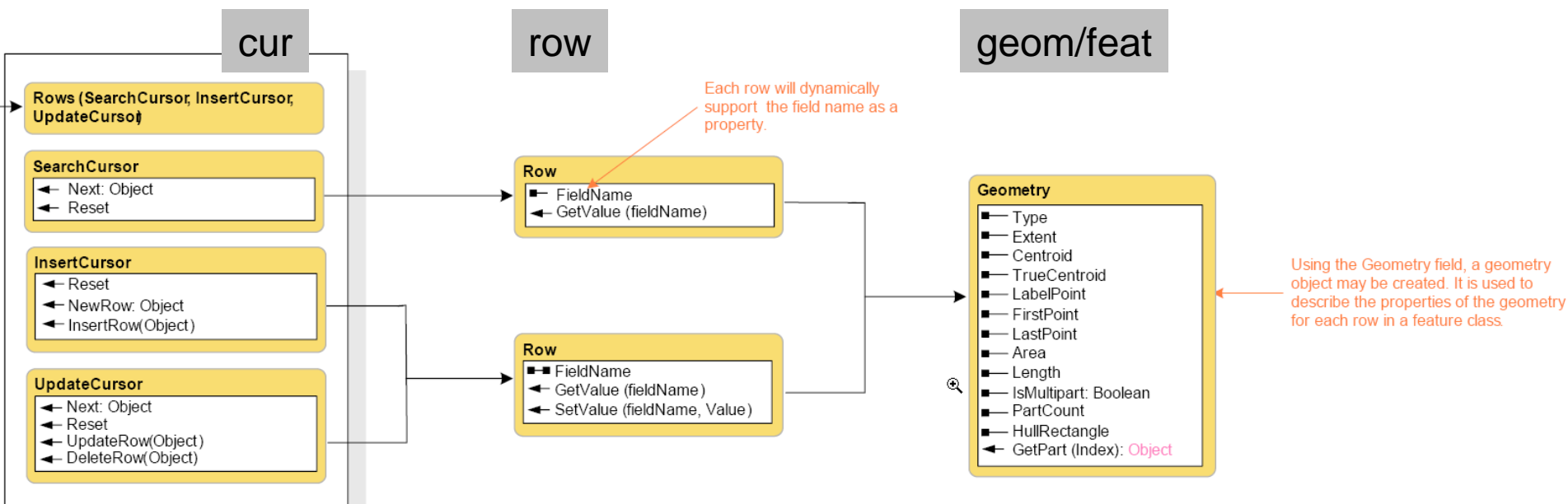
- Short review of cursors.
- Accessing geometry objects—Properties.
- Accessing geometry objects—Method.
- Point and array geometry objects.
- Creating point, line, and polygon features with geometry objects.
- Working with data sources (text files of x,y coords).
- Creating features from data sources.

Lesson 6a: Accessing Geometry Objects—Properties

- Review of uses for Cursors:
- SearchCursor—To read values in a row
- InsertCursor—To insert new rows
- UpdateCursor—To make changes to values in rows, and delete rows



Example Code: SearchCursor with Geometry field



```

Lesson6a_junk.py
15 # Create search cursor
16 cur = gp.SearchCursor("nests1990.shp")
17 row = cur.Next()
18 -while row:
19     print "The condition of the nest is:" +row.condition
20     feat = row.shape
21     print "The type of feature is: " +feat.type
22     print "\n"
23     row = cur.Next()
24
    
```

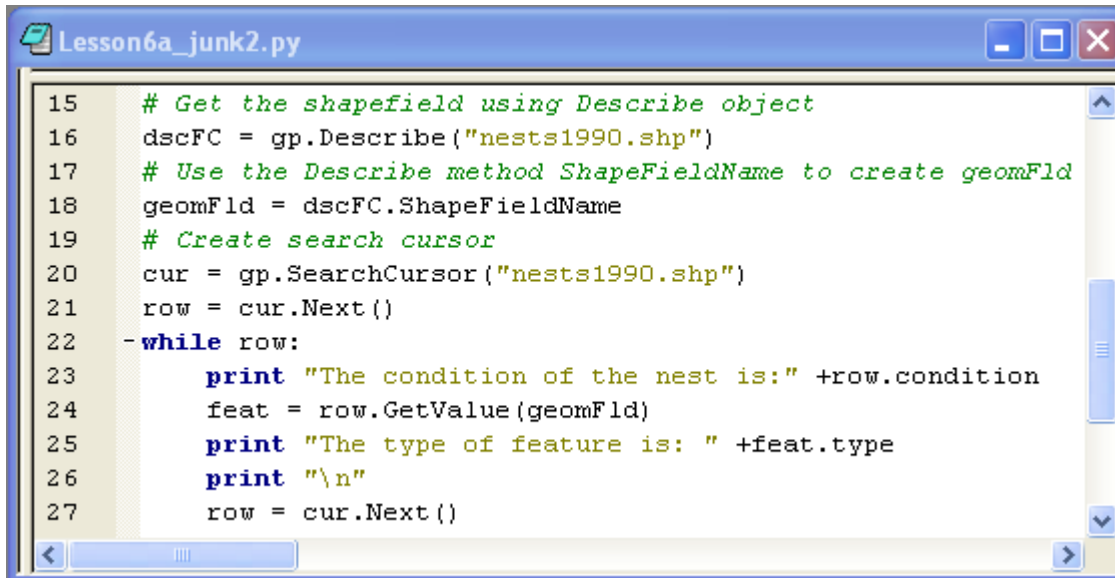
```

Interactive Window
The condition of the nest is:good
The type of feature is: point

The condition of the nest is:poor
The type of feature is: point
    
```

