Geocoding Workflows in Python

Victor Bhattacharyya

2021 ESRI
DEVELOPER SUMMIT

Outline

- Publishing locators
- arcpy.geocoding.Locator object
  - .geocode()
  - .reverseGeocode()
  - .suggest() and .geocodeWithSuggest()
  - Properties
- Geocoding Geoprocessing Tools

Publishing & Sharing Locators

Creating Geocoding Services on ArcGIS Server & Enterprise

Benefits to creating a Geocoding Service

- Easy to share with colleagues
- Off-loads the processing to ArcGIS Server
- Useable in configurable apps & ArcGIS Pro

Publishing a locator to ArcGIS Server

```python
# path to the locator you want to publish. Do NOT include .loc at the end
locator_path = r"C:\data\locators\RedlandsLocator"

# path to the AGS connection file. Do NOT include .ags at the end
ags_connection = r"C:\ags_connections\servername"

# the service name you want to publish it as
service_name = "RedlandsLocator"

current_dir = os.getcwd()

# put the sd draft files & sd files in the current working directory
sddraft_file = os.path.join(current_dir, str(service_name) + ".sddraft")
sd_file = os.path.join(current_dir, str(service_name) + ".sd")
summary = "Address locator"
tags = "address, locator, geocode"

# Create the sd draft file
analyze_messages = arcpy.CreateGeocodeSDDraft(locator_path, sddraft_file, service_name,
                                              connection_file_path=ags_connection, copy_data_to_server=True,
                                              summary=summary, tags=tags, max_result_size=20,
                                              max_batch_size=500, suggested_batch_size=150, overwrite_existing_service=True)

# stage and upload the service if the sddraft analysis did not contain errors
if analyze_messages['errors'] == {}:
    #try:
    # Execute StageService to convert sddraft file to a service definition (sd) file
    arcpy.server.StageService(sddraft_file, sd_file)
    # Execute UploadServiceDefinition to publish the service definition file as a service
    arcpy.server.UploadServiceDefinition(sd_file, ags_connection)
    print("The geocode service was successfully published")
```

Publishing a locator to ArcGIS Server

```python
# path to the locator you want to publish. Do NOT include .loc at the end
locator_path = r"C:\data\locators\RedlandsLocator"

# path to the AGS connection file. Do NOT include .ags at the end
ags_connection = r"C:\ags_connections\servername"

# the service name you want to publish it as
service_name = "RedlandsLocator"

current_dir = os.getcwd()

# put the sd draft files & sd files in the current working directory
sddraft_file = os.path.join(current_dir, str(service_name) + ".sddraft")
sd_file = os.path.join(current_dir, str(service_name) + ".sd")
summary = "Address locator"
tags = "address, locator, geocode"

# Create the sd draft file
analyze_messages = arcpy.CreateGeocodeSDDraft(locator_path, sddraft_file, service_name,
                                              connection_file_path=ags_connection, copy_data_to_server=True,
                                              summary=summary, tags=tags, max_result_size=20,
                                              max_batch_size=500, suggested_batch_size=150, overwrite_existing_service=True)

# stage and upload the service if the sddraft analysis did not contain errors
if analyze_messages['errors'] == {}:
    #try:
    # Execute StageService to convert sddraft file to a service definition (sd) file
    arcpy.server.StageService(sddraft_file, sd_file)
    # Execute UploadServiceDefinition to publish the service definition file as a service
    arcpy.server.UploadServiceDefinition(sd_file, ags_connection)
    print("The geocode service was successfully published")
```

Publishing a locator to ArcGIS Server

```python
# path to the locator you want to publish. Do NOT include .loc at the end
locator_path = r"C:\data\locators\RedlandsLocator"

# path to the AGS connection file. Do NOT include .ags at the end
ags_connection = r"C:\ags_connections\servername"

# the service name you want to publish it as
service_name = "RedlandsLocator"

current_dir = os.getcwd()

# put the sd draft files & sd files in the current working directory
sddraft_file = os.path.join(current_dir, str(service_name) + ".sddraft")
sd_file = os.path.join(current_dir, str(service_name) + ".sd")
summary = "Address locator"
tags = "address, locator, geocode"

# Create the sd draft file
analyze_messages = arcpy.CreateGeocodeSDDraft(locator_path, sddraft_file, service_name,
                                              connection_file_path=ags_connection, copy_data_to_server=True,
                                              summary=summary, tags=tags, max_result_size=20,
                                              max_batch_size=500, suggested_batch_size=150, overwrite_existing_service=True)

# stage and upload the service if the sddraft analysis did not contain errors
if analyze_messages['errors'] == {}:
    #try:
    # Execute StageService to convert sddraft file to a service definition (sd) file
    arcpy.server.StageService(sddraft_file, sd_file)
    # Execute UploadServiceDefinition to publish the service definition file as a service
    arcpy.server.UploadServiceDefinition(sd_file, ags_connection)
    print("The geocode service was successfully published")
```

