This script is designed to convert a JSON file (no_tax.json) containing spatial and attribute data into a shapefile (notax_fc1.shp). It uses Python's json library for reading and processing the JSON file and the arcpy library from ArcGIS Pro to create and manage the shapefile.

1. Load the JSON File
    - The json.load method reads the content of the no_tax.json file and loads it into a Python dictionary called tax_json.
    - The JSON structure typically includes metadata (describing the fields) and data (rows of attributes and geometries).
    - Key components:
        - tax_json['meta']['view']['columns']: Describes the fields in the dataset.
        - tax_json['data']: Contains the rows of data, including geometry (spatial information) and attributes.

```
[1]
import json
with open('no_tax.json', 'r') as file:
    tax_json = json.load(file)
```

2. Display Field Names for Verification
    - Prints the field names from the JSON file for verification.
    - This step ensures you correctly identify field names and their structure before creating corresponding fields in the shapefile.

```
fields = tax_json['meta']['view']['columns']
for field in fields:
    print(field['name'])
```

3. Convert Geometry Data
    - Spatial data in the JSON is often stored in Well-Known Text (WKT) format, which is a standard textual representation of geometric shapes.
    - The `arcpy.FromWKT` function converts these WKT strings into ArcPy geometry objects that can be written into a shapefile.

    The geometry data is located in the 8th column (`row[8]`) of the JSON's data rows. Modify this index if your JSON structure differs.

```python
import arcpy
arcpy.FromWKT(tax_json['data'][8][8])
for row in tax_json['data']:
    row[8] = arcpy.FromWKT(row[8])
for row in tax_json['data']:
    print(row)
```

4. Define Workspace and Create Shapefile
    - workspace is the folder where the shapefile will be created.
    - fcname is the name of the shapefile.
    - The script checks if the shapefile already exists using arcpy.Exists. If it does, it deletes it to avoid errors.
    - A new shapefile is created using arcpy.management.CreateFeatureclass with:
        - Geometry type: POLYGON (can be adjusted based on your data).
        - Spatial reference: EPSG 4236 (WGS 84).

```python
import os
fcname = 'notax_fc1.shp'
workspace = r'D:\project1'

fc_fullname= os.path.join(workspace,fcname)
if arcpy.Exists(fc_fullname):
    arcpy.management.Delete(fc_fullname)

arcpy.management.CreateFeatureclass(out_path= workspace, out_name=fcname, geometry_type='POLYGON', spatial_reference=4236)
```

5. Inspect Existing Fields
   Lists the existing fields in the newly created shapefile to ensure no conflicts arise when adding new fields.

```python
desc = arcpy.da.Describe(fc_fullname)
for field in desc['fields']:
    print(field.name)
```

6. Prepare Field Names and Types
    - Filters out the the_geom field (likely redundant, as geometry is handled separately).
    - Sets field types (e.g., TEXT, LONG) for each field.
    - Truncates field names to 10 characters to comply with shapefile conventions.
    - Replaces spaces and periods in field names with underscores to ensure compatibility

```python
fields = tax_json['meta']['view']['columns']
fields = [field for field in fields if field['name'] != 'the_geom']
for field in fields:
    print(field['name'])
field_type = ['TEXT', 'TEXT','LONG', 'LONG', 'TEXT','LONG','TEXT', 'TEXT','TEXT', 'TEXT','TEXT', 'TEXT','TEXT']

```

```
field_names = []
for ind, field in enumerate(fields):
    name = field['name']
    if name == 'the_geom':
        continue
    if name.lower() == 'id':
        name = f'id_{ind}'
    max_len = min(10, len(name))
    name = name[:max_len]
    field_names.append(name)
field_names = [field.replace(" ", "_") for field in field_names]
field_names = [field.replace(".", "_") for field in field_names]
for ind, field_name in enumerate(field_names):
    arcpy.management.AddField(fc_fullname, field_name=field_name, field_type=field_type[ind])
field_names.append('SHAPE@')
```

7. Add Fields to the Shapefile
    - Iterates through the processed field_names and field_type.
    - Adds each field to the shapefile using arcpy.management.AddField.

