

ArcPy: An Introduction

Danny McVey

Hannes Ziegler



esri

THE
SCIENCE
OF
WHERE®

Agenda

- About the Python language
- About ArcPy
 - List functions
 - Describe function
- Additional topics



What is Python?

"Python is an easy to learn, powerful language... (with) high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing...make it an ideal language for scripting...in many areas and on most platforms."

- python.org

Scripting language of ArcGIS

Free, cross-platform, easy to learn, widely useful, great community



Why use Python and ArcGIS?

- Automate repetitive tasks
 - Perform (repeatable) analysis , data processing, administration...
 - Extend ArcGIS Pro and Desktop
 - geoprocessing tools
 - raster functions
 - Extend ArcGIS services with geoprocessing tasks
-
- Long and continued commitment
 - since 2004
 - actively part of future plans



Python Everywhere



ANACONDA®

Language Basics

The background is a vibrant, abstract composition. It features a gradient of colors from deep purple and blue on the left to bright orange and pink on the right. There are several overlapping, semi-transparent shapes, including a large, curved orange shape on the right side and a smaller, curved pink shape above it. A thin, bright yellow line curves across the middle of the image. The overall style is modern and artistic, with a focus on bold colors and geometric forms.

Python Language Basics

- Dynamically Typed
- Common Types Supported
 - Strings
 - Numbers
 - Functions

```
a = 1
b = 2
c = a+b

print(c)
```

```
fc = r"C:\UC2019\Demos\Demos.gdb\MajorAttractions"
```



Python Language Basics

- Files

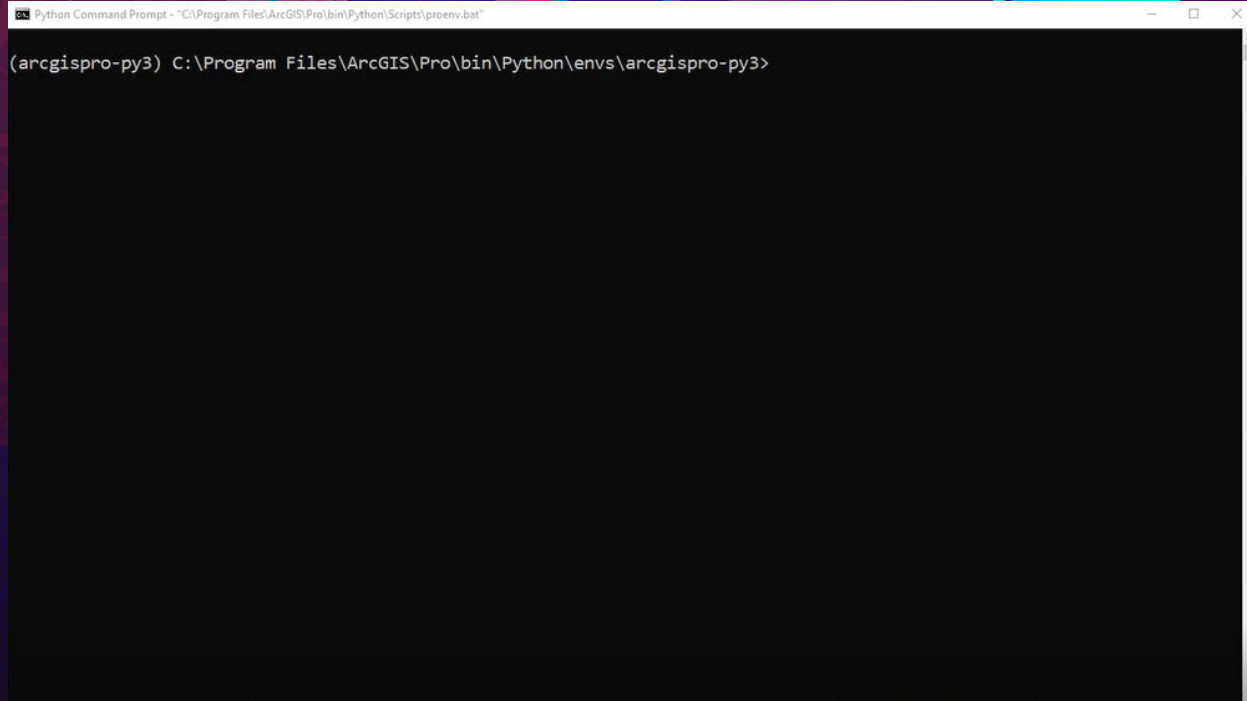
```
# Xml processing
opnXml = open(r"C:\UC2019\Demos\attractions.xml", 'r')
```

- Lists

```
#           0       1       2
flds = ["NAME", "ADDR", "EMP"]
```

- Dictionaries

```
web_map_properties = {'title': mpTitle,
                      'type': 'Web Map',
                      'snippet': mpSnippet,
                      'tags': 'PM 2.5', 'text': 1}
```

```
Python Command Prompt - "C:\Program Files\ArcGIS\Pro\bin\Python\Scripts\proenv.bat"
(arcgispro-py3) C:\Program Files\ArcGIS\Pro\bin\Python\envs\arcgispro-py3>
```

Demo: Python & ArcPy

The background of the slide is a vibrant, abstract composition of colors including purple, blue, pink, and orange. It features various geometric shapes, a grid pattern, and a stylized mountain range on the right side. A prominent yellow arc curves across the right half of the image. The text 'ArcPy' is centered on the left side in a white, sans-serif font.

