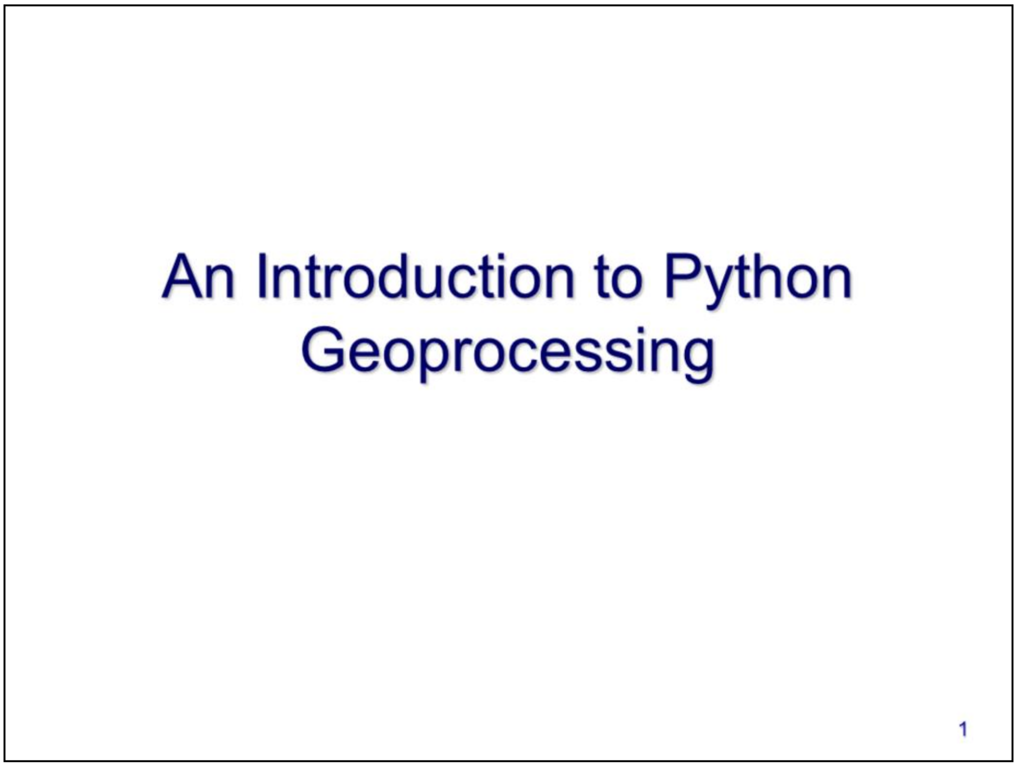
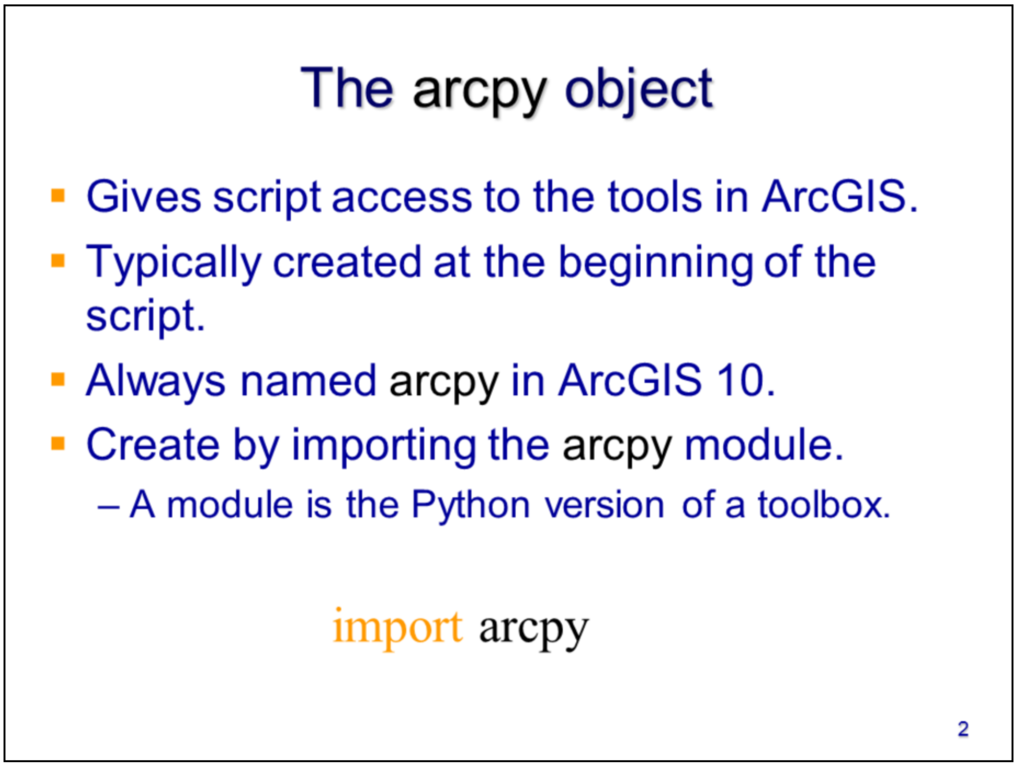


An Introduction to Python Geoprocessing

1

This video will introduce you to running ArcGIS tools within Python.

