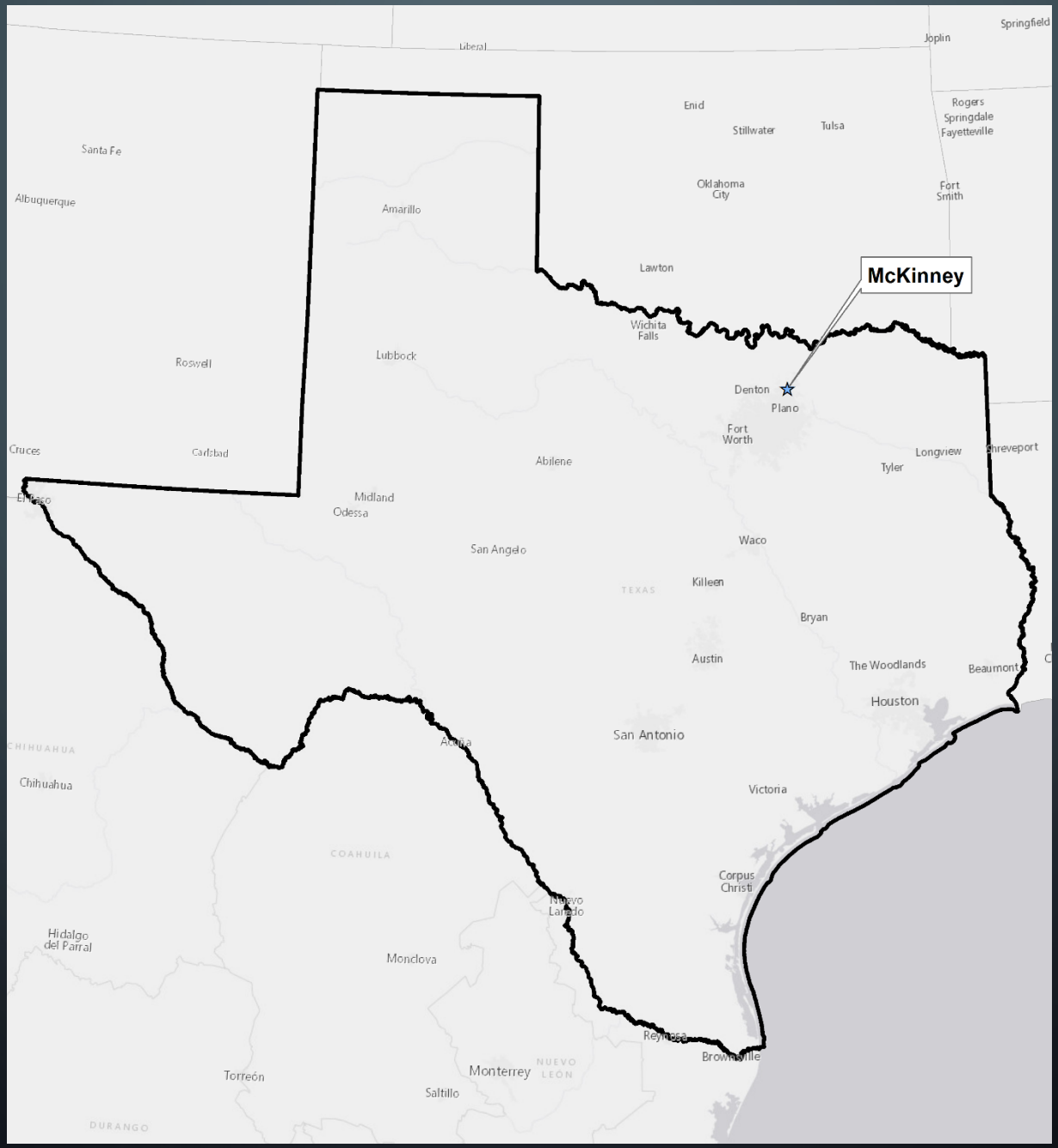
PYTHON FOR LOCAL GOVERNMENT

NEIL ROSE, GISP AND PARKER JONES

CITY OF MCKINNEY

INTRODUCTION

- Neil
  - BA in Geography and Geospatial Technology from UNC – Wilmington
  - Pursuing MGIST from Penn State University
  - 4.5 years at City of McKinney
- Parker
  - BS in Cartography and GIS from University of Wisconsin – Madison
  - 6 months at City of McKinney


