An Introduction to Python for ArcGIS Pro

Instructor: Stephen Bond (TWDB/ACC)

What’s This Course All About?

- Introduction to programming with Python
- Accessing and exploring geospatial data with the ArcPy Python package
- Running geoprocessing tools and manipulating feature data with ArcPy
- Creating custom script tools

Python

- Free and open-source, cross-platform, general purpose programming language that supports a variety of programming paradigms
- Interpreted language - does not require a build step prior to running, but requires a program (python.exe) to be installed
- Used in ArcGIS ecosystem through ArcPy and ArcGIS API for Python

Why Bother?

- Automation of ArcGIS workflows
- Complex workflows can be easier in Python than in other options like Model Builder
- Access to non-Esri tools
- Robust error handling and logging
- Documentation / Comments

ArcGIS Python Environment

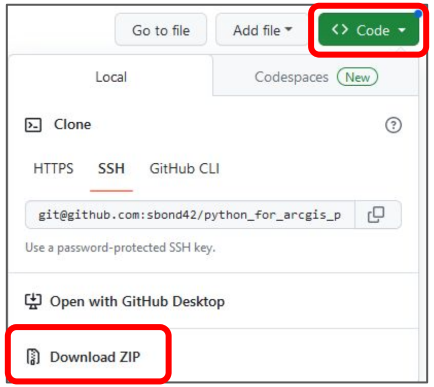
- Using Python with ArcGIS Pro requires ArcPy, which can only be used with the Python installation included with ArcGIS Pro
- Python 3
- Can be managed through ArcGIS Pro (Project > Package Manager)

Running Python

- Jupyter / ArcGIS Notebooks
- Python window in ArcGIS Pro
- IDLE (installed with ArcGIS Pro) or other Integrated Development Environment (IDE)
- Python Interactive Terminal / Python Command Prompt

Course Materials

- Notebook and sample data
- Download from:
  https://github.com/sbond42/python_for_arcgis_pro



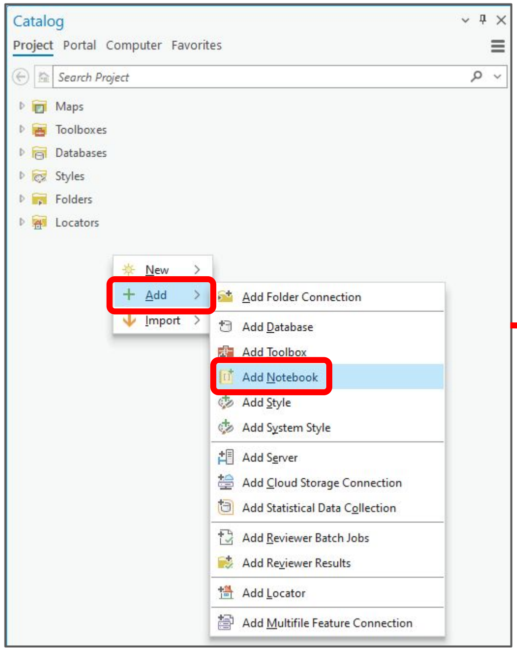
```
git@github.com:sbond42/python_for_arcgis_p
```

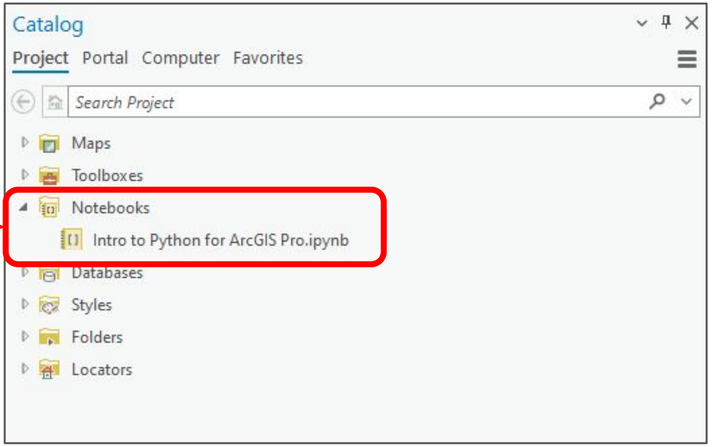
Use a password-protected SSH key.