ArcPy

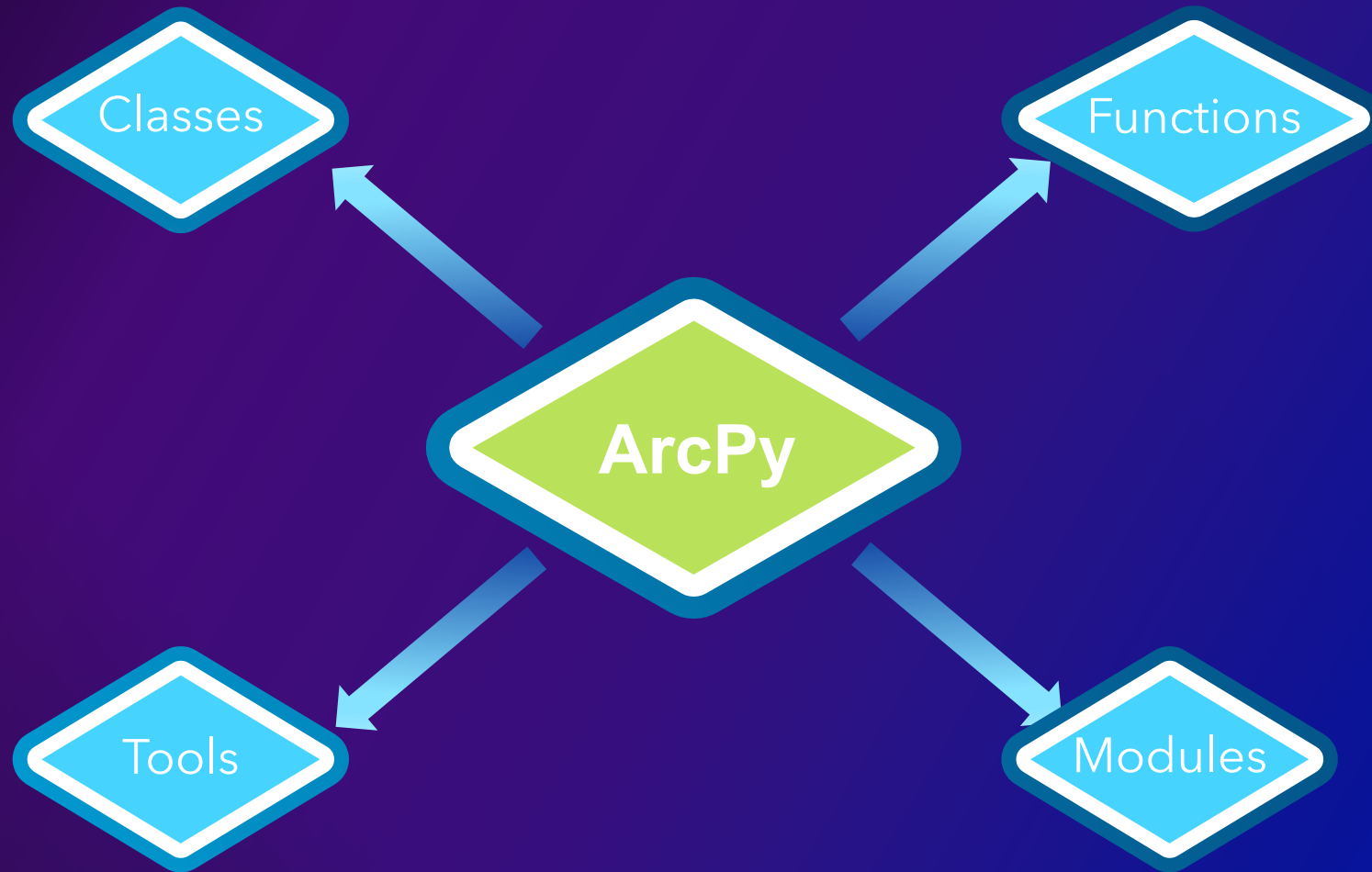
what is ArcPy?