McKinney
Springfield
Joplin
Liberal
Enid
Stillwater
Tulsa
Rogers
Springdale
Fayetteville
Santa Fe
Albuquerque
Amarillo
Oklahoma City
Fort Smith
Lawton
Wichita Falls
Roswell
Lubbock
Denton
Plano
Fort Worth
Cruces
Carlsbad
Abilene
Longview
Shreveport
Tyler
El Paso
Midland
Odessa
San Angelo
Waco
TEXAS
Killeen
Bryan
Austin
The Woodlands
Beaumont
Houston
San Antonio
Acuña
CHIHUAHUA
Chihuahua
Victoria
COAHUILA
Corpus Christi
Nuevo Laredo
Hidalgo del Parral
Monclova
Reynosa
Brownsville
Torreón
Monterrey
NUEVO LEÓN
Saltillo
DURANGO

CITY GROWTH

- 2000 population: 54,369
- 2010 population: 131,117 (141% increase from 2000)
- 2015 estimate: 155,142 (18% increase from 2010)
- Estimate as of January 1, 2018: 179,804 (16% increase from 2015)
- Rapid growth of city causing increased workload for GIS

ITINERARY

1. Problem
2. Street Name and Approval
3. FEMA Web Scraper
4. Project Notifications
5. Street Index Map Book Generator
6. Demonstrations
7. Q&A

PROBLEM

- Many day-to-day tasks that require the same process
- Limited staff and resources
- Need to streamline processes
- Desire for continuous improvement

STREET NAME AND APPROVAL

SOLUTION #1

STREET NAME AND APPROVAL PROCESS (BEFORE PYTHON)

- Receive list of proposed street names from developer
- Manually enter street names into spreadsheet
- Search existing street name feature one by one
- Search proposed street name table one by one
- Approve or deny
- Manually enter approved street names into proposed street name table one by one
- Approximately one hour per standard record plat

STREET NAME AND APPROVAL PROCESS WORKFLOW

1. Create temporary geodatabase
2. Convert input excel to table
3. Join table to street name approval table
4. Compare proposed streets to existing table entries
5. Convert table back to excel
6. Run Soundex

STREET NAME AND APPROVAL PROCESS (WITH PYTHON)

- Street2Street
  - Add proposed street names to excel template
  - Launch script using excel template as input
  - Approximately 30 seconds per standard record plat
- Approved2SDE
  - Appends approved streets from output excel to SDE table
  - Approximately 15 seconds per standard record plat

PYTHON BREAKDOWN

- create_dbconn
  - Searches for and finds SDE connection based on user

```
25      #   Function to create database connection for mapping
26      def create_dbconn():
27          username = getpass.getuser()
28          conn_path = os.path.join('C:\Users', username, 'AppData\Roaming\ESRI\Desktop10.3\ArcCatalog')
29          for sde in os.listdir(conn_path):
30              if sde.endswith('.sde'):
31                  connection = os.path.join(conn_path, sde)
32                  desc = arcpy.Describe(connection)
33                  cp = desc.ConnectionProperties
34                  if cp.server == 'MCKGISSQL1':
35                      if cp.database == 'SDE':
36                          if cp.authentication_mode == 'OSA':
37                              if cp.version == 'sde.DEFAULT':
38                                  db_conn = os.path.join(conn_path, sde)
39          return db_conn
```

PYTHON BREAKDOWN

- denial
  - Compares proposed street names to existing SDE based on join
  - Denies any street names that match existing entries

```
88      #   Denies street names based on existence
89      def denial():
90          fc = prop_street
91          fields = ['RoadName', 'StreetName', 'GIS_Approval', 'Final_Approval']
92          with arcpy.da.UpdateCursor(fc, fields) as cursor:
93              for row in cursor:
94                  if row[0] == row[1]:
95                      row[2] = 'Denied - Exists'
96                      row[3] = 'Denied'
97                      cursor.updateRow(row)
98          del cursor
99          arcpy.DeleteField_management(prop_street, 'StreetName')
100         return
```

