would like to thank our sponsors of this event:
Python Programming for GIS

MAC URISA
Spring 2014
Introduction

- John Reiser  
  Rowan University Geospatial Research Lab  
  reiser@rowan.edu
- New Jersey Geographer blog  
  jreiser@njgeo.org
- Twitter - @johnjreiser
- GitHub - johnreiser
- Working in Python since 2007
Today's Agenda

• Now until 10:30 –
  – Intro to Python & why you should be using it
  – Basics of Python language
  – Learning resources available to you

• 10:45 until Noon –
  – Working with Python and ArcGIS
  – Moving from ModelBuilder to Python
  – Extending GIS beyond what’s in ArcToolbox

• After Lunch –
  – Review of some Python projects
  – Question and Answer
Why program?

- Desktop GIS includes a considerable array of tools and components to help you perform analysis and manage your data.
- The ability to chain together tools in series, or adapt the tools to satisfy your specific needs can improve your efficiency or extend your capabilities.
- If you’re performing the same tasks repeatedly, or you find yourself performing manual fixes, you likely can reduce your effort though custom code.
Why Python?
Brief History

• Initially created by Guido van Rossum (BDFL).
• 2.0 released in 2000. 3.0 released in 2008.
• Designed from the start to be extensible.
• Community driven development.
• Adopted by the geospatial software community.
  – ArcGIS
  – QGIS
  – PostGIS
  – Countless libraries to work with spatial data
Why program in Python?

- Extensible nature of the language gives you the flexibility to change it as you see fit.
- Conversely, others may have already worked on a similar problem and released their solutions on the web for you to incorporate.
- Support for various OSs and GISs means that your tools can be easily migrated to other platforms and maintained as GIS evolves.
- A strong community supports the software and its continued development.
ArcView GIS/Avenue Developer's Guide with 3.5 Disk [Paperback]
by Amir H. Razavi
4 stars (4 customer reviews)

Available from these sellers.

9 new from $15.97  19 used from $0.01

FREE TWO-DAY SHIPPING FOR COLLEGE STUDENTS

Formats | Amazon Price | New from | Used from
--- | --- | --- | ---
Paperback | -- | $14.97 | $3.47
Paperback, January 1999 | -- | $15.97 | $0.01

There is a newer edition of this item:
ArcView GIS Developer's Guide
20 used & new from $7.43
See a problem with this suggestion? Let us know
From the ArcGIS perspective

• You might have already used Python within ArcGIS.
• Python can work within ArcGIS –
  – Custom field calculator functions
  – Scripts within ArcToolbox and Python Toolboxes
  – Interactively through the Python prompt
• Python can extend ArcGIS functionality –
  – Enable scheduled tasks
  – Improve upon ETL tasks
  – Talk to other software or components
A Whirlwind Tour
Python Interfaces

• Python Shell – running 'python' from the Command Line or the Python panel in ArcGIS.

• More commonly you’ll work within an IDE – integrated development environment.

• **IDLE** – a cross-platform Python development environment.

• **PythonWin** – a Windows only interface to Python.

• Eclipse, PyCharm, and Wing are other IDEs that support Python.

• Find the setup that works best for you.
IDLE – Development Environment

• IDLE helps you program in Python by:
  – color-coding your program code
  – debugging
  – auto-indent
  – interactive shell
Glorified Calculator

```python
2+2
4
(50-5*6)/4
5
7/3
2
7/3.0
2.3333333333333335
```
Assigning Variables

```python
>>> 4 + 3
7
>>> x = 4 + 3
>>> x / 2.0
3.5
>>> y = x / 2.0
>>> print x
7
>>> print y
3.5
```
More than just printing

- Python is an object oriented language.
- Practically everything can be treated as an object.
- Objects have specific types, such as integers or strings.
- "hello world" is a string.
- Strings are delimited using matching single or double quotes.
- Strings, as objects, have methods that return the result of a function on the string.
String Methods

- Assign a string to a variable, in this case: `hw`
  ```python
  hw = "hello world"
  ```
- `hw.title()`
- `hw.upper()`
- `hw.isdigit()`
- `hw.islower()`

```python
>>> print "hello world"
hello world
>>> hw = "hello world"
>>> hw.title()
'Hello World'
>>> hw.upper()
'HELLO WORLD'
>>> hw.isdigit()
False
>>> hw.islower()
True
```
String Methods