- a python site-package
- a python api for ArcGIS Pro, ArcGIS Server, ArcGIS Desktop

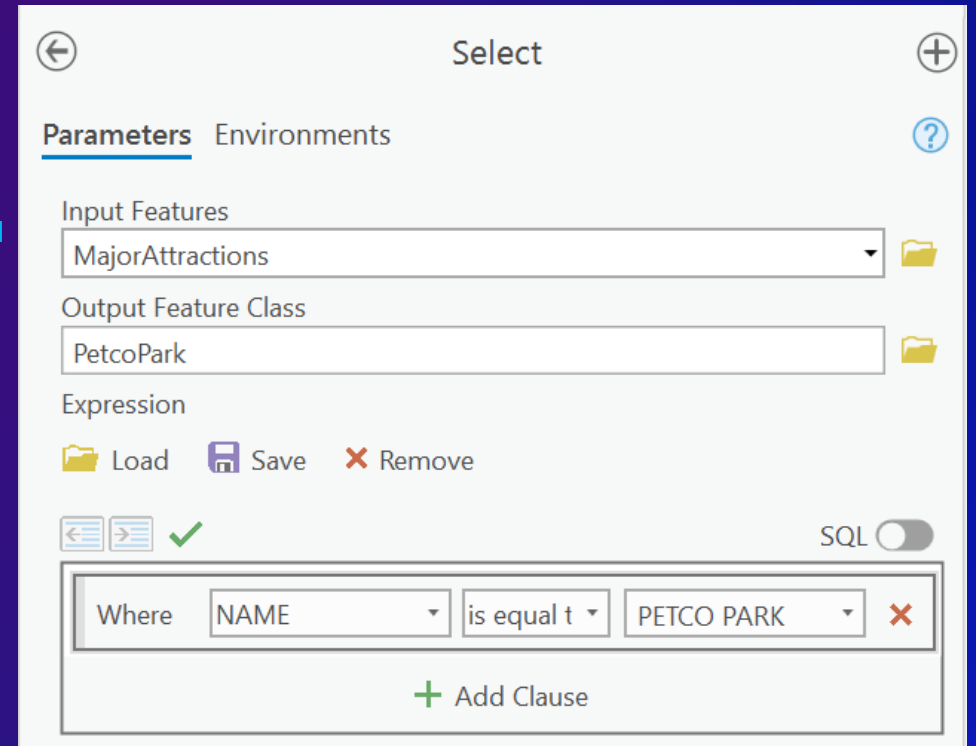
```
print(9*9)
81

import arcpy
arcpy.Buffer_analysis(...)
```





ArcPy Tools



Select

Parameters Environments

Input Features
MajorAttractions

Output Feature Class
PetcoPark

Expression
Load Save Remove

SQL

Where NAME is equal t PETCO PARK

+ Add Clause

```
arcpy.Select_analysis("MajorAttractions", "PetcoPark", "Name = 'PETCO PARK'")
```

Accessing Tools and Syntax help

Select (Analysis Tools)

Extracts features from an input feature class or input feature layer, typically using a select or Structured Query Language (SQL) expression, and stores them in an output feature class.

Syntax

Select(in_features, out_feature_class, {where_clause})

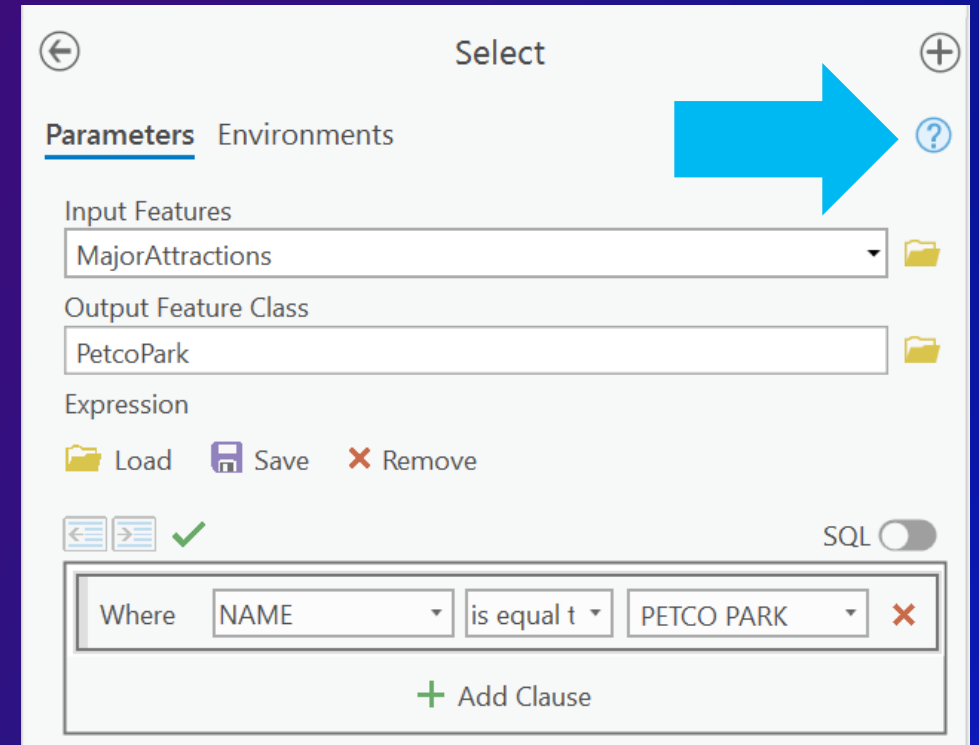
Parameter	Explanation	Data Type
in_features	The input feature class or layer from which features are selected.	Feature Layer
out_feature_class	The output feature class to be created. If no expression is used, it contains all input features.	Feature Class
where_clause (Optional)	An SQL expression used to select a subset of features. For more information on SQL syntax, see SQL reference for query expressions used in ArcGIS .	SQL Expression

