LUKE PERKINS: So welcome to programming basics for AEC. My name's Luke Perkins. I'll tell you a little bit about myself on the next slide. So I work for a company called Stewart, which is a regional multidisciplinary design firm in Raleigh, North Carolina.

And I put this map over here because anybody knows a little bit of geography about North Carolina, the little red county out there, that's not where Raleigh is, but that's where I'm from. And it's called Graham County. I'm from a tiny little town called Robbinsville. And if you don't try it now, but if you were to zoom in close enough on Google Maps, you'll hear banjo music.

So and I mention this because I know the Appalachian American accent is not pleasing to most people's ears. And it can cause some fatal errors in communication. So if we need to reboot and try something again, just hold me up and tell me, let's hear that again. So I graduate at North Carolina State in 2014. And I joined Stewart with a passion for fully utilizing design technology in land development.

You know, I got really involved in, out of this university after learning some of the tools at Stewart. And knowing that we had this Infrastructure Design Suite and we were only using like 5% of it really. And so I've made it a point to learn as much as I can and try to move our team along in utilizing this technology.

So what are we going to learn about today? This class is really centered for how to get started in the basics of programming and the fundamentals of object-oriented programming. And kind of the thought behind the class is my mentor and I sat down in the spring and we really hashed out some of the proposals that we wanted to put together. And there was this big gap that we felt like at AU and even in industry and that you have at a typical design firm you've got IT, which is very focused on networking issues and licensing and things like that.

And then you've got the designers, who either go a business development route or they really push the technology. And they want to develop tools. And they didn't go to school to really learn how to do that. But there's a need for that.

So this class, what we're going to do is the first portion of it is we're really going to dig down into the fundamentals and the basics of object-oriented programming. And then we're going to take a step back and look at how that fits into programming in general and how to make that work with AutoCAD applications. If you're here looking for very specific interfacing with certain

APIs, you're probably a little bit beyond what we're going to go over. But I think that it's important for everybody to learn kind of programmatic thinking and the reason why we need to take a look at some of the processes and see where we can implement programming in our workflows.

So some motivation for why you'd want to do that. And the first one is pretty simple, and that's increasing your efficiency. Anywhere where we can get a little bit more efficient in what we do. It turns into it turns into money. Time is money in our industry. And so making those workflows more efficient, automation, that's what we're after.

There's a huge opportunity for innovation in our industry. We are not known as a tech kind of industry. And I think that's-- if you're like me, you went to school. And you took all those engineering classes. And you kind of steered away from the software development classes because there were people that were way more interested in that. But now that I get out in the industry, I see where we can pull those people and what they know into what we're doing to get a lot of benefit out of that.

So this graphic, and it is not showing up too well, but I thought it was really interesting. It was a survey that was taken about a year ago. And it asked people, in 50 years, if they think that most of the workforce will be replaced by robots and computers and almost 2/3 of those people said yes. That the majority of our workforce is going to be replaced. But when asked if their jobs specifically was going to be replaced, only a third said yes.

So I think it's especially at AU, when you see all the-- just walking through the exhibit hall and everything that's being automated, and all the robots and computers and you can see it in our industry and in what I do every day. You know, sites are getting closer and closer to grading themselves, which will be great because I want to do that all the time. But I still think that it's important for us to constantly push the push the boundaries on what we know so that we don't get replaced. And that we know what the machine's doing behind the scenes. And that's just going to help you in troubleshooting and debugging problems is knowing a little bit more about where this stuff comes from.

So in that, we'll get to the our key learning objectives. So like I mentioned earlier, we're going to look at the basic building blocks of object-oriented programming and some of those fundamentals. We're going to discuss various programming languages and their uses. We're going to look at the basics of software development and what that process generally looks like.

And then we're going to talk a little bit about the opportunity for programming as it pertains to AutoCAD applications.

So here are some of the fundamental programming concepts that we're going to go over. And these are things that when I found-- that you really need to have a good understanding of if you want to take a piece of code and look at it and dissect it. These are some of the things that you really need to have a strong knowledge in.