PYTHON BREAKDOWN

- Soundex
  - Compares proposed street names to existing table based on linguistics (sound alike)

```
131     #   Soundex defined functions
132     def run_soundex(names, dictionary):
133         for n in names:
134             dictionary['%-10s' % n] = fuzzy.nysiis(n)
135         return
```

```
148     #   Creates the a list of potential streets to run against Soundex
149     def create_pot_list():
150         workbook = load_workbook(final_excel)
151         sheet = workbook.active
152         rowx, colx, coly = 2, 3, 11
153         while rowx <= sheet.max_row:
154             if sheet.cell(row=rowx, column=coly).value != 'Denied':
155                 pot_list.append(sheet.cell(row=rowx, column=colx).value)
156             rowx += 1
157         return
```

PYTHON BREAKDOWN

```
160     #   Creates a variety of dictionaries to use to write the Soundex information to the xlsx
161     def build_dictionaries():
162         for k, v in pot_dict.items():
163             for k2, v2 in str_dict.items():
164                 if v == v2:
165                     fin_dict[k2] = k
166
167         for k, v in fin_dict.items():
168             dup_list.append(v)
169
170         newlist = list(set(dup_list))
171
172         for i in newlist:
173             end_dict[i] = ''
174
175         for i in end_dict.items():
176             for k, v in fin_dict.items():
177                 if i == v:
178                     end_dict.update(fin_dict)
179
180         for k1, v1 in end_dict.items():
181             for k2, v2 in fin_dict.items():
182                 if v2 == k1:
183                     soundex_dict.setdefault(k1, [])
184                     soundex_dict[k1].append(k2)
185         return
```

PYTHON BREAKDOWN

```
188     #   Writes the Soundex streets to the .xlsx
189     def write_soundex():
190         workbook = load_workbook(final_excel)
191         sheet = workbook.active
192         rowx, colx, coly, colz = 2, 3, 10, 11
193         while rowx <= sheet.max_row:
194             for k, v in soundex_dict.items():
195                 if sheet.cell(row=rowx, column=colx).value == k.strip():
196                     if sheet.cell(row=rowx, column=colz).value != 'Denied':
197                         str_names = str([x.encode('UTF8') for x in v]).strip("[]'")
198                         str_names2 = str(str_names).replace(' ', '')
199                         sheet.cell(row=rowx, column=coly).value = 'Soundex: ' + str(str_names2).replace("','", ", ")
200             rowx += 1
201         workbook.save(final_excel)
202         return
```

FEMA WEB SCRAPER

SOLUTION #2

FEMA DATA ACQUISITION PROCESS (BEFORE PYTHON)

- No standard acquisition process
- Incomplete, outdated FEMA dataset
  - Last updated in 2009
  - Only contained floodplain data

FEMA DATA ACQUISITION PROCESS WORKFLOW

1. Open and navigate webdriver to locate and download data
2. Extract data and compare schema to existing FEMA features
3. Clear existing data and append newly downloaded data

PYTHON BREAKDOWN

- download_data

```
# Download data from FEMA website
def download_data():
    driver = webdriver.Chrome('Y:\GIS\GISWork\PJones\MapProjects\FEMA_WebScraper\chromedriver_win32\chromedriver.exe')
    driver.get('https://msc.fema.gov/portal/advanceSearch#searchresultsanchor')
    driver.maximize_window()

    jurisdiction_search = driver.find_element_by_id('txtjurisdictionname')
    jurisdiction_search.send_keys('MCKINNEY, CITY OF')
    time.sleep(1)
    jurisdiction_search.send_keys(Keys.ARROW_DOWN)
    time.sleep(1)
    jurisdiction_search.send_keys(Keys.ENTER)
    time.sleep(1)

    search_button = driver.find_element_by_id('mainSearch')
    search_button.click()

    wait = WebDriverWait(driver, 100)
    effective_products = wait.until(ec.visibility_of_element_located((By.ID, 'eff_root')))
    effective_products.click()

    county_products = wait.until(ec.visibility_of_element_located((By.ID, 'eff_nfhl_county_root')))
    county_products.click()

    county_products.send_keys(Keys.TAB, Keys.ENTER)

    downloaded = False
    while not downloaded:
        for download_file in os.listdir(download_directory):
            if download_file.startswith('48085C_') and download_file.lower().endswith('crdownload'):
                time.sleep(1)
            elif download_file.startswith('48085C_') and download_file.lower().endswith('.zip'):
                downloaded = True
            else:
                time.sleep(0)
    return
```

