



# Lesson 4: Enumeration & Cursor Objects

Basic Programming in ArcGIS with Python  
Workshop

RS/GIS Lab, Utah State University

With Material from ESRI

# Learning Objectives:

- To understand uses of enumeration objects
- To be able to write a simple script using enumeration
- To understand uses of cursor objects
- To be able to write a simple script using cursors
- To understand some basic debugging and coding guidelines
- To review how scripts are added to ArcToolbox

# Lesson 4a: Enumeration Methods & Objects

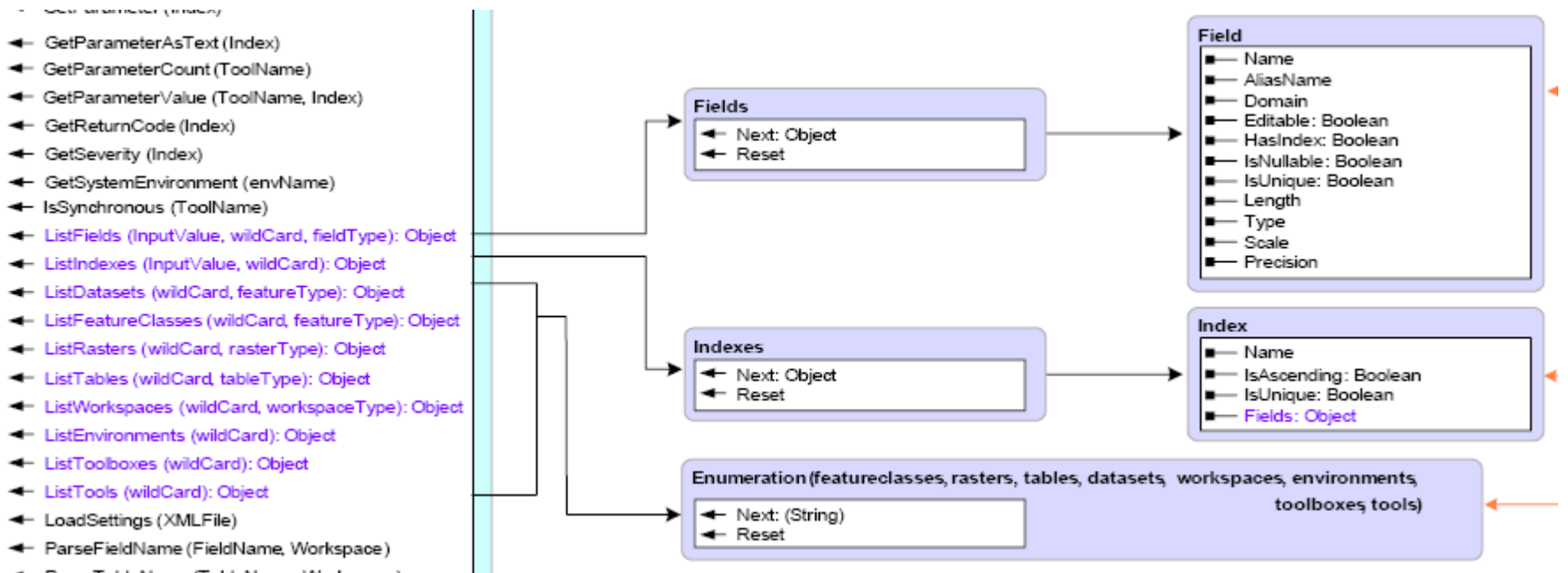
- Enumeration methods:

- “Sit on” the geoprocessor object
- Each returns an object
- Each object holds a list of data
- Object returned depends on the method called