• The string held in your variable remains the same.
• The method returns an altered string.
• Changing the variable requires reassignment:
  – `hw = hw.upper()`
  – `hw` now equals "HELLO WORLD"
Other Python Objects

• Lists (mutable sets of strings):
  - `var = [] # create list`
  - `var = ['one', 2, 'three', 'banana']`

• Tuples (immutable sets):
  - `var = ('one', 2, 'three', 'banana')`

• Dictionaries (associative arrays or "hashes"):
  - `var = {} # create dictionary`
  - `var = { 'lat': 40.20547,
             'lon': -74.76322 }`
  - `var['lat'] = 40.2054`

• Each has its own set of methods.
Lists

• Think of a list as a stack of cards, on which your information is written.
• The information stays in the order you place it in until you modify that order.
• Methods return a string or subset of the list or modify the list to add or remove components.
• Written as var[index], index refers to order within set (think card number, starting at 0).
• You can step through lists as part of a loop.
List Methods

• Adding to the List
  – `var[n] = object`
    • replaces `n` with `object`
  – `var.append(object)`
    • adds `object` to the end of the list

• Removing from the List
  – `var[n] = []`
    • empties contents of card, but preserves order
  – `var.remove(n)`
    • removes card at `n`
  – `var.pop(n)`
    • removes `n` and returns its value
Lists in ArcToolbox

You will create lists:
• Layers as inputs
• Attributes to match
• Arrays of objects

You will work with lists:
• List of field names
• List of selected features
• "Multivalue" in ArcToolbox or ModelBuilder is a list of values.
List Manipulation

• Using array slices, you can identify a subset of a list for manipulation or copying.

  a = [100, 101, 102]

  print a[1:]

  >>> [101, 102]

• Strings act as lists of individual characters, so array slicing can be used for string subselection.

  Monty Python

  [6:10]

  [−12:−7]
Tuples

• Like a list, tuples are iterable arrays of objects.
• Tuples are immutable – once created, they are unchangeable.
• To add or remove items, you must redeclare them.
• Example uses of tuples:
  – County Names
  – Land Use Codes
  – Ordered set of functions
Dictionaries

- Dictionaries are sets of key & value pairs.
- Allows you to identify values by a descriptive name instead of order in a list.
- Keys are unordered unless explicitly sorted.
- Keys are unique:
  - `var['item'] = "apple"
  - `var['item'] = "banana"
  - `print var['item']` prints only "banana"
  - the value for key 'item' was overwritten
Indentation and Blocks

• Python uses whitespace and indents to denote blocks of code.
• Lines of code that begin a block end in a colon:
• Lines within the code block are indented at the same level.
• To end a code block, remove the indentation.
• Define blocks of code that run only when certain conditions are met.
• Functions are defined in a similar fashion.
Conditional Branching

- if and else -
  
  if variable == condition:
      #do something based on v == c
  else:
      #do something based on v != c

- elif allows for additional branching

  if x:
      code runs if x evaluates to true
  elif y:
      code runs if y evaluates to true
  else: #none of the above
      code runs if neither x nor y are true
Looping with For

• For allows you to loop over a block of code a set number of times

• For is great for manipulating lists:

```python
a = ['cat', 'window', 'defenestrate']
for x in a:
    print x, len(x)
```

Results:

```plaintext
cat 3
window 6
defenestrate 12
```
Looping with For

• We could use a for loop to perform geoprocessing tasks on each layer in a list.
• We could get a list of features in a feature class and loop over each, checking attributes.
• Anything iterable can be used in a For loop.
• Just be sure not to modify the list while looping.
Modules

• Modules are additional pieces of code that further extend Python's functionality.
• A module typically has a specific function or focus, such as:
  – additional math functions (numpy, scipy)
  – databases (psycopg2)
  – network (urllib)
• Python comes with many useful modules.
• arcpy is the module we will use to load ArcGIS functionality into Python.
Modules

- Modules are accessed using import
  - import sys, os # imports two modules

- Modules can have subsets of functions
  - os.path is a subset within os

- Modules are then addressed by modulename.function()
  - sys.argv # list of arguments
  - filename = os.path.splitext("points.txt")
  - filename[1] # equals ".txt"

- Documentation on modules and the import keyword shows the power of modularization.
Files

- Files are manipulated by creating a file object:
  - `f = open("points.txt", "r")`

- The file object then has new methods:
  - `print f.readline() # prints line from file`

- Files can be accessed to read, write, or append:
  - `f = open("output.txt", "w")`
  - `f.write("Important Output!")`
Error Capture

• Check for type assignment errors, items not in a list, etc.