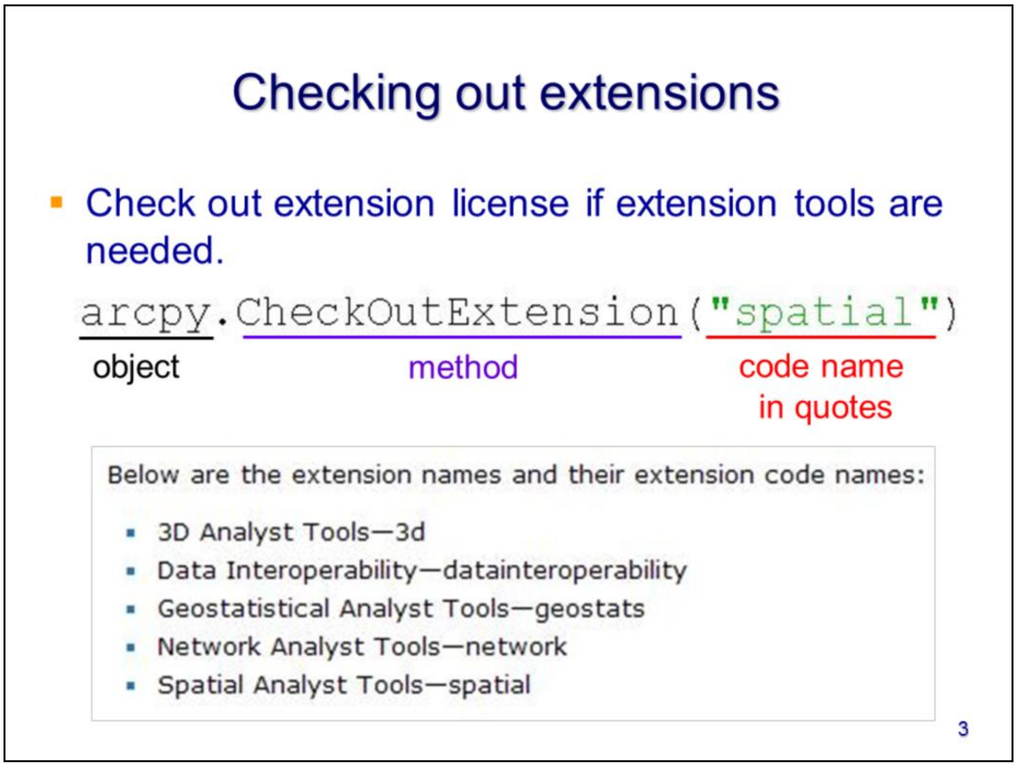

The arcpy object

- Gives script access to the tools in ArcGIS.
- Typically created at the beginning of the script.
- Always named arcpy in ArcGIS 10.
- Create by importing the arcpy module.
  - A module is the Python version of a toolbox.

```
import arcpy
```

2

With only the few basic concepts we've discussed previously, we can now learn how to use Python to run some tools in ArcGIS. Python is able to access ArcGIS's capabilities through a **module** called "arcpy." A module in Python is the equivalent of a toolbox in ArcGIS – it is a set of related tools with related purposes. In order for Python to access the arcpy module, we first need to plug-in ArcGIS to our script by typing in "import arcpy." The import command is used anytime you need to load a module in Python – this is usually done first thing in a script.



Checking out extensions

- Check out extension license if extension tools are needed.

```
arcpy.CheckOutExtension("spatial")
```

$\underbrace{\texttt{arcpy}}_{\text{object}}.\underbrace{\texttt{CheckOutExtension}}_{\text{method}}(\underbrace{\texttt{"spatial"}}_{\text{code name in quotes}})$

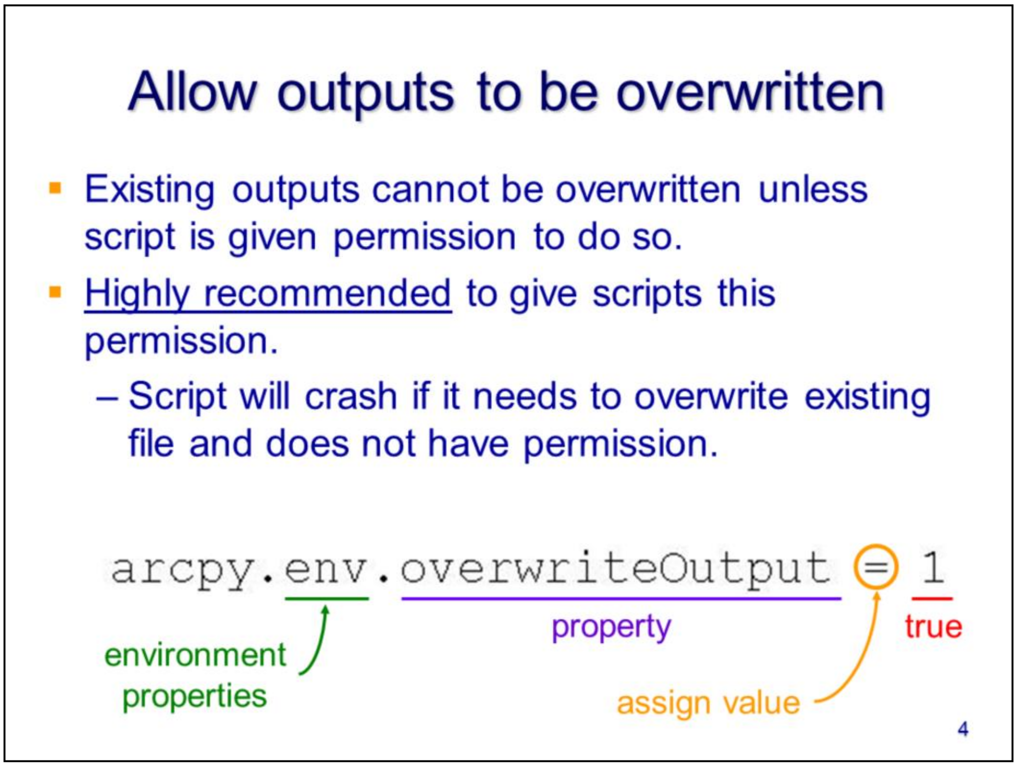
Below are the extension names and their extension code names:

- 3D Analyst Tools—3d
- Data Interoperability—datainteroperability
- Geostatistical Analyst Tools—geostats
- Network Analyst Tools—network
- Spatial Analyst Tools—spatial

3

Many of ArcGIS's tools are available through extensions – these extensions contain specialized tools which enhance the capabilities of ArcGIS. In order to use these tools, you need to have access to the appropriate extension license(s) and your script will need to check out the license.

To check out an extension license in your script, you'll need to use a method of arcpy called "CheckOutExtension". Recall that to use a method of an object, you need to specify: 1) the object name, 2) a period, 3) the method name, and 4) the method parameters. In this case, the object is *arcpy*, the method is *CheckOutExtension* and the parameter is the name of the extension that you want to use. Extensions should be referred to by their code name as indicated in the table.



Allow outputs to be overwritten

- Existing outputs cannot be overwritten unless script is given permission to do so.
- Highly recommended to give scripts this permission.
  - Script will crash if it needs to overwrite existing file and does not have permission.

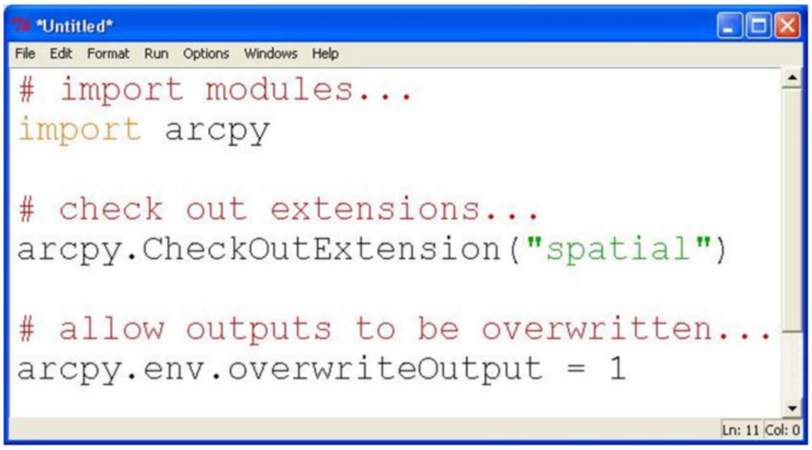
```
arcpy.env.overwriteOutput = 1
```

environment properties
property
assign value
true
4

Arcpy will not allow you to overwrite any existing GIS data unless you first give it permission to do so. It is a good idea to give this permission in most scripts because it will make debugging your script much easier. The debugging process removes errors in your script and you will likely need to run your script many times before it works correctly. Each time you rerun the script, it will need permission to overwrite any files that were created on previous runs, otherwise the script will crash.