More on the *Geometry field*

- Shapefiles and geodatabases have a *Geometry field*.
- Not always called “shape” but usually.
- From the *Geometry* field a geometry object can be created.
- Geometry object is created for a single feature/row

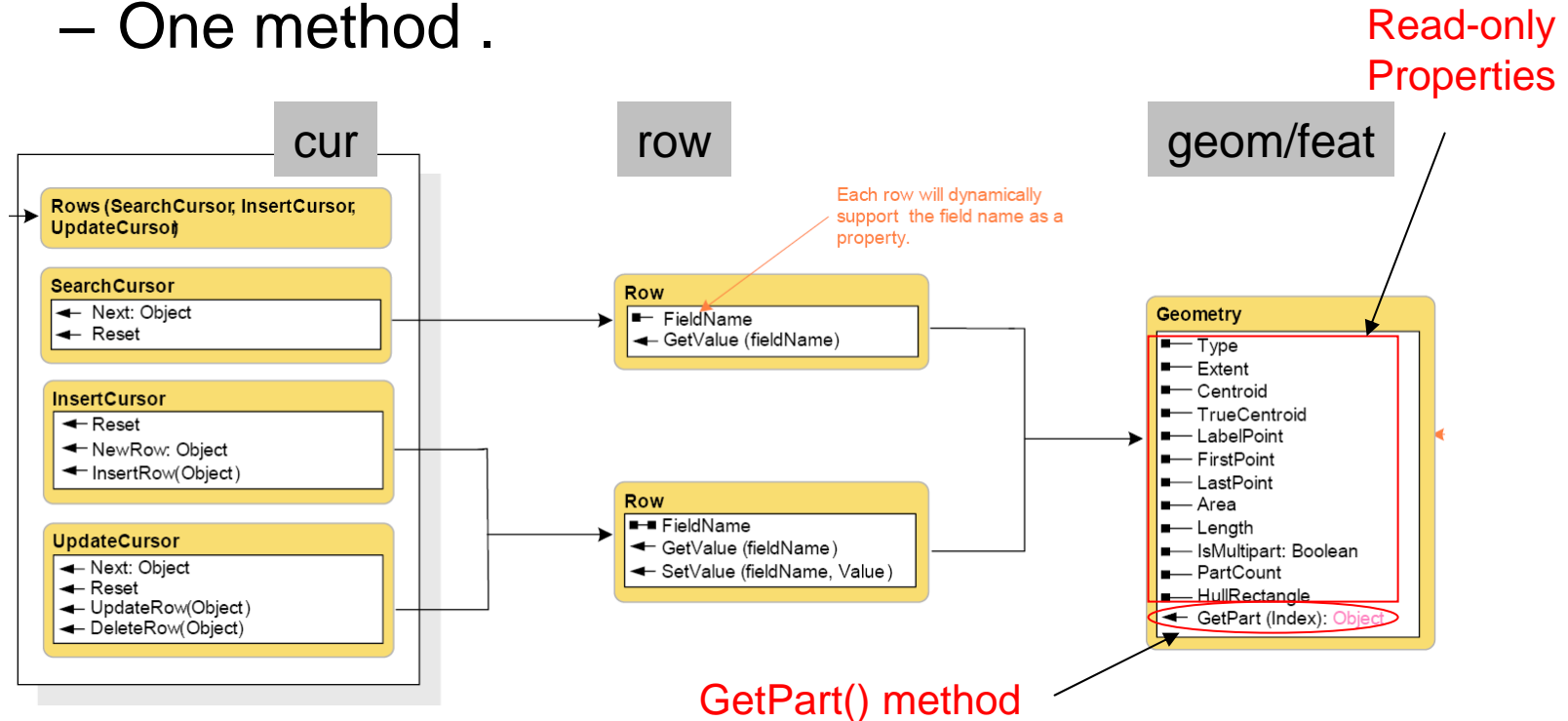


```

15  # Get the shapefield using Describe object
16  dscFC = gp.Describe("nests1990.shp")
17  # Use the Describe method ShapeFieldName to create geomFld
18  geomFld = dscFC.ShapeFieldName
19  # Create search cursor
20  cur = gp.SearchCursor("nests1990.shp")
21  row = cur.Next()
22  -while row:
23      print "The condition of the nest is:" +row.condition
24      feat = row.GetValue(geomFld)
25      print "The type of feature is: " +feat.type
26      print "\n"
27      row = cur.Next()
  