And I will say that I went to school for Civil Engineering and graduated. I took a Java class because civil engineers at NC State were only taking Fortran classes. And I didn't see a future in Fortran really, but Java was an alternative that I could take. And I got in there and there were people that had Java in high school for two years, who were just totally blowing me away.

And so I don't have a strong background in computer science. I'm not going to be a guy that can make a lot of assumptions about what you know. And I think that if you're just getting into programming, that this might be an approach that's a little bit more easier for you.

So with that, we'll jump into the first one. And I'm going to flip over to the mouse. So functions, and I'm going to sit down because I'm going to be flipping back and forth a little bit from the slideshow. So what is a function? And basically, a function is a commonly used routine. And this gets back to programmatic thinking and finding process these that are repetitive in your daily workflows and finding a place where you can implement programming.

And I'll get into why you see an erosion control plan off to the right in just a second. But first, I think it's really important that as a civil engineer, I feel like being lazy can come in handy sometimes. Because I know a lot of people who will, they'll sit there, they'll drink a cup of coffee-- things of coffee and they'll plug-in their headphones. And they'll just hammer out some work that's very repetitive instead of sitting down with a couple of people and thinking how can we find a better way to do this.

So I'm going to start with where I started basically. As an engineering intern, you have to do a lot of monotonous work every now and then. And there was a certain erosion control plan that I finally just said, I've got to find a better way to do this. So let's look at that in Excel.

OK, a little background. This is an erosion control plan. I know that everybody in here is not land development people. But basically this project had a lot of linear roads and greenways that can usually be treated by silt fence. And we had a regulatory agency that was kind of a stickler and made us size these temporary storage areas everywhere we had an inlet protection or silt fence outlet.

So it was basically going to be me building 50 of the same spreadsheet over and over again. And so I sat down with some people to try to figure out a better way to do that. So if I unhide, this is the spreadsheet that we use to size an excavated storage area. And without getting into too much detail, you basically need a drainage area. And the size of that drainage area gives you a flow. And then you can try out some parameters for sizing that basin until you have a surface area and a volume that meets the requirements.

And I was looking at the site and how many of these that I would have to size. And instead of sitting down and hammering it out, I decided to come up with a simple routine that would do this for me. So in this data tab, I've got the name of all these excavated storage areas that I'm going to have, the acreage that flows to them. And then I've got a little subroutine that if you click there, it produces all these sheets and it uses the solver function to actually go through and solve exactly what the surface area needs to be.

And I won't get in to dissecting the code for this because I want to get back on topic. But if you look at this real quick. This Visual Basic code is not a lot. And that's the thing is, I've already sent this out to other engineers in our group. And any time they have a site that's going to have to do this, it's really figuring out some drainage areas. One button mouse click and they're done. Just because I sat down and googled some stuff and figured out a few lines of code.

So that's kind of what I'm talking about when we get back to programmatic thinking. And now that wasn't a true function because it didn't have-- that was more of a subroutine in Visual Basic. And so we'll look at what a true function is in object-oriented programming.

And I'm going to use Python. I know that Python is not something that is used a lot around AutoCAD, but I think if you're a beginner, it's really good. Because just my interaction with other languages, it's like they sat down and got everything right with Python. And it reads a lot cleaner, it's a lot easier to understand. So for teaching or if you're learning a language for the first time, you know, I think Python is a good place to start.

Let me get rid of this. So in talking about functions and talking about a routine, a process that's repetitive, that we want to automate, I thought about my time in Vegas and something that's going to be repetitive and that is having a drink. So I'm going to open. [INAUDIBLE] and so making a drink. So what are the steps in making a drink?

Well, we'll have to use our imagination a little bit. Because I've just basically written out the steps. So I've got a print statement that's going to print this line that says filling shaker with ice. And next just pouring in the alcohol. Adding a mixer. Shaking it up. Pour into tumbler. And then I've got my favorite drink which is a whiskey sour.

These are the steps. And you know, for the metaphor, you know I could go in and have to print out these steps every time or I can turn this into a simple function. So in Python, if we define a function as say, whiskey sour. And we'll indent everything. So this is how Python uses indentation instead of semi-colons, for a lot of people who use a different language. And then we will return the variable whiskey sour.