Code sample

Select example 1 (Python window)

The following Python window script demonstrates how to use the Select function in immediate mode.

```
import arcpy
arcpy.env.workspace = "C:/data"
arcpy.Select_analysis("majorrds.shp", "C:/output/majorrdsClass4.shp", '"CLASS" = \'4\'')
```



Select

Parameters Environments

Input Features
MajorAttractions

Output Feature Class
PetcoPark

Expression

Load Save Remove

SQL

Where NAME is equal to PETCO PARK

+ Add Clause

ArcPy Functions

▼ General data functions
CreateScratchName
CreateUniqueName
Exists
ParseTableName
TestSchemaLock
ValidateTableName

▼ Geodatabase administration
AcceptConnections
DisconnectUser
ListUsers

▼ Geometry
AsShape
FromGeohash
FromWKB
FromWKT

```
arcpy.Exists('MajorAttractions')
```



ArcPy Classes

- Used to help code
- Create geometry
 - Points
 - Polygons
 - Spatial Reference
- Provide field information

▼ Geometry
Geometry
Multipoint
Point
PointGeometry
Polygon
Polyline

▼ General
ArcSDESQLExecute
Array
Extent
Index
NetCDFFileProperties
RandomNumberGenerator
Raster
Result
SpatialReference
ValueTable
VCS

ArcPy Modules

- **collection of related functionality**
 - `arcpy.da`
 - `arcpy.metadata`
 - `arcpy.nax`

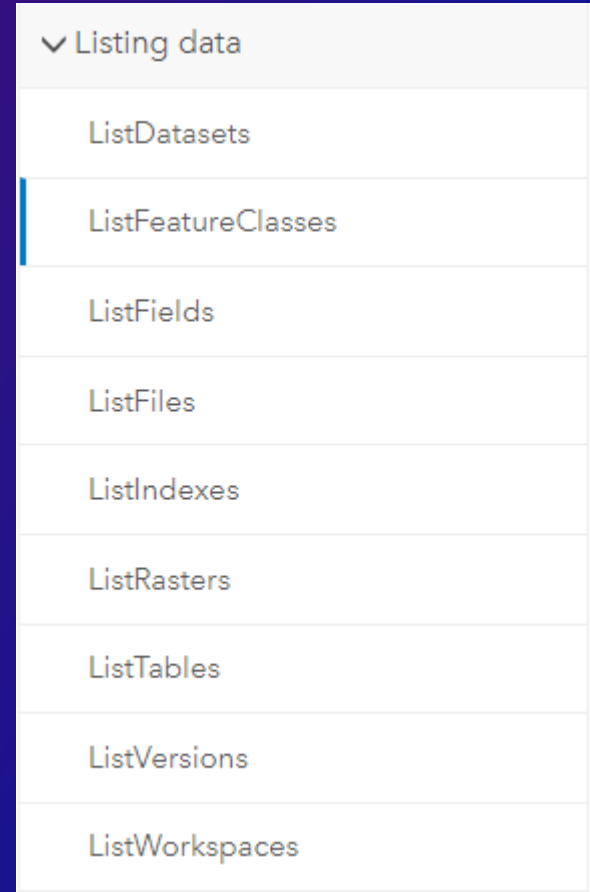


arcpy List functions

The background features a vibrant color palette of purple, pink, and orange. On the right side, there is a stylized topographic map showing terrain contours and a yellow curved line. The overall aesthetic is modern and digital.

Listing Functions

- **Return Python list**
 - String
 - Object
- **Minimal Parameters needed**



Listing data
ListDatasets
ListFeatureClasses
ListFields
ListFiles
ListIndexes
ListRasters
ListTables
ListVersions
ListWorkspaces

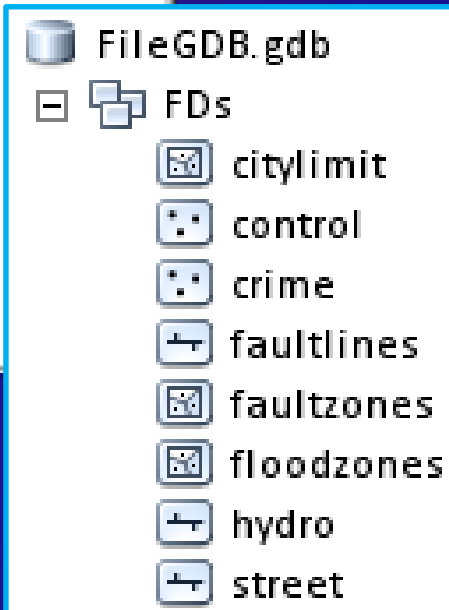
arcpy.ListFeatureClasses

- ListFeatureClasses ({wild_card}, {feature_type}, {feature_dataset})

```
# Set the workspace
arcpy.env.workspace = "C:/Data/FileGDB.gdb/FDs"

# Get a list of all feature classes
fcList = arcpy.ListFeatureClasses()

# Print the names of the feature classes
for fc in fcList:
    print(fc)
```



arcpy.da.Walk

- modeled after python's `os.walk`

```
ws = r"c:\projects\project_x"
walk = arcpy.da.Walk(ws, datatype="FeatureClass", type="Polygon")

for dirpath, dirnames, filenames in walk:
    for filename in filenames:
        feature_classes.append(os.path.join(dirpath, filename))
```