To give your script permission to overwrite existing files, you'll need to access arcpy's environment properties (i.e. **env**) and set the **overwriteOutput** property to **1** (i.e. true). The env is a toolbox within arcpy that allows you to set global parameters which can affect the operation of all the ArcGIS processes in your script.

Example code – setting up arcpy



```
# import modules...
import arcpy

# check out extensions...
arcpy.CheckOutExtension("spatial")

# allow outputs to be overwritten...
arcpy.env.overwriteOutput = 1
```

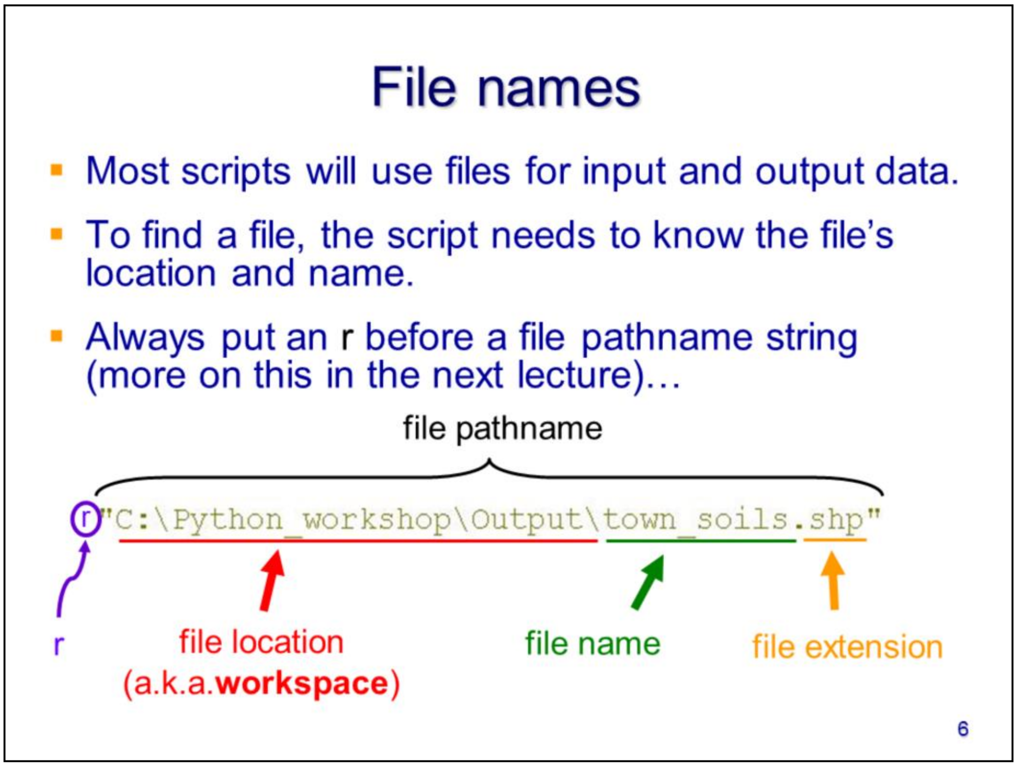
5

Let's look at an example of how the steps we've just discussed will appear in a script. Notice that the lines that start with the # symbol are colored in red – these are comment lines that the script will ignore. The lines below the comment lines are the actual script statements that contain instructions for the computer.

1) The 1st statement in the example script imports the arcpy module which gives the script access to ArcGIS's capabilities

2) The 2nd statement checks out an ArcGIS extension. If your script doesn't need any extension tools, then skip this step in your script.

3) The 3rd statement gives the script permission to overwrite existing files.



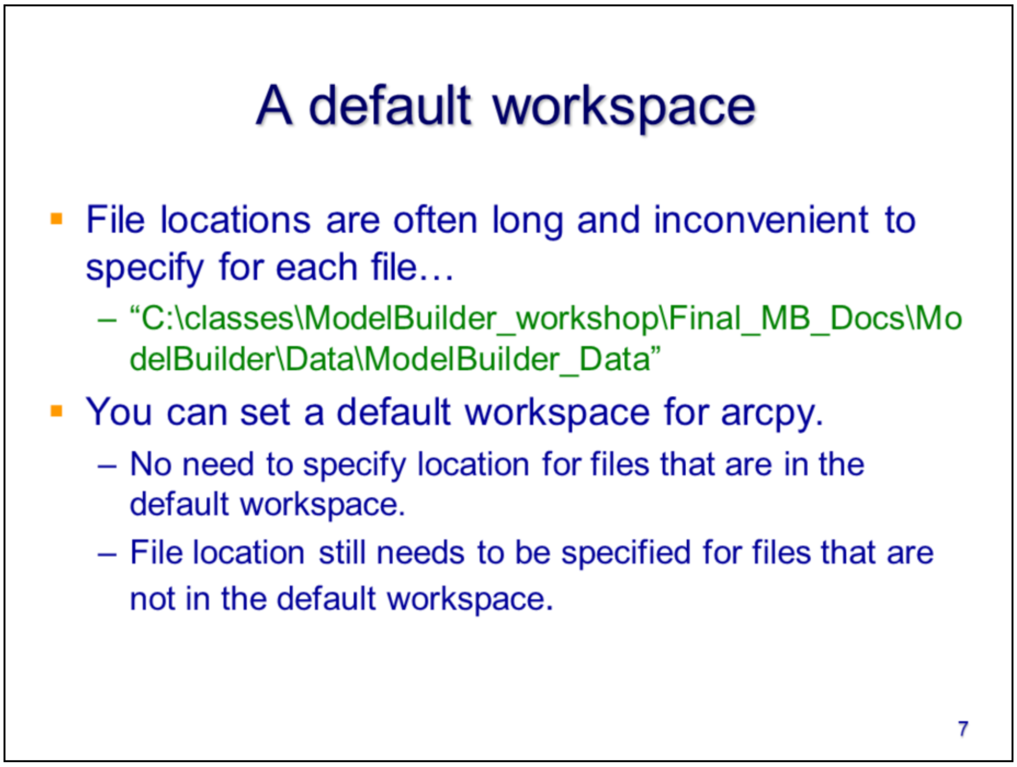
File names

- Most scripts will use files for input and output data.
- To find a file, the script needs to know the file's location and name.
- Always put an **r** before a file pathname string (more on this in the next lecture)…

file pathname
r"C:\Python_workshop\Output\town_soils.shp"
r
file location
(a.k.a.**workspace**)
file name
file extension
6

In most scripts, you will need to work with data stored in files. In a script, file names are always treated as strings and so they're enclosed in quotes. In Python, you should always put an **r** before a file name – for reasons we'll discuss in detail in the next lecture.

In this course, I'll refer to a complete file name as the "file pathname" – this contains all the information needed for the script to locate the specific file. The first part of the pathname is the "file location" which is also called the "workspace." The "file name" is the name of the specific file and the "file extension" is the letters following the period after the file name. The file extension indicates the type of file.

