Python Essentials in ArcGIS I - An Introduction

Knowledge of Python is vital for ArcGIS professionals interested in furthering their automation and analysis skills. Python allows users to create custom data management or analysis tools ranging from single functions to complex multifunction processes with validation, which can be easily reused, shared, and even executed with little to no user interaction. This workshop will introduce the Python scripting language and show how it can be used to access and automate geoprocessing tools and functionality.

http://video.esri.com/watch/74/python-essentials-in-arcgis-i-_dash_-an-introduction

---

Video Transcription

00:01 Hello and welcome to User Conference 2010.

00:05 I hope that it's been good...a good couple of hours for everybody and hope that you have some plans...

00:11 ...to get some good information out of User Conference this year.

00:16 Today we're going to introduce Python and its role in ArcGIS.

00:20 We'll show some basic Python concepts and examine what Python scripting can do to improve...

00:26 ...your geoprocessing workflows.

00:29 Just kind of as a for your information, these presentation slides are going to be available on the User...

00:35 ...Conference Proceedings Web page and the DVD that gets sent out in a couple of weeks as well as...

00:41 ... on the geoprocessing resource center, so don't feel like you have to go crazy scribbling notes, trying to keep up.

00:48 Just try to absorb what we're talking about, what we're saying, and you can get what were...get the actual slides later on.

00:56 Also, we're going to try to take questions as we're going along, so if you feel brave enough to have others hear...

01:02 ...the questions that you have for us, please raise your hands and we'll try to spot you and get to an end point...

01:10 ...then take some questions.

01:12 But we have a lot to cover so we'll have to work quickly with that.

01:16 And of course we'll take some questions at the end, just personally if you'd rather do it that way.

01:25 So, we have an expected audience for this session and I just kind of want to use this as a little bit of a, an informal survey...

01:34 ...so our first point there is that you are new to Python scripting, or at kind of a beginner level...

01:39 ...so if you can just give me a wave if you're at about that level.

01:43 Okay, that's a lot of people.

01:47 And you're in the right spot. I don't want to make that sound bad.

01:50 So I guess one prerequisite that we have is that you're comfortable in ArcGIS.

01:55 So maybe you've used geoprocessing before, maybe ModelBuilder, some of that type of functionality...

02:02 ...but you really want to start using Python.

02:04 So maybe it's another wave out of those people familiar with ArcGIS geoprocessing.

02:10 Okay, how about those that are moving to Python, maybe from a different scripting language?

02:14 So you're familiar with some programming and just want to start learning Python. Okay, a good number.

02:20 And maybe you're none of these three or you're all of them, and you're just interested in what's new in 10.

02:25 And everybody should wave for that one. Two hands.

02:32 So here's what we'll be covering today.

02:35 We'll try to answer kind of the fundamental question of, Why should I use Python?

02:40 We'll move to a short section on some general Python basics.

02:44 We'll hope to kind of get everyone to a starting point so that you know some of the terminology and functions...

02:50 ...that we'll be talking about as we go through the presentation.

02:55 We'll talk about what's new for Python scripting and ArcGIS 10, so for everybody who is waving with...

03:01 ...two hands, that will be a key part.

03:05 Of course we'll cover executing geoprocessing tools in Python.

03:10 That's really the main reason that you're going to be here at this session today is because you want...

03:14 ...to do geoprocessing in Python, so that's kind of our starting point.

03:19 When we're talking about scripting in ArcGIS, we'll deal with messaging and error handling...

03:24 ...which are important when you're beginning with Python.

03:28 We'll get into some geoprocessing tasks that you can handle with Python and cover some ArcPy...

03:33 ...functions that you can use to accomplish those tasks.

03:37 We'll cover batch processing, which is, again, a very common use for Python, as well as receiving arguments...

03:43 ...to make your scripts very flexible and easy to share with, with your, with your users, the users of your scripts.

03:52 So let's get started with some Python scripting basics.

03:58 So the first question, I guess, and it might be on some peoples' minds, some others might understand it, but...

04:04 ...what is Python?

04:05 What is, you know, what is all the fuss that we're making about Python?

04:10 In Python scripting, I guess from our stand of view, from the Esri stand of view, is the scripting language...

04:17 ...of choice for ArcGIS.

04:20 And we like it for a number of reasons, but a couple of those reasons are because it's free for you to use...

04:26 ...there's no additional cost in using it, like with some scripting languages.

04:31 It works on Windows and Linux platforms, so it's cross-platform. For those of you who may not use...

04:37 ...Windows, it will work on other platforms.

04:40 And as well it's easy for beginners to use; it's a pretty straightforward language, once you get some basics down.

04:47 So it's easy to learn, but it's also very powerful for our advanced users.

04:53 But that question, I guess, still remains. Why should I use Python in ArcGIS?

04:58 There's definitely some overhead to learning Python and how it works in ArcGIS...

05:04 ...so the question is, is it really worth the time?

05:08 The short answer to that is yes.

05:10 I might be a little bit slanted on that perspective, but it's overwhelming, yes.

05:16 The long answer, I guess, we'll cover today and in other Python sessions this week.

05:22 So, Python, just at its most basic point of reference, I guess, for ArcGIS is a way to run geoprocessing tools.

05:32 You can run just a single tool or a string of tools, so like a geoprocessing workflow.

05:38 But the strength of Python is not just in executing single tools, but rather in executing those long...

05:44 ...and advanced workflows that we know that you make.

05:48 So, that ability to develop, execute, and share custom geoprocessing workflows that make Python scripting powerful is...

05:56 ...hopefully why you're here today and why you're interested in learning about Python.

06:01 And, from a strictly cost-benefit perspective, using Python to automate some of your geoprocessing work...

06:07 ...can be an enormous time-saver.

06:09 So, just I guess from a cost perspective it's going to help with that.

06:16 So let's start with just some basic Python concepts.

06:20 What is Python code?

06:21 Python code is...can be thought of as simply text, text in a text file that can be viewed in any text editor or text viewer.

06:32 But where you actually develop Python scripts and Python code is in an IDE, or...

06:38 ...an integrated development environment.

06:41 So that's a program where you can write, save, run, debug Python code.

06:46 Some examples of IDEs are Python Win IDL, as well as Wing, are some common IDEs.

06:56 Starting with ArcGIS 10, you can also execute Python straight from inside an ArcGIS application in the Python window.

07:06 So, we'll, we'll show that in some of our demonstrations and talk about it as well.

07:10 So when we're looking at a Python file, how do we know what different kinds of lines there are?

07:15 Essentially, how do I know which lines will execute, and which won't execute?