Run Course Notebook

- Open ArcGIS Pro and show Catalog pane
- Right click in pane, select Add, the select Add Notebook
- Navigate to course folder and double-click “Intro to Python for ArcGIS Pro.ipynb”
- Double-click notebook in Catalog pane to open

Run Course Notebook



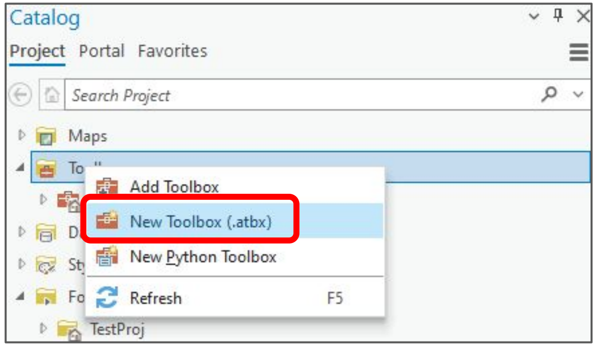


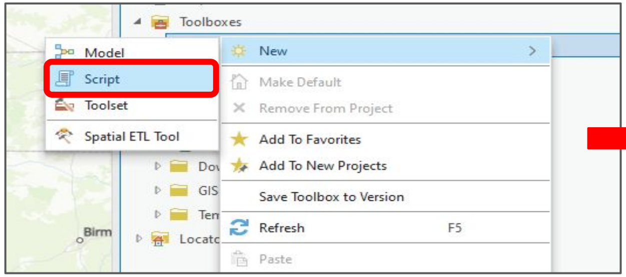
ArcGIS Pro Script Tools

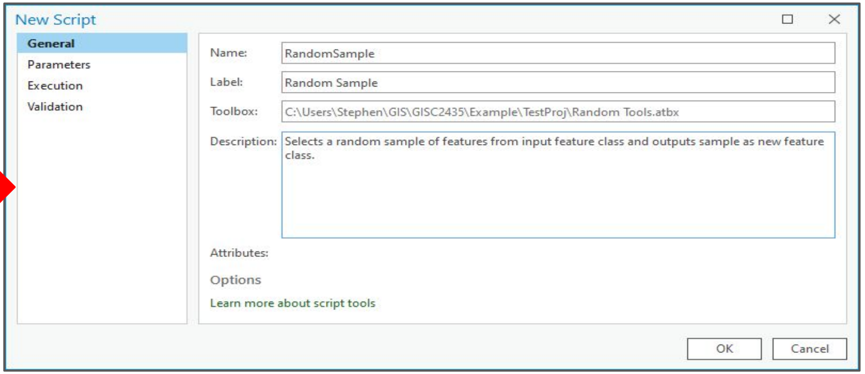
- Likely the best option if code is meant to be used with ArcGIS Pro / makes use of ArcPy
- Relatively easy to create, maintain, and share
- User-experience is generally the same as for a built-in Geoprocessing Tool
- Can be used in Model Builder

Creating a Script Tool

- Create new Toolbox
  - Call it “Random Tools” or similar
- Create new Script Tool in our new toolbox
  - Right-click on toolbox, click New, then click Script
  - Enter Name, Label, and Description