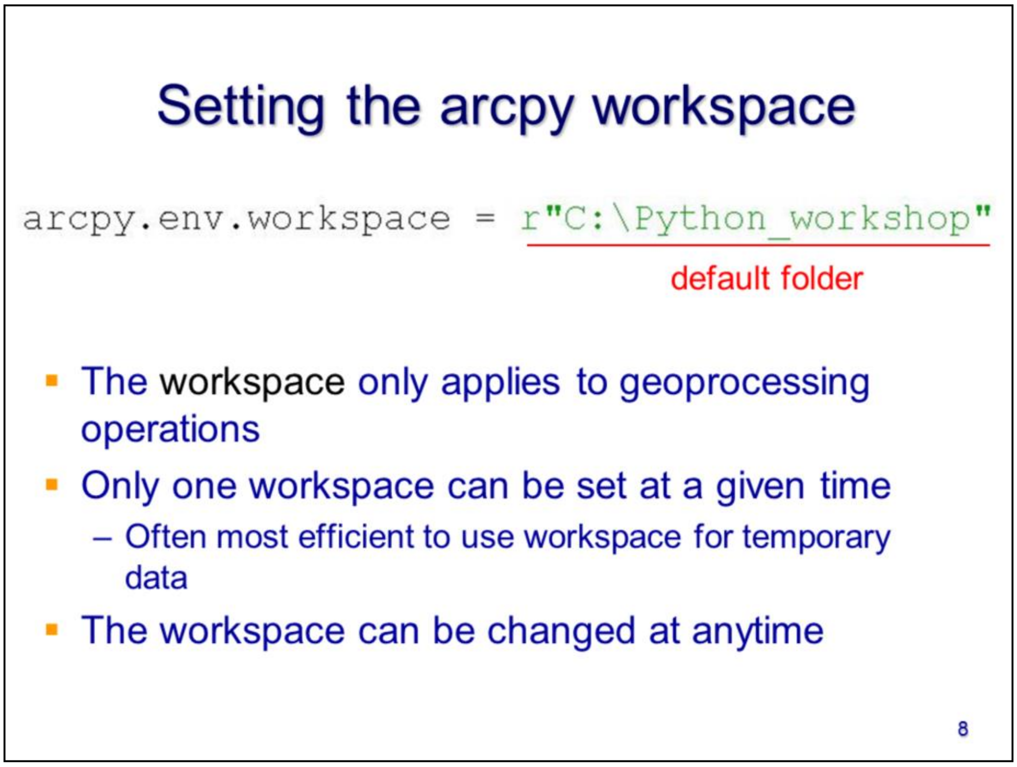


A default workspace

- File locations are often long and inconvenient to specify for each file…
  - "C:\classes\ModelBuilder_workshop\Final_MB_Docs\ModelBuilder\Data\ModelBuilder_Data"
- You can set a default workspace for arcpy.
  - No need to specify location for files that are in the default workspace.
  - File location still needs to be specified for files that are not in the default workspace.

7

When a script is working with multiple files that are stored in the same location (i.e. workspace), then we can set a default workspace for arcpy. Once the workspace is set, then you will no longer need to specify the file location for files that are located in that workspace. However, you will need to specify the workspace for files stored in other locations. Setting a default workspace can make the script more efficient to write and easier for you to read.



Setting the arcpy workspace

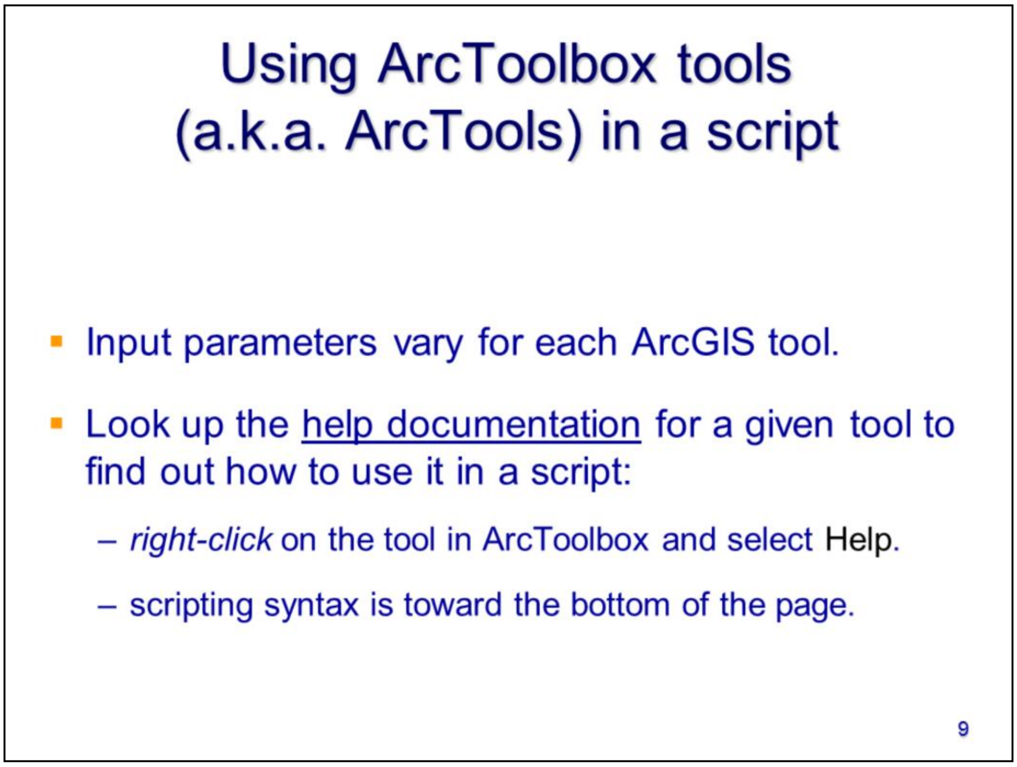
```
arcpy.env.workspace = r"C:\Python_workshop"
```

default folder

- The workspace only applies to geoprocessing operations
- Only one workspace can be set at a given time
  - Often most efficient to use workspace for temporary data
- The workspace can be changed at anytime

8

To set a default workspace for arcpy, you'll need to set the "workspace" property of arcpy's environment settings. Remember to include the **r** before the file location. Generally, you can set only one workspace for arcpy at a given time (the exception is when using the spatial analyst extension).

