Esri® ArcPy™ Cheat Sheet

What Is ArcPy?

ArcPy™ (often referred to as the ArcPy site package) provides Python access for all geoprocessing tools, including extensions, as well as a wide variety of useful functions and classes for working with and interrogating GIS data. Using Python and ArcPy, you can develop an infinite number of useful programs that operate on geographic data.

Installing ArcPy The ArcPy package is part of the default Python environment arcgispro-py3 that is distributed with ArcGIS Pro and ArcGIS Server.

To get started with a script:

```
import arcpy
```

Listing Data

ArcPy list functions are used for listing ArcGIS® data. Most of these functions list data from the current ArcPy environment workspace. See the ArcPy Environments section for setting your workspace.

List datasets	`arcpy.ListDatasets()`
List feature classes	`arcpy.ListFeatureClasses()`
List files	`arcpy.ListFiles()`
List rasters	`arcpy.ListRasters()`
List tables	`arcpy.ListTables()`
List workspaces	`arcpy.ListWorkspaces()`

A few ArcPy list functions require passing in the workspace/dataset to search.

List indexes in a specified dataset
`arcpy.ListIndexes(dataset)`

List fields in a specified dataset
`arcpy.ListFields(dataset)`

List versions the connected user has permission to use
`arcpy.ListVersions(sde_workspace)`

Alternatively, the data access module (**arcpy.da**) helps you find and access files using **Walk**, which is based on the Python built-in **os.walk**. Unlike in the list functions above, the top directory is passed as an argument.

Traverses the directory tree and returns the directory path, directory names, and file names

```
for dirpath, dirnames, filenames in arcpy.da.walk(workspace, ...):
    ...
```

Cursors

Cursors are used to access and manipulate records in a table or feature class. Cursors place a lock on the data being accessed. After using a cursor, delete the cursor object. This can be accomplished with the cursor's context manager. Once the context manager code block is exited, the cursor object is automatically deleted.

```
with arcpy.da.SearchCursor(...) as cursor:
    for row in cursor:
        ...
```

This avoids explicitly calling `del <cursor>`.

SearchCursor returns rows of attribute values

```
arcpy.da.SearchCursor(
    in_table, field_names, ...
)
```

UpdateCursor updates or deletes rows of attribute values

```
arcpy.da.UpdateCursor(
    in_table, field_names, ...
)
```

InsertCursor inserts rows of attribute values

```
arcpy.da.InsertCursor(
    in_table, field_names, ...
)
```

Cursors Field Tokens

Use cursor field tokens to access special fields or data of a dataset. Here are some useful tokens:

OID@	Object ID
GLOBALID@	Global UID
SHAPE@X	X-coordinate
SHAPE@Y	Y-coordinate
SHAPE@Z	Z-coordinate
SHAPE@M	M-value
SHAPE@XY	Centroid x,y coordinates
SHAPE@XYZ	Centroid x,y,z coordinates
SHAPE@AREA	Area
SHAPE@LENGTH	Length
SHAPE@	Geometry object
SHAPE@WKT	Well-known text representation
SHAPE@WKB	Well-known byte representation
SHAPE@JSON	Esri JSON representation

Extracting Geometries

You can extract geometries from existing features in two ways:

1) Read geometries with **arcpy.da** cursors using the SHAPE@ field token.

```
# Store geometries in a Python list
buffered_points = [
    row[0].buffer(100)
    for row
    in arcpy.da.SearchCursor(fc, "SHAPE@")
]
```

2) Setting the output parameter of a geoprocessing tool to an empty **Geometry** object will output a list of **Geometry** objects.

```
geoms = arcpy.management.CopyFeatures(
    "C:/<path>", arcpy.Geometry()
)
```

ArcPy Environments

Environment settings affect how geoprocessing tools run. These are exposed as properties on the **arcpy.env** class. Environment settings set in a script will go out of scope (reset) once the script terminates.

