Maptime MSP: Python

Tyler Johnson, GIS Specialist
Bolton & Menk

My history with Python

- Introduction was during lecture and workshop of college GIS course
  - No “real world” example or application
- First chance to put into use was several years later
  - Iowa Geocoding project – 16(!) separate structure shapefiles
- Started with ModelBuilder/Python snippets and a hand-me-down book
  - Knew ModelBuilder wasn’t the best option for custom workflows
  - Needed a way to automate very simple workflows

File Edit Format Run Options Windows Help

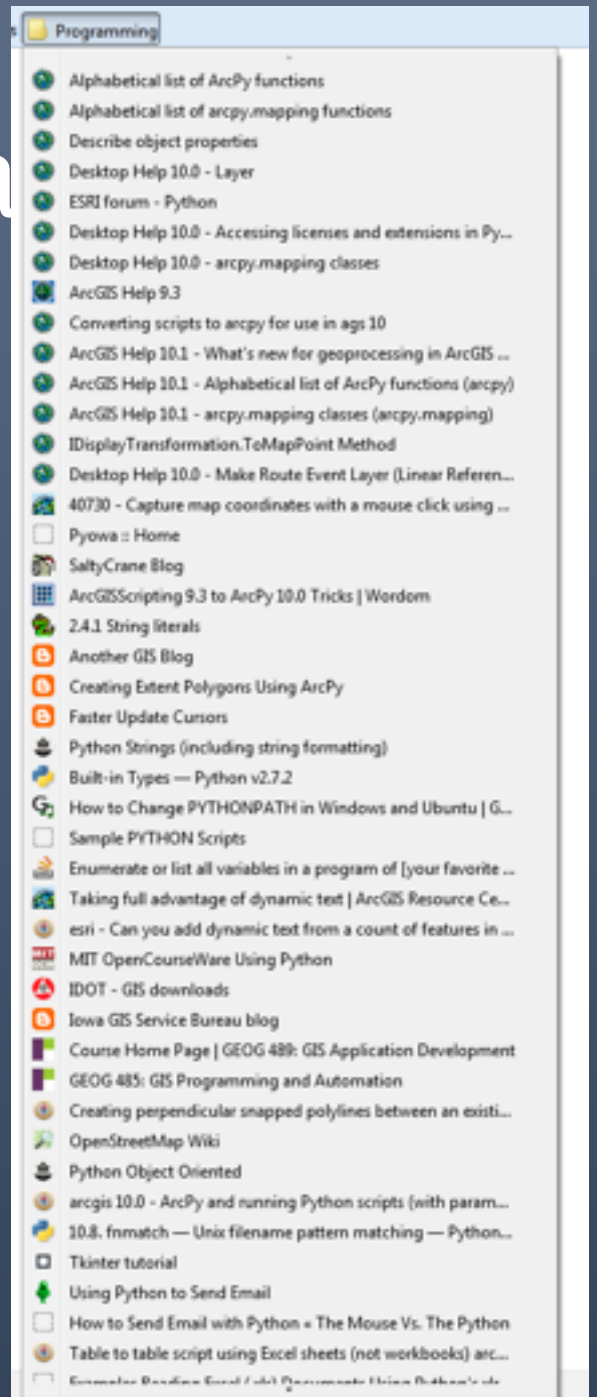
```
# ---------------------------------------------------------------------------
# This script is for use in the Geocoding project, Iowa DNR 2010-2011
#
# For the counties that were completed using separate shapefiles for each land use
# classification, this script populates the Land Use field for each shapefile, and
# then merges the 16 shapefiles into a single "Structure" shapefile.
#
# To use, add script to ArcToolbox and set a "Directory" parameter (Workspace type) as an input
#
# Structure shapefiles must be named consistently and as listed below for the script to work
# ---------------------------------------------------------------------------

import arcpy
from arcpy import env
import string

# Get working directory from parameter
struct_dir = arcpy.GetParameterAsText(0)
env.workspace = struct_dir
arcpy.AddMessage("\nWorking directory is " + struct_dir)

# Get county name from end of directory path
county_name = arcpy.GetParameterAsText(1)
arcpy.AddMessage("\nCounty name is: " + county_name)

# Calculate Land_Use fields
arcpy.AddMessage("\nCalculating Land Use fields...")
arcpy.CalculateField_management("Residential_(Individual_Dwelling_Unit).shp", "Land_Use", "\"Residential (Individual Dwelling Unit)\"", "VB", "")
arcpy.CalculateField_management("Public_Attractions_and_Landmarks.shp", "Land_Use", "\"Public Attractions and Landmarks\"", "VB", "")
arcpy.CalculateField_management("Information_and_Communication.shp", "Land_Use", "\"Information and Communication\"", "VB", "")