So now that this is a function. And you guys might have to help me because I misspell things sometimes. And if you catch it before the program does then let me know. So we'll run the module. So if I run this function whiskey sour. Boom, automatically I'm filling a shaker with ice, pouring in the alcohol, adding in the mixer. Doing all those steps that we were going to have to do manually if we were to type it out every time.

And I know this is, you have to use your imagination a little bit here about how you'd apply that. But if you want to take that a step further, and this is where you can get into some powerful stuff with programming and with functions is those steps are pretty common for other processes, right. I'm making a whiskey sour. If I'm going to take my friends with me, they might want to have other drinks.

So we don't have to just make a whiskey sour if we look at what the functions have in common. So let's change this function into fixing a drink. Fix drink. And instead of returning a whiskey sour, we'll set up two different functions. We'll define whiskey sour and we'll also have another one of my favorite drinks, old fashioned.

So under whiskey sour we'll run the fun-- will run the function fix drink. And then we'll pretend like after we run that function, it produces a variable called whiskey sour. Perfect. Whiskey sour. And old fashioned, similarly, we'll run the fixed drink function. Whisky sour. Oh, we're mixing the-- we're making it old fashioned this time. Old fashioned equal to-- and now we need to have a return for both of these.

So we will return old fashioned. And up here will return whiskey sour. So if I save this and run it. Now we can either make a whiskey sour and it runs through the steps that we had previously. Or we can make an old fashioned. Same steps and we're either creating a whiskey sour or an old fashioned.

So this is where I think-- I think that if you go through school for computer science. And if you're in an in programming and code every day, these things may seem pretty trivial. But as an engineer, I think we kind of have to take a step back and look at some of the processes that we have, where those processes overlap and things that we can streamline. Because when you get into this code, the benefit of doing this is say the process that you're looking at changes. Something in these steps change.

Well, if you were-- if you were to put all these steps under each of those functions, now you're going to have to go change your code in multiple places. So this keeps our code nice and concise, where we're overlapping in our functions. And if we needed to go back and change one of these steps, we could run either one of these functions and we would only have to change it in one place. So that's functions.

We're looking at the next slide. Objects and object-oriented programming. So objects, you know, to be I guess thorough. Even that function in Python was an object. And that might get confusing, but object-oriented programming means that everything is given a unique identification as an object. But when we think about objects as virtual pieces of information that relate to something tangible, usually an object-oriented program, we call that a class.

And classes are the nouns of our program. And those nouns can have attributes. So they might have certain properties. And they might have a method, which is essentially a function that does something to that class. It either changes the state of that class or it carries out a process on that class. So let's take a look at an example of that.

I'm going to open up objects and methods. So here I've got an object, a class that I call Casino. And it's got two methods. So similarly, we were looking at the functions. These methods are methods that do something to this class. So the first one is a little bit unique to Python in that it's an initialization method. So this is-- it's a method that helps you create this object Casino. And it's going to take in these parameters.

And I think this is another thing that really gets confusing if you're not in computer science all the time. Because you hear people talking about parameters and hear people talking about arguments, almost interchangeably. Parameters are kind of this empty box set up here in this parentheses to take in information. The information that the user decides to put in there, those are your arguments.

So our parameters are self, which is something that's a little bit unique to Python in that when you create a class object, it actually passes the object into the class. Then we're going to give each of our casinos a name and we're going to give them an allowance. How much we're allowed to spend at that casino. OK, and then we're going to have a method. So something that we can carry out on that object called gamble.

And what gamble does is it checks and sees if our allowance is greater than zero. If it is, it's going to print this statement saying that we're rolling the dice at whatever casino we name the casino. We're looking for that winner, winner, chicken dinner. And then if you're like me, you're going to take your allowance and you're going to knock off some money because I'm horrible at gambling.

So this conditional statement, the if statement. We said if the allowance is greater than zero, it's going to carry out that. If not, it's going to jump down and tell me basically I'm out of money. And I spent it all-- this last method that I'm carrying out here on-- it's most object-oriented programs have something similar. It's just taking that parameter and sticking it in between these brackets.