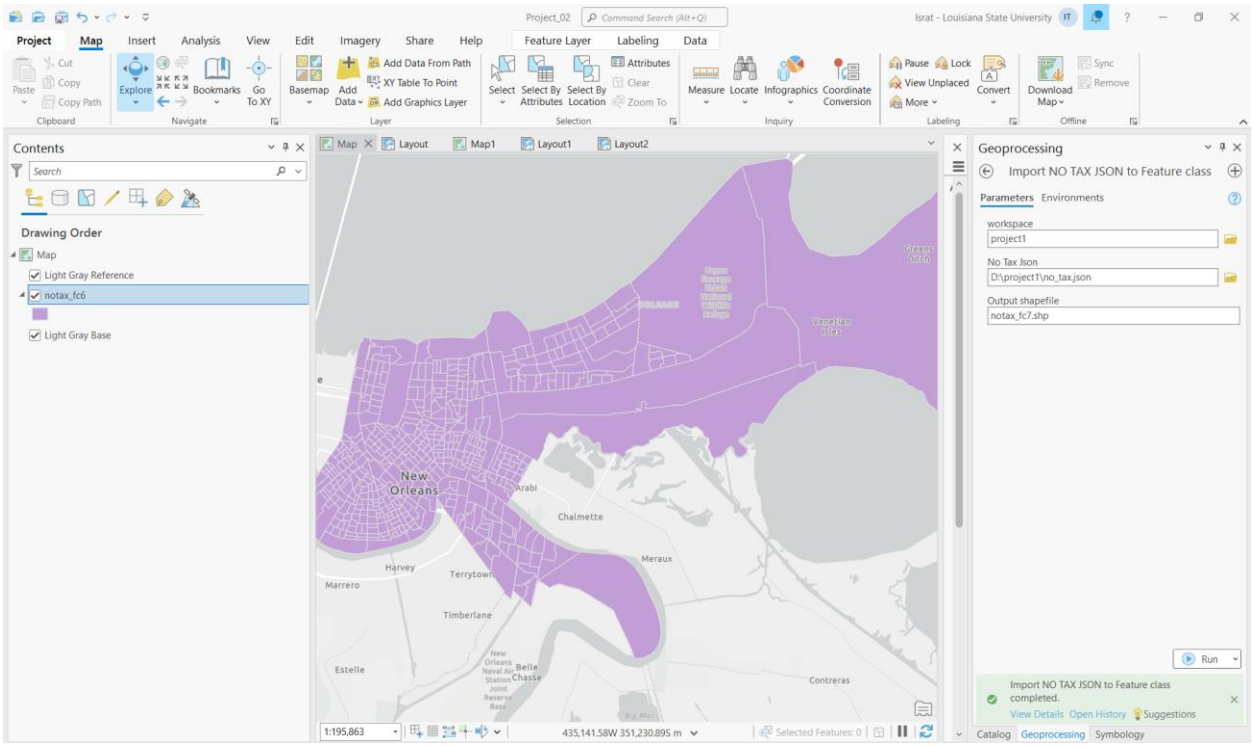
```
for ind, field_name in enumerate(field_names):
    arcpy.management.AddField(fc_fullname, field_name=field_name, field_type=field_type[ind])
field_names.append('SHAPE@')
```

8. Populate the shapefile

- Inserts rows of data into the shapefile.
- Add SHAPE@ to field_names for handling geometry.
- For each row in the JSON:
    - Exclude the geometry column during attribute processing.
    - Append the geometry object to the row.
    - Insert the row into the shapefile using arcpy.da.InsertCursor.

```
with arcpy.da.InsertCursor(fc_fullname, field_names=field_names) as cursor:
        for row in tax_json['data']:
            new_row = []
            for ind, value in enumerate(row):
                if ind == 8:
                    continue
                if value == None:
                    value = ""
                new_row.append(value)
            new_row.append(row[8])
            cursor.insertRow(new_row)
```

9. Finally developing a toolbox and use it in ArcPro


Project_02
Command Search (Alt+Q)
Israt - Louisiana State University
Project
Map
Insert
Analysis
View
Edit
Imagery
Share
Help
Feature Layer
Labeling
Data
Paste
Cut
Copy
Copy Path
Clipboard
Explore
Bookmarks
Go To XY
Navigate
Basemap
Add Data
Add Data From Path
XY Table To Point
Add Graphics Layer
Layer
Select
Select By Attributes
Select By Location
Attributes
Clear
Zoom To
Selection
Measure
Locate
Infographics
Coordinate Conversion
Inquiry
Pause
Lock
View Unplaced
More
Convert
Labeling
Download Map
Sync
Remove
Offline
Contents
Search
Drawing Order
Map
Light Gray Reference
notax_fc6
Light Gray Base
Map
Layout
Map1
Layout1
Layout2
New Orleans
Arabi
Chalmette
Meraux
Harvey
Terrytown
Marrero
Timberlane
Estelle
Belle Chasse
Contreras
Venetian Isles
1:195,863
435,141.58W 351,230.89S m
Selected Features: 0
Geoprocessing
Import NO TAX JSON to Feature class
Parameters
Environments
workspace
project1
No Tax Json
D:\project1\no_tax.json
Output shapefile
notax_fc7.shp
Run
Import NO TAX JSON to Feature class completed.
View Details
Open History
Suggestions
Catalog
Geoprocessing
Symbology