arcpy.CalculateField_management("Industrial.shp", "Land_Use", "\"Industrial\"", "VB", "")
arcpy.CalculateField_management("Health_and_Medical.shp", "Land_Use", "\"Health and Medical\"", "VB", "")
arcpy.CalculateField_management("Government_and_Military.shp", "Land_Use", "\"Government and Military\"", "VB", "")
arcpy.CalculateField_management("General_Building.shp", "Land_Use", "\"General Building\"", "VB", "")
arcpy.CalculateField_management("Energy.shp", "Land_Use", "\"Energy\"", "VB", "")
arcpy.CalculateField_management("Education.shp", "Land_Use", "\"Education\"", "VB", "")
arcpy.CalculateField_management("Commercial_or_Industrial.shp", "Land_Use", "\"Commercial or Industrial\"", "VB", "")
arcpy.CalculateField_management("Commercial.shp", "Land_Use", "\"Commercial\"", "VB", "")
arcpy.CalculateField_management("Agriculture.shp", "Land_Use", "\"Agriculture\"", "VB", "")
arcpy.CalculateField_management("Water_Supply_and_Treatment.shp", "Land_Use", "\"Water Supply and Treatment\"", "VB", "")
arcpy.CalculateField_management("Transportation.shp", "Land_Use", "\"Transportation\"", "VB", "")
arcpy.CalculateField_management("Residential_(Multiple_Dwelling_Unit).shp", "Land_Use", "\"Residential (Multiple Dwelling Unit)\"", "VB", "")
arcpy.CalculateField_management("Residential_(Mobile_Home).shp", "Land_Use", "\"Residential (Mobile Home)\"", "VB", "")
arcpy.AddMessage("\nLand Use fields calculated successfully")

# Merge separate Land Use shapefiles into one Structures shapefile
arcpy.AddMessage("\nMerging structure files...")
arcpy.Merge_management("Residential_(Individual_Dwelling_Unit).shp;Public_Attractions_and_Landmarks.shp;Information_and_Communication.shp;Industrial.shp; \
                       Health_and_Medical.shp;Government_and_Military.shp;General_Building.shp;Energy.shp;Education.shp;Commercial_or_Industrial.shp; \
                       Commercial.shp;Agriculture.shp;Water_Supply_and_Treatment.shp;Transportation.shp;Residential_(Multiple_Dwelling_Unit).shp; \
                       Residential_(Mobile_Home).shp", env.workspace + "/" + county_name + "_Structures.shp","")
arcpy.AddMessage("\nStructure shapefiles merged successfully")
```

