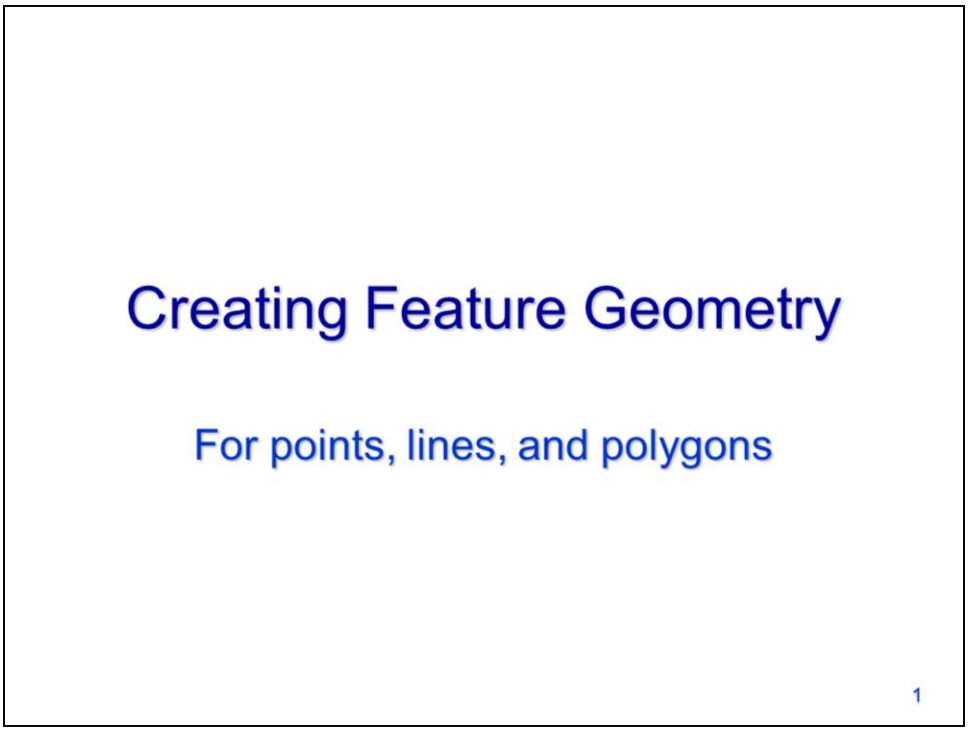


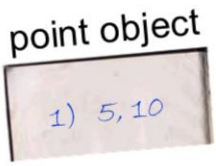
This video will discuss how to create new feature geometries for point, line, and polygon features.

Review: single-part point

- Single-part point feature…
  - Defined by single point object

FID	Shape *	Id	POINT_X
0	Point	0	500
1	Point	0	600

(5, 10)




```
>>> pnt = geom.getpart()
>>> pnt.X
5
>>> pnt.Y
10
```

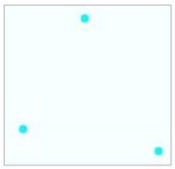
2

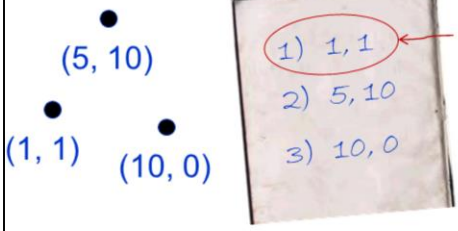
Let's begin by reviewing the storage structure for each feature type. A single-part point feature is stored as a point object since it has no need for a more complex storage structure.

Review: multi-part point

	FID	Shape *	OBJECTID
▸	0	Multipoint	0

- Multi-part point feature…
  - Each point defined by point object
  - Each point in feature contained in feature array…





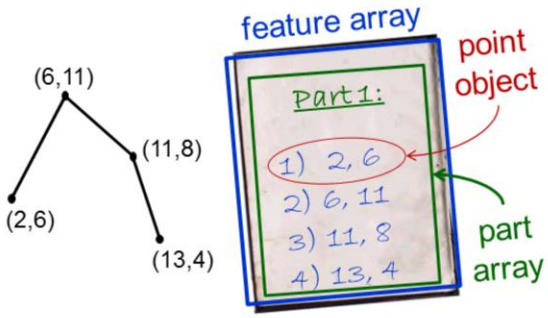

```
>>> fArray = geom.getPart()
>>> pnt = fArray[0]
>>> pnt.X
1
>>> pnt.Y
1
```

3

A multi-part point feature contains multiple points. Each point in the feature is stored in a point object. All of the point objects are stored in a feature array.