Sharing a locator to ArcGIS Enterprise

```python
# path to the locator you want to publish. Do NOT include .loc at the end
locator_path = r"C:\data\locators\SanDiegoLocator"

# the service name you want to publish it as
service_name = "SanDiegoLocator"

current_dir = os.getcwd()

# put the sd draft files & sd files in the current working directory
sddraft_file = os.path.join(current_dir, str(service_name) + ".sddraft")
sd_file = os.path.join(current_dir, str(service_name) + ".sd")
summary = "Address locator"
tags = "address, locator, geocode"

# The URL of the federated server you are publishing to
in_server = "https://machinename.domainname.com/server"
# Sign into the Portal that is federated with the in_server
arcpy.SignInToPortal("https://machinename.domainname.com/portal", "username", "password")

# Create the sd draft file
analyze_messages = arcpy.CreateGeocodeSDDraft(locator_path, sddraft_file, service_name,
                                              copy_data_to_server=True,
                                              summary=summary, tags=tags, max_result_size=20,
                                              max_batch_size=500, suggested_batch_size=150,
                                              overwrite_existing_service=True)

# Stage and upload the service if the sddraft analysis did not contain errors
if analyze_messages['errors'] == {}:
    try:
        # Execute StageService to convert sddraft file to a service definition
        # (sd) file
        arcpy.server.StageService(sddraft_file, sd_file)

        # Execute UploadServiceDefinition to publish the service definition
        # file as a service
        arcpy.server.UploadServiceDefinition(sd_file, in_server)
        print("The locator was successfully shared to Portal")
```

Sharing a locator to ArcGIS Enterprise

```python
# path to the locator you want to publish. Do NOT include .loc at the end
locator_path = r"C:\data\locators\SanDiegoLocator"

# the service name you want to publish it as
service_name = "SanDiegoLocator"

current_dir = os.getcwd()

# put the sd draft files & sd files in the current working directory
sddraft_file = os.path.join(current_dir, str(service_name) + ".sddraft")
sd_file = os.path.join(current_dir, str(service_name) + ".sd")
summary = "Address locator"
tags = "address, locator, geocode"

# The URL of the federated server you are publishing to
in_server = "https://machinename.domainname.com/server"
# Sign into the Portal that is federated with the in_server
arcpy.SignInToPortal("https://machinename.domainname.com/portal", "username", "password")

# Create the sd draft file
analyze_messages = arcpy.CreateGeocodeSDDraft(locator_path, sddraft_file, service_name,
                                              copy_data_to_server=True,
                                              summary=summary, tags=tags, max_result_size=20,
                                              max_batch_size=500, suggested_batch_size=150,
                                              overwrite_existing_service=True)

# Stage and upload the service if the sddraft analysis did not contain errors
if analyze_messages['errors'] == {}:
    try:
        # Execute StageService to convert sddraft file to a service definition
        # (sd) file
        arcpy.server.StageService(sddraft_file, sd_file)

        # Execute UploadServiceDefinition to publish the service definition
        # file as a service
        arcpy.server.UploadServiceDefinition(sd_file, in_server)
        print("The locator was successfully shared to Portal")
```