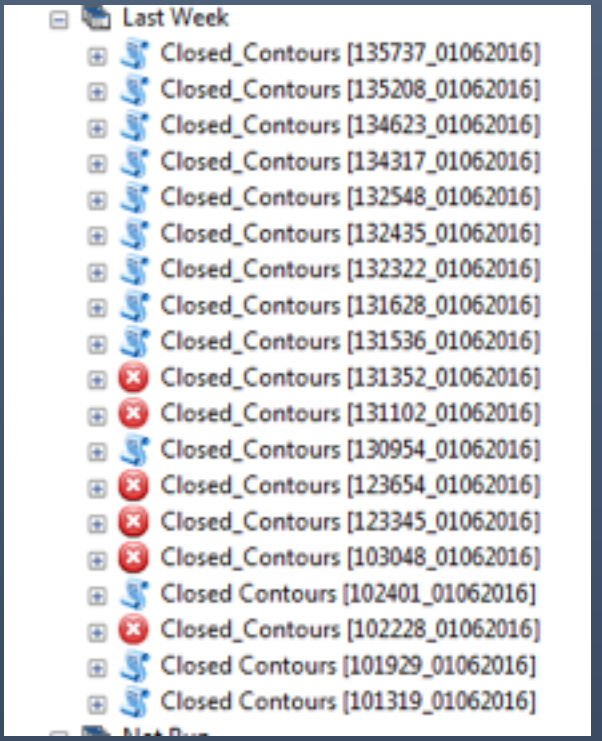
Ln: 1 Col: 0

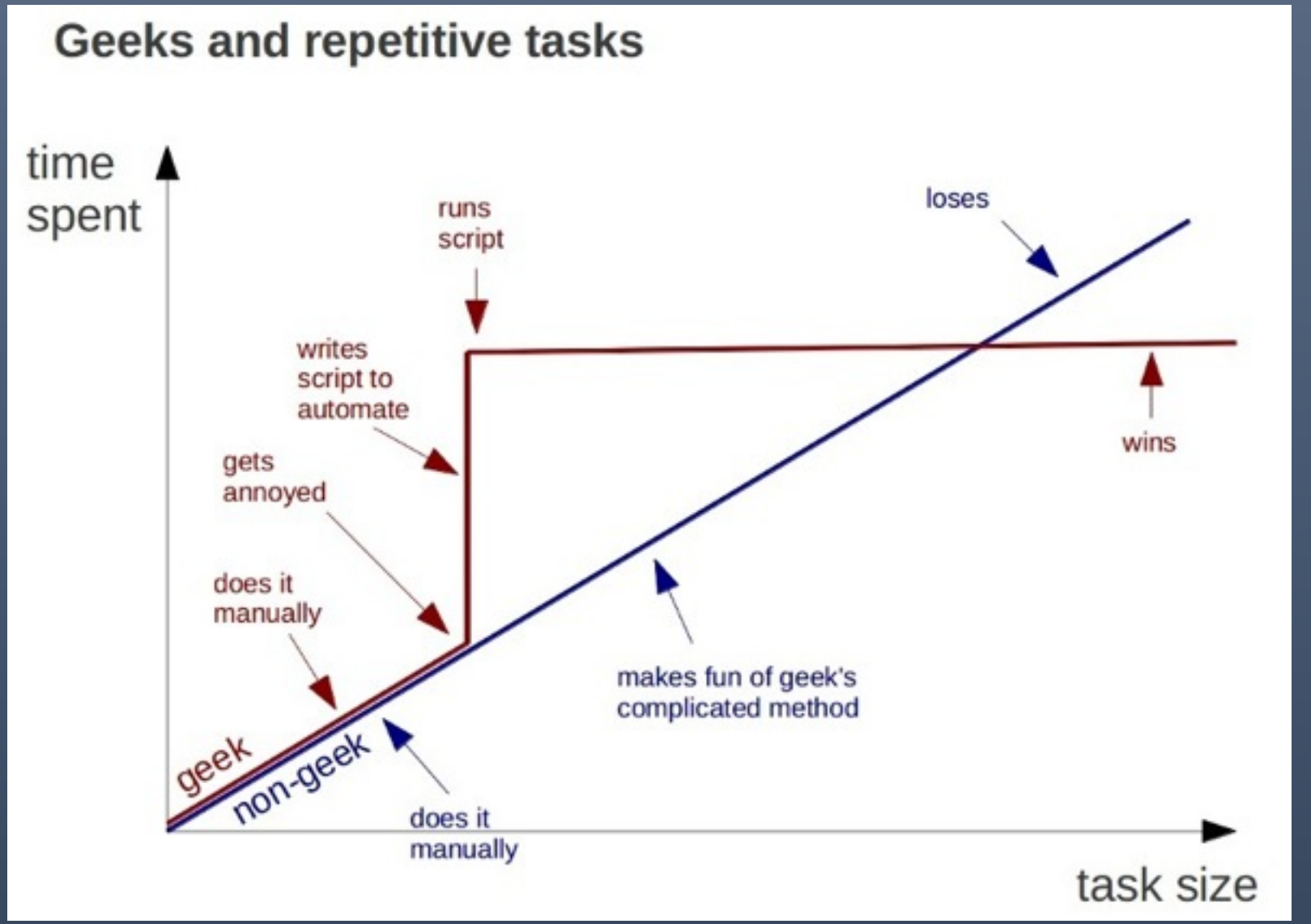
Learning the hard wa

- So many google and forum searches
- Lots of errors and warnings and crashes
- But constantly trying to find new scripts to write

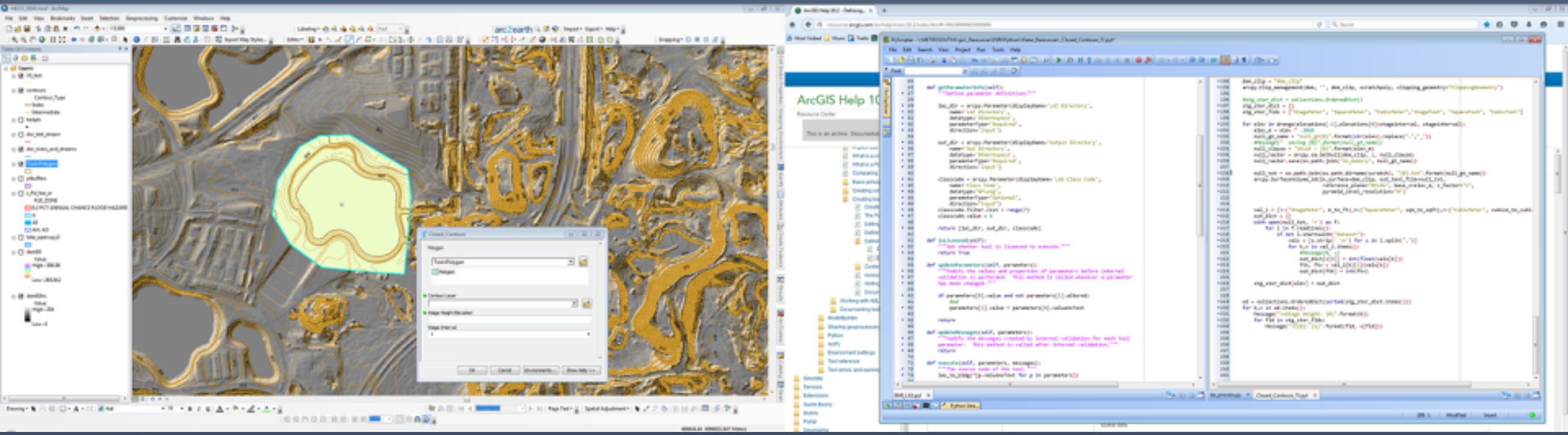
- "How to Make Mistakes in Python"






Geeks and repetitive tasks
time spent
runs script
loses
writes script to automate
wins
gets annoyed
does it manually
makes fun of geek's complicated method
geek
non-geek
does it manually
task size

Writing/testing a script



Essential bookmarks

- String formatting cookbook
  - https://mkaz.github.io/2012/10/10/python-string-format/
- Regular expression cheatsheet
  - https://www.debuggex.com/cheatsheet/regex/python
- Arcpy Geometry
  - http://resources.arcgis.com/en/help/main/10.2/index.html#//018z00000070000000
- Arcpy Parameter types
  - http://resources.arcgis.com/en/help/main/10.2/index.html#//001500000035000000

Challenges/roadblocks (mostly) overcome

- Making the case for automation and time investment to managers
- Getting colleagues to use and embrace the scripts
- Be willing to make mistakes and cause errors
- Started using functions, list comprehension, and dictionaries
- Finding the time to practice