```

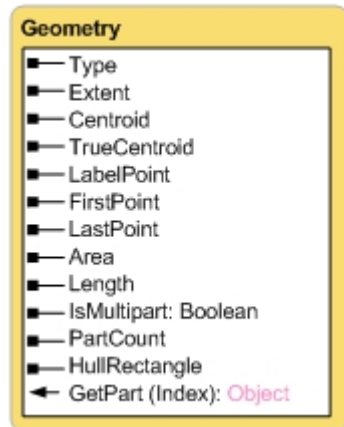
The Geometry Object

- Provides information about an individual feature.
- Properties and methods
 - Read-only properties
 - One method .



Geometry Properties

- Provides information about an individual feature.
- Properties and methods
 - Read-only properties
 - One method



Type	Null, point, multipoint, line, circulararc, ellipticarc, bezier3curve, path, polyline, ring, polygon, envelope, any, bag, multiPatch, triangleStrip, triangleFan, ray, sphere.
Extent	XMin; YMin; XMax; YMax.
Centroid	Returns the true centroid if it is within or on the feature, otherwise the label point is returned.
TrueCentroid	The center of gravity for a feature.
LabelPoint	The point at which the label is located. The LabelPoint is always located within or on a feature.
FirstPoint	The first coordinate of the feature.
LastPoint	The last coordinate of the feature.
Area	The area of a polygon. Empty for all other feature types.
Length	The length of the linear feature. Empty for point, multipoint feature types.
IsMultipart	True, if the number of parts for this geometry is more than one.
PartCount	The number of geometry parts for the feature.
HullRectangle	The coordinate pairs of the convex hull rectangle.
GetPart(optional index)	Returns an array of point objects for a particular part of geometry or an array containing a number of arrays, one for each part.

Example: Using geometry properties

- Some properties return an x,y coordinate pair.
 - *Extent, centroid, true centroid, first point, last point.*
 - Returned in the format of a string.
 - Parse it with split method.
 - Turn in to an integer with `float` then `int` methods

```
# Enter while loop for each feature/row
while row:
    # Use the geometry field 'shape' to create geom object
    feat = row.shape
    # Get the centroid x,y coords as a string
    cent = feat.centroid
    # Split the coord pair into a list of two strings
    cent = string.split(cent, " ")
    # Get the x and y string from the list
    xcent = cent[0]
    ycent = cent[1]
    # Must convert to float number first, then to int
    xc = int(float(xcent))
    yc = int(float(ycent))
```

Assignment 6a: Access Geometry Properties

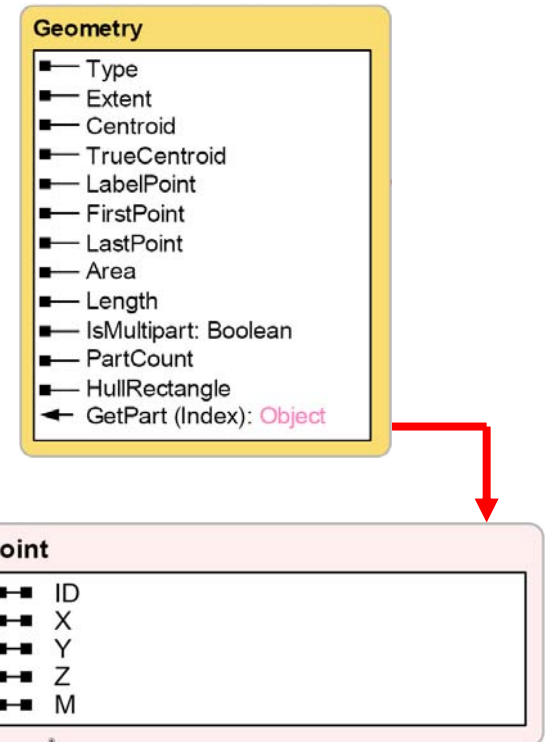
- Take a look at the datasets by opening the map document lesson6.mxd.
- Open the script lesson6a_view.py in PythonWin, and run it for the shapefiles in the Lesson6 folder (i.e. NewTrails.shp, nests1990.shp and HmRange1990.shp). You will have to change the path, and note that it requires a dataset name as an argument. The dataset name has to be exactly as it is in the folder.
- Start with the script lesson6a.py and write a script that finds which polygon in the shape file HmRange1990.shp has a centroid that is different from its true centroid (i.e. center of area vs. center of gravity).