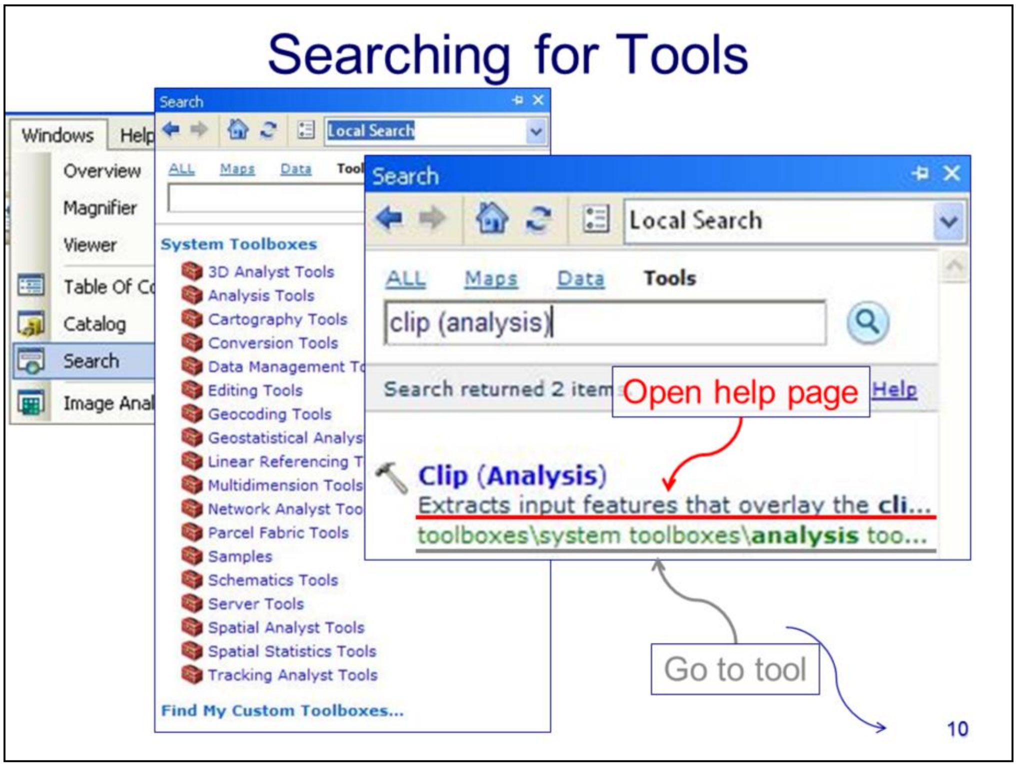


Using ArcToolbox tools (a.k.a. ArcTools) in a script

- Input parameters vary for each ArcGIS tool.
- Look up the help documentation for a given tool to find out how to use it in a script:
  - *right-click* on the tool in ArcToolbox and select Help.
  - scripting syntax is toward the bottom of the page.

9

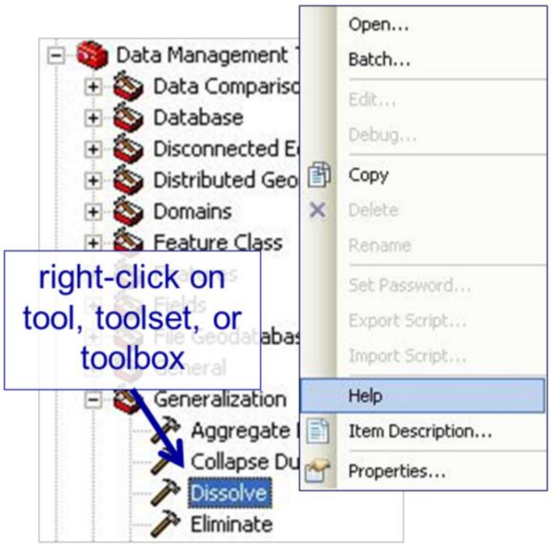
Now that we've taken care of the preliminary set setup of our script, we can start working with ArcTools (i.e. tools in ArcToolbox). There are hundreds of ArcTools that are available for use in a script and each tool requires a different set of parameters to make it work. Fortunately, ArcGIS contains thorough documentation on how to use each ArcTool and run them in a script.



Searching for Tools

To find a tool in ArcToolbox, you can use the **search** window in ArcGIS or ArcCatalog. In the search window, select the “Tool” tab and then enter the name of the tool below. The search will return a list matching the name you entered – click on the links below to open the help page or to be taken to the tool in ArcToolbox.

Tool Help Documentation

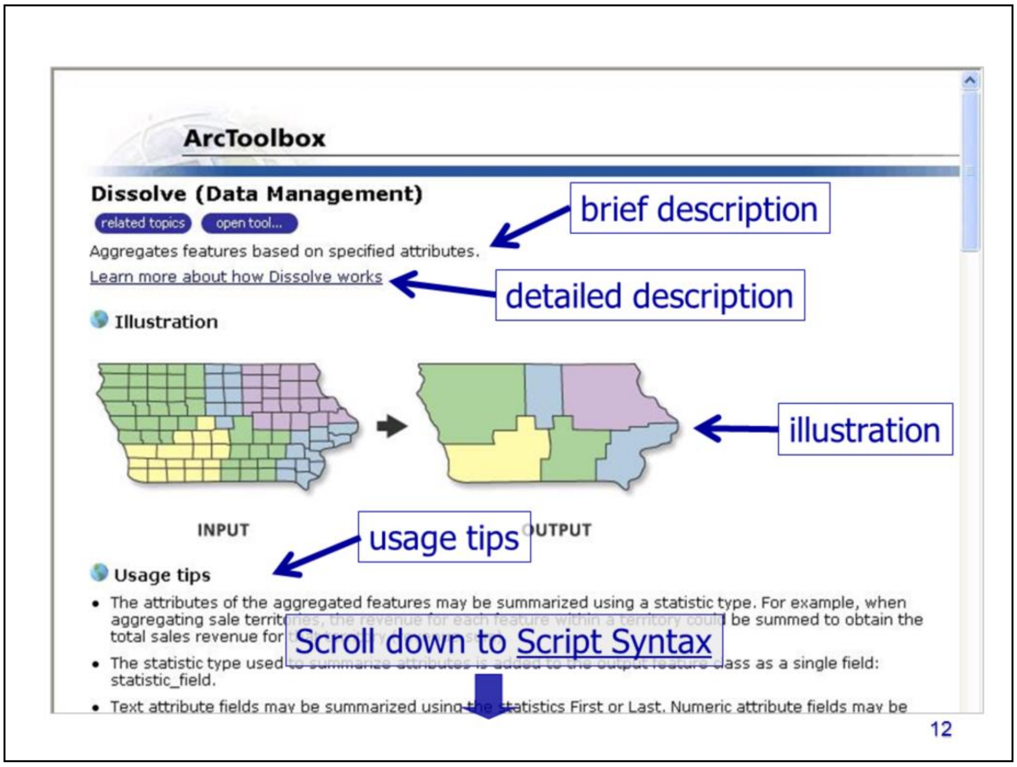


Help documentation describes…

- tool's purpose
- how tool works
- usage tips
- how to use in command line or script

11

The documentation for ArcTools can be accessed in ArcToolbox either through ArcGIS or ArcCatalog. Locate the tool of interest in ArcToolbox and right-click on it, then select Help from the menu.



Dissolve (Data Management)

Aggregates features based on specified attributes.