Python List Functions

- **index**
 - where in the list is an item
- **sort**
 - A to Z a to z
- **count**
 - how many times an item occurs in the list

```
my_list = arcpy.ListFeatureClasses()

len(my_list)
9

my_list.index('counties')
4

my_list = sort(my_list)
```



The background is a vibrant, abstract composition. It features a gradient of colors from deep purple to bright pink and orange. On the right side, there are stylized, layered shapes that resemble a topographic map or a landscape with a mountain range. A prominent yellow arc curves across the right side of the image. The overall aesthetic is modern and digital.

arcpy.Describe

Describe Function

Describe function reads data properties

Returns an object with properties

Data type

Shape type

Spatial reference

Fields

```
# Describe a feature class
desc = arcpy.Describe("C:/Data/Roads.shp")

print(desc.shapeType)
Polyline
```



Describe Function

- Programmatically understand your data
- Useful when creating tools
 - Geometry Type
 - Fields
 - SpatialReference
 - RasterBands

```
desc = arcpy.Describe("C:/demos/MajorAttractions")
if desc.shapeType == "Point":
    ...

desc = arcpy.da.Describe("C:/demos/MajorAttractions")
print(desc)
{'OIDFieldName': 'OBJECTID',
 'dataType': 'FeatureClass',
 'shapeType': 'Point',
 ...
}
```



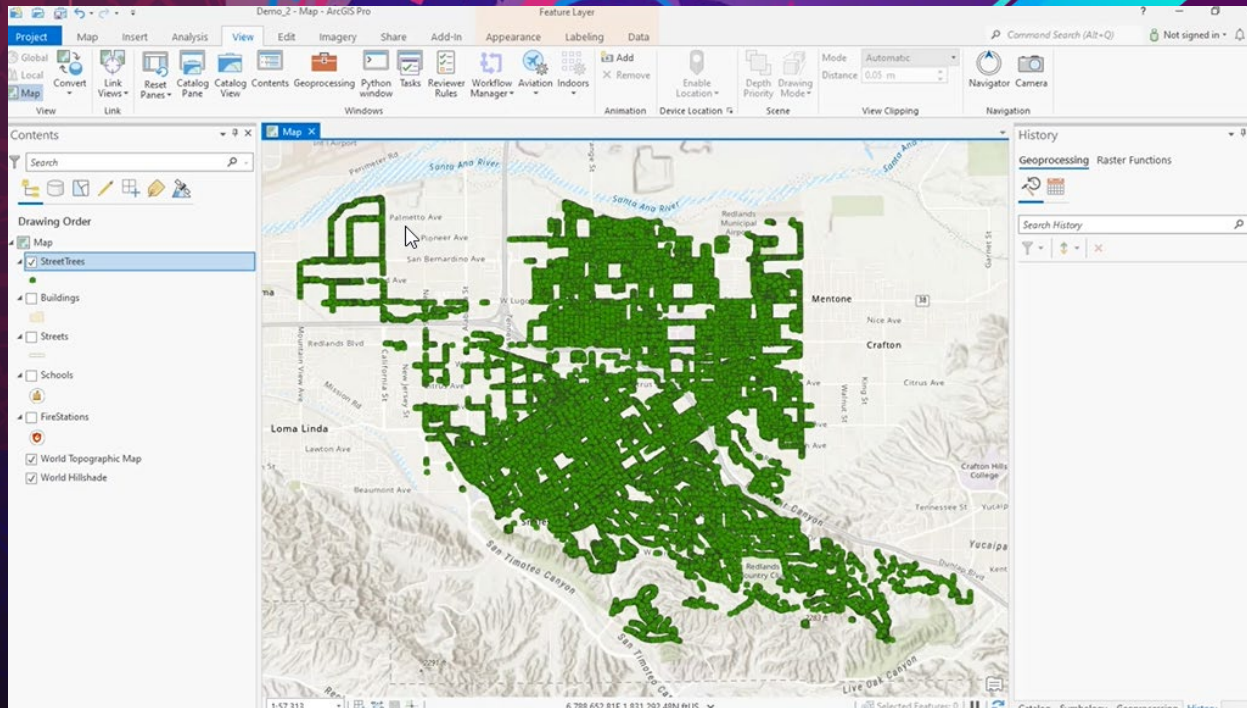
da.Describe Function

- Returns Python Dictionary
- Same Describe properties

```
desc = arcpy.da.Describe("C:/demos/MajorAttractions")

print(desc)
{'OIDFieldName': 'OBJECTID',
 'dataType': 'FeatureClass',
 'shapeType': 'Point',
 'spatialReference': <SpatialReference object at 0x19>,
 ...
}
```



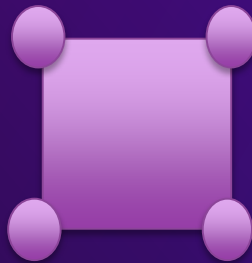


Demo: Python in Pro, List & Describe

Additional Topics

The background features a vibrant, abstract composition. On the left, there are layered, wavy shapes in shades of blue, purple, and pink. The right side is dominated by large, curved, overlapping shapes in bright orange, pink, and yellow. A thin, curved yellow line arches across the middle-right section. The overall aesthetic is modern and digital.