So it's going to take self dot name and stick it in these brackets when we print it out. So if we run this program and we create a couple of Casino objects. So Casino1 equal to-- so here, when I open up that parentheses, it's already asking me for those-- for my input values that I set up with those parameters. So what's somebody's favorite Casino? Nobody has a favorite yet? All right, well we'll just go with the Venetian because you guys are boring.

And then we'll make our allowance $200 at the Casino, at the Venetian. So instantly, if we look at the Casino, Casino1 object that we just made. And we look at the name, we should see the Venetian. And if we look at our allowance, we should see 200. If we run the gamble method on our Casino, Casino1.gamble. You'll see we're rolling the dice at the Venetian. And we're looking for that winner, winner, chicken dinner.

I know that doesn't really make sense because you roll dice and craps and you're looking for the winner, winner chicken dinner in blackjack. Let's not get into too many details. If I do that, if I run the gamble again, you'll see, oh, told you I can't type sometimes. Casino1.gamble. You see, we're rolling the dice again. But if I try it a third time, it's going to tell us the Venetian took all my money. Because that's generally how gambling goes for me.

And the next step to object and object-oriented programming is inheritance. And this is similarly to the functions and founding processes that overlap and where we can eliminate some code. Inheritance is kind of similar for classes. So essentially you're going to sit down when you're thinking of a program. And you're going to come up with these nouns for your program.

So in our program, it was casinos. And you're going to think of the attributes and methods that are going to be contained in that class. And a lot of times, you'll have something an attribute that will be unique to a certain group of casinos. So I'll demonstrate that. And we'll use the same kind of example.

So here, I've created two other classes down at the bottom. One is called the strip. So we're looking at casinos on the strip. And the other one is Fremont. And I haven't been at the Fremont yet, but I know that you can ride a zipline down Fremont. So that's something that's unique to going and visiting Fremont Street.

We wouldn't be able to put the zipline function, which basically just prints we're zooming down zip line. We wouldn't be able to put that in this class Casino that we made beforehand. Because not all casinos have a zipline that you can ride. But you're going to be able to give all of them a name and allowance and be able to gamble just like we did before.

So inheritance is basically figuring out those attributes, those methods that you have in common and then building situations where you have unique properties and inherit the more general class. So here, if you think about creating a strip Casino, a strip Casino doesn't sound right. A Casino on the strip. And where we would usually put our parameters here, we just put Casino. And since we've already defined that, it's going to know what the parameters should be for Casino. And that is inheritance.

And I think when you look through some code, this is a roadblock for a lot of people. Because it jumps around and where is this coming from. And if you really just take a second, don't freak out and look at some of this stuff, a lot of it will start making sense. So to run-- if we run this and we're going to build a couple of casinos.

So Casino1 is going to be equal to, we'll do the strip. So Casino on the strip and we'll keep using the Venetian. Our allowance is $100 this time. And we'll make another Casino, Casino2 equal to Fremont. I don't know any casinos at Fremont so this one is going to be Luke's Casino. And we'll give ourselves $100 there as well.

So at Casino1, I could run the gamble method. Or maybe not, can't run the model method, Casino1.gamble. So rolling the dice at the Venetian. And I can do that again. And similarly, as before, we're out of money.

But since we created the Casino2, which also inherited the Casino class, I can also gamble there. And notice that we're not out of money there, because it created two different instances of that Casino class. One for the strip and one for Fremont. Now, there's things that are unique to Fremont. So if I took Casino2 and ran zipline. We're zooming down the strip. We're zooming down the zipline on Fremont.

If I were to try to do this with Casino1, it's going to tell us that the strip object has no attribute zipline. And that's to be expected. And kind of, I guess the powerful piece here is similarly as before, you want to have concise code that when changes happen, you can go make them in one place. And everything that we put up in this Casino class is going to be inherited by these other two. And if anything changed up here, or if we want to add something, that's common to both of these situations, we can do that up here.