Sharing a locator to ArcGIS Enterprise

```python
# path to the locator you want to publish. Do NOT include .loc at the end
locator_path = r"C:\data\locators\SanDiegoLocator"

# the service name you want to publish it as
service_name = "SanDiegoLocator"

current_dir = os.getcwd()

# put the sd draft files & sd files in the current working directory
sddraft_file = os.path.join(current_dir, str(service_name) + ".sddraft")
sd_file = os.path.join(current_dir, str(service_name) + ".sd")
summary = "Address locator"
tags = "address, locator, geocode"

# The URL of the federated server you are publishing to
in_server = "https://machinename.domainname.com/server"
# Sign into the Portal that is federated with the in_server
arcpy.SignInToPortal("https://machinename.domainname.com/portal", "username", "password")

# Create the sd draft file
analyze_messages = arcpy.CreateGeocodeSDDraft(locator_path, sddraft_file, service_name,
                                              copy_data_to_server=True,
                                              summary=summary, tags=tags, max_result_size=20,
                                              max_batch_size=500, suggested_batch_size=150,
                                              overwrite_existing_service=True)

# Stage and upload the service if the sddraft analysis did not contain errors
if analyze_messages['errors'] == {}:
    try:
        # Execute StageService to convert sddraft file to a service definition
        # (sd) file
        arcpy.server.StageService(sddraft_file, sd_file)

        # Execute UploadServiceDefinition to publish the service definition
        # file as a service
        arcpy.server.UploadServiceDefinition(sd_file, in_server)
        print("The locator was successfully shared to Portal")
```

Documentation about Sharing a Locator

Python sample: https://esriurl.com/PythonPublishLocator

Syntax

```
CreateGeocodeSDDraft (loc_path, out_sddraft, service_name, {server_type},
{connection_file_path}, {copy_data_to_server}, {folder_name}, {summary}, {tags},
{max_result_size}, {max_batch_size}, {suggested_batch_size},
{supported_operations}, {overwrite_existing_service})
```

arcpy.geocoding.Locator

A new way to access & use locators in Python

arcpy.geocoding.Locator()

- **Locator object introduced new at ArcGIS Pro 2.6**

- **Three ways to create a Python locator object**
  - **From a path to your locator on your machine**
  - **From an AGS connection file on your machine**
  - **From a URL**

- **Python locators support:**
  - **.geocode()**
  - **.reverseGeocode()**
  - **.suggest() and .geocodeWithSuggest()**
  - **.properties (getting & setting)**

Create a locator from a path

```
### path example
locator_path = r"D:\data\locators\MyCityLocator"
locator = arcpy.geocoding.Locator(locator_path)
output = locator.geocode("Your home address", True)
print(output)
```

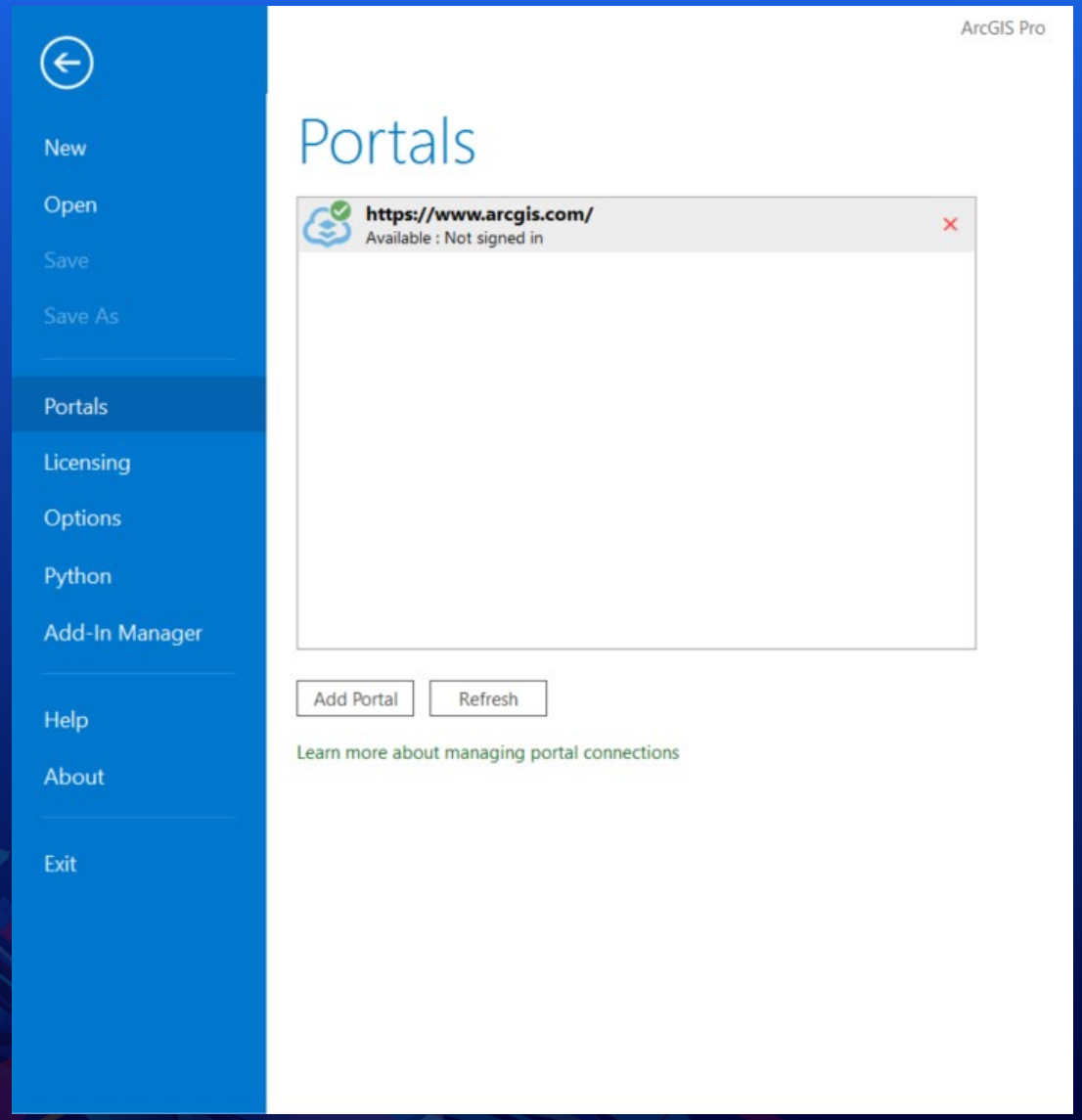
Create a locator from an AGS connection

```
### AGS connection example
locator_path = r"C:\ags_connection_files\agsFileName.ags/CAN"
locator = arcpy.geocoding.Locator(locator_path)
output = locator.geocode("Quebec", True)
print(output)
```