Utility script(s) for common tasks

- Helpful for reusability
- Keep file in common directory and import into any or all scripts

```
1 import arcpy
2 import os, sys
3
4 # import utility module (in same dir as .pyt)
5 import rev_utils
6 rev_utils.in_arcgis = True
```

```
def basename(in_dataset):
    """Return arcpy's Describe basename attribute of in_dataset."""
    in_basename = arcpy.Describe(in_dataset).baseName
    return str(in_basename)


def compare_SR(dataset1, dataset2):
    """Compare spatial references of input datasets
    Return True if match, False if no match
    """

    sr1 = arcpy.Describe(dataset1).spatialReference
    sr2 = arcpy.Describe(dataset2).spatialReference
    if sr1.factoryCode == sr2.factoryCode:
        return True
    else:
        return False


def create_SR(fcode=None):
    """
    Create spatial reference object for given factory code.
    If no parameter given, default is NAD83 UTM Zone 15N.
    """
    if not fcode:
        fcode = 26915
    sr = arcpy.SpatialReference()
    sr.factoryCode = fcode
    sr.create()
    return sr


def duplicateDict(in_dataset, *in_fields):
    """Returns a dictionary containing values and count for attributes from in_dataset's in_field.
    Parameters:
    in_dataset - Input dataset to check for duplicates
                 Provide full path to dataset, or layer name of ArcMap layer
    in_fields - field(s) in input dataset to check for duplicate values
                If multiple fields given, values for fields will be concatenated in order given
    """

    dupe_dict = {}

    valuecursor = arcpy.da.SearchCursor(in_dataset, in_fields)
    valuecount = collections.Counter("-".join(map(str, row)) for row in valuecursor)
    dupe_dict = {x[0]:x[1] for x in valuecount.items() if x[1] > 1}

    return dupe_dict


def fld_iDict(in_dataset, fld_list=None):
    """Returns a dictionary containing field names (key) and their indices (value).
    Parameters:
    in_dataset - Input dataset or table
    fld_subset - List of fields (if None, all fields are included in dictionary

    Dictionary can be used for arcpy.da cursors where field index is used to access values.
    Example: {field name: field index}
    """
```

Functions, built-ins, and list comprehension

- More efficient
- Saves space
- Cleaner code (sometimes

```
def shp_to_geoJSON(indataset, outfolder, truncatefloat=True, floatprecision=2, toJS=True):
    """
    Convert ESRI dataset to geoJSON
    Arguments:
    indataset - Input dataset (feature layer, feature class, or shapefile)
    outfolder - Folder to save json file
    truncatefloat - If true, truncate ALL float values to 2 decimal places
    toJS - Save to JS file and insert a line to set 'var' value to JSON dictionary
    """
```

```
    with arcpy.da.SearchCursor(point, ['SHAPE@',pid]) as rows:
        for r in rows:
            if isinstance(r[0], arcpy.Geometry):
                p = r[0].firstPoint
                pxy[roundCoords(p, tolerance)] = r[1]
```

```
# unique IDs for parcels
parcelIDs = list(set([int(r[0]) for r in arcpy.da.SearchCursor(adjacent_gtACRES_parcels, ["PID"])]))
```