Creating Geometry



```
pnt = arcpy.Point(-22.570,63.968)
```

```
line = arcpy.Polyline(  
arcpy.Array([arcpy.Point(-22.570,63.968),  
arcpy.Point(-22.573,63.969)  
]))
```

```
Polygon = arcpy.Polygon(  
arcpy.Array([arcpy.Point(-22.570,63.968),  
arcpy.Point(-22.573,63.969),  
arcpy.Point(-22.575,63.968)  
]))  
)
```



Geometry Operators

- **Return Values**

- Boolean
- Geometry/s

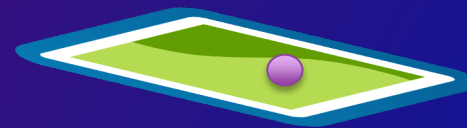
- **Relational:**

- Is a point within a polygon
- Return Boolean Values

```
sr = arcpy.SpatialReference(4326)
pt = arcpy.Point([-114.1, 51.0], sr)
```

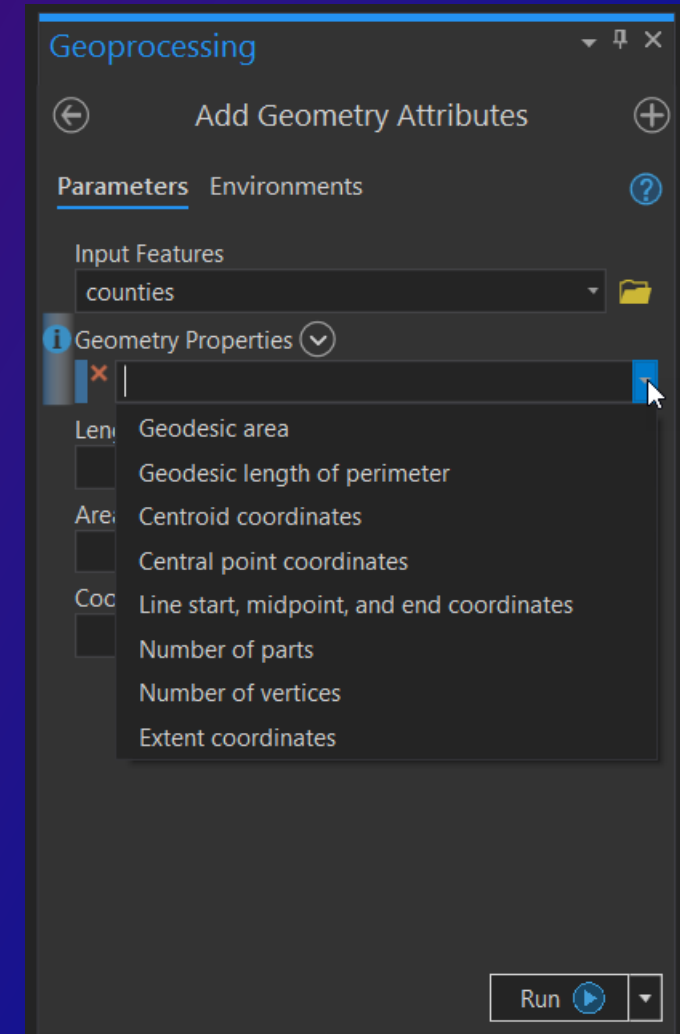
```
print(pt.within(some_polygon))
True
```

```
print(pt.buffer(5))
<Polygon object at 0x1>
```



