Using ArcObjects in Python

Mark Cederholm
UniSource Energy Services

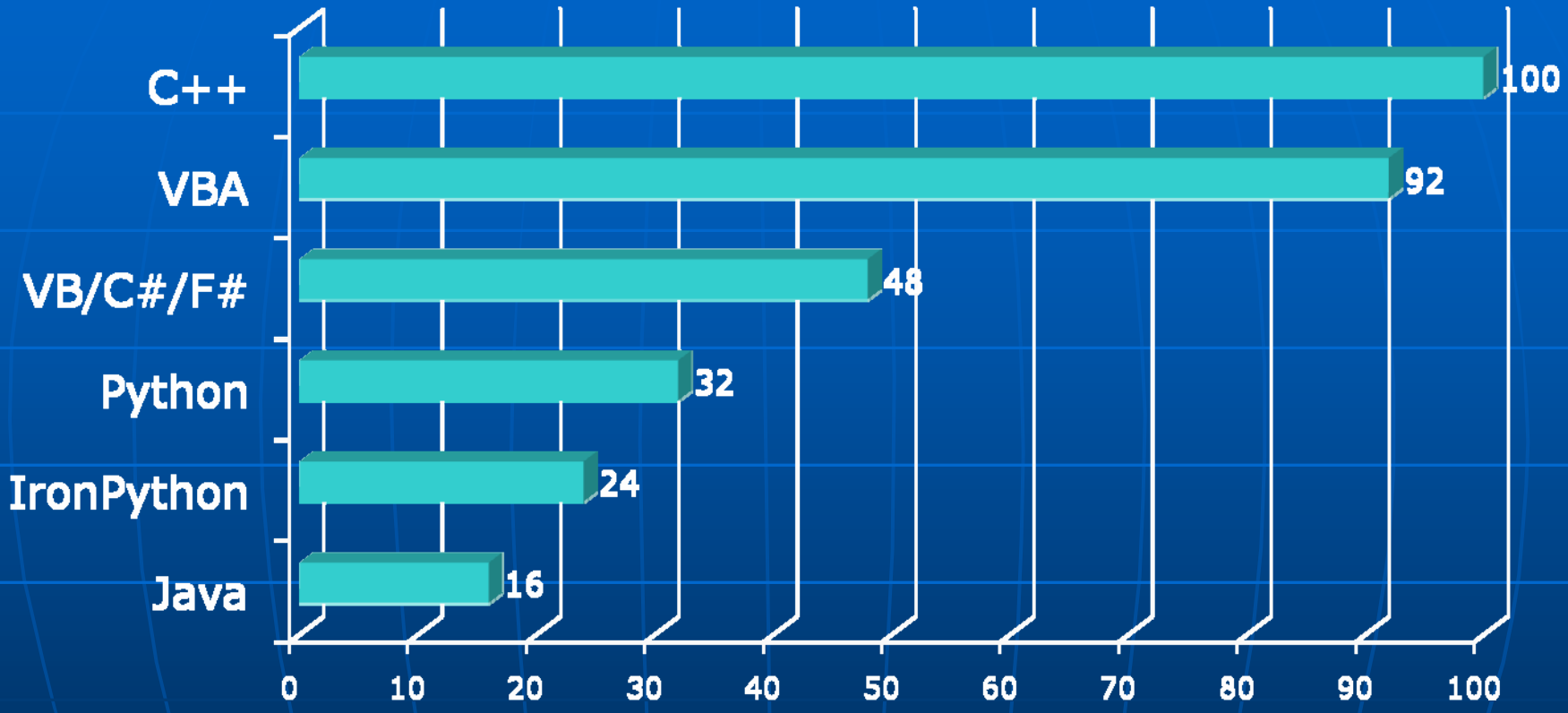
Why Python?