07:20 So, there's the lines that will execute, and anything that won't execute is usually thought of as a comment.

07:27 And a comment will always start with a pound sign. You can see it there on the example.

07:32 So that pound sign indicates that it's a comment and that line will not execute.

07:36 So comments are used kind of as placeholders or as a place where you can document the flow of your script...

07:42 ...and that line won't execute.

07:44 Everything else will try to execute.

07:47 Now, one of the very basic concepts in Python is a variable, and that's one of the building blocks...

07:53 ...that you'll use as you're writing scripts.

07:56 So the question is, what are these variables?

07:58 We'll use that word a lot today and so we need to start out with a good definition of that.

08:03 Essentially a variable is just a name that stores a value.

08:08 It's a sign using the equal sign, so you're just assigning some name equal to a raw value, and...

08:14 ...that value can be a string, sort of like a path to a dataset.

08:19 It can be a simple number.

08:20 Or it can be some other, more advanced Python data structures.

08:26 And these variables will just act as substitutes for any raw values.

08:30 You can use these variables...

08:31 ...many places in your scripts, including when you call geoprocessing tools, you can stick a variable in the...

08:37 ...geoprocessing tool, and it'll know what to do with that variable.

08:44 So, an...a very important part of Python is the ability to test conditions, so you're using branching logic to perform...

08:52 ...one operation if a condition is met, and another operation if that condition is not met.

08:58 The most common form of this is the "if else" statement, and that just reads something like, "if this is true...

09:06 ...execute these indented lines, and if it's not true, we'll execute some other lines."

09:13 So, a colon is put at the end of each condition statement and how the lines, like I said, are indented...

09:19 ...beneath that condition statement determines what will execute if the condition is met.

09:25 So there's many different operators for testing conditions.

09:28 Here we see mostly the double equal sign, and that's used to test equality, so that reads, Is equals to".

09:36 There's some other operators, like some mathematical operators, greater than, less than, less than or equal to.

09:45 There's many different operators you can use.

09:47 So here in our code sample we see we're just assigning the letter A, a string of the letter A, to a variable.

09:53 We'll then test that variable if the variable is equal to A.

09:57 We'll execute what is ever in that indented section beneath it, and that "if" or that "else" statement in the fourth line, I guess...

10:07 ...down is simply saying if the variable is not equal to A.

10:11 So that's the false part of the condition testing.

10:15 And if it's not equal to A, will do whatever is indented, so print variable is not A.

10:22 Another important thing that is built into the Python language is the ability to iterate or loop.

10:29 And this keeps us from having to reproduce any code that we want to run more than once.

10:36 And we put everything that we want to run or iterate beneath a loop statement and indent it.

10:43 So there's a couple of different kinds of ways that you can accomplish looping or iteration.

10:48 The first is a while loop and so you see just in this example we have x equal to 1, and while x is less than 5...

10:56 ...you just will go into that statement and do anything that is indented.

11:01 So in this case we'll print x and then we'll increase the iteration, the x value, by 1 to continue the iteration...

11:09 ...and this will execute until x is greater than 5.

11:14 Another kind of loop is a counted or range loop, so you can see in that middle example that we...

11:21 ...call a range statement, and we check for any values in the range starting with 1 and ending with 5...

11:28 ...and then we'll print the numbers that are inside that range.

11:33 A third kind of loop or iterator is a list loop, and that's one that we'll hit quite a bit today, and that's just you start...

11:40 ...with a list and a list is one of those advanced Python data structures that I talked about earlier.

11:49 So it can contain any numbers or data strings.

12:52 In this case it contains numbers.

11:54 And we'll just want to list, or we'll want to loop through or iterate through each item in that list...

12:00 ...and so for each number in x, our list, we will print the number.

12:09 A few other important things that we want to keep in mind as we start working with Python code is that variable names...

12:16 ...functions, geoprocessing tools that are called in Python, everything pretty much is case sensitive.

12:25 And if you start using different cases in your variables than what you originally set, you can run into this common...

12:32 ..."name is not defined" error, so that's one thing to look for.

12:37 Another thing is when you're specifying a path in Python, you should just go ahead and use forward slashes...

12:43 ...as the path separators instead of the common backslash.

12:49 There's additional characters you can use as these separators, but forward slashes work in both Windows and Linux...

12:55 ...so it's good to just stick, stick with that one.

12:59 Now, two important pieces that you can think of in Python are functions and modules.

13:06 And a function, you can kind of think of just as a tool.

13:10 It's a defined piece of functionality that will perform some specific task, so again, you can just kind of think of it like a tool.

13:18 A module is a Python file where functions live, and so often all of the functions in a module are related...

13:26 ...and these modules have to be imported into your Python scripts in order to be used.

13:31 So the example that I have there on the third bullet is, we'll use the math module and call the square root function...

13:39 ...from that math module, and provide a value and that function will then return a result value.

13:50 Okay, so now that we have some basic Python concepts in mind, let's start talking about Python scripting in ArcGIS.

13:59 So, in ArcGIS 10, geoprocessing in Python is done through ArcPy.

14:04 All geoprocessing tools are accessed, or can be accessed through ArcPy.

14:10 So this ArcPy module is a package that only provides...that not only provides access to geoprocessing tools...

14:17 ...but also several special functions, classes, and modules that will enhance the scripting experience in ArcGIS.

14:25 So there are certain functions, like the List feature classes function, that you can use to...

14:30 ...list all the feature classes in a workspace.

14:33 The Describe function to retrieve data properties and the number of printer functions to get and modify...

14:39 ...a table or feature class's attribute values.

14:44 There's a number of ArcPy classes that we can use to create complex data objects like spatial references...

14:51 ...field maps, and geometry objects.

14:53 Classes are a topic that will be covered in the next session, actually, immediately after this session.

15:01 And there are ArcPy modules that provide some extended functionality to be used with ArcGIS scripting.

15:09 So, there's a mapping module for performing automated mapping tasks, and there's a Spatial Analyst module...

15:15 ...for performing raster analysis and map algebra.

15:19 Now, ArcPy builds on the scripting experience of previous versions of ArcGIS that was known as ArcGIS scripting.

15:27 You can still use the ArcGIS scripting module in ArcGIS 10 and almost everything that we talk about today is...

15:33 ... valid using ArcGIS scripting in previous ArcGIS releases.

15:38 You'll just need to note that if you are on 9.3 or a previous release of the software before 10...

15:45 ...you'll need to refer to your documentation, because everything that we talk about today will be ArcGIS 10 specific.

15:52 But, the concepts are very similar so don't be scared off by that.

15:58 Also new at 10 is the Python window, which I mentioned earlier, and this is an embedded...