Lesson 6b: Accessing Geometry Objects—Methods

Geometry Method: GetPart() –The Point Object

- When a point feature, a single point object is returned
- The point object has properties:
 - ID, X, Y, Z, M





ID	A long value used to uniquely identify the point.
X	The horizontal coordinate of the point.
Y	The vertical coordinate of the point.
Z	The elevation value of the point.
M	The measure value of the point.

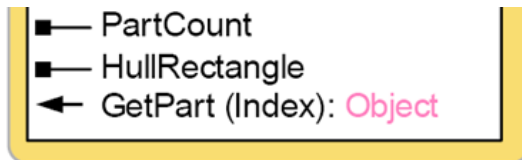


Example Code: GetPart() –The Point Object

```
# Create search cursor
cur = gp.SearchCursor("nests1990.shp")
row = cur.next()

# Enter while loop for each feature/row
while row:
    # Get the geometry object (e.g. nest point)
    feat = row.shape
    # Get a single point object
    pnt = feat.getpart()
    # Print id fld., id, x and y prop. for pnt obj
    print row.id, pnt.id, pnt.x, pnt.y
    # Go to next row (i.e. feature) in the cursor
    row = cur.next()
```

-  Pnt with x,y, Id
-  Pnt with x,y, Id
-  Pnt with x,y, Id
-  Pnt with x,y, Id



ArcGIS review: Features can have multiple parts



- Geometry field stores the geometry
 - Each geometry has at least one “part”
 - Some geometries can have more than one part
- Classic Example: Hawaii
- Methods on the geometry object
 - IsMultiPart
 - PartCount

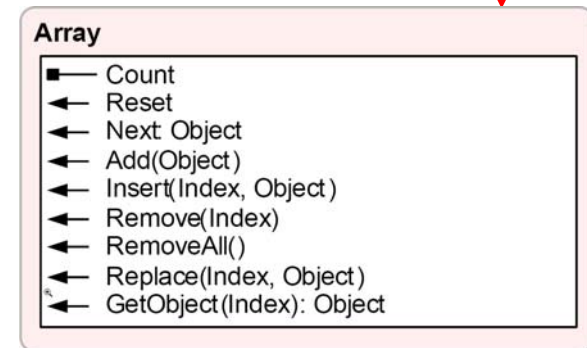
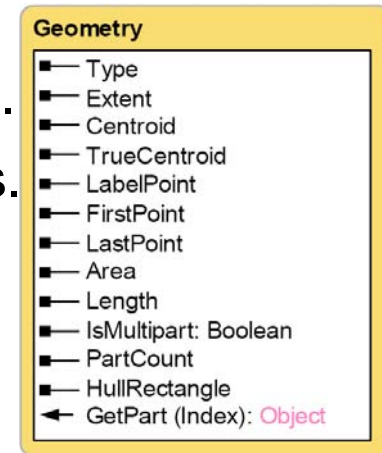
Selected Attributes of STATES

FID	Shape ^	AREA	STATE_NAME	STATE_FIPS	SUB_REGION	STATE_ABBR	POP1990	POP1999	POP99
49	Polygon	6381.227	Hawaii	15	Pacific	HI	1108229	1195992	

Record: 0 Show: All Selected Records (1 out of 51 Selected) Options

Geometry Method: GetPart() –The Array Object

- When a polygon, polyline, or multipoint feature an array object of point objects is returned if index is specified.
- If index not specified, returns an array containing an array of point objects for *each* geometry part.
- The array object stores a series of point objects.
- An array has one property and several methods.