- ArcGIS VBA support ends after 10.0
- At 10.0, ArcMap and ArcCatalog include an integrated Python shell
- Python scripting objects provided by ESRI
- IDLE is a decent development and debugging environment
- Python scripts can use ArcObjects!

Geoprocessing objects

- Ready-to-use geoprocessing objects are available for Python through arcgisscripting (9.3) and arcpy (10.0)
- At 9.3: additional functionality includes data access objects such as cursors
- At 10.0: additional functionality includes some map document automation
- Nonetheless, a great deal of functionality is only available through ArcObjects

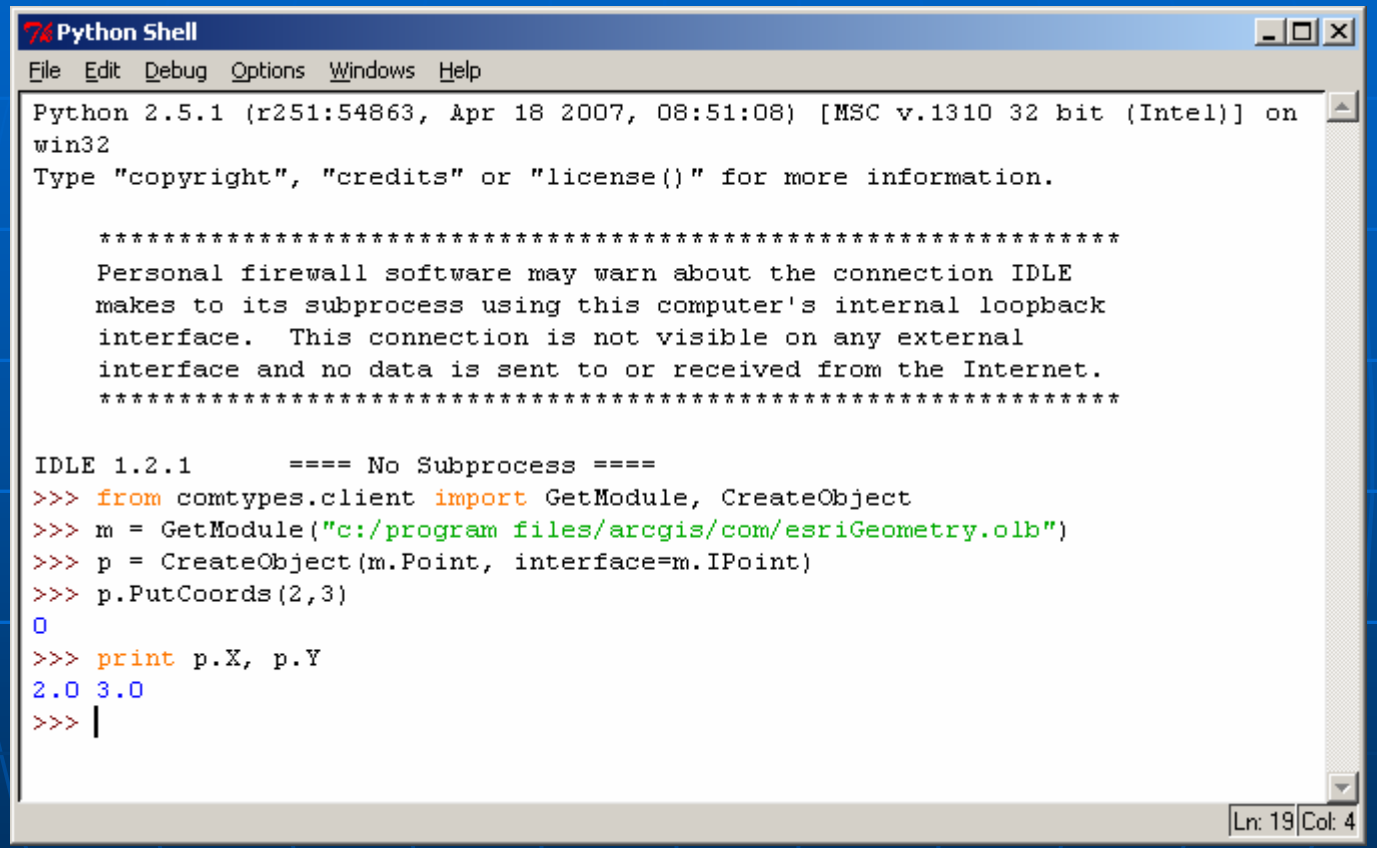
COM interop: relative speed



Benchmark = 500+K ShapeCopy operations

(ArcGIS 9.3.1 with VS2008)

Demo: Standalone scripting



The comtypes package

Available for download at:

http://sourceforge.net/projects/comtypes/

Download and run installer; or else download zip file, unzip, and enter this line at the command prompt:

```
python setup.py install
```

See also this link for documentation:

http://starship.python.net/crew/theller/comtypes/

Loading and importing modules

```
def GetLibPath():
    ##return "C:/Program Files/ArcGIS/com/"
    import _winreg
    keyESRI = _winreg.OpenKey(_winreg.HKEY_LOCAL_MACHINE, \
                              "SOFTWARE\\ESRI\\ArcGIS")
    return _winreg.QueryValueEx(keyESRI, "InstallDir")[0] + "com\\"

def GetModule(sModuleName):
    import comtypes
    from comtypes.client import GetModule
    sLibPath = GetLibPath()
    GetModule(sLibPath + sModuleName)
```

```
GetModule("esriGeometry.olb")
import comtypes.gen.esriGeometry as esriGeometry
[or]
from comtypes.gen.esriGeometry import Point, IPoint
[import * is not recommended]
```

Creating and casting objects

```
def NewObj(MyClass, MyInterface):
    from comtypes.client import CreateObject
    try:
        ptr = CreateObject(MyClass, interface=MyInterface)
        return ptr
    except:
        return None

def CType(obj, interface):
    try:
        newobj = obj.QueryInterface(interface)
        return newobj
    except:
        return None

def CLSID(MyClass):
    return str(MyClass._reg_clsid_)
```