Review: line

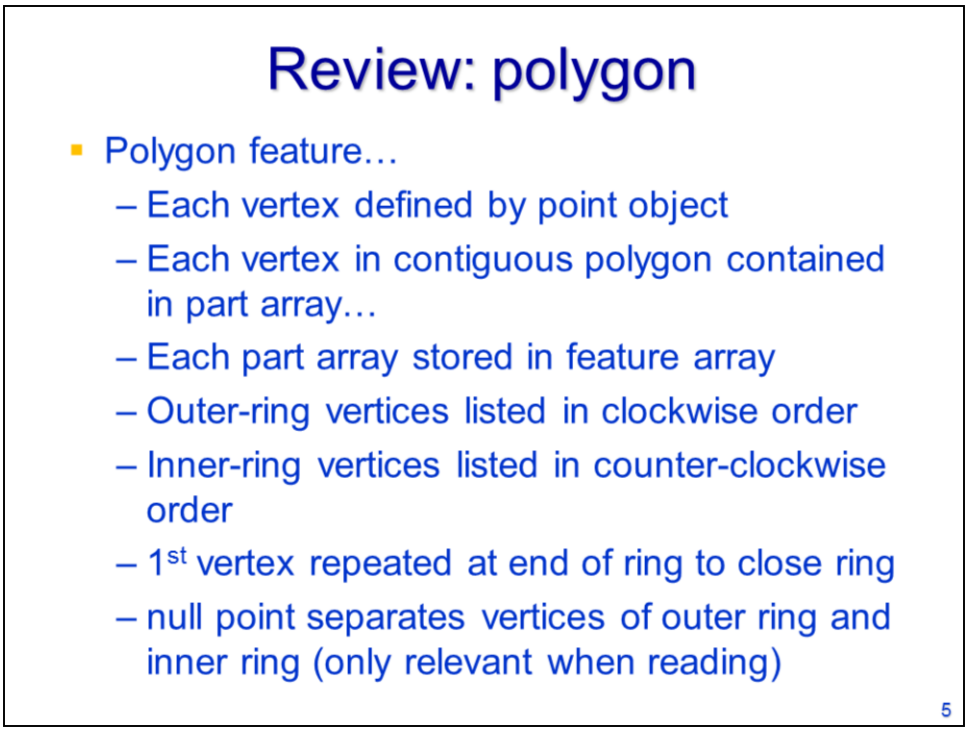
- Line feature…
  - Each vertex defined by point object
  - Each vertex in contiguous line contained in part array…
  - Each part array stored in feature array
  - Vertices listed in order




```
>>> fArray = geom.getPart()
>>> pArray = fArray[0]
>>> pnt = pArray[0]
>>> pnt.X
2
>>> pnt.Y
6
```

4

A line feature requires a more complex storage structure because it can have multiple parts and each part consists of multiple point objects. The point objects for each contiguous part are stored in a part array – the vertices are listed in the order that they are connected. The part array(s) are stored in a feature array. A line has the same storage structure even if it contains only a single part.



Polygon have the same storage structure as a line feature with vertices stored as point objects; point objects stored in part arrays, and part arrays stored in a feature array. However, polygons are more complex features than lines and so we need to follow a few additional rules:

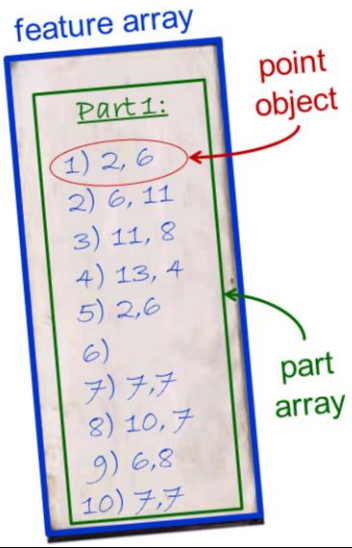
The vertices that correspond to the outer ring of a polygon must be listed in a clockwise order in the part array.