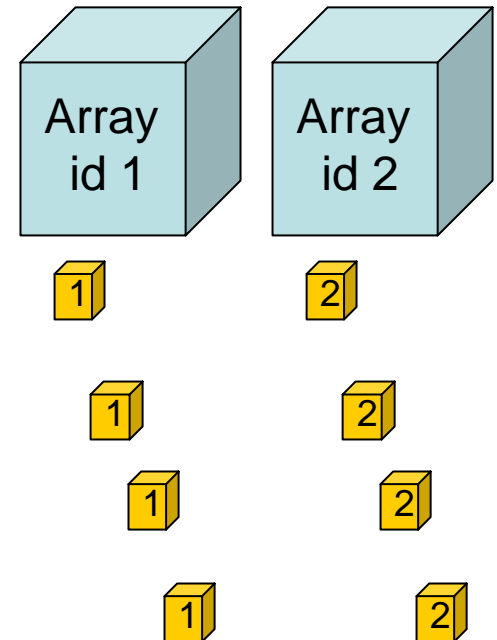


Count	The number of objects in the array.
Reset()	Resets the array to the first object.
Next()	Returns the next object in the array.
Add(Object)	Adds an object to the array in the last position.
Insert(Index, Object)	Adds an object to the array in a specific position.
Remove(Index, Object)	Removes a specific object from the array.
RemoveAll()	Removes all objects and creates an empty array.
GetObject(Index)	Returns a specific object from the array.

Example Code: GetPart() –The Array Object

```
# Create search cursor
cur = gp.SearchCursor("OldTrails.shp")
row = cur.Next()

# Enter while loop for each feature/row
while row:
    # Get the geometry object (e.g. line)
    feat = row.shape
    # Get an array of pnt objs for the 1st part (0)
    array = feat.getpart(0)
    # Get the first pnt obj in the array
    pnt = array.next()
    # Enter while loop for each pnt object
    while pnt:
        # Print id fld., id, x and y prop. for pnt obj
        print row.id, pnt.id, pnt.x, pnt.y
        # Go to the next pnt in the array
        pnt = array.next()
    # Go to the next row in the cursor
    row = cur.next()
```

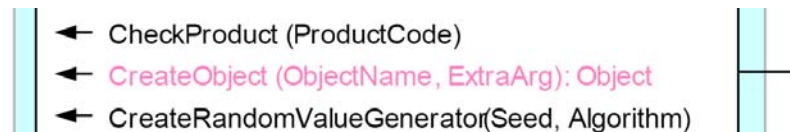


Assignment 6b: Access Geometry with GetPart() Method

- Start with the script lesson6b.py.
- Write a script that accesses the geometry objects for the polyline shapefile NewTrials.shp, and writes the vertices to a single textfile. Because this is not a point feature class, you will have to use array objects in addition to point objects to access the x,y properties.
- Hint: First get a script working that prints the row.id, x and y coordinates, then get it to work so that it writes to the text file.

Lesson 6c: Creating Geometries

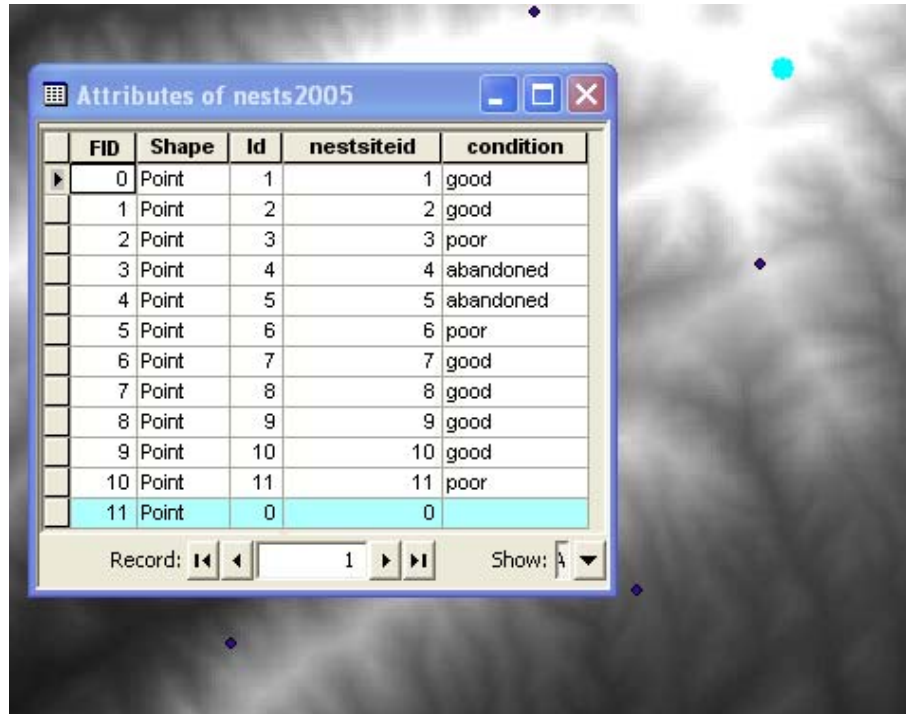
- Use CreateObject method on the geoprocessor object
- Use InsertCursor to create the new feature/row
- Basic steps:
 - 1) Create cursor object with InsertCursor
 - 2) Create geometry object* with CreateObject Method on gp
 - 3) Assign X, Y, or id properties to pnt object
 - 4) Add/insert a new row to the cursor object
 - 5) Assign the geometry object to the geometry field
 - 6) Save the “inserted” row/feature with the InsertRow() method