New Script

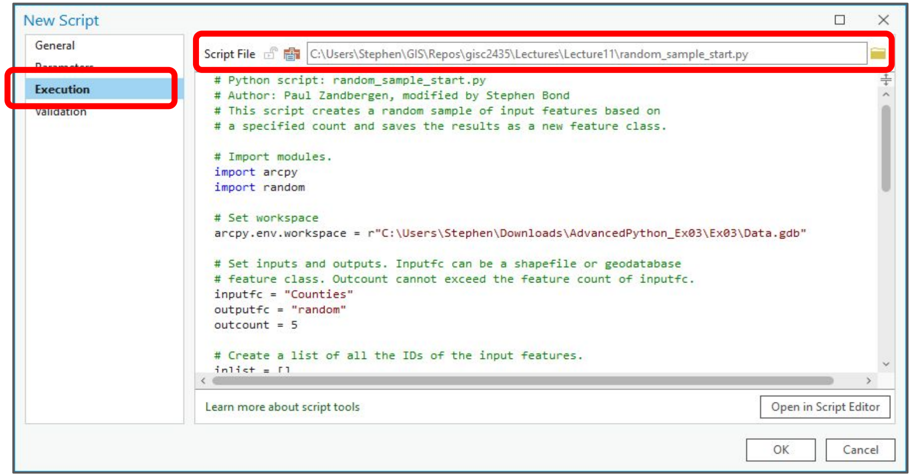
General
Parameters
Execution
Validation

Name: RandomSample
Label: Random Sample
Toolbox: C:\Users\Stephen\GIS\GISC2435\Example\TestProj\Random Tools.atbx
Description: Selects a random sample of features from input feature class and outputs sample as new feature class.
Attributes:
Options
Learn more about script tools

OK Cancel

Creating a Script Tool

- Specify the location of our script (random_sample_start.py) on the Execution tab
- Script should be kept with tool, but can be embedded
- Click OK

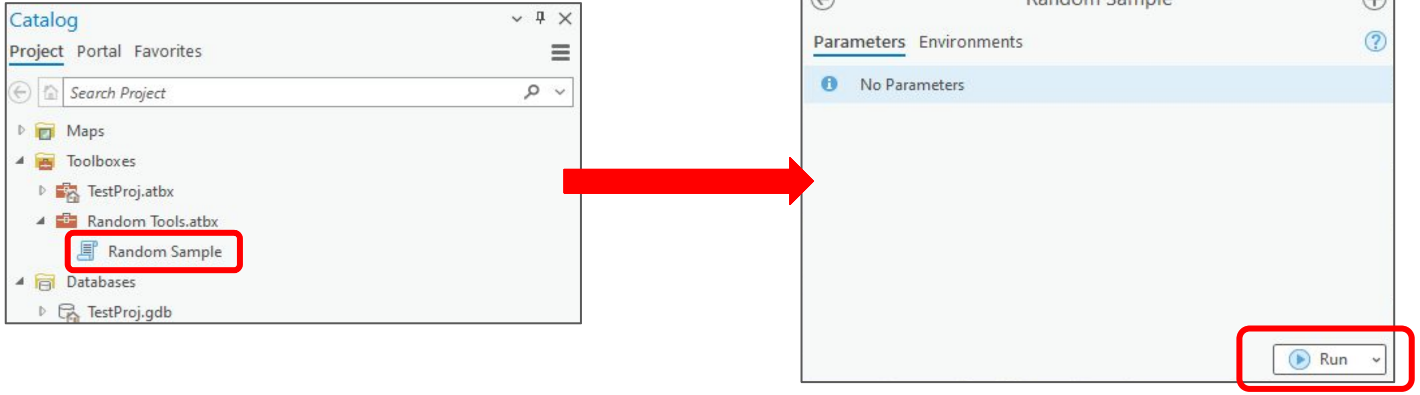


Script File C:\Users\Stephen\GIS\Repos\gisc2435\Lectures\Lecture11\random_sample_start.py

```
# Python script: random_sample_start.py
# Author: Paul Zandbergen, modified by Stephen Bond
# This script creates a random sample of input features based on
# a specified count and saves the results as a new feature class.

# Import modules.
import arcpy
import random

# Set workspace
arcpy.env.workspace = r"C:\Users\Stephen\Downloads\AdvancedPython_Ex03\Ex03\Data.gdb"

# Set inputs and outputs. Inputfc can be a shapefile or geodatabase
# feature class. Outcount cannot exceed the feature count of inputfc.
inputfc = "Counties"
outputfc = "random"
outcount = 5

# Create a list of all the IDs of the input features.
inlist = []
```