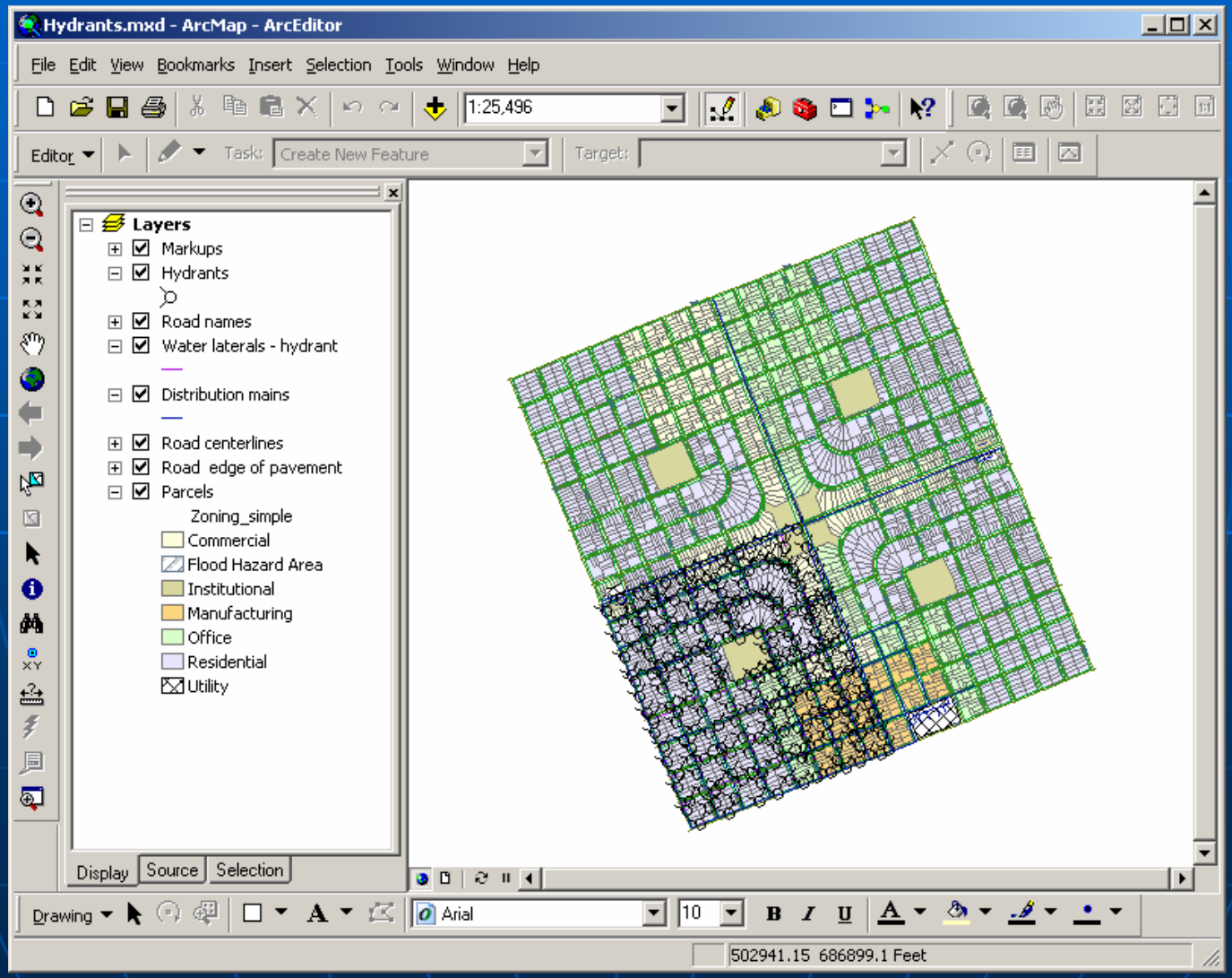
Standalone licensing

```
pInit = NewObj(esriSystem.AoInitialize, \
               esriSystem.IAoInitialize)
eProduct = esriSystem.esriLicenseProductCodeArcEditor
licenseStatus = pInit.IsProductCodeAvailable(eProduct)
if licenseStatus == esriSystem.esriLicenseAvailable:
    licenseStatus = pInit.Initialize(eProduct)
return (licenseStatus == esriSystem.esriLicenseCheckedOut)
```

TIP: Use the geoprocessing object instead

```
import arcgisscripting
gp = arcgisscripting.create(9.3)
gp.setproduct("ArcEditor")
```

Demo: Manipulating an existing ArcMap or ArcCatalog session



Retrieving an existing session from outside the application boundary

```
if not (app == "ArcMap" or app == "ArcCatalog"):
    return None
pAppROT = NewObj(esriFramework.AppROT, esriFramework.IAppROT)
iCount = pAppROT.Count
if iCount == 0:
    return None
for i in range(iCount):
    pApp = pAppROT.Item(i)
    if app == "ArcCatalog":
        if CType(pApp, esriCatalogUI.IGxApplication):
            return pApp
        continue
    if CType(pApp, esriArcMapUI.IMxApplication):
        return pApp
return None
```

Getting a selected feature

```
pApp = GetApp()
. . .
pDoc = pApp.Document
pMxDoc = CType(pDoc, esriArcMapUI.IMxDocument)
pMap = pMxDoc.FocusMap
pFeatSel = pMap.FeatureSelection
pEnumFeat = CType(pFeatSel, esriGeoDatabase.IEnumFeature)
pEnumFeat.Reset()
pFeat = pEnumFeat.Next()
if not pFeat:
    print "No selection found."
    return
pShape = pFeat.ShapeCopy
eType = pShape.GeometryType
if eType == esriGeometry.esriGeometryPoint:
    print "Geometry type = Point"
. . .
```