PYTHON BREAKDOWN

- fc_schema_compare

```
169  # Run feature comparisons of downloaded feature classes to existing SDE feature classes
170  def fc_schema_compare():
171      arcpy.env.workspace = feature_dataset
172
173      for feature_class in arcpy.ListFeatureClasses():
174          fc_path = os.path.join(arcpy.env.workspace, feature_class)
175          arcpy.env.workspace = sde_path
176          for sde_feature_class in arcpy.ListFeatureClasses():
177              sde_fc_path = os.path.join(arcpy.env.workspace, sde_feature_class)
178              if feature_class == sde_feature_class[8:]:
179
180                  if not os.path.exists(os.path.join(fema_folder, 'Compare')):
181                      os.makedirs(os.path.join(fema_folder, 'Compare'))
182                      compare_folder = os.path.join(fema_folder, 'Compare')
183                  else:
184                      compare_folder = os.path.join(fema_folder, 'Compare')
185
186                  compare_txt = os.path.join(compare_folder, str(feature_class + '.csv'))
187                  arcpy.FeatureCompare_management(fc_path, sde_fc_path, 'OBJECTID', 'SCHEMA_ONLY', '#',
188                                                  '0.003280833333 Feet', 0, 0, '#', '#', 'CONTINUE_COMPARE', compare_txt)
189                  print 'Feature class {0} has been compared'.format(feature_class)
190                  arcpy.env.workspace = feature_dataset
191      return
```

PYTHON BREAKDOWN

- clear_sde_tables

```
# Delete existing data from SDE feature classes and tables
def clear_sde_tables():
    arcpy.env.workspace = feature_dataset
    create_dbconn()

    for feature_class in arcpy.ListFeatureClasses():
        arcpy.env.workspace = sde_path
        for sde_feature_class in arcpy.ListFeatureClasses():
            sde_fc_path = os.path.join(arcpy.env.workspace, sde_feature_class)
            if feature_class == sde_feature_class[8:] and int(arcpy.GetCount_management(sde_fc_path).getOutput(0)) > 0:
                arcpy.DeleteRows_management(sde_fc_path)
                print 'SDE Feature class {0} has been cleared'.format(feature_class)
            elif feature_class == sde_feature_class[8:] and \
                    int(arcpy.GetCount_management(sde_fc_path).getOutput(0)) == 0:
                print 'SDE Feature class {0} is already empty'.format(feature_class)

    arcpy.env.workspace = temp_gdb

    for table in arcpy.ListTables():
        arcpy.env.workspace = create_dbconn()
        for sde_table in arcpy.ListTables():
            sde_table_path = os.path.join(arcpy.env.workspace, sde_table)
            if table == sde_table[8:] and int(arcpy.GetCount_management(sde_table_path).getOutput(0)) > 0:
                arcpy.DeleteRows_management(sde_table_path)
                print 'SDE Table {0} has been cleared'.format(table)
            elif table == sde_table[8:] and int(arcpy.GetCount_management(sde_table_path).getOutput(0)) == 0:
                print 'SDE Table {0} is already empty'.format(table)
    return
```

PYTHON BREAKDOWN

- append_data (QA/QC)

```
if cell_string == 'FALSE':
    arcpy.Append_management(fc_path, sde_fc_path, 'NO_TEST', '', '')
    print 'No feature comparison errors for feature class {0}, appending data to SDE'.format(feature_class)
    print ''
    os.remove(os.path.join(fema_folder, 'Compare', str(feature_class) + '.xlsx'))
    os.remove(os.path.join(fema_folder, 'Compare', str(feature_class) + '.txt.xml'))

else:
    starting_row = 2
    comparison_error = False

    while starting_row <= sheet.max_row:
        cell = sheet.cell(row=starting_row, column=3).value
        cell_string = str(cell)

        if 'precisions are different' in cell_string:
            print 'Ignoring precision difference for feature class {0}'.format(feature_class)
            starting_row += 1

        elif 'scales are different' in cell_string:
            print 'Ignoring scale difference for feature class {0}'.format(feature_class)
            starting_row += 1

        elif 'DATE lengths are different' in cell_string:
            print 'Ignoring date field length difference for feature class {0}'.format(feature_class)
            starting_row += 1

        elif 'Shape lengths are different' in cell_string:
            print 'Ignoring shape length field mismatch for feature class {0}'.format(feature_class)
            starting_row += 1
```

