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.

Returns a list of datasets `arcpy.ListDatasets()`

Returns a list of feature classes `arcpy.ListFeatureClasses()`

Returns a list of files `arcpy.ListFiles()`

Returns a list of the rasters `arcpy.ListRasters()`

Returns a list of tables `arcpy.ListTables()`

Returns a list of workspaces `arcpy.ListWorkspaces()`

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

Returns a list of the indexes in a specified dataset

```
arcpy.ListIndexes(dataset)
```

Returns a list of fields in a specified dataset

```
arcpy.ListFields(dataset)
```

Lists the 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

```
arcpy.da.Walk(top, ...)
```

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-coordinate
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

Data Conversion

Besides providing access to records and attributes of datasets with `Cursors` and `Describe`, the data access module (**arcpy.da**) also helps convert to other popular Python data formats such as the NumPy Structured Array and the Arrow Table.

Converts a feature class to 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 NumPy structured array to a table

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

Converts a table to a NumPy structured array

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

Converts a table or feature class to an Apache Arrow table

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

Extracting Geometries

You can also extract geometries from existing features in two ways:

1) 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()
)
```

2) 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@")
]
```

ArcPy Environments

Environment settings are accessed from `arcpy.env.EnvManager` can be used in a `with` statement to temporarily set an environmental variable for a section of the script. Environment settings set in a script will go out of scope (reset) once the script terminates.

Get environment:

```
print(arcpy.env.workspace)
```

Set the workspace environment:

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

Temporarily set workspace environment:

```
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.

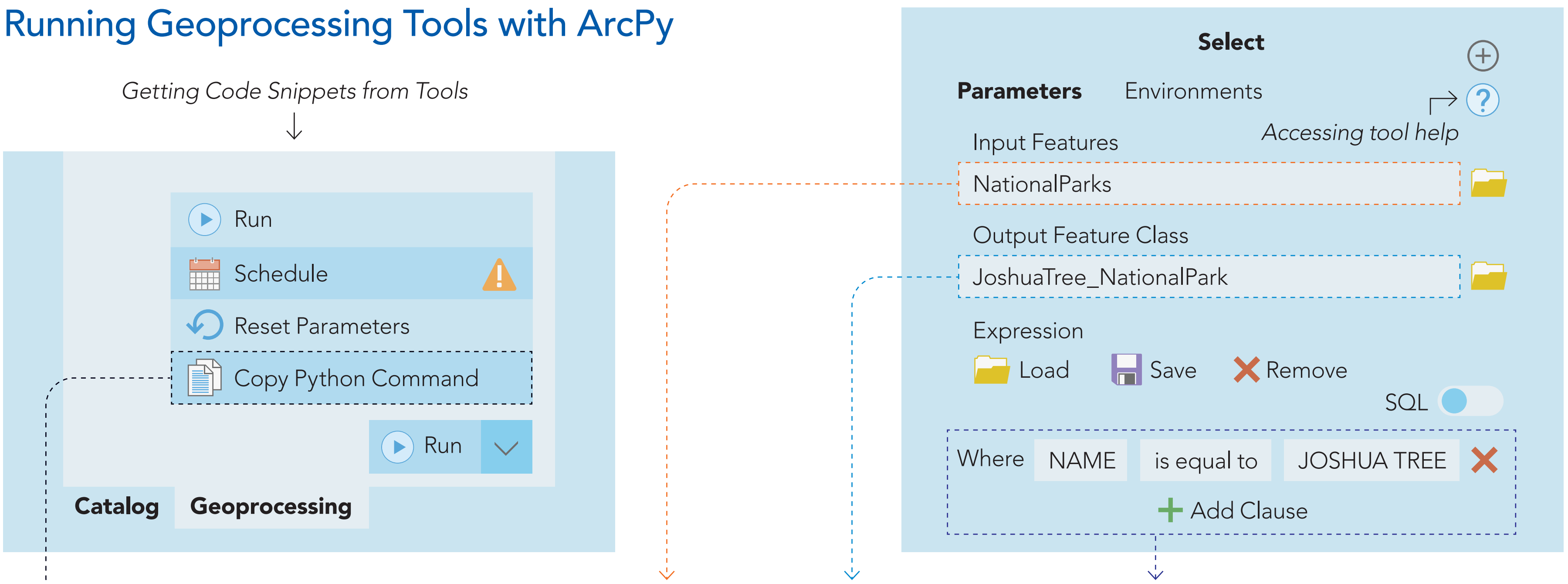
Describes a data element and returns a dictionary (slower, great for exploration)

```
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



Getting Code Snippets from Tools

Accessing tool help

Input Features: NationalParks

Output Feature Class: JoshuaTree_NationalPark

Expression: Where NAME is equal to JOSHUA TREE

```
arcpy.analysis.Select("NationalParks", "JoshuaTree_NationalPark", "Name = 'JOSHUA TREE'")
```

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)
    ]
)
```

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` objects as follows:

Create a PointGeometry object—defined by a single point object

```
arcpy.PointGeometry(P1)
```

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

```
arcpy.Multipoint(
    arcpy.Array([P1, P2, ...]
)
```

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

```
arcpy.Polyline(
    arcpy.Array([P1, P2, ...]
)
```

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]
    )
)
```

Memory Workspace

Significantly speed up processing times with the memory workspace. The memory workspace is a memory-based workspace that supports output feature classes, tables, and raster datasets. To write to the memory workspace, specify an output dataset path beginning with memory and with no file extension. The memory workspace is used for intermediate output, but final output should be persisted to disk.

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

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

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>`.

Returns rows of attribute values

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

Updates or deletes rows of attribute values

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

Inserts rows of attribute values

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

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.

Generally useful properties:

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

For example:

```
if G1.type == "point":
  ...
```

Generally 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)
```

Geometry Operators

The primary geometry objects also support the following operators to perform relational operations:

+	Intersect	`G3 = G1 + G2`
|	Union	`G3 = G1 | G2`
-	Difference	`G3 = G1 - G2`
^	Symmetric Diff.	`G3 = G1 ^ G2`
==	Equals	`G1 == G2`
!=	Not Equals	`G1 != 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.

ArcPy Modules

A module defines a collection of functions, classes, operators, and other constructs.

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

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

Workflow Manager
arcpy.wmx

THE SCIENCE OF WHERE®

G4856739