* Note: for point features you'll create a pnt obj, otherwise you create an array and pnt objs.

Example Code: Create Point Features (for an existing feature class)

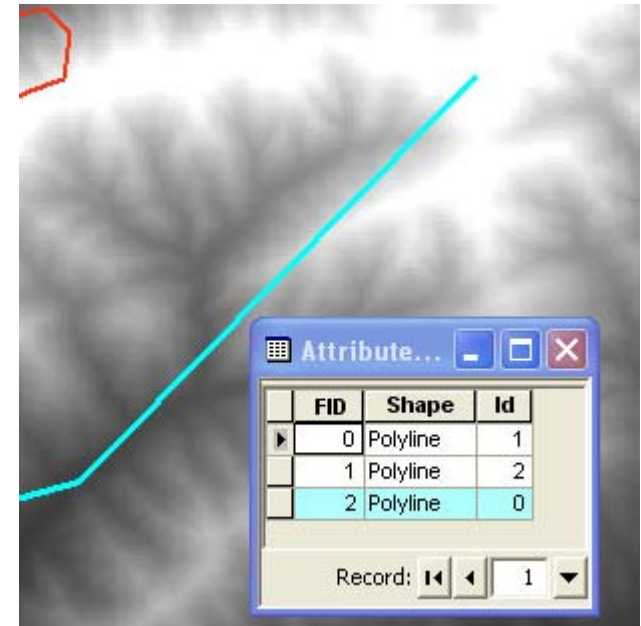
```
# Create cursor and pnt objects
cur = gp.InsertCursor("nests2005.shp")
pnt = gp.CreateObject("point")
# Assign x and y properties to pnt obj
pnt.X = 437565
pnt.Y = 4519249
# Create new row to put pnt into
row = cur.NewRow()
# Assign pnt obj to Shape field
row.Shape = pnt
# "save" the insert
cur.InsertRow(row)
```



FID	Shape	Id	nestsiteid	condition
0	Point	1	1	good
1	Point	2	2	good
2	Point	3	3	poor
3	Point	4	4	abandoned
4	Point	5	5	abandoned
5	Point	6	6	poor
6	Point	7	7	good
7	Point	8	8	good
8	Point	9	9	good
9	Point	10	10	good
10	Point	11	11	poor
11	Point	0	0	

Example Code: Create Line Features (for an existing feature class)

```
# Create cursor, array, and pnt objects
cur = gp.InsertCursor("OldTrails.shp")
array = gp.CreateObject("Array")
pnt = gp.CreateObject("point")
# Assign x and y prop. to pnt obj (vertex 1)
pnt.X = 431314
pnt.Y = 4515672
array.Add(pnt) # Add pnt obj to the array
# Assign x and y prop. to pnt obj (vertex 2)
pnt.X = 434603
pnt.Y = 4516572
array.Add(pnt) # Add pnt obj to the array
# Assign x and y prop. to pnt obj (vertex 3)
pnt.X = 437248
pnt.Y = 4519284
array.Add(pnt) # Add pnt obj to the array
# Create new row to put array into
row = cur.NewRow()
# Assign array obj to Shape field
row.Shape = array
# "save" the insert
cur.InsertRow(row))
```



Example Code: Create Polygon Features (for an existing feature class)

```
# Create cursor, array, and pnt objects
cur = gp.InsertCursor("HmRange1990.shp")
array = gp.CreateObject("Array")
pnt = gp.CreateObject("point")

pnt.X = 431314
pnt.Y = 4515672
array.Add(pnt)
pnt.X = 434603
pnt.Y = 4516572
array.Add(pnt)
pnt.X = 437248
pnt.Y = 4519284
array.Add(pnt)
# Add first x,y coordinate again
pnt.X = 431314
pnt.Y = 4515672
array.Add(pnt)

row = cur.NewRow()
row.Shape = array
cur.InsertRow(row)
```



- Because the featureclass (HmRange1990.shp) is a polygon, the array is expected to be a polygon (i.e. must close)
- Python won't know, but ArcMap will
- Use Check Geometry tool (Database Management) to check for geometry problems