• Try & Except
  try:
    a block of code that might have an error
  except:
    code to execute if an error occurs in "try"

• Allows for graceful failure
  – important in ArcGIS – built-in debugging capabilities are lacking
Resources
Books

Online Tutorials

• Learning Python:
  – Code.org
  – Codecademy

• ESRI Training offerings:
Exponentiation

All that math can be done on a calculator, so why use Python? Because you can combine math with other data types (e.g. booleans) and commands to create useful programs. Calculators just stick to numbers!

Now let’s work with exponents.

```python
eight = 2 ** 3
```

In the above example, we create a new variable called `eight` and set it to 8, or the result of 2 to the power to 3 \((2^3)\).

Notice that we use `**` instead of `*` or the multiplication operator.
Step 7: Create a list

The script's next task (indicated in the pseudocode) is to create a list of shapefile variables.

- In the script window, move your cursor to line 7, under #Create a list of shapefile variables.

```
1 #Assign variables to the shapefiles
2 park = "nd_park520.shp"
3 school = "ND_Schools454.shp"
4 sewer = "nd_sew454.shp"
5
6 #Create a list of shapefile variables
```

- Type the following line of code:

```
shapelist = [park, school, sewer]
```

- Press Enter.

In this line of code, shapelist is a variable assigned to a list of values. The list is indicated with the bracket closures [], and each item in the list is separated by a comma.
Online Communities

• Blogs
  – geospatialpython.com
  – sgillies.net
  – many more...

• Stack Exchange
  – gis.stackexchange.com
  – stackoverflow.com

• ESRI Python & Geoprocessing forums
Break
Incorporating Python into ArcGIS
Python within ArcGIS

• Python is integrated with the ArcGIS Desktop software and can be accessed in several ways:
  – Field calculator
  – ArcGIS Toolbox scripts & Python Toolboxes
  – Python Add-ins
  – Interactive Prompt

• Understanding Python can help with many operations.

• Your level of knowledge can vary; expanding your knowledge will only open up more opportunities.
The Field Calculator

- The Field Calculator (and Calculate Field tool) allow you to programmatically change the values within a layer's or table's column.
- Python can be used to calculate new data from existing fields or other properties.
Referencing Other Fields

- Fields are delimited with exclamation marks in the Python code block.
- `!block!` refers to the Block column.
- A function can be defined to do additional processing, with fields passed as parameters.
- Creating a PAMS_PIN for NJ parcel data.
def buildPAMS(mun, blk, lot, qua=None):
    """Returns a PAMS Pin from three or four cadastre fields. """
    if (qua == None) or (qua == ''):
        return "%s_%s_%s" % (mun, blk, lot)
    else:
        return "%s_%s_%s_%s" % (mun, blk, lot, qua)

pams_pin = buildPAMS(!muncode!, !block!, !lot!, !qual!)

About calculating fields
The SHAPE Field

- The SHAPE field has several methods that can be accessed within Field Calculator to retrieve attributes of the record's geometry.
- `!shape.area!`, `!shape.length!`
- Unit type is modifiable:
  - `!shape.area@SQUAREYARDS!`
- `!shape.partCount!`
- `!shape.pointCount!`
Moving from ModelBuilder to Python
ModelBuilder

• If you've used ModelBuilder in the past, you may have noticed that MB provides export to Python functionality.

• Python scripts generated by ModelBuilder can then be used and modified outside of ArcGIS.