- ← ListFields (InputValue, wildCard, fieldType): Object
- ← ListIndexes (InputValue, wildCard): Object
- ← ListDatasets (wildCard, featureType): Object
- ← ListFeatureClasses (wildCard, featureType): Object
- ← ListRasters (wildCard, rasterType): Object
- ← ListTables (wildCard, tableType): Object
- ← ListWorkspaces (wildCard, workspaceType): Object
- ← ListEnvironments (wildCard): Object
- ← ListToolboxes (wildCard): Object
- ← ListTools (wildCard): Object

## • Enumeration objects:

- A list of data (any type) without a known count
- **ListFields** returns a Fields object
- **ListIndex** returns a Indexes object
- All other Enumeration methods return an Enumeration object



- The Enumeration object:
  - Has two methods: **Next** and **Reset**

Enumeration (featureclasses, rasters, tables, datasets, workspaces, environments, toolboxes tools)

← Next: (String)  
← Reset

- **Next** returns the name of an item in the enumeration object (list)
- **Reset** returns to the item as the start of the list

- An Enumeration object example: ListRasters(wildcard, rasterType)

### Syntax

`object.ListRasters({wildCard} As String, {RasterType} As String) as Object`

{ } = optional

Part	Description
<i>object</i>	The instantiated geoprocessing object.
<i>wildCard</i>	Optional. Combination of * and characters that will help limit the results. The asterisk is the same as saying ALL. If no wildcard is specified, all rasters in the workspace will be returned.
<i>RasterType</i>	Specifies the raster format.

```
# workspace must be set!!
```

```
gp.workspace = ("C:/john/Lesson4")
```

```
# get a list of grids beginning with "elev"
```

```
rasList = gp.ListRasters("elev*", "GRID")
```

```
# loop through list and print items in list
```

```
ras = rasList.next()
```








```
while ras:
```

```
    print ras
```

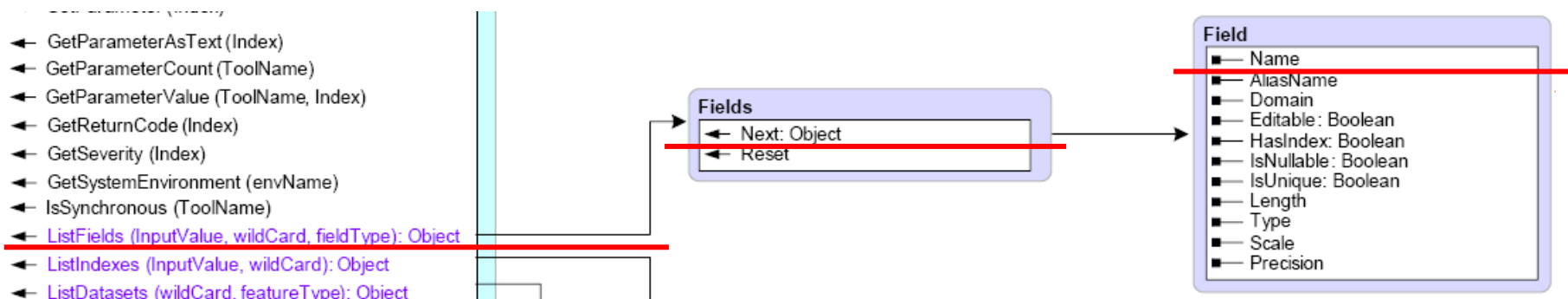
```
    ras = rasList.next()
```

```
# reset ras to top of list
```

```
ras = rasList.reset()
```

Name	
	elev30m
	elev30m10
	elev30m100
	flowline.shp
	landcover
	Lesson4.mxd
	obs.shp

- The ListFields (InputValue, wildCard, fieldType) example:
  - ListFields returns the Fields object
  - Next returns the Field object
  - The Field object returns the Name



```
fldList = gp.ListFields("flowline", "*", "String")
```

```
fld = fldList.Next()
```

```
While fld <> None:
```

```
    Print fld.Name
```

```
    fld = fldList.Next()
```