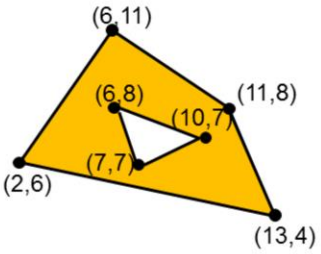
If a polygon has a hole in it, then the vertices that define the inner ring must be listed in a counter-clockwise order.

For both outer and inner rings, the first point and last point in the ring is the same.

The outer ring vertices are always specified before the inner ring(s) in the part array. A null point is used to separate the outer ring from the inner ring. The null point is added automatically by arcpy – you do not need to add the null point when creating a new geometry.

Review: polygon

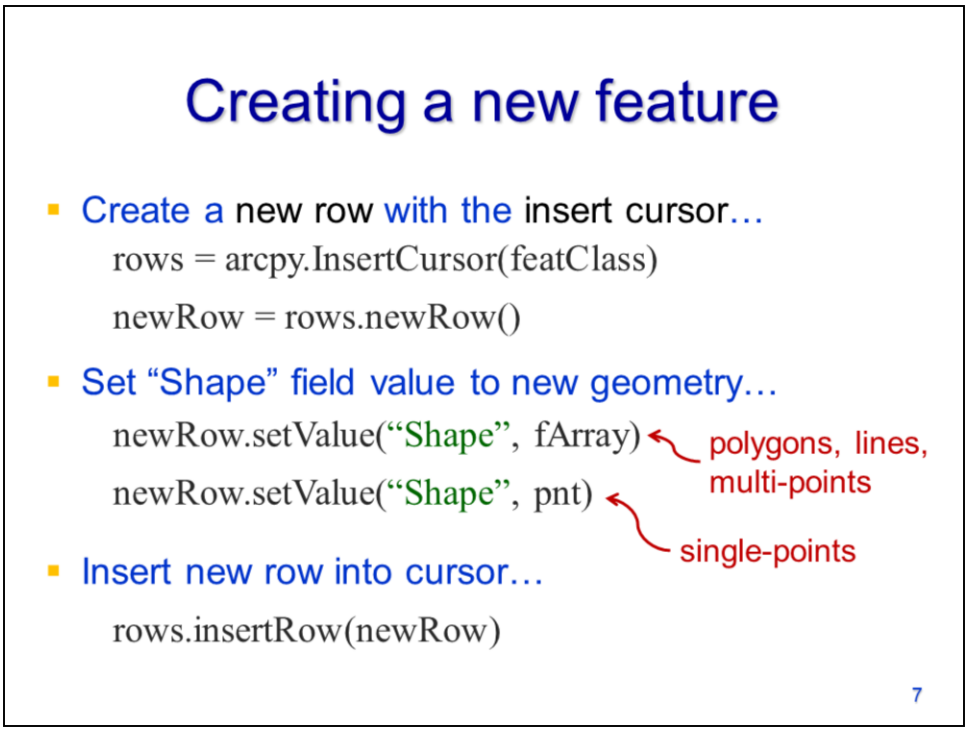





```
>>> fArray = geom.getPart()
>>> pArray = fArray[0]
>>> pnt = pArray[0]
>>> pnt.X
2
>>> pnt.Y
6
```

6

This slide illustrates the storage structure for a polygon.