16:04 ...interactive Python interface actually built into ArcGIS.

16:07 And this is a great place to explore ArcPy and start running geoprocessing tools using Python syntax.

16:14 You can also access any Python functionality that you would be able to in a Python script file right here...

16:20 ...in the Python window in ArcGIS.

16:24 So, in that window you can execute a single line or multiple lines of codes, code, as well as load any script files...

16:31 ...that you've previously worked on, or save, save your work to a text file or a Python script.

16:39 All right, so I'll start our first demonstration now.

16:42 This is my colleague David Wynne; he'll be doing the demonstrations. Dave.

16:47 Hey, thanks a lot. Is anyone here from Minnesota?

16:51 Yeah, do you recognize this?

16:53 This is Hennepin County in Minnesota; you can kind of see...Minneapolis and St. Paul down in the corner.

17:00 Can you people in the back hear me all right?

17:03 Yeah? Yeah.

17:06 No. No.

17:11 I'm just going have to talk loud, I guess. Is that better?

17:13 [Unintelligible audience response]

17:21 All right. Well.

17:30 All right, well, I think I can do this one-handed, so we'll just try that and see what happens.

17:34 All right, so one of the things we added in ArcGIS 10 is the Python window.

17:40 So if you're here inside ArcMap...

17:42 ...there's this little button up on the main toolbar, Python window.

17:47 And it's a regular window, you can move it around, you can dock it anywhere you want.

17:54 So it's divided into two sections, and the top section is where you put your code in, so any of you want to...

17:59 ...run a geoprocessing tool ,you want to do any little bit of Python, you do it there.

18:04 In the bottom half, this is where Help shows up for tools, functions, other things, in addition...

18:10 ...to messages that might return from your tools.

18:14 So, if I wanted to start running a tool I can just start by typing in "ArcPy", and then when I hit that period...

18:23 ...it gives me a list of options.

18:27 And as I go through that the list will narrow and I can pick a tool.

18:32 Now, one thing of note, when you're running tools, and Drew kind of alluded to this already...

18:37 ...is that everything is case sensitive, right?

18:40 So you need to make sure that your tool name has correct upper case and correct lower case and as well the aliases...

18:47 ...this management bit for copy features is also required.

18:53 So, kind of what the scenario is that I want to try and do here is, you'll note that Hennepin County in Minnesota...

19:02 ...is bordered by several rivers.

19:06 And if I go in a little bit closer, you notice that the river doesn't really always follow the border of the county all that well.

19:18 I want to be able to take that river and move it to the county boundary, which is more accurate.

19:27 So I'm going to do a series of tools to accomplish this.

19:30 I'm going to use Copy Features tool, and then I'm going to use a couple new editing tools that we have at 10.

19:37 So, write the bracket, I get intelligent drop-down based on choices that are available to me.

19:43 So, I'm going to copy this Crow River feature class; now let's call it River Edits.

19:54 The reason I'm doing this, the next two tools, which are both editing tools, modify the output.

19:59 So, you know, do I really know what I'm doing? Maybe not.

20:02 So, I'm going to make a copy with the Copy Features tool, and then I can work directly off of my copy instead of my original.

20:11 So, I just hit Enter, the tool will start running...okay. And you see over here [unintelligible] got a new output added to my map...

20:23 ...and that's what I'll be working with.

20:25 So the next tool I'm going to use is the Densify tool.

20:30 So again, take the tool like this, and I'm going to pick the feature class I just created.

20:41 Now, densify second parameter is a keyword-based parameter.

20:45 So I get a drop-down based on the options that are available to me.

20:48 I'm going to densify based on distance, and I'm going to enter a string for the densify...

20:55 ...distance I want to use, which is 25 meters.

20:59 As you'll note as I'm going through these parameters, the help in the bottom threw a response...

21:03 ...to where I am in my list of options.

21:06 So every time I hit a comma, basically, to move on to another parameter, you can see another parameter being highlighted.

21:15 And below there, of course, is all the tool information, more about the parameters, essentially a lot of the stuff...

21:21 ...that you'd see in the help system that's been brought directly into the ArcPy wrappers for these tools.

21:31 So, I've done all the parameters I want here.

21:33 I'm going to close that off and run that again.

21:38 And I'm going to follow it up with the Snap tool.

21:41 The reason I'm actually densifying that line is to create a more uniform pattern of vertices...

21:49 ...something that...is closer to what I have in my county boundary.

21:54 So, again I'm going to use the river edits as my input.

21:59 And before I do the next parameter I'm just going to open up this tool so we can get a better look at what it actually does.

22:08 I'm just going through the search, opening the Tool dialog.

22:13 So you know this tool has two parameters, and the first one is in input features...

22:18 ...and the second one is this little more detailed parameter called Snap environment.

22:21 Now, sort of...see the snap environment is kind of like a table almost, it's a series of values that you can enter.

22:30 Or, when you have a parameter like that, or you have a parameter that's a multivalue where you can select multiple values...

22:37 ...these can be represented in Python by Python lists.

22:41 So, that's represented with a square bracket.

22:45 And then, because this is a table there's multiple rows, so each row can again, can also be represented as a Python list.

22:53 Another square bracket.

22:55 So I want to snap to my Counties feature class, I'm going to snap to the edge, and I'm going to set a distance of 25 meters.

23:14 Square brackets to close out each list, a round bracket to close out the tool.

23:21 And this will take probably about eight to nine seconds.

23:28 Now you see, okay, that finished, but it isn't really what I was hoping for.

23:32 You can see certainly that, you know, sure it snapped in some places but obviously I missed the mark.

23:39 So, one of the nice things about Python Windows is it's very easy to recall what you've done already...

23:46 ...and reuse that functionality.

23:49 So if I just start clicking my arrow key I can go back up and start over again right from the beginning and I can run that...

23:56 ...Copy Features tool again and re-create my temporary data, and I call Densify.

24:04 Then I'm going to try, make that just a little bit tighter, and do that at 10 meters, hit Enter, and then snap.

24:17 I’m going to be very generous and snap a total of 100 meters.

24:25 Do that a few seconds again.

24:32 All right, and now I think we've actually got what we were looking for.

24:35 So this is my input I had beginnings. You can see how far off it is off the line.

24:41 And now, with a series of tools and sort of an iterative process, I'm able to get the desired result that I'm looking for.

24:58 Okay, thanks Dave.

25:02 So like we saw in that demonstration, executing a geoprocessing tool in Python is pretty straightforward.

25:09 Only one line of code is really needed to execute a tool, so first ArcPy must be imported...

25:17 ...and then you'll just enter the tool syntax.