• You will likely need to make modifications to your exported scripts:
  – MB-generated Python code is not always clear.
  – References to Layers or data via relative paths will need to be modified.
Great Starting Point

- Exporting from ModelBuilder is a great starting point for many projects.
- Declare variables/layers/workspaces using ModelBuilder UI. Start to put the pieces together.
- Export an incomplete model to Python for final development.
- The shell will be there – it will produce a valid (though potentially nonfunctional) arcpy script.
- You can then complete and extend the functionality.
# -*- coding: utf-8 -*-
# basic_arcpy_clip.py
# Created on: 2014-06-10 09:24:48.00000
# (generated by ArcGIS/ModelBuilder)
# Usage: basic_arcpy_clip <Input_Features> <Clip_Features> <Output_Feature_Class>
# Description:
# --------------------------------------------------------

# Import arcpy module
import arcpy

# Script arguments
Input_Features = arcpy.GetParameterAsText(0)
Clip_Features = arcpy.GetParameterAsText(1)
Output_Feature_Class = arcpy.GetParameterAsText(2)

# Local variables:

# Process: Clip
arcpy.Clip_analysis(Input_Features, Clip_Features, Output_Feature_Class, "")
The arcpy Module

• Importing the arcpy module provides access to the ArcGIS Engine in Python.

• Functionality provided by arcpy includes the tools provided by ArcToolbox, as well as some specialized functionality such as cursors.

• Easy, pythonic way to work with ArcGIS.

```python
import arcpy
arcpy.AddField_management("c:/data/Portland.gdb/streets", "LENGTH_MILES", "TEXT")
arcpy.CalculateField_management("c:/data/Portland.gdb/streets", "LENGTH_MILES", "!shape.length@miles!", "PYTHON_9.3")
```
Standalone arcpy Scripts

• arcpy can be called from any Python program that is using the version of Python installed with ArcGIS.
• Commonly under Windows, the version installed with ArcGIS is the only Python installed.
• Python and arcpy provides access to geoprocessing functionality and tasks, with lower overhead.
• Programs using arcpy can be run without having any of the ArcGIS Desktop suite running.
• Great for scheduled or background tasks.
arcpy Cursors

• Cursors enable you to iterate through a series of rows, insert new rows, or update existing rows.
• Cursors empower you to perform analysis on the feature level, as opposed to the layer level.
• Features can be directly accessed, enabling greater functionality than possible through most tools.
• Features can also be modified individually.
Geotagged Photos

# argv[1]: input directory of geotagged photos
# argv[2]: output directory
# argv[3]: output feature class

if os.path.exists(sys.argv[1]):
    files = []
    for f in os.listdir(sys.argv[1]):
        if os.path.splitext(f)[1].lower() == ".jpg":
            fp = os.path.join(sys.argv[1], f)
            # nested functions to retrieve EXIF GPS data
            gpsinfo = process_gps(get_exif(fp))
            if not len(gpsinfo) == 0:
                gpsinfo["name"] = f
                gpsinfo["path"] = fp
                files.append(gpsinfo)

rows = arcpy.InsertCursor(os.path.join(sys.argv[2], sys.argv[3]))
for f in files:
    row = rows newRow()
    row.NAME = f["name"]
    row.PATH = f["path"]
    pnt = arcpy.CreateObject("Point")
    pnt.x = f["x"]
    pnt.y = f["y"]
    row.SetValue("shape", pnt)
    rows.InsertRow(row)
import arcpy

def shift_features(in_features, x_shift=None, y_shift=None):
    """
    Shifts features by an x and/or y value. The shift values are in
    the units of the in_features coordinate system.
    """

    Parameters:
    in_features: string
        An existing feature class or feature layer. If using a
        feature layer with a selection, only the selected features
        will be modified.

    x_shift: float
        The distance the x coordinates will be shifted.

    y_shift: float
        The distance the y coordinates will be shifted.

    """

    with arcpy.da.UpdateCursor(in_features, ['SHAPE@XY']) as cursor:
        for row in cursor:
            cursor.updateRow([[row[0][0] + (x_shift or 0),
                               row[0][1] + (y_shift or 0)]]
                           )

    return
Extending ArcGIS tools using Python
Python Add-Ins

- At 10.1, ESRI introduced Python Add-Ins.
- Replacing VBA as the method for creating UI tools.
- ESRI provides an Add-In Wizard to help create these tools.
- ESRI training: "Creating Desktop Add-ins Using Python"
Download Imagery

• NJ OIT-OGIS's WMS services are great, but what if you want the source tiles?

• Python Add-Ins allow you to pull the data down, extract it and load it into ArcGIS.