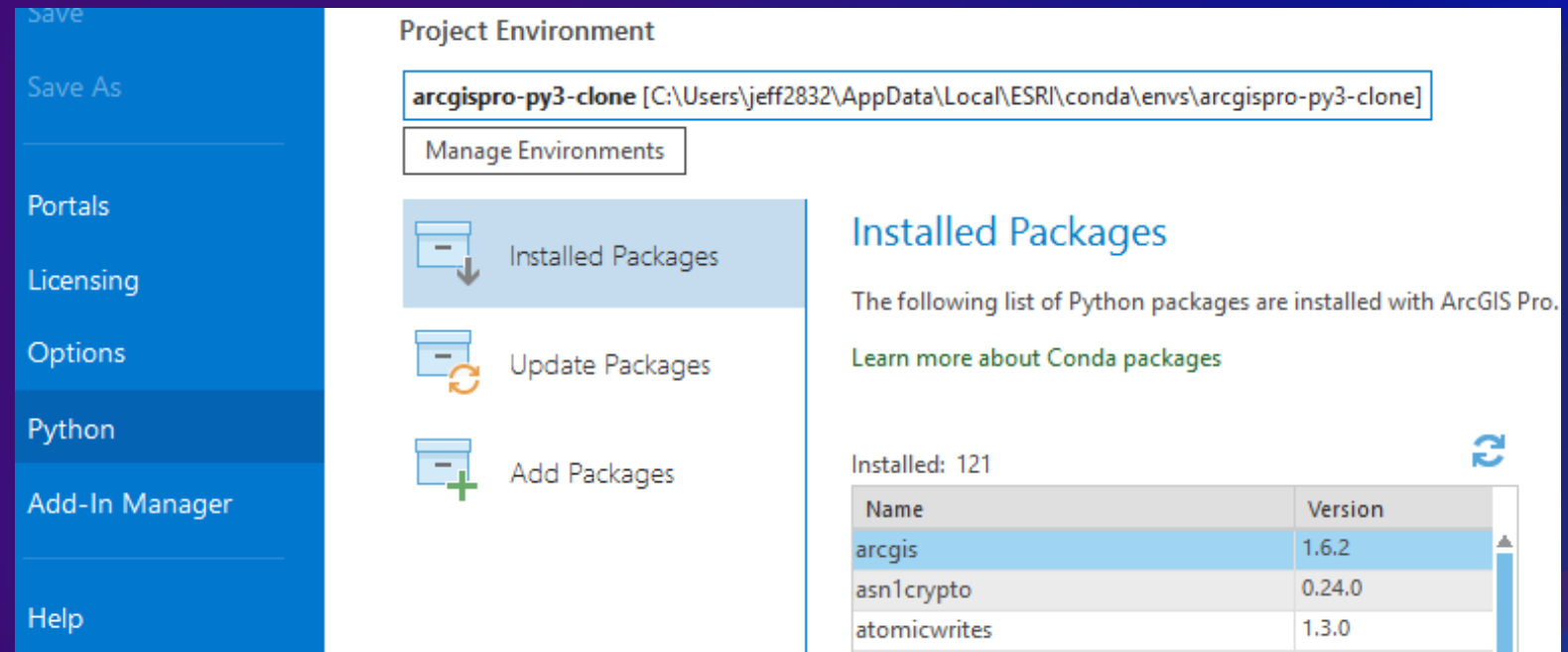
Extending ArcGIS Pro with gp tools

- build rich familiar dialog
 - interactive dialog
 - messaging
 - progress bar
- deep integration with ArcGIS Pro/ArcMap app
 - active map's layers
 - add result to display
- see "ArcPy: Building Geoprocessing Tools " tech workshop



Adding python Packages

- **Conda Environments**
 - Default is read only
 - Clone to add a package
 - Access to 1600 + packages
 - Python.exe with everyone



The screenshot shows the 'Project Environment' window in ArcGIS Pro. The left sidebar contains menu items: Save, Save As, Portals, Licensing, Options, Python (highlighted), Add-In Manager, and Help. The main area displays the environment name 'arcgispro-py3-clone' and a 'Manage Environments' button. Below are three options: 'Installed Packages' (selected), 'Update Packages', and 'Add Packages'. The 'Installed Packages' section shows a list of 121 installed packages with a table of the first three: arcgis (1.6.2), asn1crypto (0.24.0), and atomicwrites (1.3.0).

Project Environment

arcgispro-py3-clone [C:\Users\jeff2832\AppData\Local\ESRI\conda\envs\arcgispro-py3-clone]