Get workspace property:

```
ws = arcpy.env.workspace
```

Set workspace property:

```
arcpy.env.workspace = "C:/<path>"
```

Temporarily set workspace using **arcpy.EnvManager** context manager:

```
with arcpy.EnvManager(workspace="C:/<path>"):
    ...
```

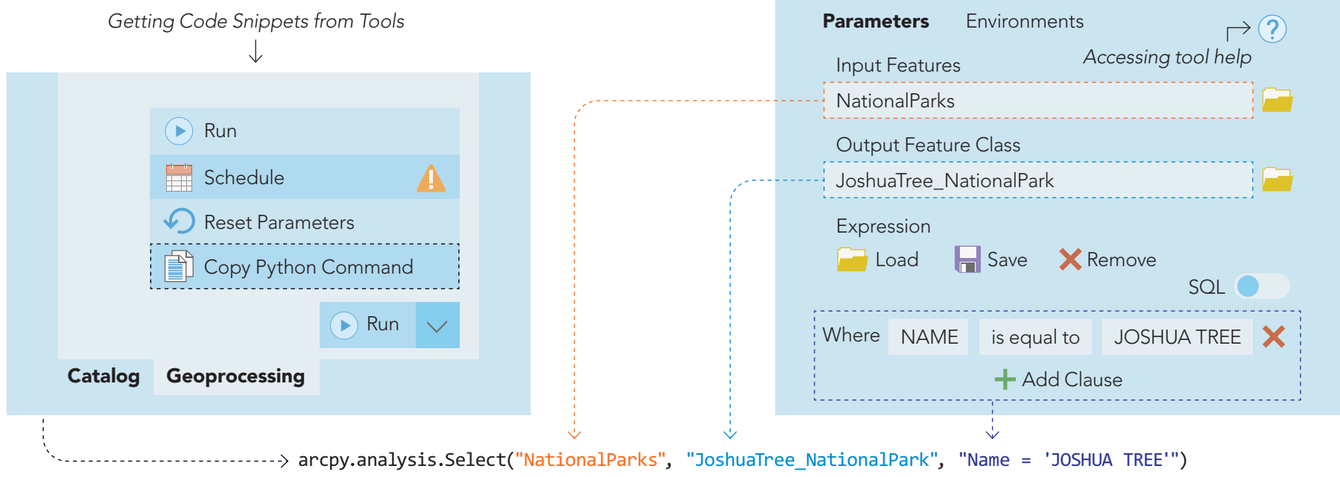
Describing data

Query the properties of an object such as data type, fields, and indexes. Returned values can be used to conditionally operate on the data. There are two varieties of **Describe** shown below; the former is slower but great for exploring properties, while the latter is faster and therefore useful in production code.

Describes a data element and returns a dictionary (slower, great for exploration)
`arcpy.da.Describe(value)`

Describes a data element and returns an object (faster, great for production code)
`arcpy.Describe(value)`

Running Geoprocessing Tools with ArcPy



All geoprocessing tools can be accessed as functions from the module corresponding to its respective system toolbox (see the right sidebar).

Constructing Geometries

Geometry objects define a spatial location and an associated geometric shape. The primary geometry objects are **PointGeometry**, **Multipoint**, **Polyline**, and **Polygon**. The basic building blocks of geometry objects are **Point** objects.

```
p1 = arcpy.Point(-22.00, 64.00)
```

Complex geometries are defined by an **Array** of **Point** objects.

```
a1 = arcpy.Array(
    [
        arcpy.Point(-22.00, 64.00),
        arcpy.Point(-22.50, 53.85)
    ]
)
```

It is best practice to define a spatial reference when creating geometry objects.

```
sr = arcpy.SpatialReference(4326)
```

Note: Do not confuse **Point** and **Array** objects with the primary geometry objects. The latter are constructed from the former.

The primary geometry objects can be constructed from **Point** and **Array** (and optional spatial reference) objects as follows:

Create a PointGeometry object—defined by a single point object
`arcpy.PointGeometry(p1, sr)`

Create a Multipoint object—an ordered collection of points defined by an array of point objects

```
arcpy.Multipoint(
    arcpy.Array([p1, p2, ...]), sr
)
```

Create a Polyline—a path defined by an array of point objects

```
arcpy.Polyline(
    arcpy.Array([p1, p2, ...]), sr
)
```

Create a Polygon object—a closed shape defined by an array of points (To close the shape, the last point in the array must equal the first.)

```
arcpy.Polygon(
    arcpy.Array([p1, p2, p3, ..., p1)], sr
)
```

Geometry Operators

Primary geometry objects support the following relational operators:

+	Intersect	`g3 = g1 + g2`
|	Union	`g3 = g1 | g2`
-	Difference	`g3 = g1 - g2`
^	Symmetric Diff.	`g3 = g1 ^ g2`
==	Equals	`g1 == g2`
!=	Not Equals	`g1 != g2`

Data Conversion

Convert tables to other popular Python data formats.