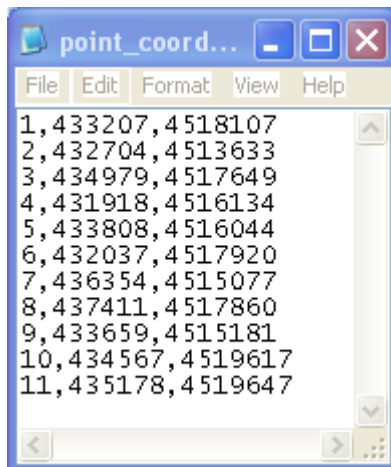
Creating a New Feature Class

- Use CreateFeatureClass_management tool before creating cursor, pnt, and/or array objects.
- Usage: CreateFeatureClass_management (out_path, out_name, geometry_type, template, has_m, has_z, spatial_reference, config_keyword, spatial_grid_1, spatial_grid_2, spatial_grid_3)
- Where out_path is the workspace, out_name is the name of the feature class, geometry_type is POINT, MULTIPOINT, POLYGON, or POLYLINE, and spatial_reference can be defined by the spatial reference of another feature class.
- Example:

```
# Create a new polyline feature class
Gp.CreateFeatureClass_management(r"C:\Python\Lesson6",
                                r"C:\Python\Lesson6\NewTrails.shp",
                                "polyline",
                                " ", " ", " ",
                                r"C:\Python\Lesson6\nests1990.shp")
```

Working with Source Data (i.e. from text file)

- Coordinate pairs must be stored on a single line.
- 1st position = ID, 2nd position = X coord., 3rd position = Y coord.
- Format for points, multipoint, lines, and polygons differs.

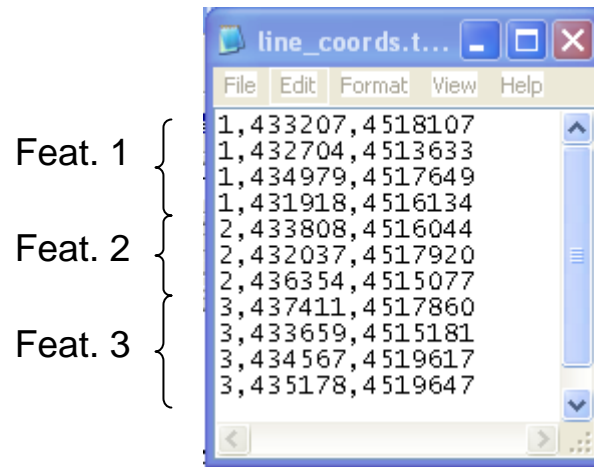


Point features

-One point per id

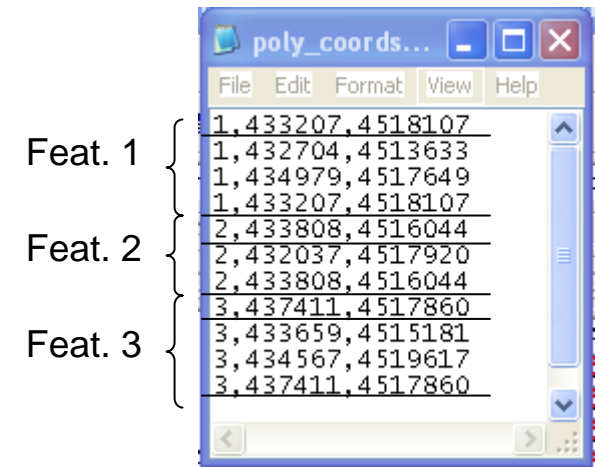
Multipoint features

-Many points per id



Line features

- IDs distinguish features



Polygon features

-IDs distinguish features

-Last point same as first

Working with Source Data

- Basic steps (for point data):

- 1) Create feature class with CreateFeatureClass_management
- 2) Create cursor and point objects
- 3) Open the text file
- 4) Loop through the textfile using readlines method
- 5) Use split method, then assign ID, X, and Y prop. to each pnt obj.
- 6) Add/insert a new row to the cursor object
- 7) Assign the geometry object to the geometry field
- 8) Save the “inserted” row/feature with the InsertRow() method
- 9) Close the text file

```
Step 2 { # Create Point, and Cursor objects
        pnt = gp.CreateObject("Point")
        cur = gp.InsertCursor("nests2007.shp")
Step 3 { # Open the text file
        input = open(txtFile, "r")
Step 4 { # Loop through the coordinate values
        for line in input.readlines():
Step 5 {         values = string.split(line, ",")
                pnt.id = values[0]
                pnt.x = values[1]
                pnt.y = values[2]
```