Manage Environments

Installed Packages

Update Packages

Add Packages

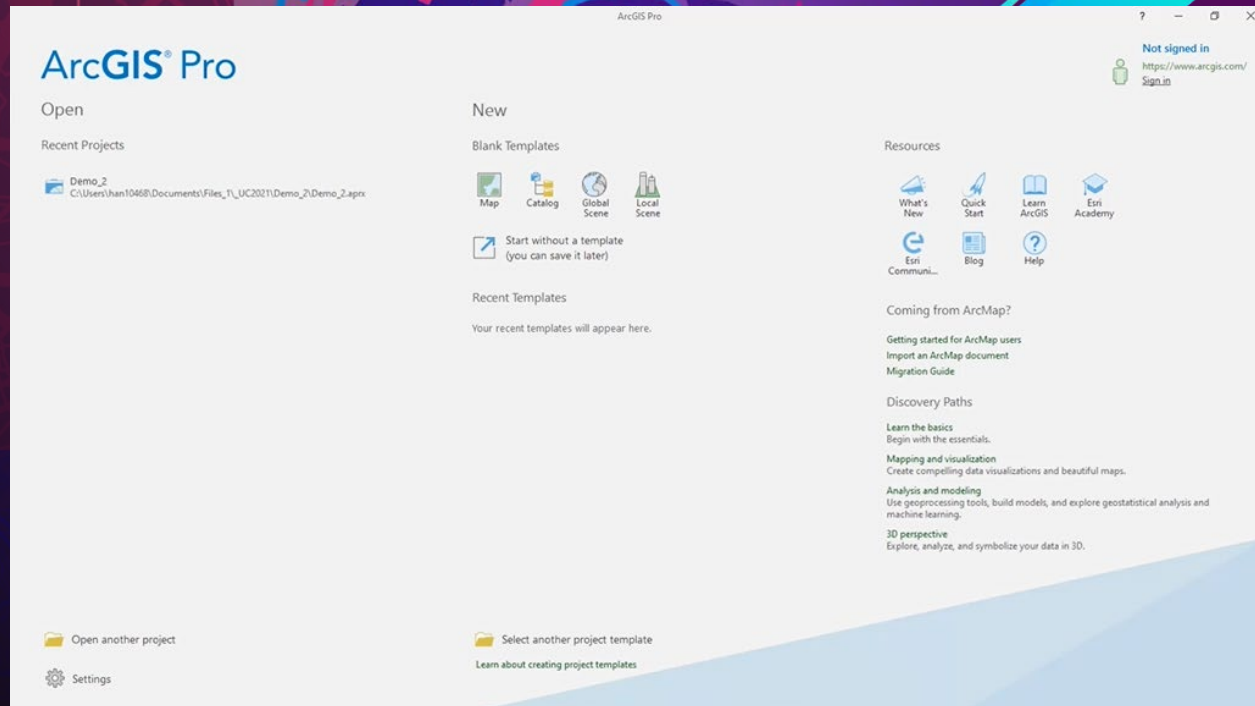
Installed Packages

The following list of Python packages are installed with ArcGIS Pro.

[Learn more about Conda packages](#)

Installed: 121

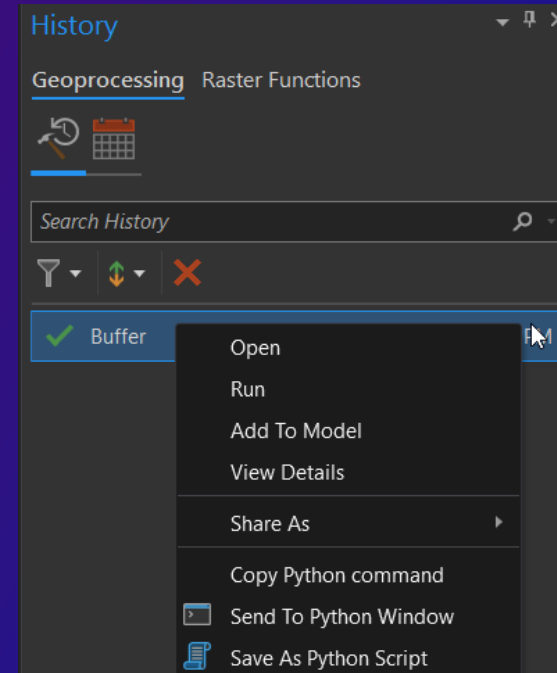
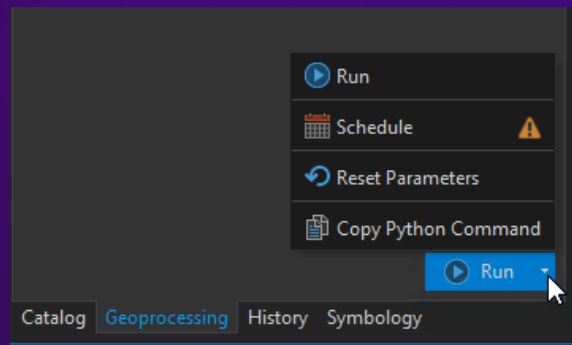
Name	Version
arcgis	1.6.2
asn1crypto	0.24.0
atomicwrites	1.3.0



Demo : Adding and using Modules

Tips on getting started

- **Export Model Builder to Python**
- **Copy python command from tools**



Tips on getting started

- **Won't always work the first time**
 - Write it, run it, break it
 - Get comfortable with tracebacks

```
>>> x=2
>>> x+y
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'y' is not defined
>>> y=3
>>> x+y
5
>>>
```



Other Python Technical Workshops at UC 2021

- [ArcGIS API for Python: An Introduction](#)
- [ArcGIS: Python Overview](#)
- [Geospatial Deep Learning Using the ArcGIS API for Python](#)



The background is a vibrant, abstract composition. It features a mix of colors including deep purple, magenta, bright pink, and orange. There are various geometric and organic shapes, such as curved lines, overlapping planes, and a grid-like pattern in the upper right. The overall effect is dynamic and modern.

We want your feedback

Click on the [Session Survey](#) link
below this video window



esri[®]

THE
SCIENCE
OF
WHERE[®]

Presentation Title

Presenter Names



esri

THE
SCIENCE
OF
WHERE®





The background is a vibrant, abstract composition. It features a gradient of colors from deep purple and blue on the left to bright orange and pink on the right. There are several overlapping, semi-transparent shapes, including a large, curved orange shape on the right side. A thin, bright yellow arc curves across the middle-right portion of the image. In the bottom right corner, there is a stylized, low-poly mountain range in shades of orange and yellow. The overall aesthetic is modern and energetic.

Section Header

Section Subhead



Demo Title

Presenter(s)



esri[®]

**THE
SCIENCE
OF
WHERE**[®]