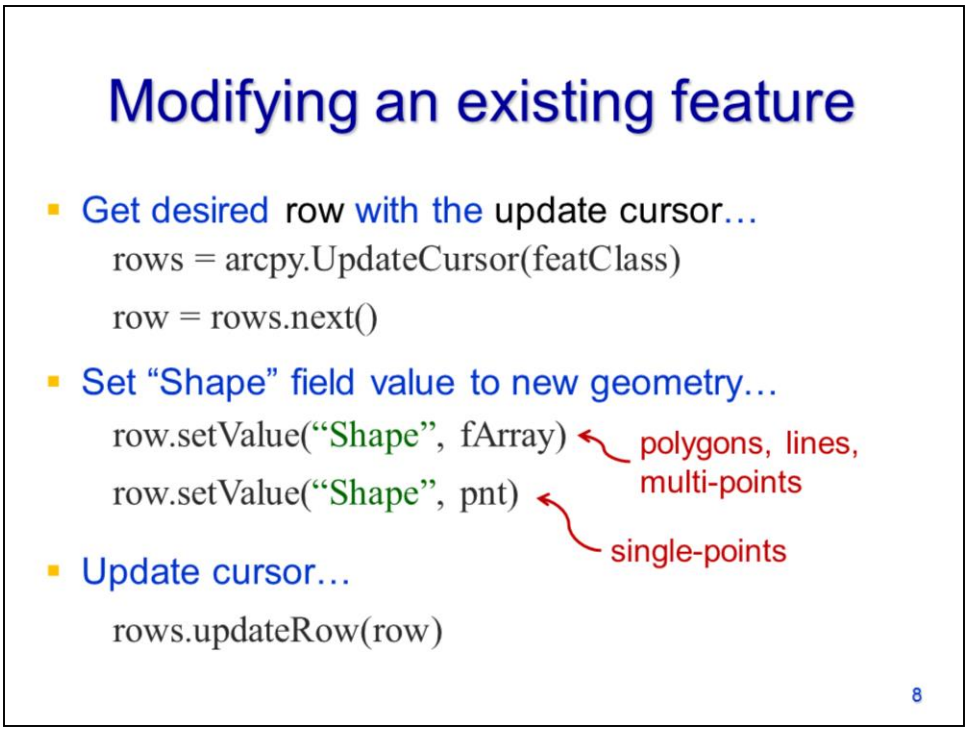


To create a new feature,

First create a new row object using the **newRow** method of the InsertCursor. The insert cursor must be create using an existing feature class.

The geometry of the row object can be specified by using the setValue method on the "Shape" field. The "value" for the "Shape" field is the feature array for polygons, lines, and multi-part points. The "Shape" field value is a point object for single-part point features.

After the setValue method has been used to specify the feature geometry as well as any field values, then the new row object can be added to the cursor by using the **insertRow** method.

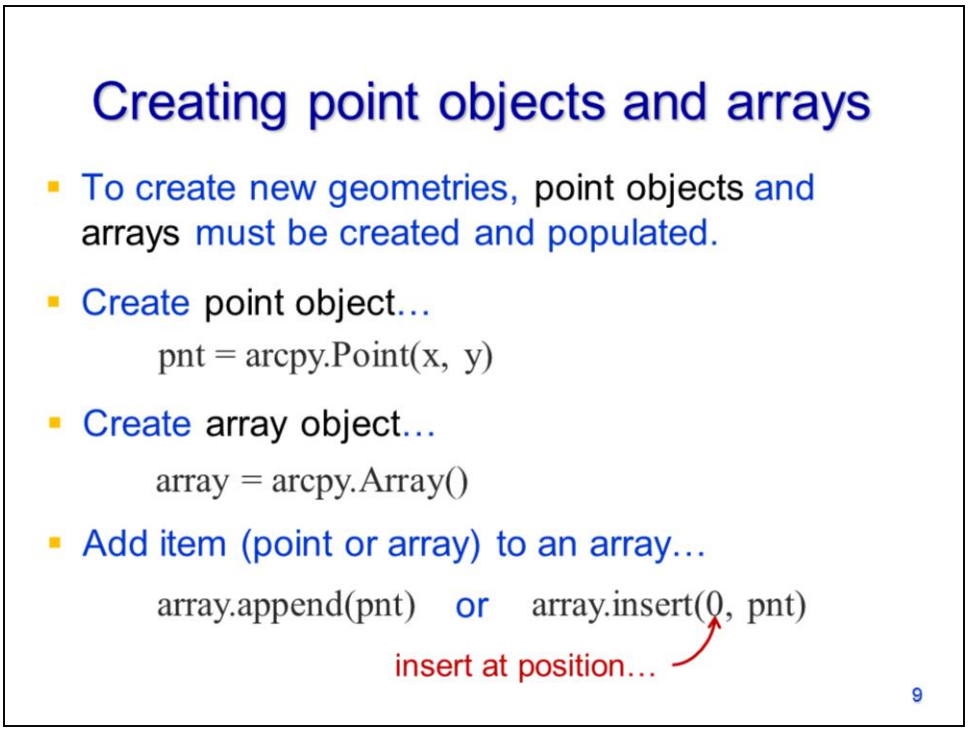


To modify the shape of an existing feature,

Use the **UpdateCursor** to retrieve the row object corresponding to the feature that will be modified.