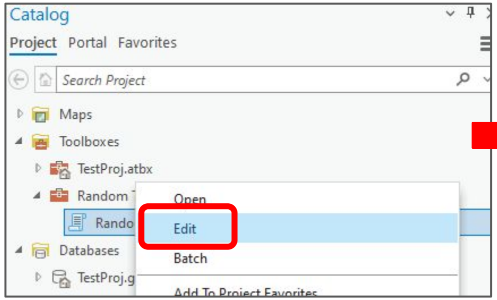
Creating a Script Tool

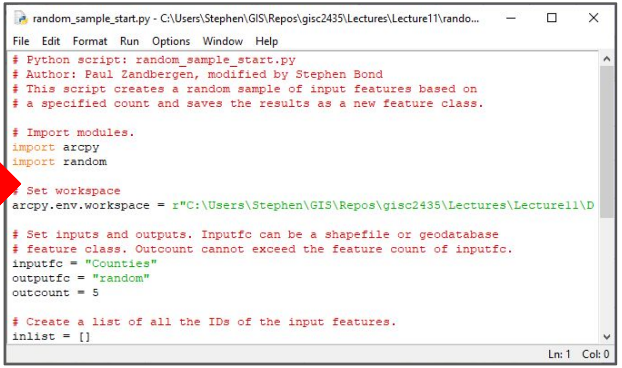
- Double-clicking on our new tool should bring up the familiar Geoprocessing Tool window
- Clicking Run should run the tool and produce output of 5 randomly selected features



Adding Parameters

- No parameters, so inputs, outputs and other values must be set directly in the code. This isn't very user-friendly
- Right-click on the Script Tool and select Edit to bring up the tool's code in an editor (IDLE by default)



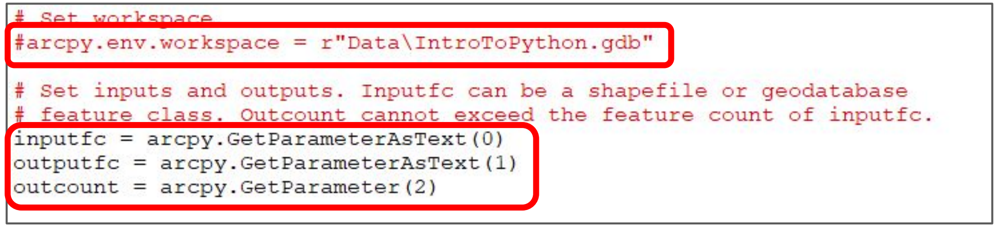

random_sample_start.py - C:\Users\Stephen\GIS\Repos\gisc2435\Lectures\Lecture11\rando...
File
Edit
Format
Run
Options
Window
Help
Ln: 1
Col: 0

```
# Python script: random_sample_start.py
# Author: Paul Zandbergen, modified by Stephen Bond
# This script creates a random sample of input features based on
# a specified count and saves the results as a new feature class.

# Import modules.
import arcpy
import random

# Set workspace
arcpy.env.workspace = r"C:\Users\Stephen\GIS\Repos\gisc2435\Lectures\Lecture11\D

# Set inputs and outputs. Inputfc can be a shapefile or geodatabase
# feature class. Outcount cannot exceed the feature count of inputfc.
inputfc = "Counties"
outputfc = "random"
outcount = 5

# Create a list of all the IDs of the input features.
inlist = []
```

Adding Parameters

- Parameters are set in the script with *arcpy.GetParameterAsText()* and *arcpy.GetParameter()*
  - The values passed into these functions correspond to the rows on the Script Tool Parameters tab
  - *GetParameterAsText* passes input to script as string
  - *GetParameter* passes input to script as object
- Make the following changes to the code and save:

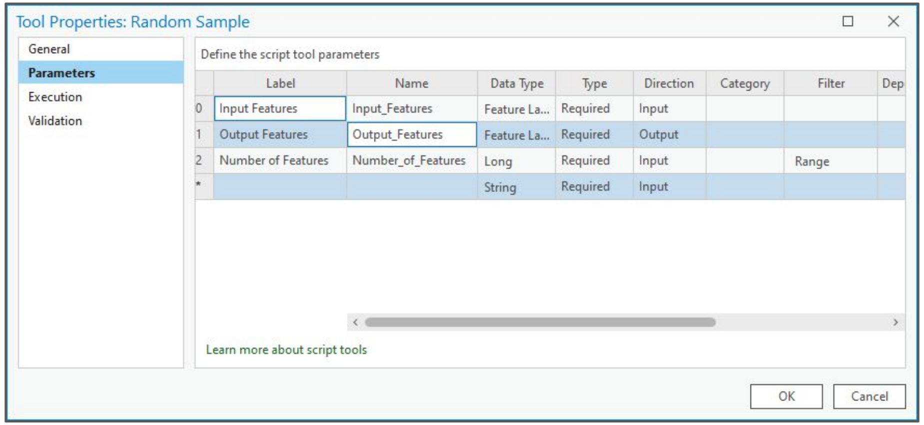
Comment out or delete



```
# Set workspace
#arcpy.env.workspace = r"Data\IntroToPython.gdb"

# Set inputs and outputs. Inputfc can be a shapefile or geodatabase
# feature class. Outcount cannot exceed the feature count of inputfc.
inputfc = arcpy.GetParameterAsText(0)
outputfc = arcpy.GetParameterAsText(1)
outcount = arcpy.GetParameter(2)
```

Adding Parameters

- Right-click on tool and select Properties
- Go to the Parameters tab
- The ID of each parameter line corresponds to the value passed in the GetParameter/GetParameterAsText functions in the script



Tool Properties: Random Sample

Define the script tool parameters

	Label	Name	Data Type	Type	Direction	Category	Filter	Dep
0	Input Features	Input_Features	Feature La...	Required	Input			
1	Output Features	Output_Features	Feature La...	Required	Output			
2	Number of Features	Number_of_Features	Long	Required	Input		Range	
*			String	Required	Input			

Adding Parameters

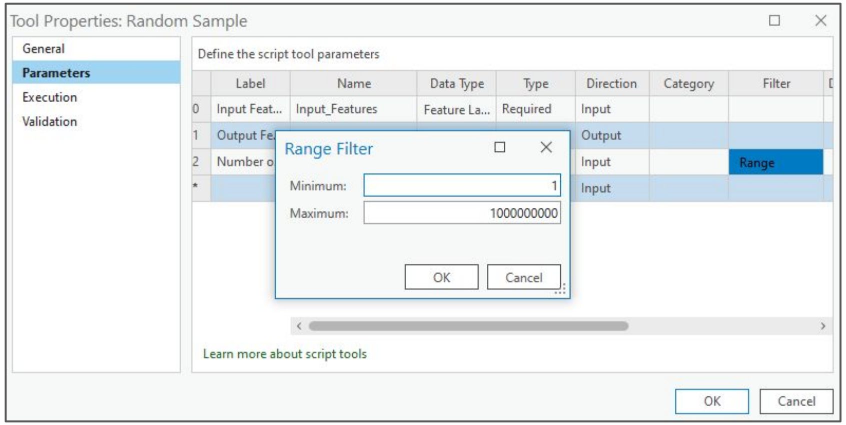
- Set parameters as follows:
- **Parameter 0** (this is *inputfc = arcpy.GetParameterAsText(0)* in the code*)*
    - Label: Input Features (label shown in tool window)
    - Name: Input_Features (auto-populated, for use in code)
    - Data Type: Feature Layer (can be layer in map or feature class)
    - Type: Required
    - Direction: Input

Adding Parameters

- **Parameter 1** (*outputfc = arcpy.GetParameterAsText(1)*)
    - Label: Output Features
    - Name: Output_Features
    - Data Type: Feature Layer
    - Type: Required
    - Direction: Output

Adding Parameters

- **Parameter 2** (*outcount = arcpy.GetParameter(2)*)
  - Label: Number of Features
  - Name: Number_of_Features
  - Data Type: Long (this is an integer value - will not run with if text or float values are input)
  - Type: Required
  - Direction: Input
  - Filter: Range



Tool Properties: Random Sample

Define the script tool parameters

	Label	Name	Data Type	Type	Direction	Category	Filter
0	Input Feat...	Input_Features	Feature La...	Required	Input		
1	Output Fe				Output		
2	Number o				Input		Range
*					Input		