Learn more about how Dissolve works

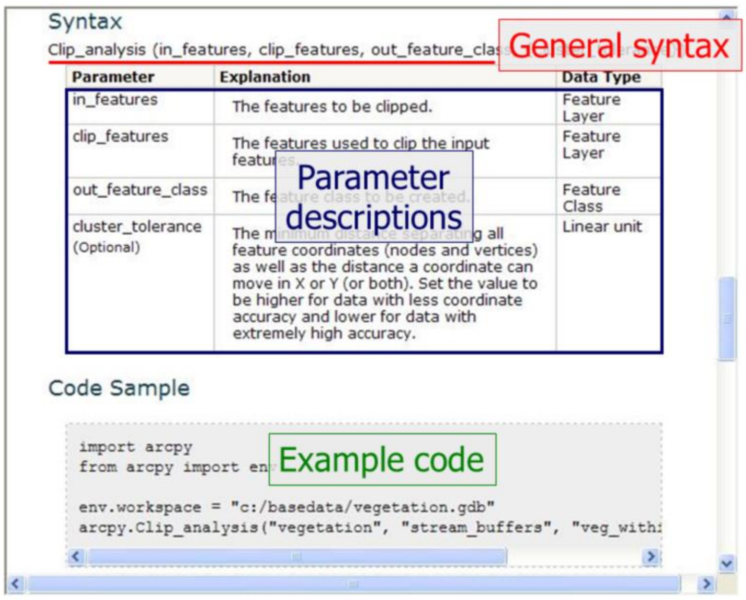
Illustration

Usage tips

- The attributes of the aggregated features may be summarized using a statistic type. For example, when aggregating sale territories, the revenue for each feature within a territory could be summed to obtain the total sales revenue for
- The statistic type used to summarize attributes is added to the output feature class as a single field: statistic_field.
- Text attribute fields may be summarized using the statistics First or Last. Numeric attribute fields may be

The tool help page will include a description of the tool and tips on how to use it properly. There will be a section for Python scripting at the bottom of the page.

Tool script syntax



Syntax

Clip_analysis (in_features, clip_features, out_feature_clas

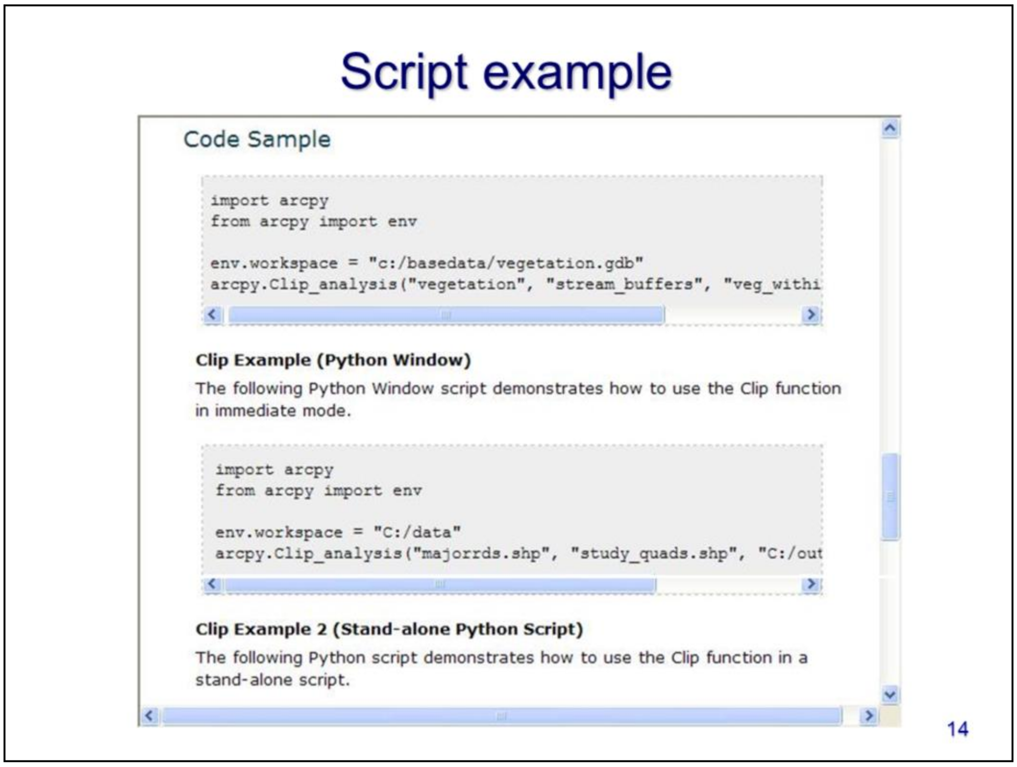
Parameter	Explanation	Data Type
in_features	The features to be clipped.	Feature Layer
clip_features	The features used to clip the input features.	Feature Layer
out_feature_class	The feature class to be created.	Feature Class
cluster_tolerance (Optional)	The minimum distance separating all feature coordinates (nodes and vertices) as well as the distance a coordinate can move in X or Y (or both). Set the value to be higher for data with less coordinate accuracy and lower for data with extremely high accuracy.	Linear unit

Code Sample

```
import arcpy
from arcpy import en

env.workspace = "c:/basedata/vegetation.gdb"
arcpy.Clip_analysis("vegetation", "stream_buffers", "veg_withi
```

13

The “Script Syntax” section includes the **general syntax** statement, a description of the tool’s parameters, and an example code from a Python script that implements the tool.



Script example

Code Sample

```
import arcpy
from arcpy import env

env.workspace = "c:/basedata/vegetation.gdb"
arcpy.Clip_analysis("vegetation", "stream_buffers", "veg_withi
```

Clip Example (Python Window)

The following Python Window script demonstrates how to use the Clip function in immediate mode.