• Allows for interactive selection within your map frame, with the data automatically added to the map once the download is complete.
NJ Services

Python-based Add-in for ArcGIS Desktop. Adds the following tools to your ArcMap environment:

- Open NJ-GeoWeb
- Download 2012 Imagery

You do not need to download any shapefiles or geodatabases for these tools to work. Using Python and available web services, all the necessary data will be pulled in dynamically.

https://github.com/RowanGeolab/arcgisPythonAddins
Moving Beyond ArcGIS using Python
Make ETL Easy

• Extract, transform, load processes can be tedious, especially if they are complex and frequent.
• Python can help assist you with bringing in outside data, conforming it to your needs and loading it into your database/storage.
• Python can connect to the web and databases, as well as process and convert GIS data.
• Python can also be run from Scheduled Tasks/cron, enabling you to fully automate these operations.
Web Retrieval & Extraction

• Several modules exist to enable easy downloading of remote resources.
• urllib (plus urllib2, urllib3, httplib, etc...)
• zipfile is a module for working with PKZIP files.
• Let's look at some code to automate downloads and extraction.
import urllib, os, sys, zipfile

def download(url, name=""):  
    if(name == ":"):  
        name = url.split('/\')[-1]
    webFile = urllib.urlopen(url)
    localFile = open(name, 'w')
    fname = localFile.name
    localFile.write(webFile.read())
    webFile.close()
    localFile.close()
    return fname

baseurls = {
    'parcels': r"http://njgin.state.nj.us/download2/parcels/parcels_mdb_{COUNTY}.zip" } 

# to grab NJGIN's copy of MODIV (assessment) data, comment out above and uncomment below.
#baseurls = {
#    'parcels': r"http://njgin.state.nj.us/download2/parcels/parcels_mdb_{COUNTY}.zip",
#}
for dt in baseurls.keys():
    for county in counties:
        url = baseurls[dt].replace("{COUNTY}", county)
        fn = url.split('/\')[-1]
        if(os.path.exists(fn)):
            print county, dt, "zip file already downloaded."
        else:
            fn = download(url)
            print county, "downloaded."

zipf = zipfile.ZipFile(fn, "r")
names = zipf.namelist()
if(len(names)==1):
    if(not os.path.exists(dt+names[0])):
        outz = open(dt+names[0], "wb")
        outz.write(zipf.read(names[0]))
        outz.close()
        print county, dt, "extracted."
    else:
        print dt+names[0], "already exists. Skipped."
else:
Fiona

• Fiona is a Python wrapper for the GDAL/OGR library.
• https://pypi.python.org/pypi/Fiona
• Enables you to convert/project/process a multitude of spatial data formats.
• Expanding on the previous example, the downloaded data can then be processed, converted to/from a specific format, reprojected, etc.
psycopg2

• psycopg2 module provides a Pythonic way of querying a PostgreSQL database.
• PostgreSQL-specific, enables accessing ArcSDE or PostGIS-formatted data.
• Full, secure access to your databases.
• Very easy to use the records returned from a database query in your Python programs.
Unconventional Exporting