Converts a feature class to a NumPy structured array

```
arcpy.da.FeatureClassToNumPyArray(
    in_table, field_names, ...
)
```

Converts a NumPy structured array to a point feature class

```
arcpy.da.NumPyArrayToFeatureClass(
    in_array, out_table, shape_fields, ...
)
```

Converts a table to a NumPy structured array

```
arcpy.da.TableToNumPyArray( in_table, field_names, ... )
```

Converts a NumPy structured array to a table

```
arcpy.da.NumPyArrayToTable( in_array, out_table )
```

Converts a table or feature class to an Apache Arrow table

```
arcpy.da.TableToArrowTable(in_table)
```

Memory Workspace

Significantly speed up processing times with the memory workspace. To write to the memory workspace, specify an output dataset path beginning with "memory" and with no file extension. Use the memory workspace for intermediate output.

```
outfc = r"memory\tempOutput"
```

Geometry Properties and Methods

Geometry objects have properties, methods, and operators that can be used to query and manipulate the geometry. Availability varies by geometry type.

Some useful properties:

type	Type of geometry
spatialReference	Spatial reference object
pointCount	Number of points in geometry

For example:

```
if g1.type == "point":
    ...
```

Some useful methods:

buffer(distance)	Buffer the geometry
distanceTo(other)	Distance to another geometry
within(geometry)	Is geometry within another

For example:

```
distance = g1.distanceTo(g2)
```

Geoprocessing Tool Results

Most geoprocessing tools return a **Result** object. The **Result** object maintains information about a tool operation after it has completed. You can use the **Result** object directly as input to another tool, or extract the outputs, messages, and parameters.

```
r1 = arcpy.analysis.Buffer(infc, outfc, 100)
```

Get the number of outputs
`r1.outputCount`

Get a given output
`r1.getOutput(index)`

Get a given input
`r1.getInput(index)`

Get the geoprocessing tool messages
`r1.getMessages()`

Note: Geoprocessing tools in the Image Analyst (**arcpy.ia**) and Spatial Analyst (**arcpy.sa**) modules return a **Raster** object.

Debugging Python Scripts

We recommend using the ArcGIS Pro Debugger extension for Visual Studio Code (VSC) to leverage VSC's native Python debugging experience while developing script tools (.atbx, .pyt) for ArcGIS Pro. To learn more about this and other options for debugging, scan the QR code:

Note: Whenever the characters <> encase a word, it marks a placeholder for an object or value to be replaced by the user.

ArcPy Modules

Charts
arcpy.charts

Data Access
arcpy.da

Mapping
arcpy.mp

Metadata
arcpy.metadata

Network Analyst
arcpy.nax and arcpy.na

Sharing
arcpy.sharing

ArcPy modules representing an ArcGIS Pro toolbox:

3D Analyst
arcpy.ddd

AllSource
arcpy.intelligence

Analysis
arcpy.analysis

Aviation
arcpy.aviation

Business Analyst
arcpy.ba

Cartography
arcpy.cartography

Conversion
arcpy.conversion

Crime Analysis and Safety
arcpy.ca

Data Interoperability
arcpy.di

Data Management
arcpy.management

Data Reviewer
arcpy.reviewer

Defense
arcpy.defense

Editing
arcpy.edit

GeoAI
arcpy.geoai

GeoAnalytics Desktop
arcpy.geoanalytics

GeoAnalytics Server
arcpy.gapro

Geocoding
arcpy.geocoding

Geostatistical Analyst
arcpy.ga

Image Analyst
arcpy.ia

Indoor Positioning
arcpy.indoorpositioning

Indoors
arcpy.indoors

Knowledge Graph
arcpy.kg

Linear Referencing
arcpy.lr

Location Referencing
arcpy.locref

Maritime
arcpy.maritime

Multidimension
arcpy.md

Network Analyst
arcpy.nax, arcpy.na

Network Diagram
arcpy.nd

Oriented Imagery
arcpy.oi

Parcels
arcpy.parcels

Public Transit
arcpy.transit

Raster Analysis
arcpy.ra

Ready To Use
arcpy.agolservices

Reality Mapping
arcpy.rm

Server
arcpy.server

Space Time Pattern Mining
arcpy.stpm

Spatial Analyst
arcpy.sa

Spatial Statistics
arcpy.stats

Standard Feature Analysis
arcpy.sfa

Territory Design
arcpy.td

Topographic Production
arcpy.topographic

Trace Network
arcpy.tn

Utility Network
arcpy.un

G7408447

ArcGIS Pro 3.6+
Document version 3R0