Creating session objects with IObjectFactory

If manipulating a session from outside the application boundary, use IObjectFactory to create new session objects:

```
pApp = GetApp()
pFact = CType(pApp, esriFramework.IObjectFactory)
pUnk = pFact.Create(CLSID(esriCarto.TextElement))
pTextElement = CType(pUnk, esriCarto.ITextElement)
```

TIP: At 10.0, you can run a script within the session's Python shell and create objects normally; use AppRef to get the app handle

```
pApp = NewObj(esriFramework.AppRef, esriFramework.IApplication)
```

UIDs and Enumerations

```
pApp = GetApp()
. . .
pID = NewObj(esriSystem.UID, esriSystem.IUID)
pID.Value = CLSID(esriEditor.Editor)
pExt = pApp.FindExtensionByCLSID(pID)
pEditor = CType(pExt, esriEditor.IEditor)
if pEditor.EditState == esriEditor.esriStateEditing:
    pWS = pEditor.EditWorkspace
    pDS = CType(pWS, esriGeoDatabase.IDataset)
    print "Workspace name: " + pDS.BrowseName
    print "Workspace category: " + pDS.Category
```

Multiple Return Values

```
iEdgeEID, bReverse, oWeight = pForwardStar.QueryAdjacentEdge(i)
```

Nothing, IsNull, and None

- Supply None as an argument representing Nothing:

```
iOpt = esriCarto.esriViewGraphics + \
       esriCarto.esriViewGraphicSelection
pActiveView.PartialRefresh(iOpt, None, None)
```

- Use boolean testing to check for a null pointer, and is None to check for a null DB value:

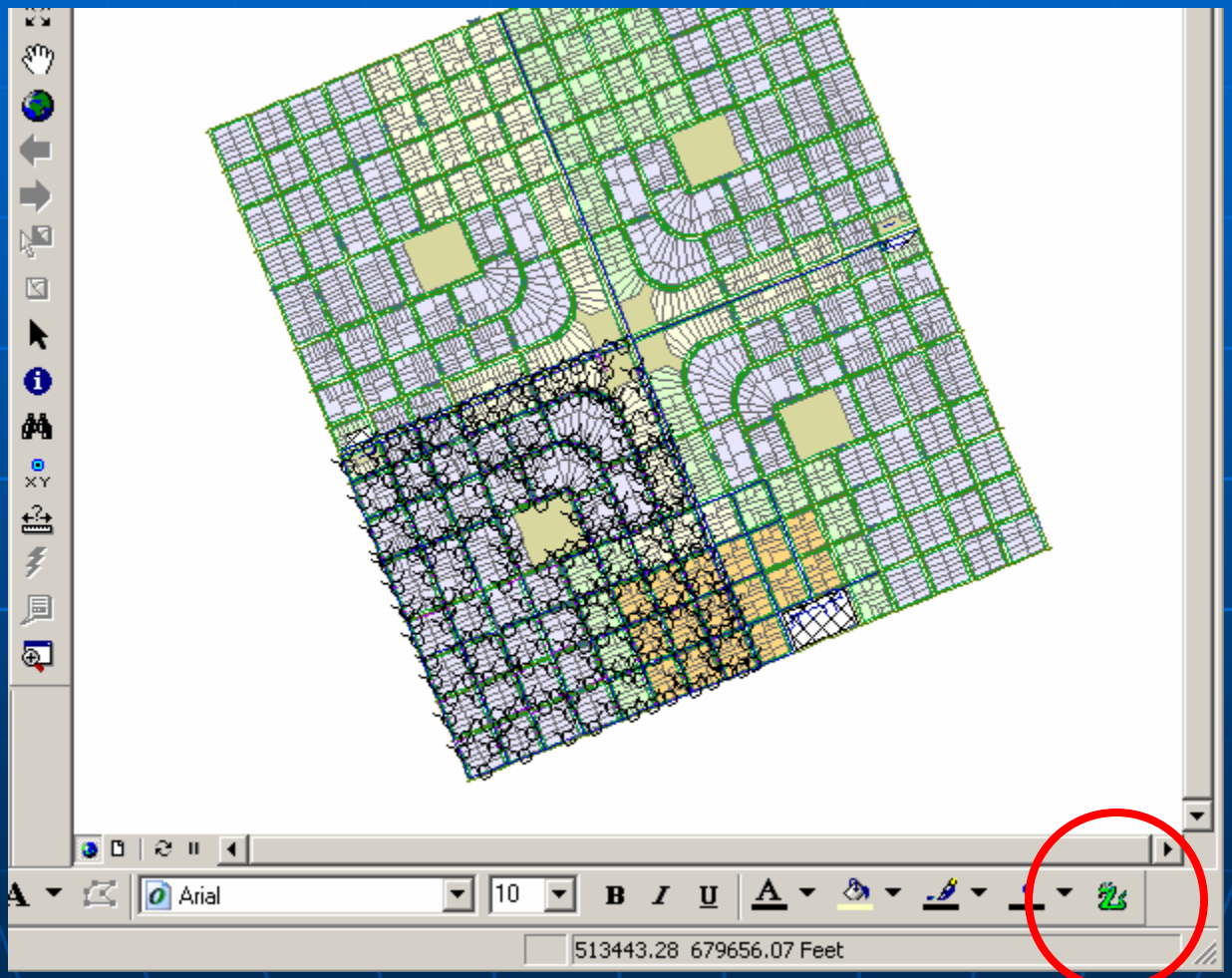
```
pCursor = pTab.Search(pQF, True)
pRow = pCursor.NextRow()
if not pRow:
    print "Query returned no rows"
    return
Val = pRow.Value(pTab.FindField(sFieldName))
if Val is None:
    print "Null value"
```