- Python can help with report generation, producing CSV, XLS, and/or PDF files.
- CSVs are very easy to read and write.
- `xlrd` and `xlwt` enable reading and writing of Excel files.
- Python makes tedious tasks easy.
- Do you want to make 1,600 Excel files?
Watersheds and Impervious Surface
<table>
<thead>
<tr>
<th>HUC 14 Watershed</th>
<th>HUC 14 Subwatershed</th>
<th>HUC 14 Code</th>
<th>HUC 14 Area in Municipality</th>
<th>HUC 14 Total Area</th>
<th>Percentage Share of Municipality Area</th>
<th>Impervious Surface in HUC 14 in Municipality</th>
<th>Total Impervious Surface of HUC 14</th>
</tr>
</thead>
<tbody>
<tr>
<td>Still Run / Little Ease Run</td>
<td>Little Ease Run (above Academy)</td>
<td>02040206120010</td>
<td>2144.8</td>
<td>6262.12</td>
<td>35.78%</td>
<td>180.57</td>
<td>465.44</td>
</tr>
<tr>
<td>Mantua Creek</td>
<td>Chestnut Branch (above Sewell)</td>
<td>02040202130030</td>
<td>1417.95</td>
<td>5014.33</td>
<td>23.66%</td>
<td>428.37</td>
<td>960.09</td>
</tr>
<tr>
<td>Mantua Creek</td>
<td>Mantua Creek (above Rt 47)</td>
<td>02040202130010</td>
<td>1194.1</td>
<td>3873.2</td>
<td>19.92%</td>
<td>317.86</td>
<td>849.36</td>
</tr>
<tr>
<td>Still Run / Little Ease Run</td>
<td>Still Run (above Silver Lake Road)</td>
<td>02040206120030</td>
<td>648.27</td>
<td>4216.82</td>
<td>10.83%</td>
<td>125.72</td>
<td>255.01</td>
</tr>
<tr>
<td>Raccoon Creek / Birch Creek</td>
<td>Raccoon Ck (Rt 45 to incl Clemen)</td>
<td>02040202160020</td>
<td>322.18</td>
<td>5173.36</td>
<td>5.37%</td>
<td>29.88</td>
<td>334.2</td>
</tr>
<tr>
<td>Raccoon Creek / Birch Creek</td>
<td>Raccoon Ck (above Clemen Run)</td>
<td>02040202160010</td>
<td>264.33</td>
<td>4180.99</td>
<td>4.41%</td>
<td>9.69</td>
<td>147.55</td>
</tr>
<tr>
<td>Mantua Creek</td>
<td>Mantua Creek (road to Sewell)</td>
<td>02040202130020</td>
<td>1.5</td>
<td>5150.94</td>
<td>0.03%</td>
<td>0.07</td>
<td>988.74</td>
</tr>
</tbody>
</table>

Area values in acres.
Source: NJ MAP, derived from NJ DEP 2007 LU/LC data.

---

<table>
<thead>
<tr>
<th>HUC 14:</th>
<th>02040203120010</th>
<th>Percent Impervious Surface:</th>
<th>7.45%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subwatershed Name:</td>
<td>Still Run / Little Ease Run</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Watershed Name:</td>
<td>Little Ease Run (above Academy)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>County</th>
<th>Municipality</th>
<th>Code</th>
<th>Municipal Area in HUC 14</th>
<th>Municipality Total Area</th>
<th>Percentage Share of HUC 14 Area</th>
<th>Impervious Surface in Municipality in HUC 14</th>
<th>Total Impervious Surface of Municipality</th>
<th>Municipal Contribution to HUC Impervious Surface</th>
</tr>
</thead>
<tbody>
<tr>
<td>GLOUCESTER</td>
<td>GLASSBORO BORO</td>
<td>0806</td>
<td>2144.8</td>
<td>5994.13</td>
<td>34.25%</td>
<td>180.57</td>
<td>1092.16</td>
<td>38.80%</td>
</tr>
<tr>
<td>GLOUCESTER</td>
<td>CLAYTON BORO</td>
<td>0801</td>
<td>2071.15</td>
<td>4848.25</td>
<td>33.07%</td>
<td>140.6</td>
<td>443.84</td>
<td>32.21%</td>
</tr>
<tr>
<td>GLOUCESTER</td>
<td>MONROE TWP</td>
<td>0811</td>
<td>1790.69</td>
<td>30026.8</td>
<td>26.44%</td>
<td>100.56</td>
<td>2210.26</td>
<td>21.60%</td>
</tr>
<tr>
<td>GLOUCESTER</td>
<td>WASHINGTON TWP</td>
<td>0819</td>
<td>257.57</td>
<td>13788.73</td>
<td>4.11%</td>
<td>42.8</td>
<td>3061.5</td>
<td>9.20%</td>
</tr>
<tr>
<td>GLOUCESTER</td>
<td>ELK TWP</td>
<td>0804</td>
<td>7.89</td>
<td>12395.28</td>
<td>0.13%</td>
<td>0.91</td>
<td>429.89</td>
<td>0.20%</td>
</tr>
</tbody>
</table>