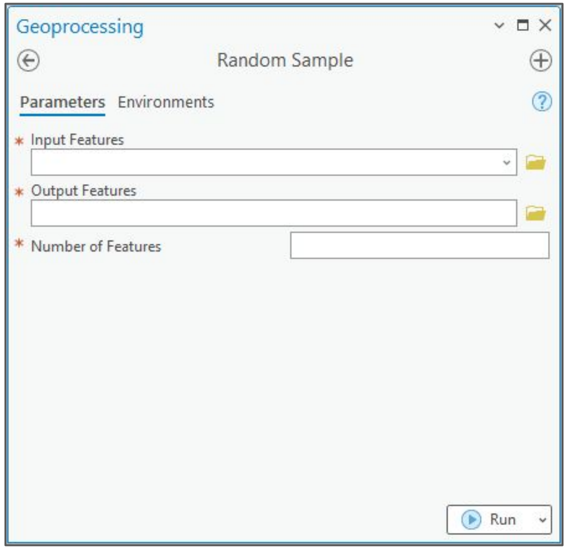
Range Filter

Minimum: 1

Maximum: 1000000000

Parameters Added

- Our tool window should now look like this:



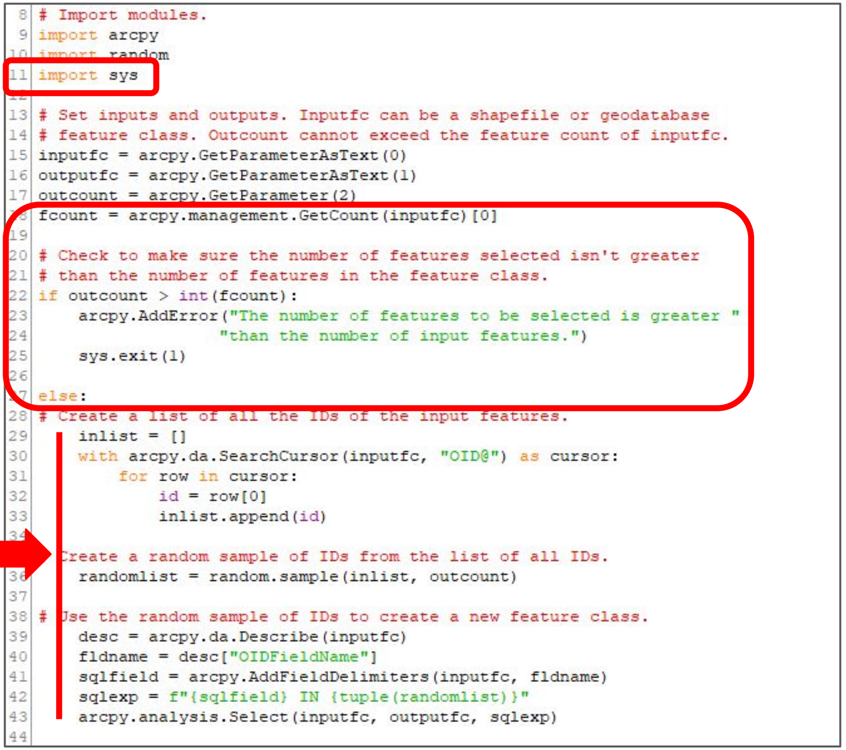
* Input Features
* Output Features
* Number of Features

Tool Messages

- Can’t just use print() to show information
- Can add messages to tool with the following:
    - arcpy.AddMessage()
    - arcpy.AddWarning()
    - arcpy.AddError()
- We used parameter filtering to limit what can be entered for number of features, but it doesn’t stop use from entering a number greater than the number of features in the input, so let’s fix that.