25:21 Now, the tool syntax is going to be an important thing, and it must always be followed with this convention...

25:28 ...and it's ArcPy.toolname_toolboxalias.

25:34 And then the parameters are specified at the end there.

25:38 So, each geoprocessing tool has some input and output parameters that must be specified for the tool to execute.

25:45 So here in my example we see, I import ArcPy, I set an environment, and then I execute a geoprocessing tool...

25:54 ...the Buffer Analysis tool.

25:57 Now as you're getting used to using Python to execute geoprocessing tools, there are some good places where...

26:03 ...you can go to see tool syntax and make sure that you're actually using a tool correctly in Python.

26:09 One of those is the image that's on the screen there, and that's the Results window.

26:14 If you right-click on a tool in the Results window, you can copy that tool's syntax as a Python snippet...

26:21 ...and so it will have all of the datasets actually right in line there, so when you copy it you can just...

26:27 ...paste that into a script tool or into the Python window for reexecution.

26:33 There's also functionality in ModelBuilder to export a model to a Python script.

26:38 So if you have some established workflows that are already models, a starting place for you is to export your models to scripts.

26:48 You can also drag tools from the ArcToolBox window or the Search window that Dave showed right into the...

26:54 ...Python window and it'll automatically drop in the correct tool syntax for you.

27:01 You can look at the tool documentation so each of the tools has a Help page that has both syntax information...

27:07 ... that show us how the tool is used in Python, as well as a more exhaustive script example that uses that tool.

27:17 You can also call the arcpy.usage function to get help and syntax information about a tool.

27:23 So you see there, arcpy.usage and then I just specify the name of the tool that I want help for.

27:32 Now, while setting environments in ArcPy is also very straightforward, it has many rewards in making your work easier...

27:39 ...and adding some additional control when you're running geoprocessing workflows.

27:44 For instance, there's the workspace environment, and if you're familiar with geoprocessing...

27:49 ...environments, you know that all of the data that you have, if you set a workspace environment...

27:56 ...you can reference that data by just its name instead of having to specify a full path.

28:03 And doing this in Python scripting is very helpful since it makes your script cleaner and easier to manage.

28:09 You don't have all these long paths in your script.

28:12 You can just specify the workspace once and then just use the names of your data instead of the full path.

28:19 There's also other environments, like output coordinate system and extent, that can be used in different ways...

28:26 ...and you can look those up if you're interested.

28:31 Now, when geoprocessing tools are run, messages are always raised about tool execution...

28:38 ...so when you're running just some geoprocessing tools from a tool dialog in ArcMap or ArcCatalog...

28:44 ...messages are displayed in that Geoprocessing Messages window where they can be followed to see how...

28:50 ...the tool is executing and there's a little progress bar in that.

28:54 Messages can also be seen in the Results window, and there's three different types of messages that tools will return.

29:03 That's informative messages, so that's just things like...that tell you when the tool execution started...

29:09 ...how long execution took, and if the process succeeded.

29:13 There's warning messages. That's kind of a step up from informative messages, and these warning messages...

29:18 ...make you aware of any potential problems that might have occurred during tool execution.

29:24 And there's, then of course, error messages, that indicate that there was a severe problem and that the tool actually failed.

29:33 So if the tool's executed in the ArcGIS Python window, and you may have...seen this in Dave's demonstration...

29:40 ...when you execute the tool, there's that special window pane that displays any messages that you have right there in the...

29:47 ...Python window; it was on the bottom part, so you can see any messages that occurred there.

29:52 Messages can also be read and displayed in Python, but you always need to explicitly call them...

29:58 ...using the ArcPy Get Messages function.

30:02 So, you can specify whether you want to return all messages, just the informative messages, warning messages...

30:08 ...or error messages, using the syntax that I have there in the slide.

30:15 So here's an example of using Get Messages to call geoprocessing tool messages.

30:20 So we'll want to execute a geoprocessing tool, in this case, the Buffer tool.

30:26 And then immediately after we execute the tool, we'll want to print the execution messages.

30:32 So that's as easy as saying "Print arcpy.getmessages."

30:36 And this will return all of the messages that may have occurred since we didn't specify a number...

30:41 ...there in the Get Messages call.

30:45 So, as you start working with Python, you're going to definitely run into some errors, and this can be pretty frustrating.

30:52 I had a starting point as a Python scripting users, and it was frustrating to me, so I'm sure it will be...

30:59 ...frustrating to any new users.

31:03 Errors can occur because a geoprocessing tool isn't being used right.

31:07 There might be typos in your script, or the script has some Python syntax errors.

31:14 Now, printing geoprocessing messages is a simple way to find errors that occurred when the tool...

31:19 ...was being executed in Python.

31:22 But Python error handling provides a more intelligent method for dealing when errors when they do occur.

31:29 One of the most common and useful forms of error handling in Python is the Try-Except statement.

31:35 So you can think of it this way. You will try to do something, like execute a geoprocessing tool...

31:41 ...or perform some computation, and if that fails, which you can think of as an acception, you can do some other steps.

31:51 And just one thing to make clear is if the operation in the Try statement does not fail, so if it executes correctly...

31:59 ...all of the code that's in that Except statement is just skipped over.

32:03 That, that Except statement is only for if something wrong happened with the code that was executed in the Try statement.

32:10 So Try-Except statements are really pretty easy to incorporate into your scripts.

32:14 You just, you know, can copy in a Try and Except and indent your code correctly, so they're easy to incorporate...

32:22 ...and you should really always use them in your scripts, since you might get into the case where you don't even...

32:29 ...know why the tool isn't...why the tool is failing, or why some Python error is being raised, and you...

32:37 ...might not even know about it unless you use some kind of error-handling technique like a Try-Except statement.

32:44 So here's just another short example showing...using the Try-Except statement.

32:51 So we will try to run the Buffer tool, but if an error occurs during tool execution because of perhaps a typo or some bad syntax...

32:59 ...or I'm using the tool incorrectly, the script will move then into the Except statement instead of simply failing and...

33:06 ...ending the script execution, so you'll get some information.

33:09 And, in this case, in my Except statement, I print the statement that Buffer failed, so an obvious message there...

33:15 ...and then also why the...why the Buffer tool failed, by getting any of the error messages using ArcPy Get Messages.

33:24 All right, I'll pass it back over to Dave for a demonstration of error handling and messaging.

33:33 All right, now, in the last demo, we worked on creating a little workflow inside of ArcMap.

33:40 But you know at some point you want to, you might want to be able to take, you know, work that you completed and flushed out...

33:46 ...and got to work, and then take it kind of to another level.