Area values in acres.
Source: NJ MAP, derived from NJ DEP 2007 LU/LC data.
with psycopg2.connect("dbname=muniprofile user=gisadmin host=gis.rowan.edu") as conn:  
    with conn.cursor() as cur:  
        cur.execute(list_huc)  
            for huc in cur:  
                w = xlwt.Workbook()  
                ws = w.add_sheet(huc[2])  
                ws.write(0, 0, "HUC 14:", style=s_title)  
                ws.write(0, 1, huc[2], style=s_title)  
                ws.write(1, 0, "Subwatershed Name:", style=s_title)  
                ws.write(1, 1, huc[0], style=s_title)  
                ws.write(2, 0, "Watershed Name:", style=s_title)  
                ws.write(2, 1, huc[1], style=s_title)  
                ws.write(3, 2, "Percent Impervious Surface:", style=s_title)  
                ws.write(3, 3, huc[3]/huc[4], xlwt.easyxf(num_format_str='0.00%'))  
                ws.write(4, 2, "Impervious Surface Acreage:", style=s_title)  
                ws.write(4, 3, huc[3], style=s_title)  
                ws.write(5, 2, "Total Acreage:", style=s_title)  
                ws.write(5, 3, huc[4], style=s_title)  
headers = ("County", "Municipality", "Code", "Municipal Area in HUC 14", "Municipality Total Area", "Percentage Share of HUC 14 Area", "Impervious Surface Acreage")  
for h in headers:  
    ws.write(0, 0, h, style=s_header)  
    ws.write(5, 0, h, style=s_header)  
    ws.write(5, 3, huc[4], style=s_header)  
    ws.write(5, 3, huc[4], style=s_footer)  
try:  
  for value in (record[3], record[2], record[5], record[7], record[11], (record[7]/record[9]), record[6], record[10], (record[6]/huc[3])):  
    if (cc == 5 or cc == 8):  
        ws.write(rc, cc, value, style=s_body)  
    else:  
        ws.write(rc, cc, value, style=s_body)  
        cc = cc + 1  
except decimal.InvalidOperation:  
  for value in (record[3], record[2], record[5], record[7], record[11], (record[7]/record[9]), record[6], record[10], 0):  
    if (cc == 5 or cc == 8):  
        ws.write(rc, cc, value, style=s_body)  
    else:  
        ws.write(rc, cc, value, style=s_body)  
        cc = cc + 1  
        rc = rc + 1  
ws.write(0, 0, "Area values in acres.", style=s_footer)  
ws.save(outfile)
Lunch
Some Custom Tools
Census
RowanGeolab / ArcGISCensusDownload

Census download tools for ArcGIS

6 commits 1 branch 0 releases 1 contributor

branch: master

ArcGISCensusDownload / +

fixed screenshot

johnroiser authored on Apr 23

<table>
<thead>
<tr>
<th>File</th>
<th>Description</th>
<th>Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>CensusDownload.pyt</td>
<td>added a few more census tables to the script</td>
<td>a month ago</td>
</tr>
<tr>
<td>LICENSE.txt</td>
<td>added license and to-do list</td>
<td>a month ago</td>
</tr>
<tr>
<td>README.md</td>
<td>fixed screenshot</td>
<td>a month ago</td>
</tr>
<tr>
<td>TODO.md</td>
<td>added license and to-do list</td>
<td>a month ago</td>
</tr>
<tr>
<td>arcgispythontoolbox.png</td>
<td>updated readme with default tables and screenshot</td>
<td>a month ago</td>
</tr>
</tbody>
</table>

Census Download Toolbox for ArcGIS

A Python Toolbox for ArcGIS Desktop (specifically ArcMap) to make quick retrieval of Census information.

Census API Key
Python Toolbox

• New in ArcGIS 10.1, Python Toolboxes are pure-Python files that appear as ArcToolbox toolboxes.
• Tools are defined as subclasses.
• A toolbox containing several independent tools can exist as one text-based file.
• Great format for transparency and version control.
• Uses .pyt as a file extension.
chupaESRI
chupaESRI

About

ChupaESRI is a Python module/command line tool to extract features from ArcGIS Server map services.

Name?
sh-3.2$ ~/scripts/EsriJSON2Pg.py http://amus.bernco.gov/ArcGIS/rest/services/AdvancePublic/MapServer/63/query "host=localhost dbname=parcels user=gisadmin" new mexico.bernalillo amus.bernco.gov
Requesting 0 <= objectid <= 999
Requesting 1000 <= objectid <= 1999
Requesting 2000 <= objectid <= 2999
Requesting 3000 <= objectid <= 3999
Q & A
Thanks for attending!

- Feel free to follow up with questions: jreiser@njgeo.org @johnjreiser on Twitter