WriteOnly and indexed properties

```
pNewField = NewObj(esriGeoDatabase.Field, \
                   esriGeoDatabase.IField)
pFieldEdit = CType(pNewField, esriGeoDatabase.IFieldEdit)
pFieldEdit._Name = "LUMBERJACK"
pFieldEdit._Type = esriGeoDatabase.esriFieldTypeString
pFieldEdit._Length = 50
pFieldsEdit._Field[1] = pNewField
pOutTable = pFWS.CreateTable(sTableName, pOutFields, \
                             None, None, "")
iField = pOutTable.FindField("LUMBERJACK")
print "'LUMBERJACK' field index = ", iField
pRow = pOutTable.CreateRow()
pRow.Value[iField] = "I sleep all night and I work all day"
pRow.Store()
```

TIP: Use geoprocessing tools to create tables and add fields

Demo: Extending ArcGIS Desktop



Creating a COM object

1. Create an IDL file defining the object and its interfaces
2. Compile with the MIDL compiler (part of the Windows SDK download) to produce a TLB file:
   ```
   midl DemoTool.idl
   ```
3. Implement the class and category registration in a Python module
4. Register the com object:
   ```
   python DemoTool.py –regserver
   ```

WARNING: The file/module name in step 4 is case sensitive!

Some final tips:

- When in doubt, check the wrapper code:
  Python25/Lib/site-packages/comtypes/gen
- Avoid intensive use of fine-grained ArcObjects in Python
- For best performance, use C++ to create coarse-grained COM objects
- Use geoprocessing objects and tools to simplify supported tasks – watch for performance, though
- Read the desktop help to check out available functionality in arcgisscripting (and arcpy at 10.0)

Questions?

- Mark Cederholm
  mcederholm@uesaz.com
- This presentation and sample code may be downloaded at:

http://www.pierssen.com/arcgis/misc.htm