PYTHON BREAKDOWN

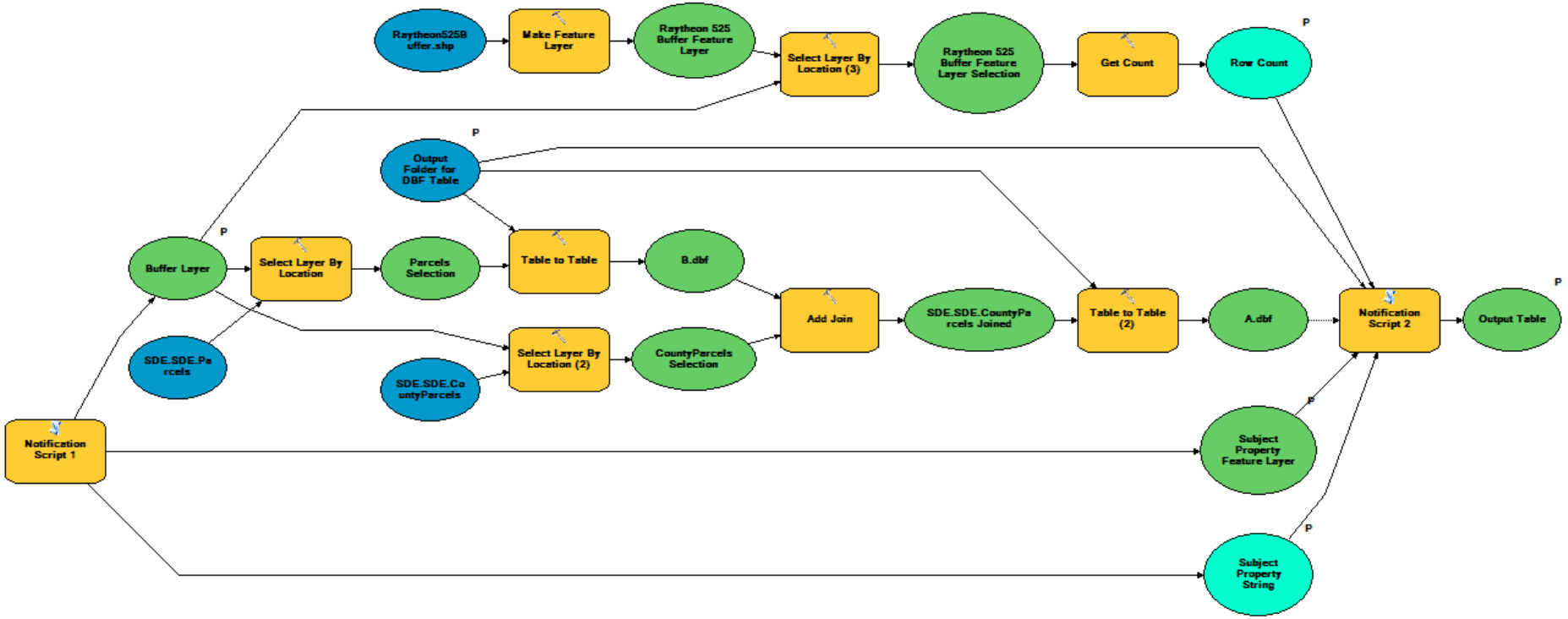
- append_data (QA/QC), continued

```
    elif 'GeometryDef' in cell_string:
        print 'Ignoring grid size mismatch for feature class {0}'.format(feature_class)
        starting_row += 1

    elif 'Table fields do not match' in cell_string:
        starting_row += 1

    elif 'Tables have different number of fields' in cell_string:
        print 'Feature comparison error: Number of fields in ' + feature_class + ' does not match number ' \
              'of fields on SDE. Data will not be appended.'
        print ''
        error_list.append(feature_class)
        comparison_error = True
        break

    else:
        print 'Feature comparison error: ' + cell_string + '. ' + feature_class + ' will not be ' \
                                                                                  'appended to SDE'
        print ''
        error_list.append(feature_class)
        comparison_error = True
        break

if not comparison_error:
    arcpy.Append_management(fc_path, sde_fc_path, 'NO_TEST', '', '')
    print 'No critical feature comparison errors for feature class {0}, ' \
          'appending data to SDE'.format(feature_class)
    print ''
    os.remove(os.path.join(fema_folder, 'Compare', str(feature_class) + '.xlsx'))
    os.remove(os.path.join(fema_folder, 'Compare', str(feature_class) + '.txt.xml'))
```