33:52 So you can always click inside the Python window and you can Save As, and you can save it out to the Python file.

34:03 And this is that file; it's basically just everything that I did in that last session.

34:08 Now, I don't know how you think this is going to work just as is.

34:12 Well, no, of course not, there's always something.

34:16 So, when you're working inside ArcMap and you work inside the Python window...

34:20 ...ArcPy is just there, it's imported for you.

34:24 Import any kind of functionality in...in Python...an import statement, import ArcPy.

34:33 Now, of course I went through a couple of different renditions so I'm going to delete that out too.

34:41 And, if I just try and run this, I can go up here and click on this little triangle button, it's called Go...

34:48 ...and that'll just execute script.

34:57 And give me a second and...

35:06 All right, well, hey, something didn't work. Of course.

35:11 Well, as Drew was sort of explaining there, there's some pretty simple mechanisms in Python to deal with...

35:16 ...handle errors so I'm going to start making this script a little more detailed by adding a try.

35:24 I can indent my code, and I type my except, and then I'm going to print out my messages.

35:36 Now, of course, when it comes to messages, you really only care when it fails.

35:41 It works, you know, hey, good, but if it fails that's what you're interested in, right?

35:46 So that's what the 2 is, print me any error messages.

35:50 One thing I sort of forget sometimes to do but you could also type in on a Try-Except, you can add an else.

35:56 So, you know, the way try-except works, it's going to try anything in that block; and if that doesn't work it's...

36:03 ...going to print anything, or do anything inside the except.

36:07 Well, if everything in the try works, then it will go down to the else, and I can say hey...some kind...

36:17 ...of message that, hey, this didn't work.

36:23 Now, when you start writing code, you know there's going to be several lines, and you go to run it...

36:30 ...and just nothing happens, you know.

36:34 Well, hey, down in the bottom there's this little tiny message, and it says "Fail to run script syntax error."

36:41 Well, you can also sort of confirm whether or not you have any syntax errors just by clicking on this little check button.

36:48 If I click that, you notice the cursor jumped down to the Except line and simply all I did was forget that colon.

36:57 So now that's all fixed up. Let's run this again.

37:01 Okay, now I actually got proper error message, so what happened here?

37:10 Okay, it just doesn't recognize that one of my datasets exists at all. And well, okay, why is that?

37:16 Well, I'm working inside [Arc]Map, I was working mostly with the layers.

37:20 Here I am in Python when it has no concept of where ArcMap is and what was in there.

37:25 So I need to able to reference those layers in some other way.

37:31 And one way I can do that is by setting geoprocessing environments.

37:36 So if you need to set an environment, it's arcpy.env, it's this environment class...

37:41 ...and it contains all of the environments.

37:44 So, I'm going to set my workspace to...

38:03 And, for good measure, you know, once in a while, you...you're going to run a script once, you run it again...

38:08 ...hey, it didn't run the second time, and that is because you're trying to overwrite something that you did the first time.

38:14 So in Python, you want to overwrite datasets and feature classes. You set this Overwrite output option to True.

38:24 So now if I run this, and this time I'm going to click on this Run button and that's going to allow me to step through...

38:29 ...on a line-by-line basis.

38:32 So I hit that, this little dialog pops up.

38:35 I want to supply any arguments I can. I don't.

38:39 I want to step through the debugger, though. Just click that option and hit OK.

38:45 And by hitting either this button up here, Step Over button, or just hitting F10, I can step through on a line-by-line basis.

39:02 Well, somewhere I made another mistake, and I can't find it.

39:08 You see it?

39:17 Well, in any case. What should happen, and this is maybe another example of the error...oh, yes, Drew caught it.

39:29 I have a very long, complicated folder structure.

39:33 Sometimes keeping it simple, right, makes it easier.

39:36 Okay, so let's go back there again.

39:39 Click on my Run button. I'm going to step through in the debugger.

39:44 F10, set my workspace and my overwrite output options, run copy features.

39:53 Run Densify and run Snap, which if you remember took about eight seconds.

40:04 And when, hey, that all worked, it drops down inside the else, prints out that last message down in the bottom...

40:11 ...hit F10 one more time, and I'm done.

40:16 Three lines! Well, I think I'm pushing my luck so I'm going to turn it back to Drew.

40:29 Thank you, Dave, and you need to work on that folder structure.

40:37 All right, so at this part of the session we're going to start talking about some actual hands-on, kind of real-world...

40:45 ...geoprocessing tasks that you might want to kind of get started working on, in Python.

40:54 So in the ArcGIS help, there's a topic that's pretty much just a flat list of all of the functions that are available through ArcPy.

41:04 So, I am going to launch to the Web help for ArcGIS, and this is the topic that I was saying, the alphabetical list...

41:18 ...of ArcPy functions, and I just want to kind of introduce some of the ArcGIS 10 help structure...

41:26 ...before we get to moving on that.

41:28 So, everything geoprocessing will now be in this professional library, and so that's kind of the first folder that...

41:35 ...you'll want to get into.

41:37 Then under Professional Library, we have a geoprocessing book.

41:41 Under this geoprocessing book, there's several sections; one of them is All About ArcPy...

41:50 There's also this Geoprocessing with Python book that contains some additional information for...

41:56 ...working with scripting, working with geoprocessing in Python.

42:01 So under this ArcPy site package book we have another section that's just about all the functions.

42:08 Now this is some of the other things that I was talking about that Python, or that ArcPy is composed of...

42:14 ...these functions classes, modules, the bottom three are modules, but we want to just get started with some functions.

42:23 So here's the topic and you can reference that if you're ever curious about some of the functionality...

42:27 ...that's available through ArcPy.

42:34 So, ArcPy functions.

42:37 These are very useful in Python scripts, and they were actually designed to support and...

42:42 ...enhance the geoprocessing workflows that you want to do in Python.

42:48 So we've already seen how the Get Messages function is used to call geoprocessing messages within a script.

42:56 There's another group of functions that is a very powerful group, and this can be used in many geoprocessing...

43:04 ...workflows related to batch processing, and so these are the Listing Data functions, and they're just used to do things...

43:14 ...like make a list of all feature classes in a workspace, or all rasters in a folder, or all fields in a table or a feature class.

43:24 There's also another function, the Describe function, that allows us to get the properties of almost any kind of data...

43:31 ...so we'll talk about that today as well.

43:35 Now, if you think about it, there's a manual way in ArcGIS to do each of these tasks, whether that's getting...

43:41 ...the properties of some data; you can right-click on a feature class in ArcCatalog and get, you know, any properties...

43:48 ...that you can think of.