Use the row object's **setValue** method to specify the geometry for the "Shape" field.

Use the **updateRow** method to update the cursor for the modified feature.



Before we can set the shape of a new or modified feature, we must first create the geometry for that feature.

To create a point object, use arcpy's **Point** method.

To create an array, use arcpy's **Array** method.

To add either point objects or arrays to an array, use either the **append** or **insert** methods.

Creating new single-point features

```
arcpy.CreateFeatureclass_management(
   r"C:\NRE_5585\Results", "newPoints.shp", "POINT")
rows = arcpy.InsertCursor(r"C:\NRE_5585\Results\newPoints.shp")
newRow = rows.newRow()
for x, y in xyLst:
   pnt = arcpy.Point(x, y)
   newRow.setValue("Shape", pnt)
   rows.insertRow(newRow)
del rows, newRow
```

Example list

xyLst = [[50,100],[20,150],[80,60]]

10

This slide shows an example of a script that creates a single-part point feature dataset. The first step in creating the dataset is to use arcpy's **CreateFeatureclass** tool. Note that this tool requires you to specify the workspace and the basename of the output file as separate parameters.

In the next step, an insert cursor is create for the new empty feature class. Then a new row is created for the cursor.

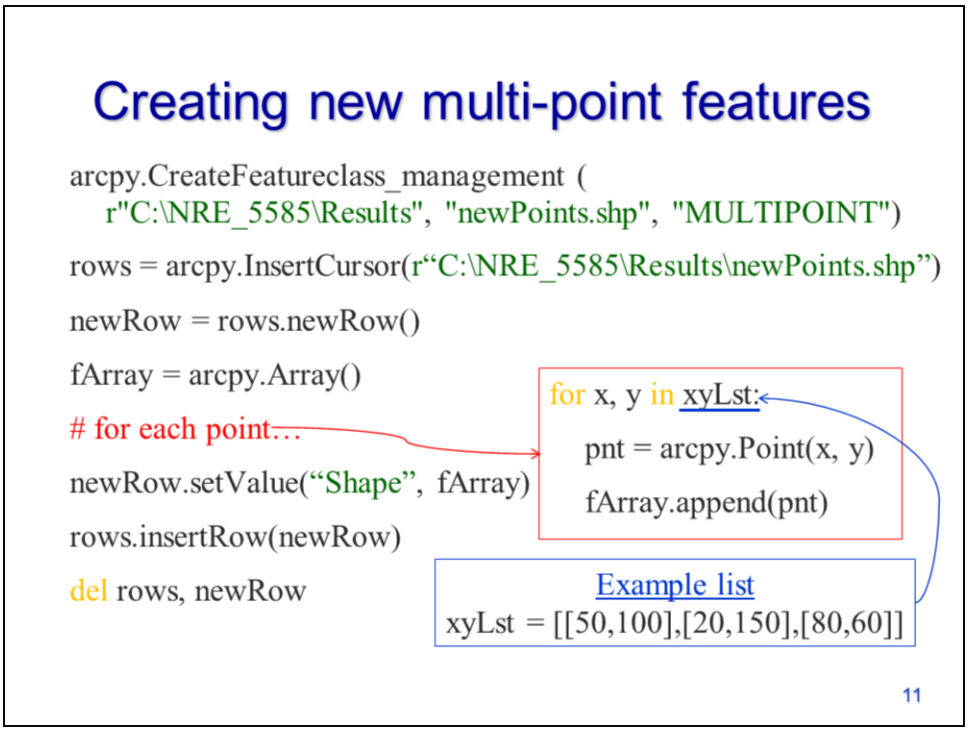
A for loop is used to iterate through a list containing the X and Y coordinates for each point feature that is to be created.