Create a locator from a URL

```
### URL example
arcpy.SignInToPortal("https://machinename.domainname.com/portal","username","password")
url = "https://machinename.domainname.com/server/rest/services/FolderName/ServiceName/GeocodeServer/ItemName"
locator = arcpy.geocoding.Locator(url)
output = locator.geocode("New York City", False)
print(output)
```

A note about Portal Sign-In



~~arcpy.SignInToPortal()~~

Using the World Geocoding Service

```
### World Geocoding service example
arcpy.SignInToPortal("https://www.arcgis.com/", username='username', password='password')
url = "https://geocode.arcgis.com/arcgis/rest/services/World/GeocodeServer/ArcGIS World Geocoding Service"
locator = arcpy.geocoding.Locator(url)
output = locator.geocode("380 New York St. Redlands, CA 92373", True)
print(output)
```

Using locator.geocode()

```
### World Geocoding service example
arcpy.SignInToPortal("https://www.arcgis.com/", username='username', password='password')
url = "https://geocode.arcgis.com/arcgis/rest/services/World/GeocodeServer/ArcGIS World Geocoding Service"
locator = arcpy.geocoding.Locator(url)
candidates = locator.geocode("Disneyland, CA", True)
print(candidates)
```

Using locator.reverseGeocode()

```
### World Geocoding service example
arcpy.SignInToPortal("https://www.arcgis.com/", username='username', password='password')
url = "https://geocode.arcgis.com/arcgis/rest/services/World/GeocodeServer/ArcGIS World Geocoding Service"
locator = arcpy.geocoding.Locator(url)

###Create input PointGeometry object
spatial_reference = arcpy.SpatialReference(4326)
point = arcpy.Point(-117.196, 34.058)
point_geometry = arcpy.PointGeometry(point, spatial_reference)

reverse_candidates = locator.reverseGeocode(point_geometry, True)
print(reverse_candidates)
```

Using locator.suggest()

```
### World Geocoding service example
arcpy.SignInToPortal("https://www.arcgis.com/", username='username', password='password')
url = "https://geocode.arcgis.com/arcgis/rest/services/World/GeocodeServer/ArcGIS World Geocoding Service"
locator = arcpy.geocoding.Locator(url)
suggest_candidates = locator.suggest("Disney")
geocode_candidates = locator.geocodeWithSuggest(suggest_candidates[0], True)
print(geocode_candidates)
```