43:50 You can do batch processing by right-clicking on a geoprocessing tool in ArcToolBox window and...

43:55 ...selecting the batch mode, or by using some ModelBuilder.

44:00 However, these ArcPy functions that we'll talk about provide a way to automate all of these tasks...

44:08 ...that would traditionally have to be done manually.

44:10 And that's really the value of Python scripting is that...ability to automate, automate your work and make it easier.

44:20 So, one of the most common scripting tasks is batch processing, and you can think of that as just running...

44:26 ...a geoprocessing operation, whether that's a single tool or a workflow, running it multiple times.

44:33 So, a simple example of this is clipping every feature class in a workspace to some common study area boundary.

44:40 Now, instead of running the Clip Tool from the tool dialog, you know, 5 times or 10 times, once for each...

44:46 ...feature class, you can write a Python script with just a very few lines that performs this batch operation.

44:55 So batch processing tasks often use the ArcPy list methods that I showed in the previous slide to create a list of data...

45:03 ...and then perform some operation on all of the data that's in that list.

45:07 And there are several of these list functions, like List Feature Classes, list Rasters, List Fields, and several others.

45:17 So here's a short code sample showing how the List Feature Classes method is used to create a list...

45:22 ...of all feature classes in a feature dataset.

45:25 So we just need to need to set a workspace environment.

45:28 Many of the List functions will require that a workspace be set.

45:34 So we specify a workspace, and then we'll create a new variable, FC List, which will be equal to the list of all...

45:41 ... polygon feature classes in that feature dataset.

45:45 And you see that List Feature Classes has parameters, the star in the polygon word, and that first parameter...

45:51 ...that's the star, is just used to filter the feature classes that will be included in the list by their names.

45:58 And so with that star there, that's just a wildcard character that, no matter what the name of the feature class...

46:03 ...it will be returned in the list.

46:05 You can do something like s, and then the star, and anything that starts with the letter s would be returned in the list.

46:11 Only those that start with the letter s, actually.

46:15 In that second parameter, you're able to specify a shape type that you want to use as a filter.

46:22 So if we put "polygon" there, only the polygon feature classes will be returned in the list.

46:28 We'll then use a list loop there in the bottom of the sample, so for "FC" in FC List, FC List is that list that we created...

46:39 ...and we'll just want to print the name of that FC.

46:42 And so if we would do this, it would return these three feature classes that are highlighted...

46:46 ...and those are polygon feature classes in that feature dataset.

46:54 So, another one, and I'll try to get through this slide quickly so we can see another demonstration.

47:00 But, in this one, we'll use the List Fields function to create a list of all of the fields that are in a table...

47:07 ...or in this case a feature class, a shapefile feature class.

47:11 So we'll loop through each of the fields, so four fields in Fields, Fields is the list.

47:17 We want to print the field name and the field type, and the objects that are returned in list fields are...

47:23 ...a little bit different than in our previous list example, List Feature Classes.

47:29 List Fields actually returns a group of field objects, and these field objects have a number of different properties...

47:36 ...like name, type, alias name, that you can use to get the specific information about that field.

47:45 All right, I'll turn it over to Dave for a demonstration of batch processing.

47:51 All right, well, a few weeks ago I was perusing the ArcGIS Online and I found this dataset had 7...800...

47:58 ...900 feature classes, a range of trees in North America, and personally this is very exciting, you know...

48:08 I'm all about trees when I'm not about GIS.

48:11 So, anyway, I copied this down and I'm sort of concerned to work through that ever since.

48:17 One of the things I thought I could do here was sort of take that big mass of feature classes and do something...

48:24 ...more productive with them.

48:28 So, really all I'm going to do with, in the first part, is just sort of walk through that geodatabase and list out a series...

48:38 ...of features classes in that, and then I'm going to see what I can do with them.

48:42 So, I'm just going to step through this code on a line-by-line basis.

48:49 I'm going to import ArcPy. I'm going to import something called OS.

48:53 This is a module, sort of Python, sort of gives you access to operating system-level tools.

48:59 It's very good for working with paths, and, well, paths in particular.

49:05 Now, this is a little trick I like to use.

49:07 You saw in the last demo how he totally blew getting the path right.

49:10 Well, this little thing here will always give you the location where your Python script sits.

49:20 So, if I just type that in down here...so, you know, you see how I can use that...

49:24 ...that I always know where my location is, I can build that into other stuff, and I don't have to physically type it in and usually mess it up.

49:35 So I'm going to use this because in my folder I have my Python script, and then I also have a series...

49:40 ...of geodatabases, so I can use them relatively.

49:44 I'm going to set my overwrite output option to True.

49:48 And then I'm going to set my workspace, and my workspace I'm going to set to a combination of my home directory...

49:58 ...and this geodatabase I found on ArcGIS Online.

50:05 And the reason I'm doing that is most of the List functions sort of need to be educated.

50:09 Well, you know, what do you want me to list feature classes on?

50:13 I'm basically telling it, educate, use this feature...this workspace.

50:20 So of course I said there was a lot of feature [unintelligible] that, so I'm going to put in a key word, this “Ilex” a wild card...

50:26 ...and that's going to return lists of all the feature classes that are basically holly tree species.

50:36 Capture that list. The list is a lot of different things you can do with it, and one of it is simply the length of it.

50:45 So this message prints that out; there are 13 feature classes returned from the feature classes.

50:50 And then a list, of course, allows...lends itself very nicely to iterative type processes, so I can walk through that list...

50:57 ...and a photo loop, one at a time; you know, just start printing those off.

51:05 Now, of course, you know, just walking through something and printing it really, you know, isn't that exciting.

51:10 So...and this other code down here that I wrote, we'll do something a little bit more.

51:18 Now the one thing Drew kind of spent a little time talking about lists, but if you want to just look at a list, you can just type...

51:25 ...it in here, like right here at the bottom, and it will reveal its contents.

51:30 You want to sort that, you know, or you want to append something to it.

51:36 Lists allow certain sort of indexing and slicing, so if I want to get a specific value from a list, I can just put in a number...

51:43 ...that'll give me the first one. If I want to get a range out, put in a number, colon, and then another number...

51:50 ...and that'll give you a series of feature classes out of that list.

51:55 In any case, I'm going to go up here and I'm going to put a break point on this line here, and I'm just going to click this Run button...

52:08 ...and that'll run through everything I've already done again a second time and come down here.

52:14 Now, usually when you run geoprocessing tools, and here I'm creating Create File Geodatabase...

52:20 ...you don't really care about what is, like, the results object.

52:23 Every tool, when it runs, returns a results object.

52:26 When I was running through that workflow before, copy features, snap and densify, it didn't matter that much.