Tool Messages

- Make the following edits to the tool code and save:

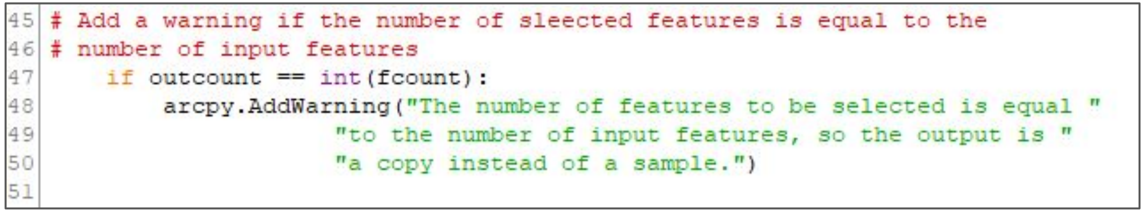


```
# Import modules.
import arcpy
import random
import sys

# Set inputs and outputs. Inputfc can be a shapefile or geodatabase
# feature class. Outcount cannot exceed the feature count of inputfc.
inputfc = arcpy.GetParameterAsText(0)
outputfc = arcpy.GetParameterAsText(1)
outcount = arcpy.GetParameter(2)
fcount = arcpy.management.GetCount(inputfc)[0]

# Check to make sure the number of features selected isn't greater
# than the number of features in the feature class.
if outcount > int(fcount):
    arcpy.AddError("The number of features to be selected is greater "
                   "than the number of input features.")
    sys.exit(1)

else:
# Create a list of all the IDs of the input features.
    inlist = []
    with arcpy.da.SearchCursor(inputfc, "OID@") as cursor:
        for row in cursor:
            id = row[0]
            inlist.append(id)

# Create a random sample of IDs from the list of all IDs.
    randomlist = random.sample(inlist, outcount)

# Use the random sample of IDs to create a new feature class.
    desc = arcpy.da.Describe(inputfc)
    fldname = desc["OIDFieldName"]
    sqlfield = arcpy.AddFieldDelimiters(inputfc, fldname)
    sqlexp = f"{sqlfield} IN {tuple(randomlist)}"
    arcpy.analysis.Select(inputfc, outputfc, sqlexp)
```

Indent everything under *else*

Tool Messages

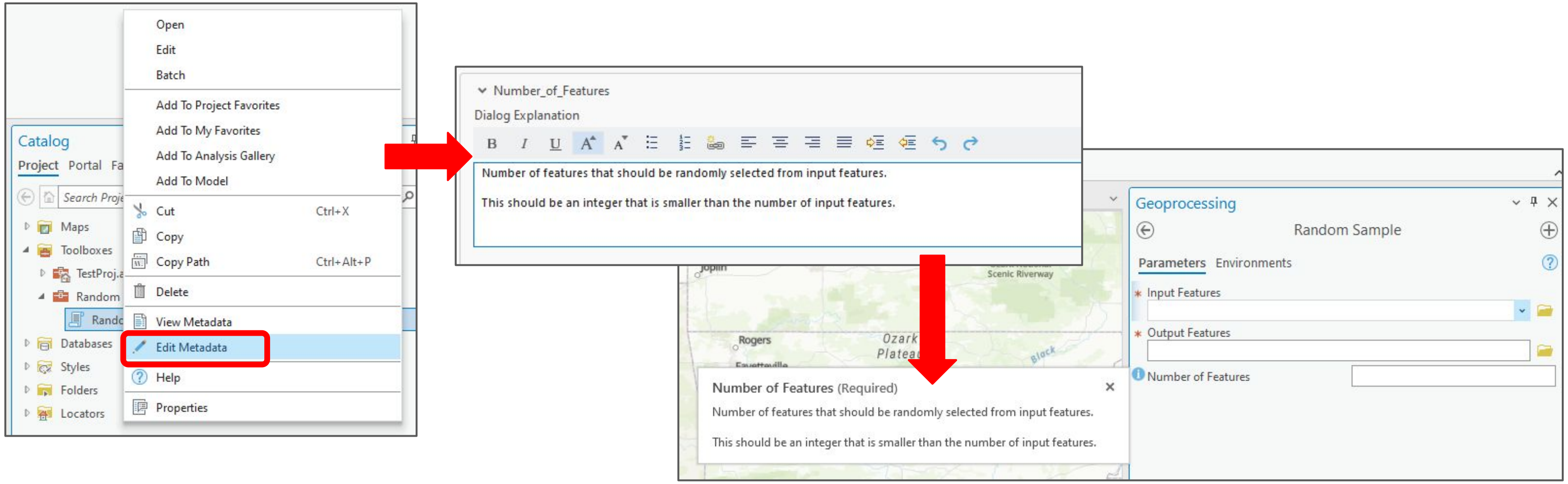
- We should also add a warning if the number of sample features is equal to the number of input features (the output just becomes a copy when this happens)
- Add the following to the end of the code, indented (since it’s part of the else statement)



```
# Add a warning if the number of sleected features is equal to the
# number of input features
    if outcount == int(fcount):
        arcpy.AddWarning("The number of features to be selected is equal "
                         "to the number of input features, so the output is "
                         "a copy instead of a sample.")