## Lesson 4a ACTIVITY: Complete the Code

- As a class, using Desktop Help and the model diagram complete the code:
  - List all the .dbf files in a workspace that begin with “1990”  
`90dbfList = gp.ListTables( "1990*", "dBASE" )`
  - List all the CAD files in a workspace that begin with “rd”  
`cmpCADList = gp.ListDatasets( "rd*", "CAD" )`
  - List all the tools from the conversion toolbox  
`conToolList = gp.ListTools( "*_conversion" )`

# Assignment 4a: Write a script using enumerations

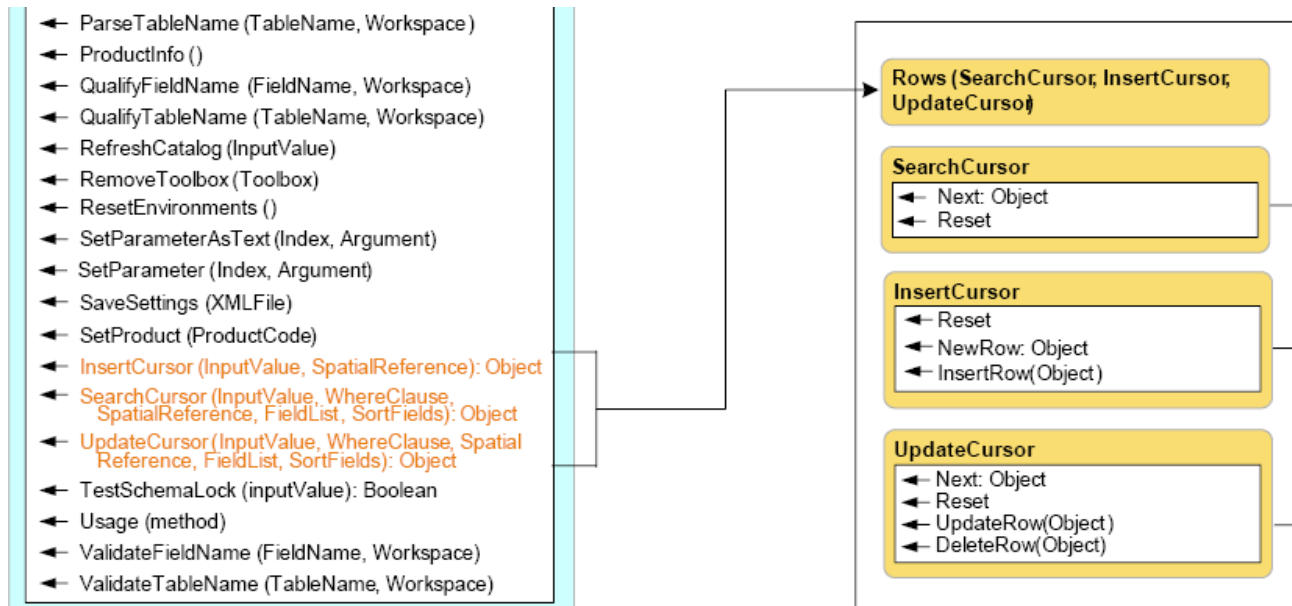
- Write a script that does the following:
  - Makes an enumeration (list) of all the featureclasses in the Lesson4 workspace that begin with “nests.”
  - Using a While loop, print the names of all the feature classes.
  - Write your script from scratch (or copy what you need from a previous lesson to get started).
  - Be sure to comment your code.