Python toolboxes and validation

- Handy, all-in-one script that can contain toolbox and tool setup, as well as the code to execute
- Makes sharing easy
- Sometimes not as handy when testing while writing

```
class Tool(object):
    def __init__(self):
        """Define the tool (tool name is the name of the class)."""
        self.label = "Closed_Contours"
        self.description = ""
        self.canRunInBackground = False
        self.elevations = []
        self.elevation_layers = []

    def getParameterInfo(self):
        """Define parameter definitions"""
        polygon = arcpy.Parameter(displayName='Polygon',
            name='Polygon',
            datatype='GPFeatureRecordSetLayer',
            parameterType='Required',
            direction='Input')
        polygon.value = r'C:\_temp\GP_Tasks.gdb\Boundary_UTM_15N'

        contours = arcpy.Parameter(displayName='Contour Layer',
            name='Contours',
            datatype='GPFeatureLayer',
            parameterType='Required',
            direction='Input')

        elev = arcpy.Parameter(displayName='Stage Height Elevation',
            name='Elevation',
            datatype='GPString',
            parameterType='Required',
            direction='Input')

        interval = arcpy.Parameter(displayName='Stage Interval',
            name='Interval',
            datatype='GPString',
            parameterType='Required',
            direction='Input')
        interval.filter.list = ['0.5','1','2']
        interval.value = 1

        return [polygon, contours, elev, interval]

    def isLicensed(self):
        """Set whether tool is licensed to execute."""
        return True

    def updateParameters(self, parameters):
        """Modify the values and properties of parameters before internal
        validation is performed.  This method is called whenever a parameter
        has been changed."""
        if not parameters[2].altered:
            if parameters[1].value and parameters[1].altered:
                scratchpoly = "in_memory\\scratchpoly"
                arcpy.CopyFeatures_management(parameters[0].value, scratchpoly)
                clipcontours = "in_memory\\scratchcontours"
                arcpy.Clip_analysis(parameters[1].value, scratchpoly, clipcontours)
                with arcpy.da.SearchCursor(clipcontours, ("Shape@","Elevation")) as rows:
                    for row in rows:
                        self.elevations.append(int(row[1]))
                if self.elevations:
```

Dictionaries

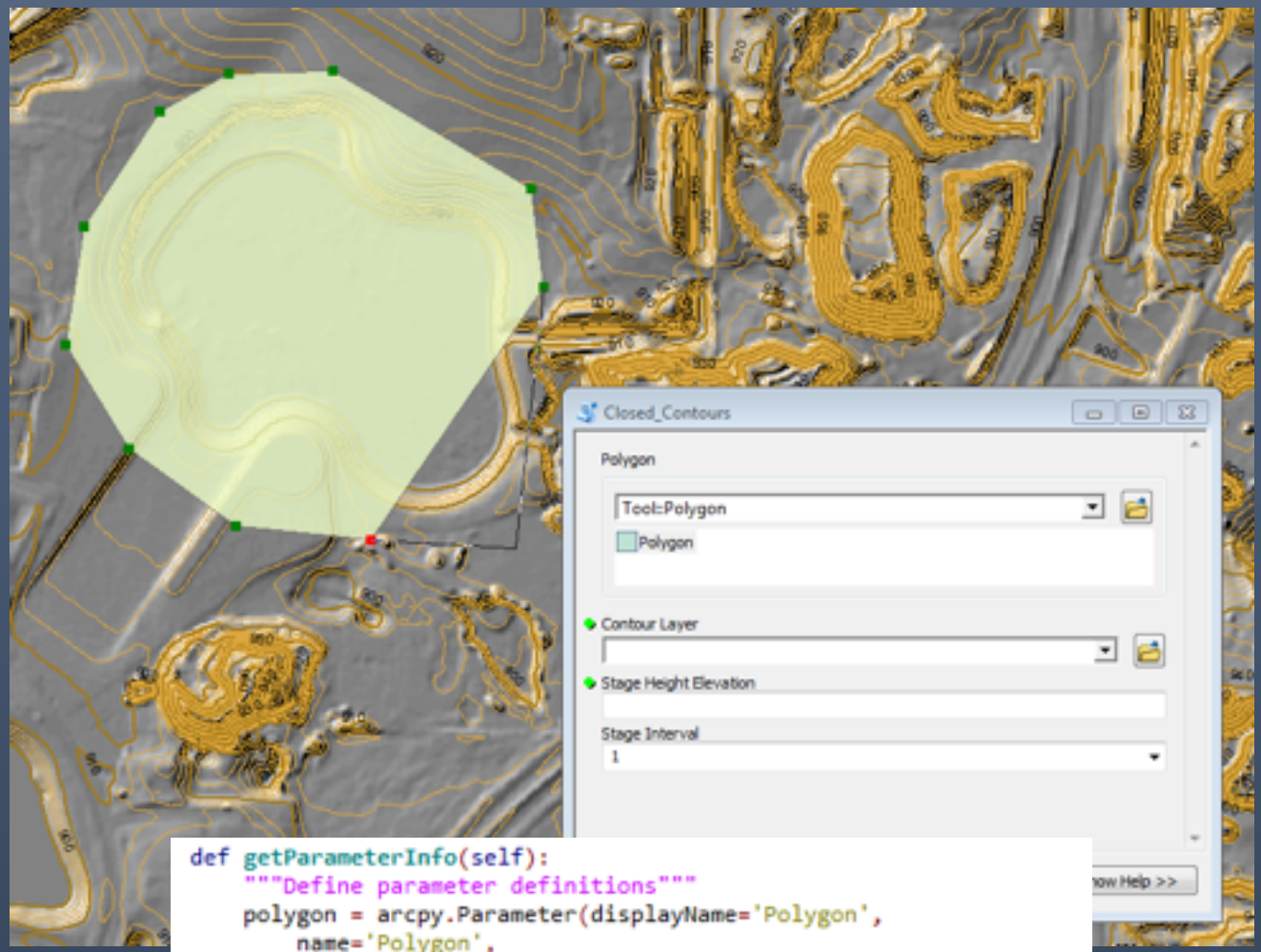
- Very handy object type!
- Keys can even be geometry objects
- Can even store function names to call
- Use the collections module's 'OrderedDict' class if order is important

```
start_invert_dict = {r[0]:r[1] for r in arcpy.da.SearchCursor(pipe_shp, [structs_fldnm, start_invert_fldnm])}
end_invert_dict = {r[0]:r[1] for r in arcpy.da.SearchCursor(pipe_shp, [structe_fldnm, end_invert_fldnm])}
```

```
val_i = {1:("StageMeter", m_to_ft),4:("SquareMeter", sqm_to_sqft),6:("CubicMeter", cubicm_to_cubicft)}
out_dict = {}
with open(null_txt, 'r') as f:
    for l in f.readlines():
        if not l.startswith("Dataset"):
            vals = [s.strip(' \n') for s in l.split(",")]
            for k,v in val_i.items():
                #Message(k, v)
                out_dict[v[0]] = int(float(vals[k]))
                ftk, ftv = val_i[k][1](vals[k])
                out_dict[ftk] = int(ftv)
```