PROJECT NOTIFICATIONS

SOLUTION #3

PROJECT NOTIFICATIONS (BEFORE CURRENT SCRIPT)



PROJECT NOTIFICATIONS (CURRENT SCRIPT)

1. Input parameters
   - Case number or address
   - Map type
   - Output folder
2. Input validation
3. TOC cleanup
4. Select and buffer subject property
5. Map element modification
6. Postcard generation
7. PDF generation

PYTHON BREAKDOWN

- verify_case

```
#   Function to verify inputs exist before running any other functions
def verify_case():
    if map_type in ['Location Map', 'Notification Map']:
        layer_conn = db_path('LandBase', 'DevelopmentProjects_Active')
        feature = arcpy.MakeFeatureLayer_management(layer_conn, 'Search Layer')
        field = 'CaseNumber'
    elif map_type == 'Alcohol and Gaming Map':
        layer_conn = db_path('LandBase', 'AddressPoints')
        feature = arcpy.MakeFeatureLayer_management(layer_conn, 'Search Layer')
        field = 'SitusAdd'

    item_list = []

    with arcpy.da.SearchCursor(feature, (field,)) as cursor:
        for row in cursor:
            clean_row = str(row[0]).upper()
            if initial_input == clean_row:
                item_list.append(clean_row)
    if not item_list:
        sys.exit(note('{0} has not been found. Ending script.'.format(initial_input)))
    else:
        note('{0} has been located. Creating the {1} and associated materials.'.format(initial_input, map_type))
    del cursor
    arcpy.Delete_management('Search Layer')
    return
```

PYTHON BREAKDOWN

- sel_subj

```
107     #   Function to select subject property and make layer in TOC
108     def sel_subj(field, name):
109         for lyr in arcpy.mapping.ListLayers(mxd):
110             if lyr.name == name:
111                 lyr.definitionQuery = field + " = '" + initial_input + "'"
112                 lyr.visible = True
113         if map_type in ['Location Map', 'Notification Map']:
114             set_lyr('SubjectProperty')
115         refresh_mxd()
116         return
```

PYTHON BREAKDOWN

- sel_parcel

```
119     #   Select by Location from Address Points to Parcels
120     def sel_parcel():
121         layer_conn = db_path('LandManagement', 'CountyParcels_ETJ')
122         parcels = layer_conn
123         arcpy.MakeFeatureLayer_management(parcels, 'parcel_lyr')
124         subj_parcel = arcpy.SelectLayerByLocation_management('parcel_lyr', 'intersect', 'SubjectAddress')
125         arcpy.MakeFeatureLayer_management(subj_parcel, 'SubjectParcel')
126         with arcpy.da.SearchCursor('SubjectParcel', ('PROP_ID',)) as cursor:
127             for row in cursor:
128                 property_id.append(row[0])
129                 copy_lyr('CountyParcels_ETJ', 'SubjectProperty')
130                 for lyr in arcpy.mapping.ListLayers(mxd):
131                     if lyr.name == 'SubjectProperty':
132                         lyr.definitionQuery = "PROP_ID" + " = '" + str(row[0]) + "'"
133                         lyr.visible = True
134         set_lyr('SubjectProperty')
135         for lyr in arcpy.mapping.ListLayers(mxd):
136             if lyr.name == 'SubjectAddress':
137                 arcpy.mapping.RemoveLayer(df, lyr)
138         arcpy.Delete_management('parcel_lyr')
139         arcpy.Delete_management('SubjectParcel')
140         del cursor
141         refresh_mxd()
142         return
```

PYTHON BREAKDOWN

- zoom2scale and set_lyr

```
226     #   Function to set zoom extent and scale for the mxd
227     def zoom2scale(feature):
228         lyr = arcpy.mapping.ListLayers(mxd, feature, df)[0]
229         ext = lyr.getExtent()
230         df.extent = ext
231         df.scale = df.scale * 2
232         return
233
234
235     #   Function to set pre-defined layer symbology
236     def set_lyr(feature):
237         if feature == 'SubjectProperty':
238             sym_lyr = r'S:\MCKGIS\Notification\Workspace\SubjectProperty.lyr'
239         elif feature == 'Buffer':
240             sym_lyr = r'S:\MCKGIS\Notification\Workspace\Buffer.lyr'
241         arcpy.ApplySymbologyFromLayer_management(feature, sym_lyr)
242         refresh_mxd()
243         return
```