# Lesson 4b: Cursor Methods & Objects

- Cursor methods access attributes/values in table cells:
  - “Sit on” the geoprocessor object
  - Each returns an object (an enumeration/list of row objects)
  - Cursors work with rows in a table or feature class attribute table
    - Add and delete rows
    - Read values in rows
    - Modify values in rows

## Cursor Methods return Cursor Objects:

- **InsertCursor** returns the InsertCursor object
  - Used to insert new rows
- **SearchCursor** returns the SearchCursor object
  - Used to read values in a row
- **UpdateCursor** returns the UpdateCursor object
  - Used to make changes to values in rows, and delete rows



## SearchCursor object:



- **Next** returns an individual row object
- To find out the value of a field for individual row (**row**), use:

- **Fieldname (a property)**

```
Print row.GNIS_Name
```

Or

- **GetValue (fieldName) (a method)**

```
Print row.GetValue("GNIS_Name")
```

- **Reset** takes you back to the start of the cursor

## SearchCursor example:

```
gp.workspace = ("C:/john/Lesson4")
cur = gp.SearchCursor ("nests1990")
row = cur.Next()
while row <> None:
    print row.nestsiteid
    print row.condition
    row = cur.Next()
del cur
```

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

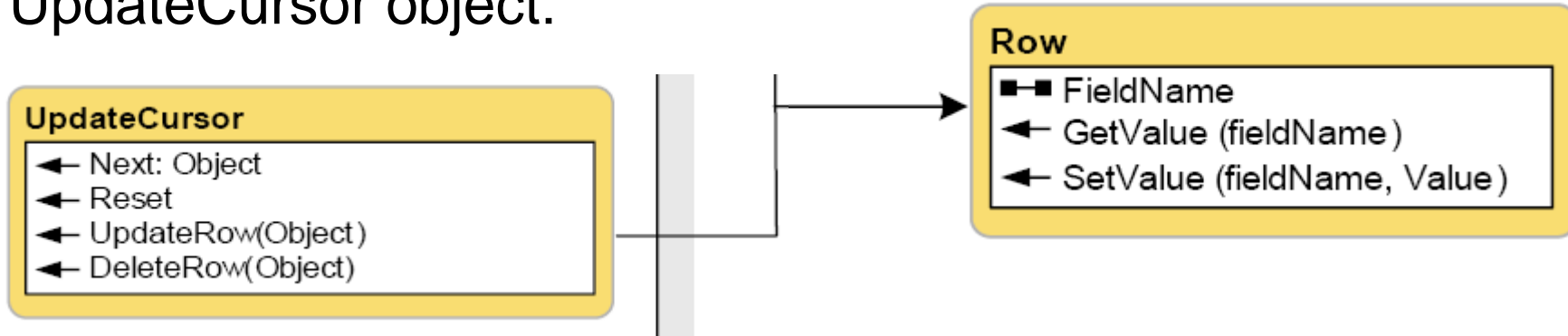
0	Point	0	1	good
---	-------	---	---	------

### Notes:

- Because `row` is not a string, but an object, we use `None` for the conditional
- We could also have use the `GetValue` method
 

```
print row.GetValue("netsiteid")
print row.GetValue("condition")
```
- We delete the cursor object at end of the while construct

## UpdateCursor object:



- **Next** returns an individual row object
- To find out the value of a field for individual row (**row**), use:
  - **Fieldname** Reads/writes a value from/to a field
  - **GetValue (fieldName)** Reads a value from a field
  - **SetValue (fieldName, value)** Writes a value to a field
- **UpdateRow** passes modified row into the table/feature class
- **DeleteRow** deletes a row