```

Tool Documentation

- Add tooltips and help documentation by right-clicking on tool and selecting Edit Metadata



Number of features that should be randomly selected from input features.

This should be an integer that is smaller than the number of input features.

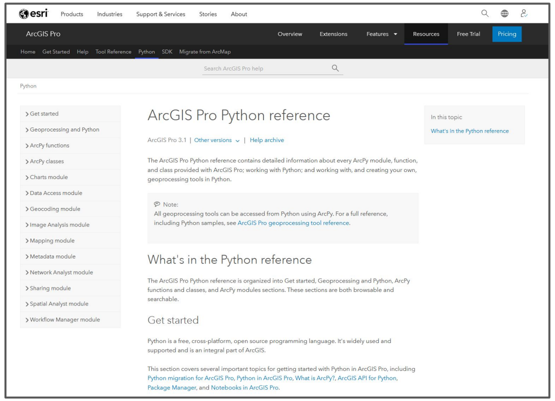
Number of Features (Required)

Number of features that should be randomly selected from input features.

This should be an integer that is smaller than the number of input features.

More Information

- Course folder contains “Running Python with ArcGIS Pro.pdf” document
- Esri documentation and books



- Get started
- Geoprocessing and Python
- ArcPy functions
- ArcPy classes
- Charts module
- Data Access module
- Geocoding module
- Image Analysis module
- Mapping module
- Metadata module
- Network Analyst module
- Sharing module
- Spatial Analyst module
- Workflow Manager module

ArcGIS Pro Python reference

ArcGIS Pro 3.1 | Other versions | Help archive

The ArcGIS Pro Python reference contains detailed information about every ArcPy module, function, and class provided with ArcGIS Pro; working with Python; and working with, and creating your own, geoprocessing tools in Python.

Note:
All geoprocessing tools can be accessed from Python using ArcPy. For a full reference, including Python samples, see ArcGIS Pro geoprocessing tool reference.

What's in the Python reference

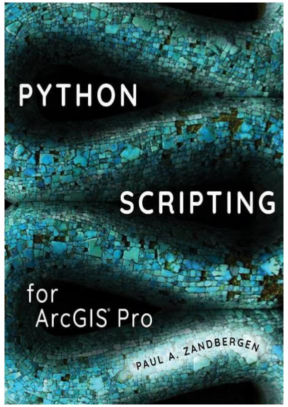
The ArcGIS Pro Python reference is organized into Get started, Geoprocessing and Python, ArcPy functions and classes, and ArcPy modules sections. These sections are both browsable and searchable.

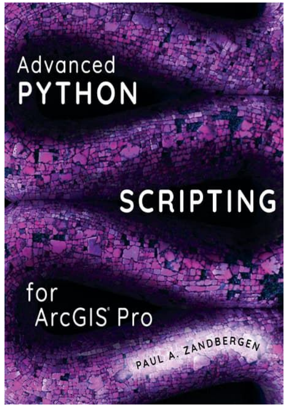
Get started

Python is a free, cross-platform, open source programming language. It's widely used and supported and is an integral part of ArcGIS.

This section covers several important topics for getting started with Python in ArcGIS Pro, including Python migration for ArcGIS Pro, Python in ArcGIS Pro, What is ArcPy?, ArcGIS API for Python, Package Manager, and Notebooks in ArcGIS Pro.

In this topic
What's in the Python reference





Thanks!

stephen.bond@austincc.edu