So what's next? Loops and conditions. We looked at conditions a little bit earlier with that if-then statement. But loops and conditions perform a lot of the logic that you're going to see in the programming. And loops are very powerful in that they can iterate, you know, they can go through hundreds and thousands of pieces of information in a split second. So this is where the powerful tools come in. You know, I don't have any huge databases that I can show that off. But we're going to look at some simple situations of using a for loop and a while loop.

So using a for loop. So here, I've got a list or an array. Which is basically just a container with multiple objects. And then I've got a for loop. And for loops look like a little bit different in Python. And I kind of like the way they're laid out in Python. And so this location has a bunch of places that you could potentially see a show.

For a theater in a location, so we're creating this theater variable. For a theater in a location, we're going to print, I saw an awesome show. And we're going to insert whatever that theater is into our brackets. So it's going to look through-- this for loop is going to look through this array. And for each index in the array, it's going to pop that entry into theater and print out that statement. So if we run this module, we should see I saw a show. And it looped through our array and inserted those casinos into our statement.

If we were to look at the while loop, not to get super political, but as long as Trump's in office my girlfriend is unhappy. So we're going to set Trump equal to true in the year 2016. So this while loop checks and sees. While Trump, so Trump is equal to true. So while true, we're going to print my girlfriend is unhappy in whatever year this is. Until, then we're going to add, we're going to add one to the year after my girlfriend is unhappy. And then we're going to check the next year.

So if the year is ever 2020, she's going to be happy because he's up for re-election. And we're going to set Trump equal to false. So if I run this loop, we see it's 2016, my girlfriend's unhappy. It's 2017, she's unhappy all the way until 2020. And now, she's happy because he's up for re-election. I'm not looking for anybody's opinion. I'm not going to give mine, it's just my girlfriend.

OK, so to get out of the code for a little bit and talk about stuff more generally. Environments, modules, and packages. So we were just in a programming environment called idle, which just basically ships when you download Python. And what an environment is it's a space that makes it easier for you to program.

So you saw things changing colors as I was programming to let me know that I had an object that I was meaning to produce. That's part of what an environment does for you. It's a place and there's a lot of different programming environments. And there's many debates online about what's the best for certain programs. So whatever language you're looking at, you can look that up and find the environment that you think suits your needs the best.

Modules and packages. So basically, modules and packages, they seem like magic at first. Because you're going to import this package, this module, and it's going to be able to do something amazing. And a lot of people don't look beyond, you know, just its functionality. Well as long as I import this guy, it'll do the work for you. And they don't ask the question of what exactly is that.

And really, it's just more code. It's the same stuff that you're producing. If you're in Python and you are you're importing Python modules, if you were to go look at that, it's built of the same-- it's built on the same stuff that we just went over. It's just stuff that's already been taken care of for you. So you don't have to go do something. You don't have to go reinvent the wheel when it's something that's commonly done, that everybody needs to do often.

And when you look at AutoCAD for that, it really depends on what kind of programming that you want to do with AutoCAD. And we'll talk about the options that you'll have. But one of the main programming environments is going to be Visual Studio. It's a Microsoft product that you'll download. And when you set up the templates for programming an AutoCAD add-in, you're going to get a lot of those packages, you're going to get a lot of those libraries, I think it's called in that case. They have a lot of those tools there for you.

And so I would suggest that spend some time. Go do some research and looking at some of that stuff that's there for you. So a lot of programming in these add-ins is finding stuff and piecing it all together instead of just totally coming up with something that's brand new for you. A good resource to get set up in the Windows Visual Studio is the My First Plug-in. It's in the auto cast-- AutoCAD development developers kind of website.

And it's a good tutorial, it'll take you through setting up that environment, importing all the AutoCAD stuff that you need to take off with that template and have all those tools. There's also a talk on that tomorrow at this time. I'm not sure where that's located, but if you check your schedules I think you'll find it.

So programming languages and their purposes. Just real generally, I'm going to talk about web languages just for a second. Because more and more things increasingly are moving to the web. I know AutoCAD, you know is kind of set in stone. There's things that we'll talk about, those on the next slide. But things like InfoWorks and some of these web-based applications that AutoCAD is coming out with now, you're going to have to interact with those a little bit differently than you are the traditional programs.