Arcpy's Point method is used to create a point object with the coordinates extracted from the xyLst.

The point object is set as the value for the "Shape" field.

The new row object is inserted into the cursor.

The cursor and row objects are deleted after the loop is complete.

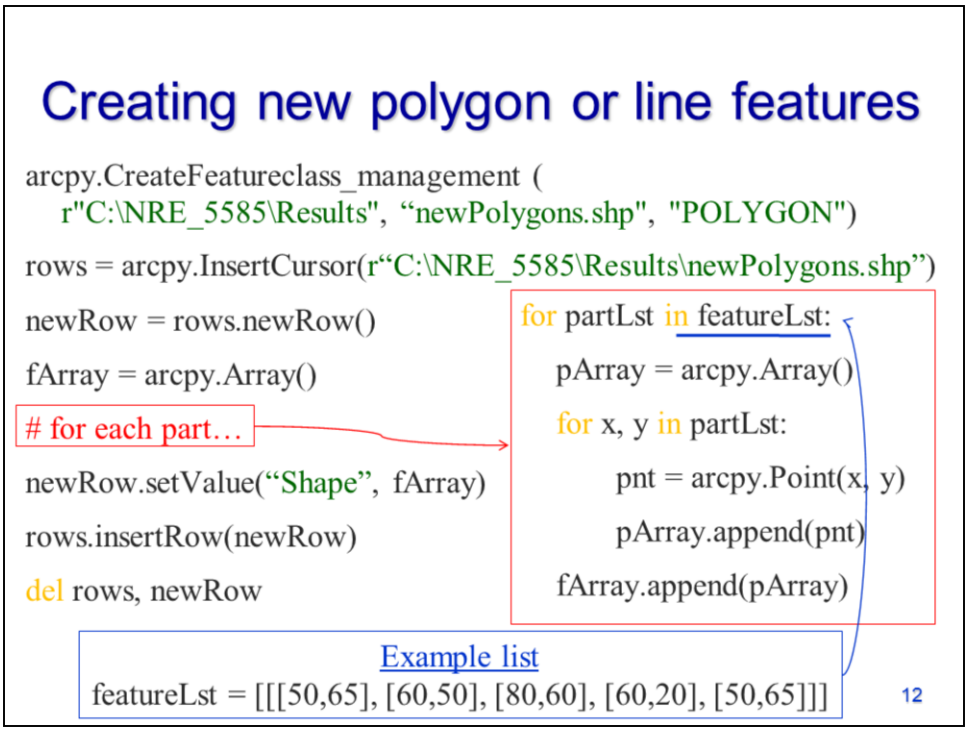


This script example shows how to create a new multi-part point feature class. A new "MULTIPOINT" feature class is created and the insert cursor is used to create a new row object.

In this case, a feature array is created to store the multiple points that may be associated with a given feature.

For each feature, a loop is used to iterate through the list of coordinates. In this loop, the point object is appended to the feature array.

The feature array is specified as the value for the "Shape" field and the new row is inserted into the feature class. Note that this script will only add a single feature to the feature class.