52:33 But I'm going to use...I'm going to capture that, and then this result has a number of properties and methods...

52:42 ...and one of the things it has is this Get output.

52:46 And this is, for this case, just a shortcut for me so that I can get the full path of this new geodatabase.

52:57 And the next few lines I'm going to run to make Feature Layer tool, because I have a feature class of the states...

53:03 ...of the U.S., but I don't want to clip my tree database off of the entire United States, I want to base it off of a specific state.

53:13 So I'm writing this query, and I'm going to pick out from my layer just the state of South Carolina.

53:21 Now you notice to this point, most of the time when I have had lines, it's all been based on one line.

53:27 The way I've kind of spread it out here is just sort of a nicer way in Python of making it a little bit more readable.

53:34 Certainly this treats it as it's the same line. You can imagine if I tried to actually put this on one line.

53:40 You know, it's very difficult to explain to anyone what's actually happening.

53:48 So if I continue to step through...create my layer, then I'm going to walk through that list...

53:58 ...supply each item in that list individually to the Clip tool.

54:05 I'm going to clip it with that layer I just created.

54:09 And then I'm creating a path on the fly with the feature class name, and that'll be my output.

54:20 Hit F10.

54:21 Now when Drew was talking about Get Messages earlier, that reveals all your messages to you.

54:26 I just want something that kind of gives me a little bit of indication that things are moving along so I'm...

54:31 ...using a slightly different method called Get Messages, and that basically takes all the messages...

54:36 ...and just gets a specific one.

54:38 So, I'm going to print out the first one.

54:42 You see the first tool succeeded.

54:48 Second, and if I just hit F5, that will continue to run through that list and clip each of them in time...

54:55 ...until the list has been exhausted.

55:07 See the messages will continue to pump into the bottom half.

55:17 And to prove this is actually working...if I go up to ArcMap...

55:29 ...now it's done.

55:37 So I can turn these on or off as I want.

55:43 Oops. Try that again.

55:52 And I step through, then you can see that each of these layers has been clipped to the boundaries of South Carolina.

56:07 Okay, thanks, Dave.

56:11 All right. Another function that you'll get a lot of usage out of, and that's very good to include and often necessary...

56:19 ...in your scripts, is the Describe function, and that's used to get the properties of really any kind of data.

56:27 So, when data is described using the Describe function, an object is returned that has a number of dynamic properties...

56:34 ...that are determined by the type of data that was actually described.

56:39 So among these properties can be things like data type, shape type, the spatial reference of a feature class...

56:46 ...the extent of the features, or simple things like the path to that dataset.

56:54 So, again, I will launch the help system, and so again we're under this functions book in the ArcPy site package...

57:06 ...and this is just a description of what the Describe function is and how it can be used.

57:13 Here is a list of all of the different data types that can be described, so there's things like feature classes, tables...

57:20 ...any other kind of data really that you can think of to use in ArcGIS.

57:24 And, at the most basic, when you describe a piece of data, you'll get the Describe Object properties...

57:31 ...and so these will be those basic things, like the path to the dataset, the name of the dataset...

57:38 ...any children the dataset might have, the data type, and the extension of the data.

57:44 So, those aren't very interesting.

57:46 They can be useful, but those properties are a little bit dry.

57:52 So if we would...wanted to describe something else, like a feature class, we can just go to the feature class...

57:57 ...properties here, and here's a listing of all of the properties that you get when you describe a feature class.

58:03 So you get feature type, information about the m and z values, spatial index, the field that's the shape, the geometry field...

58:14 ...as well as the shape type, so whether the feature class is polygon, line, point, what type of geometry.

58:21 And up here in the summary section, we also see that when you describe a feature class, you have access...

58:26 ...to the table properties and dataset properties.

58:29 So if we look at that we can see that when you describe a feature class, you also get these table properties...

58:39 ... as well as some dataset properties.

58:50 So here's an example of actually using the Describe function in a script.

58:56 We'll start by creating a new variable, Desk, that will equal the Describe object of the feature class specified.

59:03 We'll then use some of the conditional logic that we introduced earlier to print some different statements based on...

59:09 ...what the input's shape type property is found to be.

59:13 So, depending on what the Describe object, Desk, shape type property is found to be, will print an appropriate message...

59:20 ...stating the feature class shape type.

59:23 So after we create that Describe object, we'll look at it, that shape type attribute, and say if the shape type is equal...

59:30 ...to a polyline, we'll just do something simple like print, "The FC is a line feature class."

59:37 Else-if, the describe object's shape type property, is polygon, will print the FC as a polygon feature class.

59:45 Else neither of the above conditions are true, so we'll print, "The feature class is not a line or polygon."

59:55 Now, as you start to create scripts, something that you may want to do is to make your script...

59:59 ...capable of receiving arguments.

1:00:02 And when I say arguments, this thing that I'm talking about, you can just think of as an input, an input to a script.

1:00:10 In all of our previous examples, we were using hard-coded input values, so hard-coded paths...

1:00:16 ...or hard-coded numbers that we wanted to use.

1:00:19 If we wanted to reuse that code that we've already written but specify some different inputs or change some...

1:00:25 ...other settings, we would have to go in and actually manually edit to change the paths or those values.

1:00:33 And that's kind of where arguments come in, in making a script more flexible and easy to share with others...

1:00:39 ...because they allow for values to be entered by the user and path to the tools and functions in the script...

1:00:45 ...instead of actually hard coded right in the script.

1:00:49 Now in your ArcGIS scripts, you can use the Get Parameter as Text function to read arguments into your script.

1:00:58 Arguments that are read using Get Parameter as Text has a zero-based index, so that's just saying that the first argument...

1:01:05 ...has an index of 0, the second argument has an index of 1, and we'll see that on the next slide.

1:01:11 Now, an important reason to use arguments is because it allows you to connect a Python script to an ArcGIS script tool...

1:01:19 ...so you'll be able to specify any inputs and outputs for your script actually in ArcGIS in a script tool dialog...

1:01:27 ...then pass those values to the Python script for execution.

1:01:32 So another short example showing how to receive arguments.

1:01:37 We see that I just create two new variables, Input FC and Output FC equal to...

1:01:42 ...arcpy.getparameterastext, and we see the first one has an index of 0 and the second has an index of 1.

1:01:50 Now we can use these variables that are coming from script arguments in a geoprocessing tool...

1:01:55 ...the same way we would if they were a variable set to a hard-coded string or value.

1:02:00 So, we have ArcPy Clip, the Clip tool, and we use one of the argument variables Input FC, as the input feature class.

1:02:08 We then use a hard-coded value for the clip features, and then back to another argument for the output feature class.