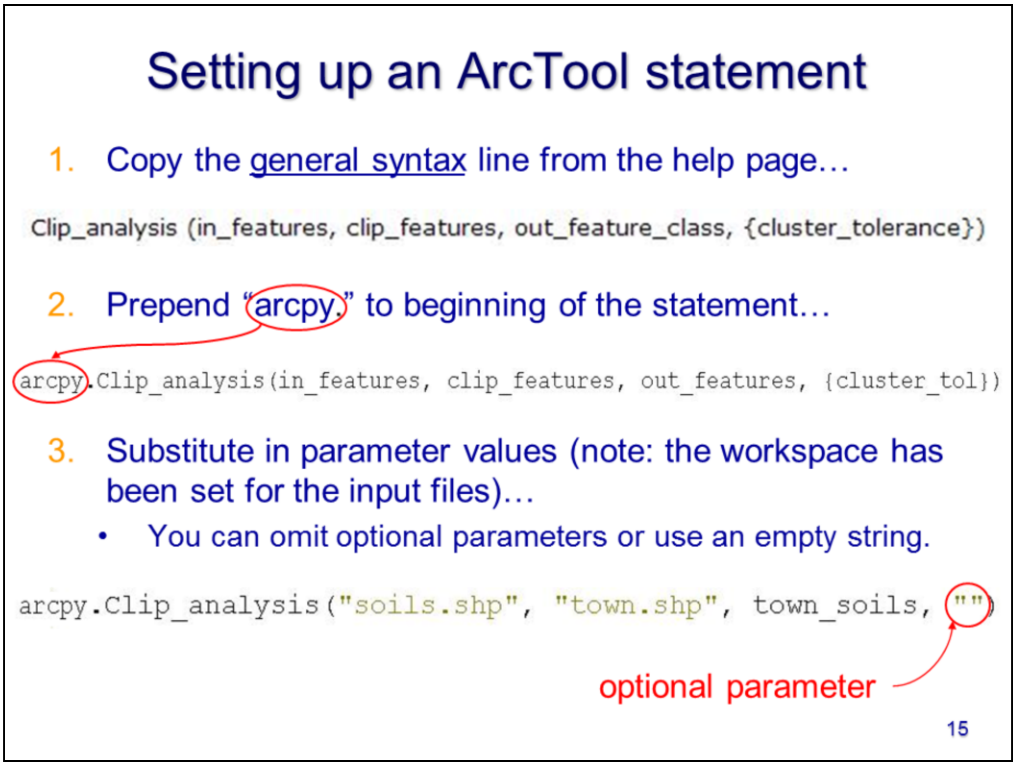
```
import arcpy
from arcpy import env

env.workspace = "C:/data"
arcpy.Clip_analysis("majorrds.shp", "study_quads.shp", "C:/out
```

Clip Example 2 (Stand-alone Python Script)

The following Python script demonstrates how to use the Clip function in a stand-alone script.

14

In this course, we will focus exclusively on “stand-alone” scripts – be sure to use the correct sample code as your guide. The Python window is for use within ArcGIS and the syntax may be somewhat different.



Setting up an ArcTool statement

1. Copy the general syntax line from the help page…

```
Clip_analysis (in_features, clip_features, out_feature_class, {cluster_tolerance})
```

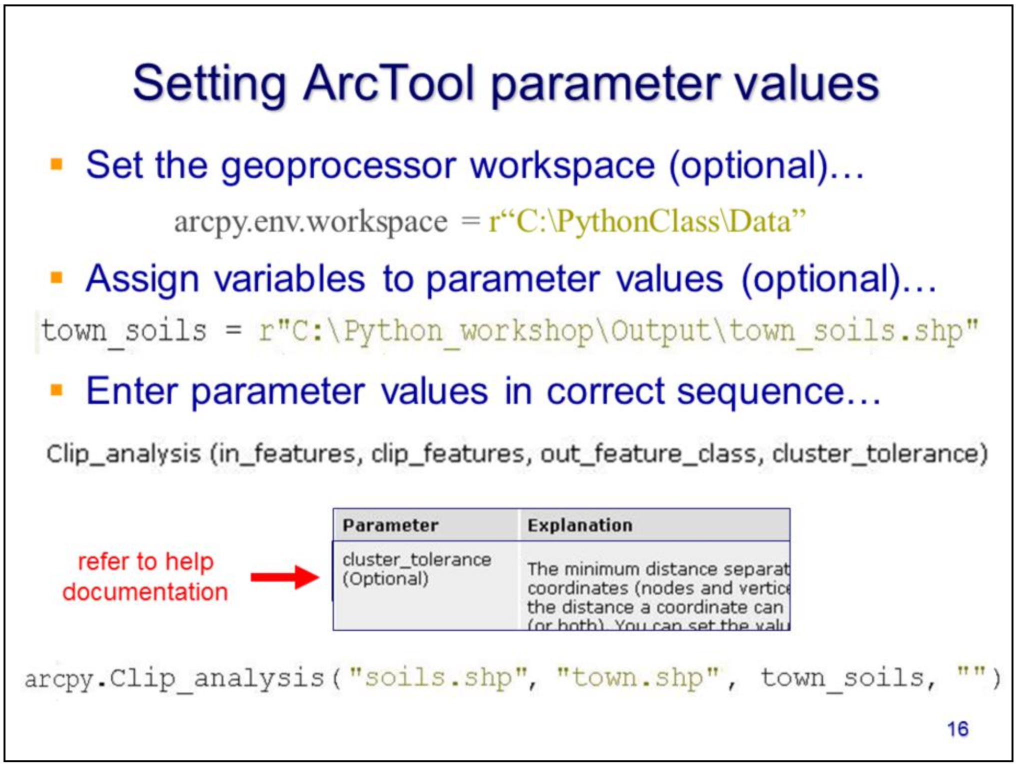
2. Prepend "arcpy." to beginning of the statement…

```
arcpy.Clip_analysis(in_features, clip_features, out_features, {cluster_tol})
```

3. Substitute in parameter values (note: the workspace has been set for the input files)…
   - You can omit optional parameters or use an empty string.

```
arcpy.Clip_analysis("soils.shp", "town.shp", town_soils, "")
```

To implement an ArcTool in Python, I recommend copying the general syntax line directly from the help page into your script – this will avoid any typos and also help you to put the parameter values in the correct order. The parameter values should be specified in the same order as the parameters listed in the general syntax line – the order allows arcpy to match the values to the correct parameters. Empty quotes can be specified for optional parameters if you want to use the default setting.



Setting ArcTool parameter values

- Set the geoprocessor workspace (optional)…

```
arcpy.env.workspace = r"C:\PythonClass\Data"
```

- Assign variables to parameter values (optional)…

```
town_soils = r"C:\Python_workshop\Output\town_soils.shp"
```

- Enter parameter values in correct sequence…

```
Clip_analysis (in_features, clip_features, out_feature_class, cluster_tolerance)
```

refer to help documentation

Parameter	Explanation
cluster_tolerance (Optional)	The minimum distance separat coordinates (nodes and vertic the distance a coordinate can (or both). You can set the valu