Using locator.properties

- locator.spatialReference
- locator.supportedRoles
- locator.supportedCountries
- locator.matchOutOfRange*
- locator.minMatchScore*
- locator.categories*
- locator.reverseGeocodeFeatureTypes*
- locator.numberOfThreads*
- locator.maxCandidates*
- locator.defaultSuggestCandidates*
- locator.maxSuggestCandidates*
- locator.minCandidateScore*
- locator.capabilities
- locator.multilineInputFields
- locator.singlelineInputField

- For more info: https://esriurl.com/PythonLocator

** denotes the property can be set.*

Setting a property

```python
### properties example
locator_path = r"D:\data\locators\Atlanta"
locator = arcpy.geocoding.Locator(locator_path)
min_score = locator.minCandidateScore
print(min_score)
output = locator.geocode("1171 Peidmont blvd atlanta, ga", True)
print(output)

### raise the minCandidateScore
locator.minCandidateScore = 95
print(locator.minCandidateScore)

output = locator.geocode("1171 Peidmont blvd atlanta, ga", True)
print(output)
locator.minCandidateScore = 95
print(locator.minCandidateScore)
```

```
60
[{'Match_addr': '1171 Piedmont Ave NE, Atlanta, GA, 30309', 'Status': 'M', 'Score': 94.24, 'LongLabel': '1171 Piedmont A
ve NE, Atlanta, GA, 30309', 'ShortLabel': '1171 Piedmont Ave NE', 'Addr_type': 'StreetAddress', 'Type': '', 'PlaceName':
 '', 'Place_addr': '1171 Piedmont Ave NE, Atlanta, GA, 30309', 'Phone': '', 'URL': '', 'Rank': 20.0, 'AddBldg': '', 'Add
95
[]
95
```

Setting a property vs .updateLocator()

```
### raise the minCandidateScore for this Python run
locator.minCandidateScore = 95
print(locator.minCandidateScore)
```

```
### raise the minCandidateScore for this Locator until it is set again
locator.minCandidateScore = 95
locator.updateLocator()
```

[{'Shape': <Point (-71.21929999999998, 46.812730000000045, #, #)>, 'Status': 'T', 'Score': 100.0, 'Match_addr': 'Québec'
, 'LongLabel': 'Québec, CAN', 'ShortLabel': 'Québec', 'Addr_type': 'Locality', 'Type': 'City', 'PlaceName': 'Québec', 'P
lace_addr': 'Québec', 'Phone': '', 'URL': '', 'Rank': 3.0, 'AddBldg': '', 'AddNum': '', 'AddNumFrom': '', 'AddNumTo': ''
, 'AddRange': '', 'Side': '', 'StPreDir': '', 'StPreType': '', 'StName': '', 'StType': '', 'StDir': '', 'BldgType': '',
'BldgName': '', 'LevelType': '', 'LevelName': '', 'UnitType': '', 'UnitName': '', 'SubAddr': '', 'StAddr': '', 'Block':
'', 'Sector': '', 'Nbrhd': '', 'District': '', 'City': 'Québec', 'MetroArea': 'Communauté Métropolitaine de Québec', 'Su
bregion': 'Québec', 'Region': 'Québec', 'RegionAbbr': 'QC', 'Territory': '', 'Zone': '', 'Postal': '', 'PostalExt': '',
'Country': 'CAN', 'LangCode': 'FRE', 'Distance': 0.0, 'X': -71.21929999999998, 'Y': 46.812730000000045, 'DisplayX': -71.
21929999999998, 'DisplayY': 46.812730000000045, 'Xmin': -71.38729999999998, 'Xmax': -71.05129999999997, 'Ymin': 46.64473
0000000045, 'Ymax': 46.980730000000044, 'ExInfo': ''}, {'Shape': <Point (-71.74874224599995, 53.382603420000066, #, #)>,
 'Status': 'T', 'Score': 100.0, 'Match_addr': 'Québec', 'LongLabel': 'Québec, CAN', 'ShortLabel': 'Québec', 'Addr_type':
 'Locality', 'Type': 'State or Province', 'PlaceName': 'Québec', 'Place_addr': 'Québec', 'Phone': '', 'URL': '', 'Rank':
 6.0, 'AddBldg': '', 'AddNum': '', 'AddNumFrom': '', 'AddNumTo': '', 'AddRange': '', 'Side': '', 'StPreDir': '', 'StPreT
ype': '', 'StName': '', 'StType': '', 'StDir': '', 'BldgType': '', 'BldgName': '', 'LevelType': '', 'LevelName': '', 'Un
itType': '', 'UnitName': '', 'SubAddr': '', 'StAddr': '', 'Block': '', 'Sector': '', 'Nbrhd': '', 'District': '', 'City'
: '', 'MetroArea': '', 'Subregion': '', 'Region': 'Québec', 'RegionAbbr': 'QC', 'Territory': '', 'Zone': '', 'Postal': '
', 'PostalExt': '', 'Country': 'CAN', 'LangCode': 'FRE', 'Distance': 0.0, 'X': -71.74874224599995, 'Y': 53.3826034200000
66, 'DisplayX': -71.74874224599995, 'DisplayY': 53.382603420000066, 'Xmin': -81.81274224599994, 'Xmax': -61.684742245999
95, 'Ymin': 43.318603420000066, 'Ymax': 63.446603420000066, 'ExInfo': ''}, {'Shape': <Point (-71.36076335999996, 46.8392