So HTML is pretty much the content on websites. It's the text. It's all the pieces of a website that you see on the front-end. I didn't mention CSS on here, but CSS kind of gives those objects that you see a certain style, a certain format. It lays out the page the way you want to see it. And then JavaScript is kind of the actions. It's the menu bar drop downs. It's the scroll over stuff that's popping out. It's the actions on the website.

Python. I threw Python on the web languages because it's becoming a popular web programming language. But it's definitely used for other stuff as well. PHP is more of a web-only kind of back-end. And when I'm talking about back-end and front-end, I'm talking the HTML, the JavaScript, that's the stuff that kind of controls what you're seeing on that page.

The back-end, when you make your picks and clicks, it's kind of doing the navigating, it's interacting with the database. And those are these languages, PHP, Ruby, Python are popular back-end programming languages.

So languages in AutoCAD and their purpose. So Auto LISP, which there was a course. I talked to some people who just came out of an Auto LISP course, is the macro programming for AutoCAD. And what that's going to do is basically help you set up common function routines, options for commands that you go through a lot. And set up that process and help you streamline it.

Visual Basic, C#, these are .NET languages that are going to be the plug-in development languages. So plug-ins are going to help you create new functionality. It's something that you can't just take care of with a command that's already in AutoCAD. It's not just a series of commands like a macro. It's something that you've identified as I really wish that I could do this. There's not a command for this. These are the languages that you're going to be in that's going to help you accomplish that task.

So what-- so we went over plug-ins a little bit. And macros being that Auto LISP. Scripts are a little bit different in that, I guess the major discrepancy between macros and scripts. Macros you're going to be able to like, you're going to put those on a toolbar and you're going to be able to click a button and run that that macro.

A script is something that you're going to have on your server. It's a file that you can run. I think at Stewart we have a lot of scripts that run and just, they go change a lot of variables in AutoCAD. So a little bit different in that it's not something, it's not a button that you're going to create that you click on in AutoCAD to carry out a certain function. But they work similarly. And then the plug-in, as I was saying, is something that adds functionality to your AutoCAD software. I wonder if that's working.

So API, you heard me mention API earlier. And a lot of you beginners, if you're running around to some of these programming talks, you're going to hear a lot about API. API for this, API for that. And what is API? And it's an acronym that stands for Application Programming interface. Which, what does that mean?

Well, it's basically when the developer creates a piece of software, and they want to let the user interact with that. Then they're going to build an API that's going to set the infrastructure for how the programmer on the outside is going to be able to interact with the program that the developer created. I think the easiest way that I've kind of found to explain this is you've got a power plant and you've got little houses that are hooked up to the power plant. You need that infrastructure to be there before you can build your house.

So if the database or whatever you're looking at connecting to is that power plant, you really need the infrastructure to be there. That's the API, before you can interact with it. And there's a ton of APIs out there for different products. And it really doesn't take more than a Google search to figure out. What kind of language, what kind of environment is best to work with the API that's specific to what you're going to be working with?

So some of the basics of software development. I think the biggest thing from my experience is really spending a lot of time upfront communicating with people. Communicating with your team, your partner, whoever you're going to sit down and create some of this content with, before you jump right in and start trying to solve everyone's problem.

These are some of the simple steps. So like I mentioned, communication a lot upfront and weeding out some of those problems early on. Coming up with a plan. You know, and I work with a lot of landscape architects day to day. And they're the best at drawing stuff out on paper. And that's huge. I feel like if you can just diagram out your thoughts and other people are just standing around a table, you know, you'll get a lot out of that without realizing it.

And then you're kind of going to get into the design of your piece of software and the construction. And you know, I've gotten maintainability, maintainability, maintainability under that construction. And part of that is going back to having really concise code. You know, overlapping your classes and your functions, like we were mentioning before, as much as you can. And then deployment.

And deployment is, there's never going to be a really good time to deploy. Whatever you're going to deploy, and this is kind of our frustration usually a lot with the tech industry is there always shipping out something that's half done. You know, AutoCAD 2017 comes up. And it's got a bunch of crashes and stuff right, when you get it. And you're waiting on all these hot fixes, all these patches.