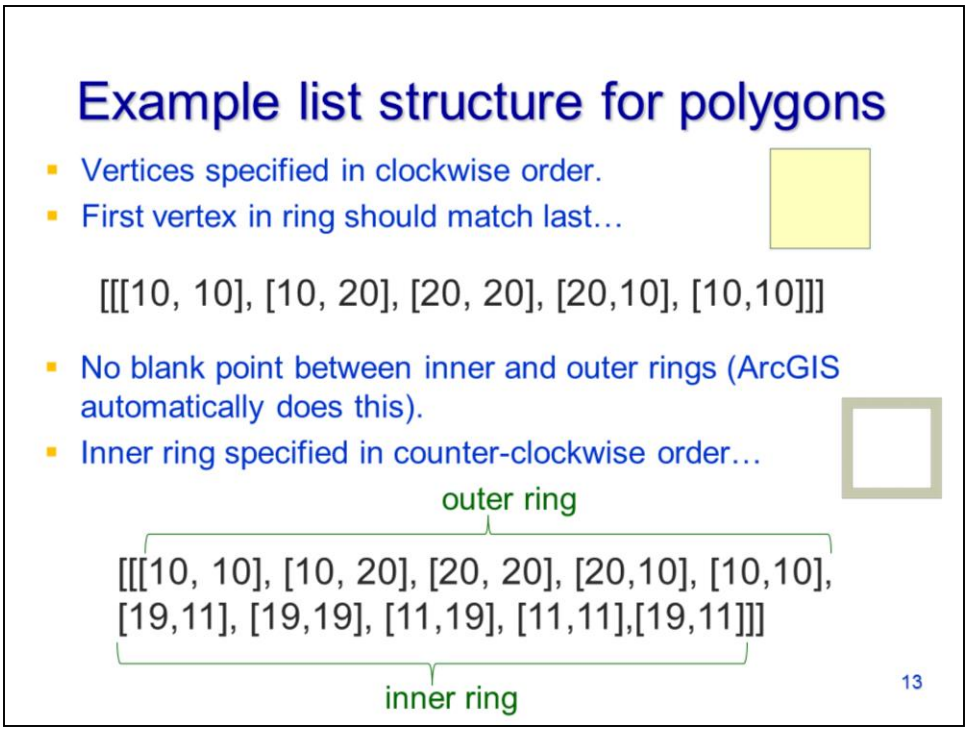


The script shown in this example can create either a polygon or a line feature class. The script in this case creates a polygon feature class – for polygon features, the "POLYGON" feature type must be specified in the **CreateFeatureclass** tool; otherwise a line feature class will be created.

As in the previous script example, the feature array is created just after creating the new row object – it should only be created once for a given features.

A for loop is used to iterate through each part in the feature. The example list shows the structure needed for a polygon feature – which we'll examine closer on the next slide. For each part in the feature, a new part array is created. A nested loop then creates a point object for each point and appends the point object to the part array. After all the points have been processed for a given part, the part array is appended to the feature array – note this is done inside the "part loop" but outside of the "point loop".

After the feature array has been fully populated, the feature array is set as the "Shape" field value and the row object is inserted into the cursor. Note that this example script can create only a single polygon feature.



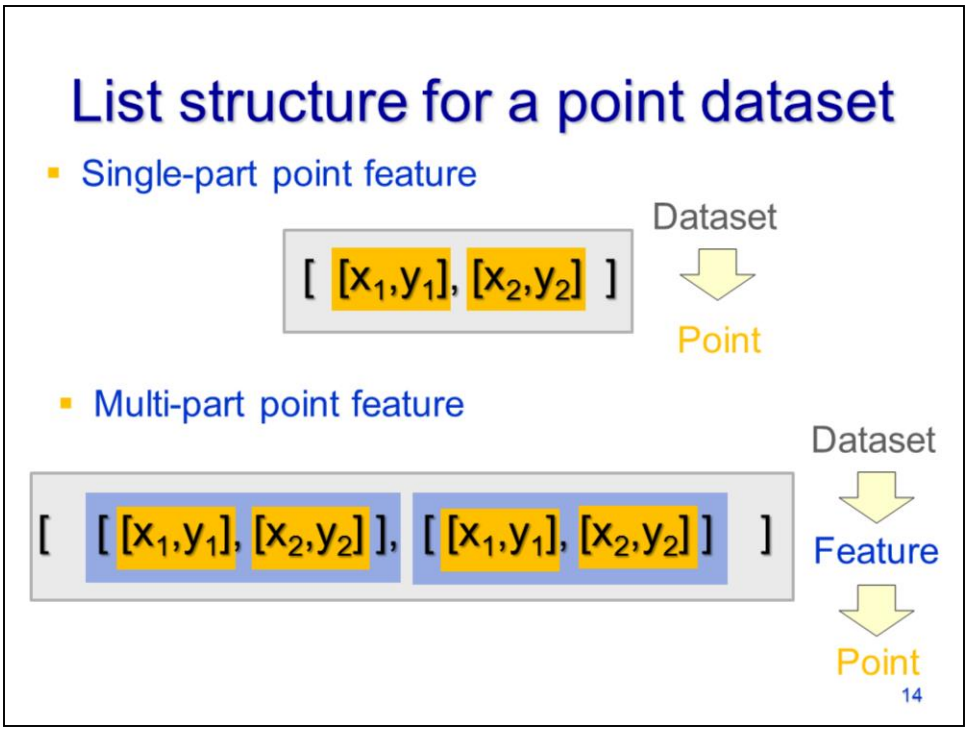
Let's take a closer look at the list structure needed to contain the vertices for a single polygon feature.