## UpdateCursor example:

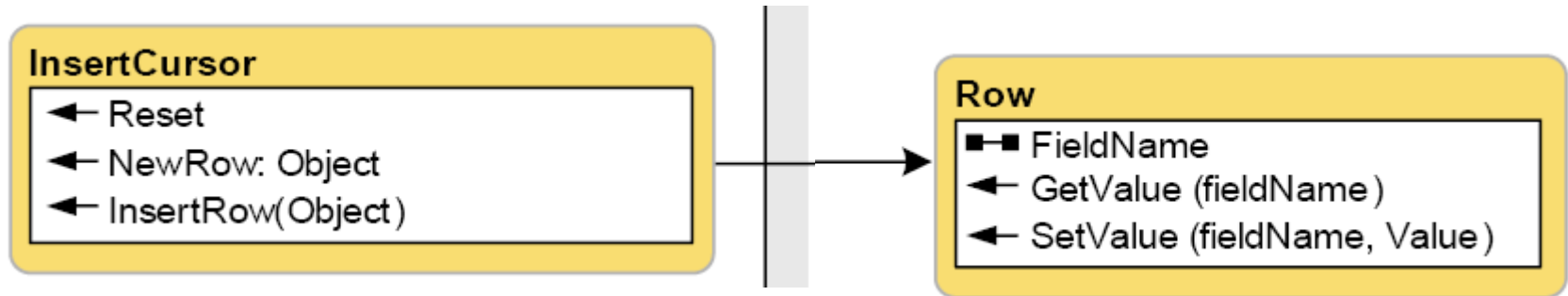
```
gp.workspace = ("C:/john/Lesson4")
cur = gp.UpdateCursor ("nests1990", "nestsiteid > 5")
row = cur.Next()
while row:
    row.condition = string.replace(row.condition, "poor", "good")
    cur.UpdateRow(row)
    row = cur.Next()
del cur
```

### Notes:

- This code: Begins with the first row that has a `nestsiteid > 5`, replaces condition that is "poor" with "good" for each row, finally updates the row change.
- The `string.replace` method requires the string module be imported (not shown in code).
- We delete the cursor object at end of the while statement (to free the variable & memory).

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

## InsertCursor object:



- **NewRow** creates a new row object
  - **FieldName** Reads/writes a value from/to a field
  - **GetValue (fieldName)** Reads a value from a field
  - **SetValue (fieldName, Value)** Writes a value to a field
- **InsertRow** inserts the row into the table or feature class
  - Automatically fills the OID field

## InsertCursor example:

```
gp.workspace = ("C:/john/Lesson4")
cur = gp.InsertCursor("nestdata1990.dbf")
row = cur.NewRow()
row.nestsiteid = 11
row.height = 90
row.count = 3
cur.InsertRow(row)
```

### Notes:

- This code: Inserts a new row to the table "nestdata1990.dbf" and fills the nestsiteid, height, and count fields.
- Once all the values are populated, the row is inserted into the table.
- NOTE! If this were a feature class, the *shape* field would have to be added as well.

	NESTSITEID	HEIGHT	COUNT
▶	1	79	3
	2	67	2
	3	99	4
	4	56	5
	5	56	3
	6	54	2
	7	56	1
	8	89	3
	9	45	4
	10	34	2

# Assignment 4b: Write a script using cursor methods and objects.

- Write a script that does the following:
  - Uses a SearchCursor method to print “nestsiteid” field from all the rows in the nests1990.shp file.
  - When you get that working, use the Select\_analysis tool (method) to make a separate shapefile for each nestsite.
  - When you get that working, use the ViewShed\_sa tool (method) to create a viewshed for each nestsite.
  - Think about: How could you use the code you wrote in assignment 4a with this code to create separate viewsheds for every nest site for all shapefiles.

# Debugging and Coding Tips

- Test code in PythonWin, not ArcToolbox
- Check Python syntax in PythonWin with Check button
- Don't forget arguments (if you have them)
- Check error message in Interactive Window
- Try to understand the code
- Narrow errors to a block or line of code (using commenting)

# Coding Guidelines

- Use consistent variable naming conventions (e.g. myIntegerVar = 1)
- Script names should follow similar naming conventions (e.g. myScript.py)
- Use indentation to show structure (Python enforces this).
- Use arguments in a script
- If a script has arguments, you should have code that checks the inputs have been given by the user (when attaching script in ArcToolbox we can provide a default input).
- Use comments (over commenting your own scripts, esp. when you're beginning is not a bad idea).
- Include a header to each script telling the script's name, a little about how it works, it's requirements, who wrote it and when it was written.
- Other guidelines...?