PYTHON BREAKDOWN

- text_update

```
#   Function to clean up map text elements
def text_update():
    if map_type == 'Location Map':
        for elm in arcpy.mapping.ListLayoutElements(mxd):
            if elm.name == 'MapType':
                elm.text = 'Location Map'
                elm.elementPositionX = 4.5
                elm.elementPositionY = 1.75
            if elm.name == 'Case':
                elm.text = initial_input
                elm.elementPositionX = 4.5
                elm.elementPositionY = 1.25
            if elm.name == 'AG_Buffer':
                elm.elementPositionX = 10.0
                elm.elementPositionY = 0.75
            if elm.name == 'N_Buffer':
                elm.elementPositionX = 10.0
                elm.elementPositionY = 0.75
```

PYTHON BREAKDOWN

- postcard

```
#   Function to create the Postcard map for Notification maps
def postcard():
    arcpy.env.workspace = r'S:\MCKGIS\Notification\Workspace'
    pc_png = str(output_folder + '\\' + initial_input + '.png')
    pc_mxd_path = str(output_folder + '\\' + initial_input + 'postcard.mxd')
    mxd.saveACopy(pc_mxd_path)
    pc_mxd = arcpy.mapping.MapDocument(pc_mxd_path)
    pc_mxd.activeView = 'PAGE_LAYOUT'
    for elm in arcpy.mapping.ListLayoutElements(pc_mxd):
        if elm.name in ['VicinityMap', 'Filepath', 'MapType', 'Case', 'Date', 'Disclaimer',
                        'Scale Bar', 'Scale Text', 'N_Buffer', 'AG_Buffer', 'SubjProp', 'Logo']:
            elm.elementPositionX = 12.0
            elm.elementPositionY = 5.5
        if elm.name == 'North Arrow':
            elm.elementHeight = 0.85
            elm.elementPositionX = 8.125
            elm.elementPositionY = 0.3
        if elm.name == 'Neatline':
            elm.elementHeight = 10.75
            elm.elementPositionX = 4.25
            elm.elementPositionY = 5.5
        if elm.name == 'Layers':
            elm.elementHeight = 10.55
            elm.elementPositionX = 4.25
            elm.elementPositionY = 5.475
    refresh_mxd()
    pc_mxd.save()
    arcpy.mapping.ExportToPNG(pc_mxd, pc_png, 'PAGE_LAYOUT')
    del pc_mxd
    os.remove(pc_mxd_path)
    return
```

STREET INDEX

SOLUTION #4

STREET INDEX MAP BOOK PROCESS (BEFORE PYTHON)

- Export title page mxd and data driven pages mxd manually
- Copy table of street names from ArcGIS into Excel
- Format cells in Excel, then paste into Word document
- Adjust size and position of tables as necessary
- Export Word document to PDF
- Combine title page PDF, data driven pages PDF, and index PDF into single document.

STREET INDEX MAP BOOK PROCESS (WITH PYTHON)

1. Generate street index table
2. Automatically export title page and data driven pages to PDF
3. Write street index table directly to PDF using ReportLab PDF library
4. Combine files into single PDF

PYTHON BREAKDOWN

- Index_template

```
def index_template(canvas, doc):
    canvas.saveState()

    # The word Index in upper, center of page
    canvas.drawCentredString(5.5 * inch, 8 * inch, "Index")
    # Title in upper-right corner of page
    canvas.setFont('Times-Roman', 7)
    canvas.drawRightString(10.5 * inch, 8 * inch, 'City of McKinney Street Atlas')
    # Draw horizontal lines
    canvas.line(0.5 * inch, 7.9 * inch, 10.5 * inch, 7.9 * inch)
    canvas.line(0.5 * inch, 7.625 * inch, 10.5 * inch, 7.625 * inch)
    canvas.line(0.5 * inch, 0.6 * inch, 10.5 * inch, 0.6 * inch)
    # Draw vertical lines
    canvas.line(3.83 * inch, 0.6 * inch, 3.83 * inch, 7.9 * inch)
    canvas.line(7.17 * inch, 0.6 * inch, 7.17 * inch, 7.9 * inch)
    # Add column titles
    canvas.setFont('Times-Roman', 8)
    canvas.drawString(2 * inch, 7.7 * inch, "Page")
    canvas.drawString(2.54 * inch, 7.7 * inch, "Grid")
    canvas.drawString(5.33 * inch, 7.7 * inch, "Page")
    canvas.drawString(5.87 * inch, 7.7 * inch, "Grid")
    canvas.drawString(8.66 * inch, 7.7 * inch, "Page")
    canvas.drawString(9.2 * inch, 7.7 * inch, "Grid")

    canvas.setFont('Times-Roman', 7)
    canvas.drawRightString(10.5 * inch, 0.45 * inch, 'City of McKinney GIS Department')
    canvas.restoreState()
    return
```