The list for a single polygon or line feature will have 3 levels – the top level contains all feature data, the 2nd level contains data for a single part, and the bottom level contains data for a single vertex. At the 2nd level, the vertices the outer ring of given polygon part are specified in a clockwise order with the 1st and last point being the same.

If the polygon has an inner ring, then it will be added after the outer ring in the part list. Do not add a null point between the outer and inner rings – ArcGIS will do that itself. The vertices for the inner ring are arranged in a counter-clockwise order.

This is an example list for a single polygon feature that contains both an outer and an inner ring.

The list that contains the vertices for a single polygon feature



Feature information usually stored in a list before constructing the point objects and arrays required build a feature's geometry. Here, we'll look at the hierarchical list structures needed to store the vertices for each type of feature class.

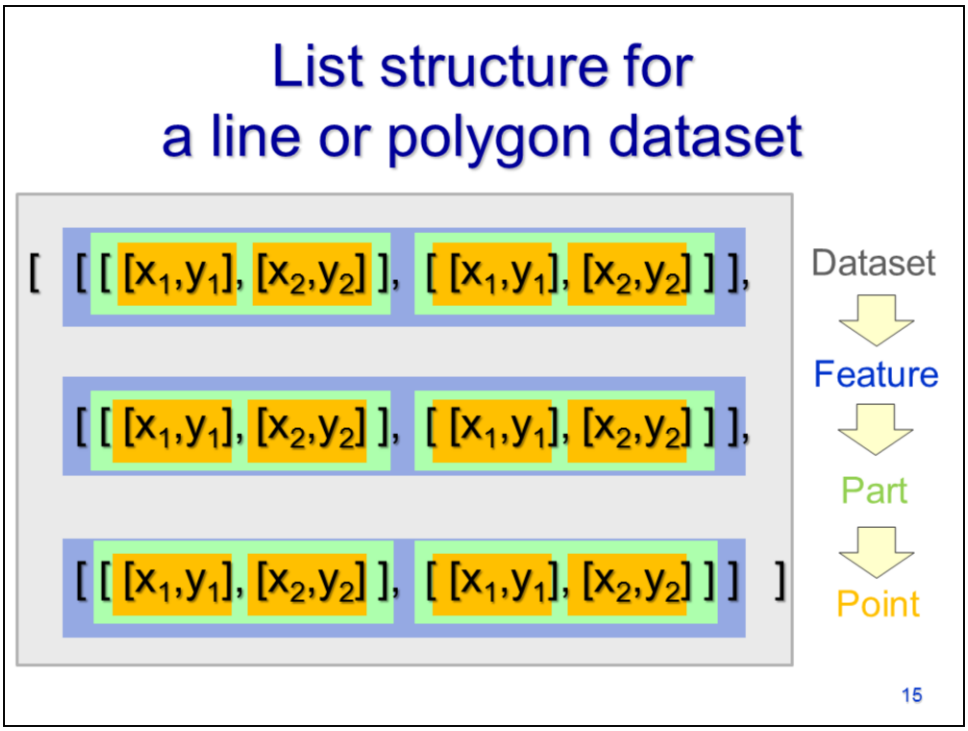
The single-part point feature class has the simplest list structure with only 2 levels in its hierarchy. The top level is the dataset list which contains one or more features which consist of a single point list. Each point list contains an X and Y coordinate

The multi-part point feature class requires 3 levels:

The top level is the dataset list which contains one or more feature lists

The feature list contains one or more point lists.

The point list contains an X and Y coordinate.



The list structure for lines and polygons has 4 levels and care must be taken to close the brackets for lists at each level.

The top level is the dataset list which contains one or more feature lists,

The feature lists contain one or more part lists,

The part lists contain multiple point lists,

The point lists contain an X and Y coordinate.

The dataset list should start with 4 left brackets and end with 4 right brackets.