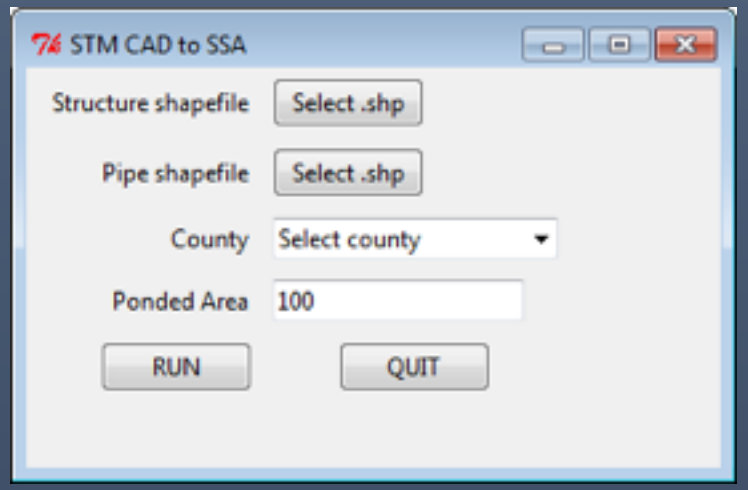
```
utDict = {}
with arcpy.da.UpdateCursor(mcd_mergedUT, ("County_num", "Shape@", "CountyGNIS"), ut_clause) as ucursor:
    for urow in ucursor:
        arcpy.AddMessage("{0} {1}".format(urow[0], urow[2]))
        if not urow[0] in utDict.keys():
            utDict[urow[0]] = [urow[1], ]
        else:
            polylist = utDict[urow[0]]
            polylist.append(urow[1])
        ucursor.deleteRow()

for cnum, polys in utDict.items():
    outpolys = polys[0]
    if len(polys) > 1:
        for polygon in polys[1:]:
            outpolys = outpolys.union(polygon)
    utDict[cnum] = outpolys
```