PYTHON BREAKDOWN

- create_index

```
def create_index():
    index_start = time.time()
    pdf_styles = getSampleStyleSheet()
    pdf_elements = []

    current_letter = ''
    flag = ''
    street_names = os.path.join(temp_gdb, 'Streets_sorted')
    with arcpy.da.SearchCursor(street_names, ['DISPLAY', 'PAGE', 'GRID']) as cursor:
        for row in cursor:
            name_strip = row[0]
            first_letter = name_strip[0]
            if first_letter.isdigit() or name_strip[2].isdigit() or name_strip[0:3] == 'CR '\
                    or name_strip[0:3] == 'FM ':
                if not current_letter:
                    current_letter = "0"
                    pdf_elements.append(Paragraph('   ', pdf_styles['Heading2']))
                    pdf_elements.append(Paragraph('Numbered Roads', pdf_styles['Heading2']))
                    pdf_elements.append(Paragraph('   ', pdf_styles['Heading2']))

            if first_letter == 'A' and flag == '':
                flag = '0'
                pdf_elements.append(Paragraph('Named Roads', pdf_styles['Heading2']))
                pdf_elements.append(Paragraph('-- A --', pdf_styles['Heading2']))
                pdf_elements.append(Paragraph('   ', pdf_styles['Heading2']))

            elif first_letter.isalpha() and first_letter != "A" and first_letter != current_letter\
                    and name_strip[0:3] != 'CR ' and name_strip[0:3] != 'FM ':
                current_letter = first_letter
                pdf_elements.append(Paragraph(' ', pdf_styles['Normal']))
                pdf_elements.append(Paragraph('-- %s --' % first_letter, pdf_styles['Heading2']))
                pdf_elements.append(Paragraph(' ', pdf_styles['Heading2']))
```

PYTHON BREAKDOWN

- create_index, continued

```
            data = [[row[0], row[1], row[2]]]

            style = [('VALIGN', (0, -1), (-1, -1), 'CENTER')]
            s = getSampleStyleSheet()
            s = s['Normal']
            s.fontSize = 6
            s.fontName = 'Courier'

            data2 = [[Paragraph(cell, s) for cell in row] for row in data]
            t = Table(data2, colWidths=[1.25 * inch, 0.55 * inch, 1.2 * inch])
            t.setStyle(style)
            if row[1] != '' and row[2] != '':
                pdf_elements.append(t)

column1 = Frame(0.67 * inch, 0.5 * inch, 3 * inch, 7.2 * inch)
column2 = Frame(4 * inch, 0.5 * inch, 3 * inch, 7.2 * inch)
column3 = Frame(7.33 * inch, 0.5 * inch, 3 * inch, 7.2 * inch)

pdf_elements.append(NextPageTemplate('ThreeColumns'))
pdf_elements.append(PageBreak())

doc.addPageTemplates(PageTemplate(id='ThreeColumns', frames=[column1, column2, column3],
                                  onPage=index_template))

doc.build(pdf_elements)
del cursor
index_finish = time.time() - index_start
notes('Street index PDF created, elapsed time: {0}'.format(time.strftime('%H:%M:%S', time.gmtime(index_finish))))
return
```

PYTHON LIBRARIES USED

- PyPDF2
- fuzzy
- pyexcel
- openpyxl
- selenium
- reportlab

DEMONSTRATION

QUESTIONS

NEIL ROSE, GISP

- 972-547-7427
- nrose@mckinneytexas.org

PARKER JONES

- 972-547-7421
- pjones@mckinneytexas.org