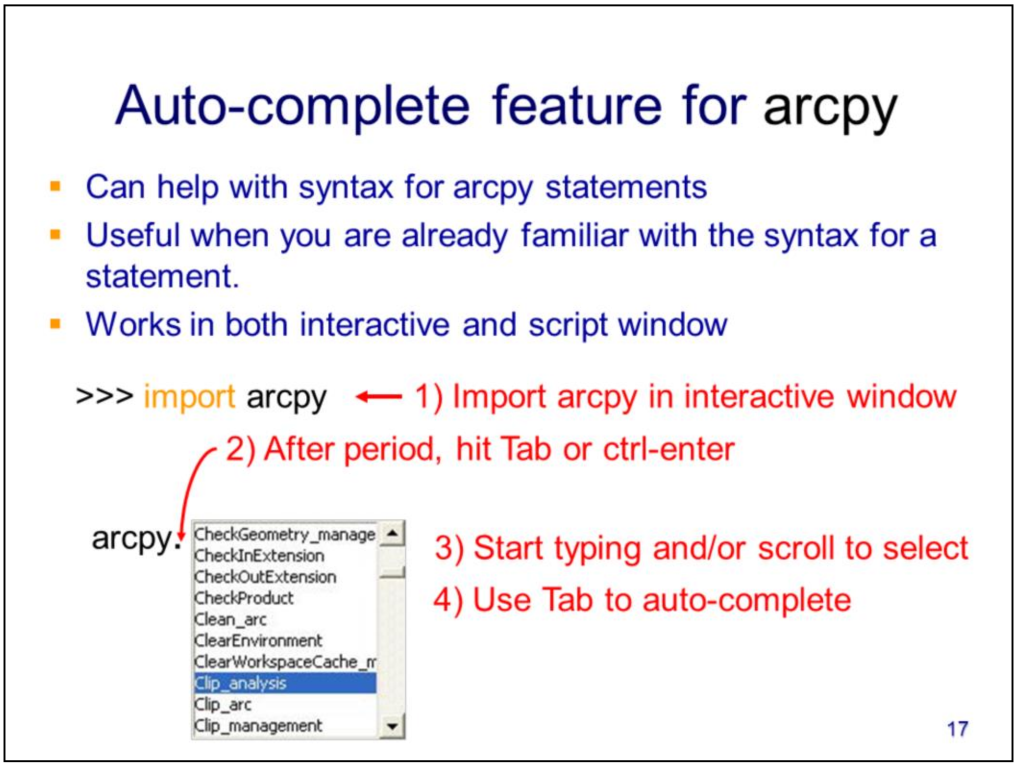
```
arcpy.Clip_analysis("soils.shp", "town.shp", town_soils, "")
```

16

Let's step through the process of writing an ArcTool statement in Python.

1) The first step sets the default arcpy workspace and is optional. I've set the arcpy workspace to my "input data" folder so, in the ArcTool statement, I won't need to specify the file location for any files located in this folder.
2) The output file in this example is not in the arcpy workspace so I've assigned it a variable which will make it more convenient to refer to this file in the ArcTool statement.
3) To write the actual ArcTool statement, first copy the general syntax line from the help documentation page to your script. Then add **arcpy.** to the beginning of the statement to create the proper syntax. For each parameter, check the description in the help page to understand the values required for each parameter, then enter the parameter values in place of the parameter names:
    1) In this case, the first parameter is the "input_features" and is a required. For this parameter, I've specified "soils.shp" – I don't need to include the file location because the file is located in the arcpy workspace that I set on a previous line.
    2) The second parameter is the "clip_features" which is also required. I've specified "town.shp" which is also located in the arcpy workspace.
    3) The 3rd parameter is the "output_features" which is required – here, I've specified the town_soils variable which I defined on a previous line.

4) The final parameter, “cluster_tolerance”, is optional so I can choose to ignore the parameter. In this case, I’ve use empty quotes so that the default parameter value will be used. Alternatively, I could have omitted the quotes since there are no further parameters that I want to specify.



Auto-complete feature for arcpy

- Can help with syntax for arcpy statements
- Useful when you are already familiar with the syntax for a statement.
- Works in both interactive and script window

```
>>> import arcpy
```

1) Import arcpy in interactive window

2) After period, hit Tab or ctrl-enter

```
arcpy.
```

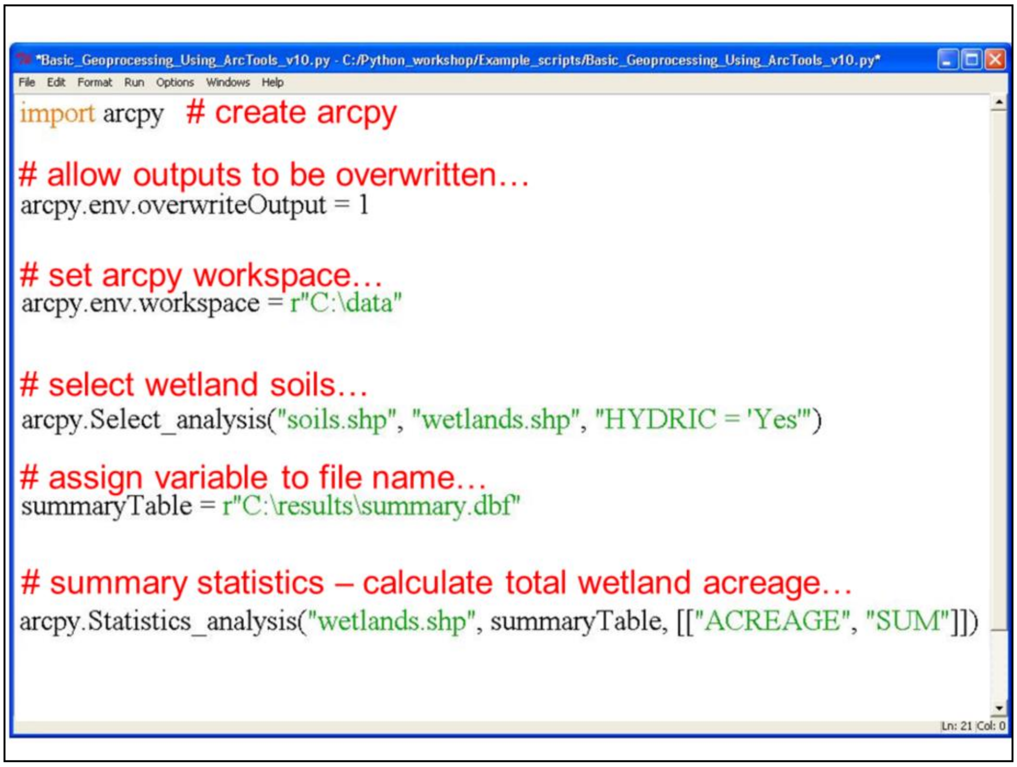
CheckGeometry_manage
CheckInExtension
CheckOutExtension
CheckProduct
Clean_arc
ClearEnvironment
ClearWorkspaceCache_m
Clip_analysis
Clip_arc
Clip_management

3) Start typing and/or scroll to select

4) Use Tab to auto-complete

17

Python has an auto-complete feature which can help you enter statements more quickly and without typos. To use the auto-complete for arcpy statements, you will first need to import arcpy in the Python Shell. Once arcpy has been imported, auto-complete will work in both the Python Shell and the script window.


Basic_Geoprocessing_Using_ArcTools_v10.py - C:/Python_workshop/Example_scripts/Basic_Geoprocessing_Using_ArcTools_v10.py
File Edit Format Run Options Windows Help
Ln: 21 Col: 0

```
import arcpy   # create arcpy

# allow outputs to be overwritten…
arcpy.env.overwriteOutput = 1

# set arcpy workspace…
arcpy.env.workspace = r"C:\data"

# select wetland soils…
arcpy.Select_analysis("soils.shp", "wetlands.shp", "HYDRIC = 'Yes'")

# assign variable to file name…
summaryTable = r"C:\results\summary.dbf"

# summary statistics – calculate total wetland acreage…
arcpy.Statistics_analysis("wetlands.shp", summaryTable, [["ACREAGE", "SUM"]])
```

In the script window, enter your script. Note that scripts run from the top statement down – so lines above can affect lines below, but not vice versa.