Using arcpy.geocoding. Locator to do geocoding

Victor

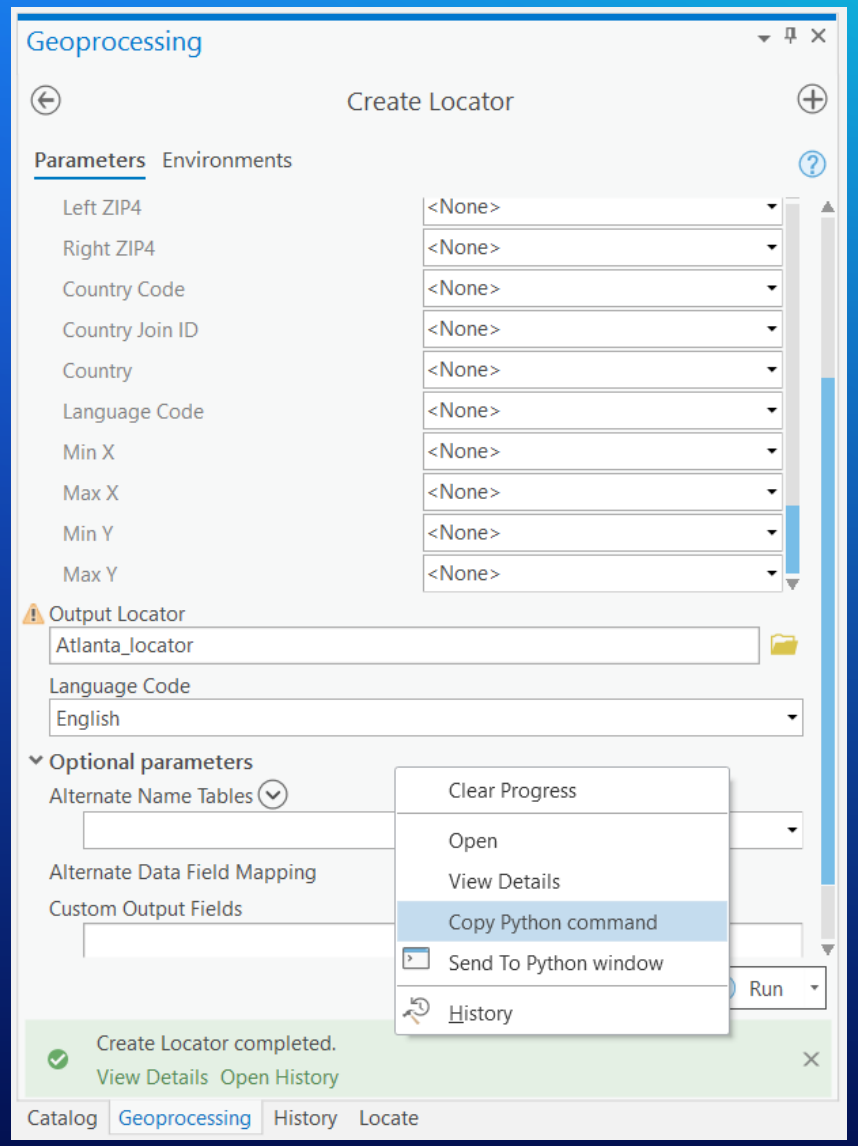
Geocoding using Geoprocessing Tools

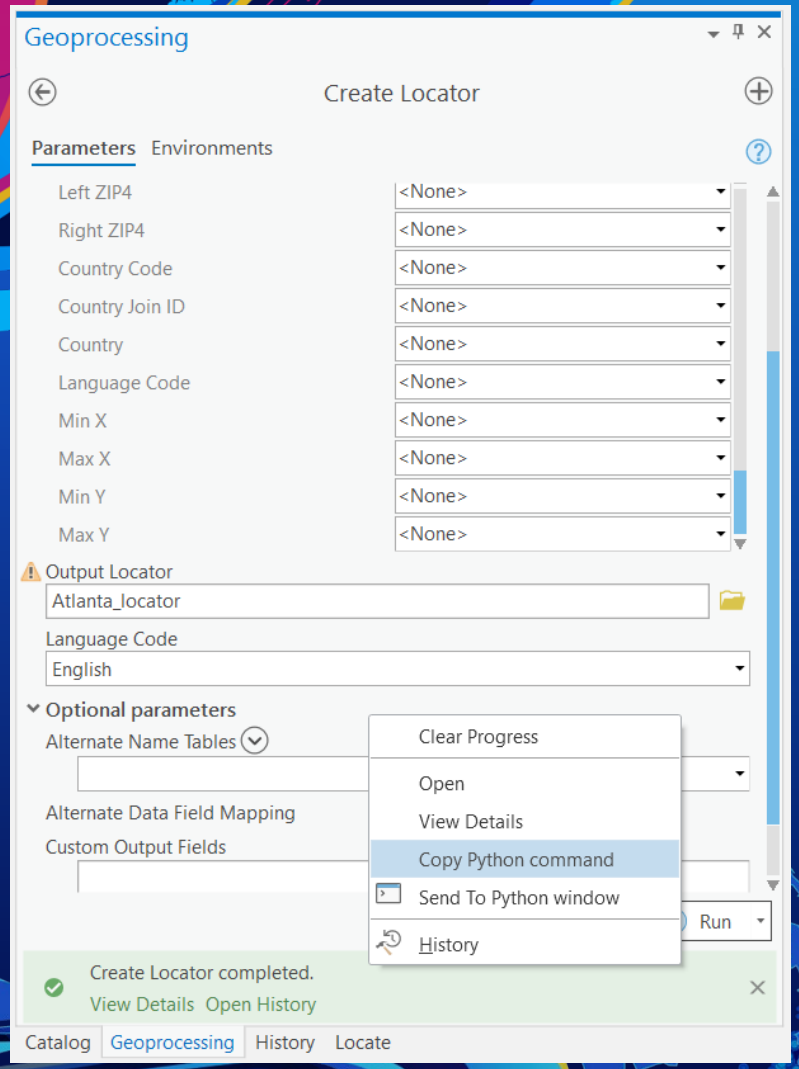
Copy Python Snippet from Pro

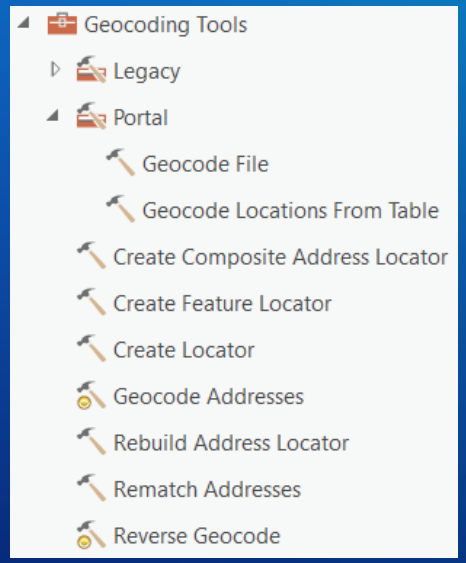
Geocoding Geoprocessing Tools in Python

- The best way to automate your GP processes is to:

1. Run the tool in ArcGIS Pro
2. Right-click -> “Copy Python command”
3. Paste into your favorite IDE
4. Edit the parameters so that they are full paths
5. Run your new Python script!







Geocoding GP Tools in Python

Victor

All code snippets available!

- https://esriurl.com/PythonGeocoding

- Includes:
  - Publishing & Sharing (https://esriurl.com/PythonPublishLocator)
  - arcpy.geocoding.locator module examples (https://esriurl.com/PythonLocator)

Please provide your feedback for this session by clicking on the session survey link directly below the video.

esri®

THE SCIENCE OF WHERE®

Copyright © 2021 Esri. All rights reserved.