Feature set parameters




```
def getParameterInfo(self):
    """Define parameter definitions"""
    polygon = arcpy.Parameter(displayName='Polygon',
        name='Polygon',
        datatype='GPFeatureRecordSetLayer',
        parameterType='Required',
        direction='Input')
    polygon.value = r'C:\_temp\GP_Tasks.gdb\Boundary_UTM_15N'
```

Tkinter/GUIs




```
            struct_var.set('Structure shapefile')

    def openFile(var_to_set, idir):
        fname = tkFileDialog.askopenfilename(parent=root,
                                             initialdir=idir.get(),
                                             title="Select shapefile",
                                             filetypes=[('shp files', '*.shp')]) ## filename not filehandle

        if not fname:
            return
        else:
            var_to_set.set(fname)
            idir.set(os.path.dirname(fname))

    root = Tk()
    root.title("STM CAD to SSA")
    root.geometry("500x200")

    mainframe = ttk.Frame(root, padding="6 0 12 12")
    mainframe.grid(column=0, row=0, sticky=(N, W, E, S))
    mainframe.columnconfigure(0, weight=1)
    mainframe.rowconfigure(0, weight=1)

    initial_dir_val = "H:\\"
    dir_var = StringVar(value=initial_dir_val)

    ttk.Label(mainframe, text='Structure shapefile').grid(column=1, row=0, columnspan=1, sticky=E)
    struct_var = StringVar(value='Select .shp')
    struct_button  = ttk.Button(mainframe, textvariable=struct_var, command=lambda f=struct_var,i=dir_var: openFile(f,i)).

    ttk.Label(mainframe, text='Pipe shapefile').grid(column=1, row=2, columnspan=1, sticky=E)
    pipe_var = StringVar(value='Select .shp')
    pipe_button  = ttk.Button(mainframe, textvariable=pipe_var, command=lambda f=pipe_var,i=dir_var: openFile(f,i)).grid(c

    ttk.Label(mainframe, text='County').grid(column=1, row=4, columnspan=1, sticky=E)
    county_var = StringVar(value='Select county')
    countybox = ttk.Combobox(mainframe, textvariable=county_var, values=mncounties)
    countybox.grid(column=2, row=4, sticky=W)

    pond_var = StringVar(value=100)
    ttk.Label(mainframe, text='Ponded Area').grid(column=1, row=6, columnspan=1, sticky=E)
    pond_entry = ttk.Entry(mainframe, textvariable=pond_var)
    pond_entry.grid(column=2, row=6, sticky=W)

    runbutton = ttk.Button(mainframe, text="RUN", command=runfn).grid(column=1, row=8, sticky=E)
    quitbutton = ttk.Button(mainframe, text="QUIT", command=root.destroy).grid(column=2, row=8)

    for child in mainframe.winfo_children(): child.grid_configure(padx=5, pady=5)
    root.bind('<Return>', runfn)
    root.mainloop()


if __name__ == "__main__":

    gui()
```

Ongoing challenges

- Finding the time to practice
- Keeping up with new workflows, new releases, new technologies
  - ArcGIS Pro and Python 3.4
  - Python toolboxes
  - Python add-ins
  - REST endpoints
- Future direction/goals
  - Tkinter and GUIs
  - Using classes

Python resources

- Python website: http://www.python.org/
- Python package index: https://pypi.python.org/pypi
- ArcGIS 10.2 online help: http://resources.arcgis.com/en/help/main/10.2/index.html#/Alphabetical_list_of_ArcPy_functions/03q30000007s000000
- Python Geonet: https://geonet.esri.com/community/developers/gis-developers/python
- Python snippets: https://geonet.esri.com/docs/DOC-1927
- Free resources: http://www.blog.pythonlibrary.org/2013/07/11/free-books-and-other-free-resources-about-python/
- Another GIS Blog's python posts: http://anothergisblog.blogspot.com/search/label/Python
- Stack Overflow's GIS/Python section: http://gis.stackexchange.com/questions/tagged/python

Thank you!

Questions/comments?

tylerjo@bolton-menk.com