Example Code: Creating a Point Feature Class

```
# Set variable for new shapefile
newFC = r"C:\john\Lesson6_results\nests2007.shp"

# Create a new feature class
gp.CreateFeatureClass_management
    (os.path.dirname(newFC),
     os.path.basename(newFC),
     "point", "", "", "",
     r"C:\john\Lesson6_results\nests1990.shp")

# Create Point and Cursor objects
pnt = gp.CreateObject("Point")
cur = gp.InsertCursor(newFC)

# Open the text file
input = open(txtFile, "r")

# Loop through the 'paragraph' line by line
for line in input.readlines():
    values = string.split(line, ",")
    pnt.id = values[0]
    pnt.x = values[1]
    pnt.y = values[2]
    row = cur.NewRow() # New row into table/cursor
    row.shape = pnt   # Assign pnt obj prop.to shape
    cur.InsertRow(row) # Save the row/feature
```

1

Pnt.id, pnt.x, pnt.y

2

Pnt.id, pnt.x, pnt,y

3

Pnt.id, pnt.x, pnt.y

4

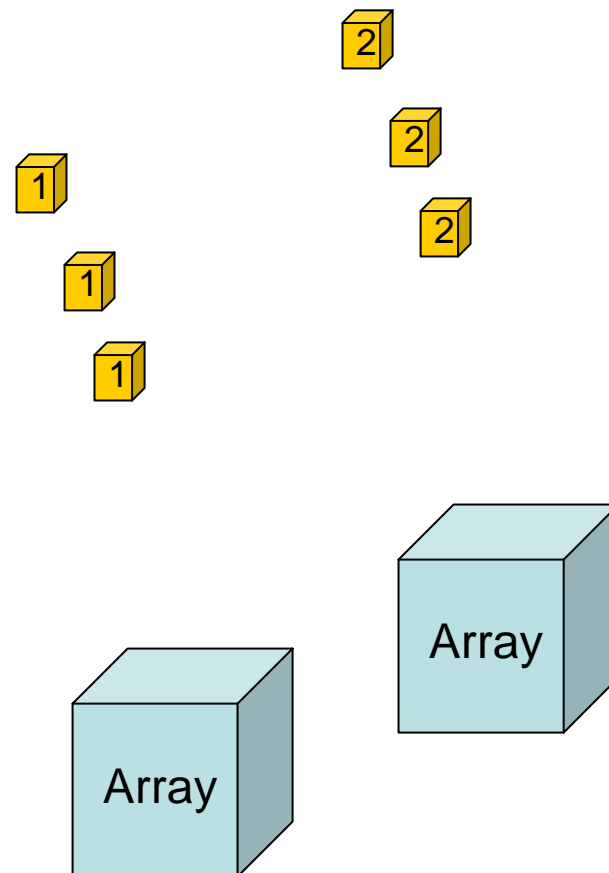
Pnt.id, pnt.x, pnt.y

```
# Open the text file
input = open(txtFile, "r")
tmpID = -1 # initialize to keep track of IDs

# Loop through the coordinate values ('paragraph')
for line in input.readlines():
    values = string.split(line, ",")
    pnt.id = values[0]
    pnt.x = values[1]
    pnt.y = values[2]
    print pnt.id, pnt.x, pnt.y
    # First loop only
    if tmpID == -1:
        tmpID = pnt.id
    # If the ID has changed create a new row/feature
    # and set value to geom field for last array
    if tmpID != pnt.id:
        row = cur.NewRow()
        # assign geometry from array to geom field
        # for the last array
        row.SetValue(shpFld, array)
        cur.InsertRow(row)
        # remove all pnts from last array
        array.RemoveAll()
    # Add pnt to the array
    array.Add(pnt)
    tmpID = pnt.id

# Insert the last feature to the feature class
row = cur.NewRow()
row.SetValue(shpFld, array)
cur.InsertRow(row)
```

Example Code: Creating a Polyline Feature Class



Assignment 6c: Creating New Features with Geometry Objects.

- Start with the script `lesson6c.py`.
- Write a script that:
 - Creates a new shapefile called `nests2007.shp`
 - Reads in the coordinates from the textfile `2007nests_coords.txt` to create `pnt` objects from which point features are inserted into the new shapefile (i.e. `nests2007.shp`)
- Suggestions: Most of the structure (i.e. pseudo code) is provided in the script you will start with. Think about what the code is doing (even where it is given) and use the handout (slides) to help you with writing the code. Add additional comments where necessary.