1:02:15 So there's no rules about being able to only use arguments and no more hard-coded values.

1:02:21 You can use either/or in your scripts and in the functions and tools inside of them.

1:02:29 All right, let's go back for another demonstration of using Describe and implementing arguments in your scripts.

1:02:36 Yeah, all right, so I'm going to go back to the script I had in the second demo, and we're going to try...

1:02:41 ...and make it more generic.

1:02:43 So, in each of this, each of the demos I've had so far, I've mostly been working with scripts that are...

1:02:49 ...almost kind of like a macro, right?

1:02:51 There's the paths have been hard coded, there's been lots of assumptions that I've probably been making I'm not even aware of.

1:02:58 So, I'm going to take some of those...that script and sort of evolve it to a point where it's something that could be adaptable...

1:03:04 ...and I could use with other datasets.

1:03:07 So, as Drew showed, one of the things we can do, [unintelligible] parameter, is, use the Get Parameter...

1:03:19 ...as Text method based with an index value.

1:03:25 So I'm going to define three parameters, one for my input feature class; I'm going to assign another one to a...

1:03:36 ...snap feature class, and I'll update the value...my index; and then an out feature class.

1:03:52 I'm going to remove this workspace environment because really, you know, I'm going to be passing in full...

1:03:58 ...full feature classes with paths. I'm not necessarily going to want them to be in the same location anyways.

1:04:04 I'm going to strip that out and then I'm going to take these variables that I've created and substitute them...

1:04:11 ...instead of these hard-coded strings.

1:04:15 So I'm going to copy my input to my output, I'm going to densify that output, and I'm going to snap that output...

1:04:27 ...and then the last one was, the feature class I actually was snapping to, County.

1:04:36 Now, Drew kind of got into the Describe a little bit, too.

1:04:42 And describe allows you to make decisions based on what your data is.

1:04:47 So, as the script is right now, it's still pretty vulnerable to somebody putting in something that I didn't intend to...

1:04:55 ...so really this workflow will only makes sense if I'm trying to snap maybe lines or polygons to another line or polygon.

1:05:03 You know, if I put in an annotation feature class that are points, maybe that's just something I want to stop right up front.

1:05:10 So you can use code like this.

1:05:16 And all I'm really doing here is I'm creating a...Describe object of my input feature class, whatever that might be...

1:05:24 ...and then I'm going to evaluate the feature type.

1:05:28 And the logic here says basically if it isn't a polygon or a polyline feature class, then that isn't what I want...

1:05:35 ...and then I'm going to come here and I'm going to throw an error.

1:05:41 But that's really not the direction I want to go; I want to try something just slightly different.

1:05:50 So one of the nice things about creating Python scripts that are adaptable and take input arguments is that...

1:05:56 ...you can then take those... That's why I wasn't at the UC last year, if anyone was looking for me.

1:06:06 You can create Python script tools.

1:06:19 Any time you have a toolbox, and actually back up one step, maybe you're in a workspace...

1:06:25 ...you can click on here and get a new toolbox.

1:06:29 From Toolbox you can click in there, you can add a script.

1:06:36 So I'm going to create a tool with that script that I've been using in mine and I'm going to call it Snap the Polygons...

1:06:47 ...something generic, and the name is what you would use, say, if you were calling to it in Python.

1:06:53 The label...I'll call it the same thing, that's what you would use if say, on the Property Tool dialog, or if you added...

1:07:01 ...this tool the ModelBuilder, that would be on top.

1:07:06 I'm going to locate that file...

1:07:13 ...and hit Next. And then on this screen, this is where I define my parameters.

1:07:17 So I have three parameters. I'm going to define the first one, call it Input Features.

1:07:24 And I'm going to use Feature Layer; so why layer and not feature class?

1:07:29 Well, if I specify it as feature class, that means I can only use full paths...

1:07:36 ...in feature classes to the data.

1:07:37 Say I had added that feature class to the map, previously I couldn't just drag and drop that layer in.

1:07:42 So, feature layer allows me to use feature classes and layers.

1:07:50 Instead of Snap Features, I'm going to also give that a feature layer.

1:07:57 And I'm going to define an output feature class.

1:08:03 And that will be a feature class.

1:08:07 There's a number of properties down here below; for this scenario, I don't have to change too many.

1:08:12 My last one, this is definitely not an input, this is an output, so I will identify that.

1:08:18 And then when I went through that exercise in the script of using Describe, what I was trying to do was basically...

1:08:24 ...filter out feature types that weren't acceptable.

1:08:28 Well, you can also do that when you create a script tool in the wizard.

1:08:33 So for my input features, I click down here. The filter's currently none.

1:08:38 I can use a feature class filter, and I can check off types that are not acceptable to me.

1:08:45 Hit Apply, Finish.

1:08:50 Now I have this fully functional tool; this will behave in the same manner that any geoprocessing tool behaves.

1:08:58 So, you know, before I was working with this river called Crow River, well, Hennepin County is also bounded by...

1:09:06 ...the upper reaches of the Mississippi.

1:09:08 So in this case, I'm going to snap the Mississippi layer, I'm going to snap it to the county, sort of like before.

1:09:22 You can see how it's off a little bit right here.

1:09:25 I run that tool, based on the code that I've written, hopefully this will come back in 5, 10 seconds.

1:09:40 And that is my output. It achieved I wanted to do but with different data.

1:09:53 Great. Thanks, Dave.

1:09:56 All right, we just have a couple more slides.

1:09:58 Here's a resources slide that we wanted to just call some special attention to.

1:10:05 The ArcGIS Resource Center is a central location for finding all of our online documentation, help forums...

1:10:13 ...and that, these were just, they're brand-new sites that were just put live a few weeks ago, and so that's a good place to check...

1:10:20 ...if you have any questions about ArcGIS.

1:10:24 We have a specific geoprocessing resource center, then that contains a script gallery, some blogs...

1:10:30 ...as well as presentations related specifically to Python.

1:10:36 So that's a good place to check.

1:10:38 There are a few Python reference books that we recommend above some other ones, so if you're interested in learning more...

1:10:47 ...just about Python as a language, these are a couple good books to check out.

1:10:51 There's a lot of information as well on the Python organization Web site, so you can go there and dive into python.org...

1:11:01 ...there is a free tutorial, a very, very long tutorial, actually, that you can download and work through.

1:11:12 Okay, thank you for attending today; if you're interested in filling out an evaluation form and haven't gotten one...

1:11:19 ...there's a student assistant over by the door so you can stop and see her.

1:11:26 Yep.

1:11:27 Oh, thank you.