And you know, seeing it from both points of view. You really, sometimes you just got to get stuff out the door to be able to figure out everything that's wrong with a certain product. And then the last is maintenance. And it's just making sure whatever you're doing is relevant. And knowing that just creating something and pushing it out there doesn't mean that it's the right solution all the time. And it'll constantly have to be updated, especially as technology is moving as fast as it is.

And then something that I've learned from a lot of my buddies who work at startups is this concept of the MVP or the minimum viable product. And I think people that work in code and engineers, a lot get caught up in the functionality. They want to be able to show off the tools and something very specific that takes a lot a lot of time and resources to be able to develop. When really, you can show, you can hide a lot of that functionality and create something that's appealing to people first. And get buyoff before you really spend a lot of time doing something behind the scenes.

So that's what this pyramid is talking about is we really don't want to focus on just the functionality when you're getting into the minimum viable product. You want to be able to have your user's experience that emotional design up front as soon as they see it. They're like, OK, this is something that we need to spend some time and resources to figure out.

And the next thing that's really big that we found out is know what Autodesk is working on. A lot of times you'll identify something, a process or some functionality that's missing from your workflow. And you can spend a lot of time working on how to fix that for your group. And then you'll find out that Autodesk just come out with something that does that.

And the best example I've got is the sizing gravity piped networks within Civil 3D. You know, me and my mentor worked on being able-- you know, what's the quickest way to size those parts without having to do a full hydraulic analysis in a certain piece of software. And it was like six months later. They've built the gravity piped network design into Civil 3D. And it's kind of a one click button that sizes your pipes.

So really, do the research, get involved in some of the features, portals and know what Autodesk is working on. And that'll save you a lot of headache and a lot of darn because it's something that I could have done myself but it's already been taken care of.

So with that, if there's any questions, and like I said, earlier I know a lot of people are looking for how to work with an interface with a specific API. If you go, if you go talk to the guys at the answer bar, they're going to be able to answer a lot of those questions in there. A lot of courses that are designed for that specifically.

This was my first talk at AU so I would love to hear some feedback on how I did. You know, how I went through the whole process on how this works. But I don't know if you guys fill out that survey, I will be able to look at how it comes out. So I'm looking forward to your feedback.

So are there any questions for me?

AUDIENCE: Is there a Python API for AutoCAD?

LUKE PERKINS: No. And yes. They're not for AutoCAD, but Autodesk, I'm trying to think of the name of-- it's not in the Civil industry. But increasingly, there will be more tools that Autodesk produces that will have a Python API. You're not going to see it in AutoCAD or Civil 3D or any of those big guys, I don't think. But the web-based stuff that's coming out now, increasingly there will be more and more Python API.

AUDIENCE: But you were programming [INAUDIBLE].

LUKE PERKINS: Visual Basic or C#.

AUDIENCE: How would you compare the Python to C# if you were wanting to--

LUKE PERKINS: I mean, I'm super biased towards Python. I feel like it's much simpler. And it made more sense for me to learn as an engineer because it has more applications outside of building add-ins for AutoCAD is great. And I can figure out the C# that I need to know to be able to do some of that stuff.

But Python, you can-- if you want to spin up a web site real quick, you can use Python. If you want to do, you know, high end data analytics, not something I've got into yet, but I know that Python can do that. So I stress learning Python up front just so I could go in different ways with it.

AUDIENCE: I just wanted to make a comment there. Python, it depends on the product. Fusion, Maya and Revit have Python APIs. I'm not sure about AutoCAD.

LUKE PERKINS: I don't think--

AUDIENCE: [INAUDIBLE].

LUKE PERKINS: Yeah.

AUDIENCE: --is kind of expanding. There's a Javascript API for AutoCAD.

LUKE PERKINS: Yeah.

AUDIENCE: So they're kind of expanding out. Different languages for different [INAUDIBLE].

LUKE PERKINS: It's been identified as like one of the languages of the future. So if you're just getting into programming, you know, I think it's a good place to start.

Anybody else? All right, well, thank you, guys.

[APPLAUSE]