Module 1 Lecture Transcripts

Contents

Module 1 Introduction

Ron Guymon: Isn't this pretty? As we gaze upon this incredible vista of Bryce Canyon, it's easy to see the value in understanding the landscape. But what is business analytics? Just as understanding this landscape helps us appreciate its beauty and history, business analytics helps us understand the landscapes of businesses.

And it's a skill that most business professionals acknowledge is important. But why is data so important for business analytics? I interviewed Scott Warner, the founder of a social media influencing company. And asked him several questions about how data is used in business and with the businesses with which he works.

I think his responses were insightful, let's listen to a few of them.

Scott Warner: I started gig-back in January of 2013. It was originally set up to help musicians, artists, both big and small, get their brand out to the masses, help them share their music with the world predominantly through the social media channels.

Social media is changing the world, I think it's the future of advertising in every sense of the word. And so it really fascinated me and I have always had a tremendous love for the arts or for music. As time went on, we realized that what we were creating for artists to help them build their brand and monetize and align with new fans and find their biggest fans, we realized that it worked for everybody.

And so the gig platform really helps everyone with trying to share their art, I guess you could say, or their passion with the world using different social media strategies as well as helping them understand the data behind it. Information, understanding knowledge around data is crucial to any business really building what it is that they're trying to build, you need to know and understand, see that stuff.

And so that's what we're really focused on doing, is helping businesses or influencers build their brand and understand it using data. Inside of any business, data is imperative. You shouldn't be making any decisions unless you have data behind it. And of course you're gonna have to start from scratch at one point and you're gonna have to learn trial and error.

But once you start creating data, you have to understand it. And depending on the metrics of your business, once you know what those metrics are, you need to build data sets around it. And so as a business owner, it's imperative that you understand what

your objectives are because once you understand those objectives, you can start putting data sets together.

And so, for us, a lot of people really struggle knowing when to post on social media during the day. They don't understand what seems to be working or not working. They don't understand, click through rates. They don't understand what seems to catch the eye. And so, a lot of the data that we focus on building is to understand those important pieces of information.

So I've had a chance to work with tons of companies and they're all interested in data. One of the things that many of them struggle with is they really have not defined what it is that they're trying to accomplish or do. They don't understand their objectives. And so a lot of times I'll spend time getting to know what their business is about, to then identify what data sets they should be building around or seek to better understand.

And so I think companies that don't understand what their objectives are or how they can win, you'll never be able to build great data sets around it. I've always been fascinated by companies that have learned how to utilize data to their advantage. There's a lot of companies out there that really struggle with data, but there are few that have figured out how to capitalize on data.

And because they understand the data around the objectives that they've put together as a team, sales go skyrocketing. Being more efficient with your employee base. I've seen companies that have really dialed in how many employees they need on the floor based on data, based on who's coming into their store and when they're there, when they're not there.

Cuz there are tons of companies that pay countless dollars for employees that really are not needed at that hour. And so understanding the data behind why you need employees at this time or not really can help you save or make a lot of money.

Ron Guymon: I suspect that Scott's feelings are not limited to just his industry, but is a growing sentiment in most business organizations.

In this module we will start by introducing the role of data analytics in business. We'll review some examples as well as a framework for making decisions with data. We also want to get you started performing business analytics as quickly as possible. Thus, in this module you'll also learn about a powerful programming language, Python, that has been adapted so that it can perform many data analytic tasks.

You'll also learn about how to get started using Python. Specifically, we'll show you how to install it and how to run code with it, and importantly, how you can get started using a common integrated development environment or IDE JupyterLab. While we hope that you can install Python and use it on your own machine, we know that there are some people that won't be able to do so.

For that reason, we'll also introduce you to Google Colaboratory, which allows you to easily use Python in a browser environment that is similar to JupyterLab without having to install anything on your computer. Google Colab is more than just a backup for if you do not have JupyterLab. It also has some useful features that are not available in JupyterLab, such as sharing files with others using just a link.

You'll learn how to perform some basic operations in Python, like reading in a data file, adding new columns to it, and saving the results as a different data file. You'll also learn how to use some basic mathematical operations and how to create variables. It's important to clarify that this isn't a programming course in the traditional sense.

We're not aiming to teach you every detail of Python, just as we're not aiming to explore every single detail on the trails in Bryce Canyon. Instead, we'll focus on the essential Python skills that you need for data analysis. So, for those of you who have a programming background, you'll notice that there are some things that we deliberately left out of this course because we didn't feel like it was needed at this point.

However, we hope that by the end of this module, even if you're a beginner, you'll know enough to start using Python for reading and writing data.

Overview of Business Analytics

Ron Guymon: This is pretty beautiful scenery behind me, right? Well, I'll come back to that in a minute, but first I wanna tell you about a story that's relevant to business analytics. Moneyball is a book by Michael Lewis that's had a big impact on how I think about data driven decisions in a business setting.

Even in the age of artificial intelligence, this story remains a relevant example of how we can augment our hunches with data driven decisions. Just as the Oakland A's use data to find undervalued players, allowing them to compete with wealthier teams, we can use business analytics to uncover hidden opportunities within a business landscape, identifying undervalued areas for growth.

Much like discovering a unique rock formation within Bryce Canyon, one of the battles that the team struggled with was how much to rely on the intuition of scouts versus the results of data analyses. The movie has some pretty humorous parts where the scouts are talking about whether the players look like an athlete, how the ball sounds when a player hits it, and even what the players' girlfriends look like.

Presumably, the scouts were trying to process the intangibles to identify players that would do well even if they didn't fit the stereotypical image of a good baseball player. The general manager, Billy Beane, had a tough time convincing the scouts that the most important thing was how often a player was able to get on base.

This story has some lessons that can be applied to many business environments. If you haven't read the book or watched the movie, I'd recommend it as an entertaining way to learn about the importance of making room for empirical inquiry in a culture of hunches. Now, hunches aren't all bad.

The capability of the human mind and intuition is amazing. For example, in a few moments we're able to combine information about the way a person talks, looks, speaks, smells, and moves, and then make judgments about whether that person will be a good fit with the firm's culture. Or at least we think so.

Often, we are probably correct. However, our capacity is limited and susceptible to weaknesses in certain situations. When used correctly, data analyses can augment our mental capacity to help us recognize our mental weaknesses that arise when we rely only on our hunches. Applying data science, business data, communication, and aligning business processes with analytic findings is known as business analytics.

The goal of business analytics isn't always to eliminate all intuition and judgment, but to consider how data can be used to inform tactical and strategic business decisions. Business analytics can help confirm or disconfirm a hunch. It can also help us spot settings where we wouldn't have recognized important patterns without a lot of effort.

It can also lead us to questions that we wouldn't have considered and even help us formulate new business strategies. In general, the way business analytics works is that it uses historical data to help us answer the following questions, what happened? Why did it happen? What is happening now?

What will happen next if there's no intervention? And what intervention should be taken to achieve the most desirable outcome? Now, since Moneyball was written, sports

analytics has become a big deal in nearly every sport. More broadly, data analytics seems to influence every business and every aspect of life.

In fact, it is the way AI tools like ChatGPT and Gemini are created. From health analytics related to our exercise, sleep, and nutrition to the analyses of where our ancestors are from and the shows that we watch, business analytics plays a major role. Because more companies are adopting data driven decision making tools, it's becoming less acceptable to say I think the best course of action is, and more important to say after looking at the data, I think the best course of action is this.

As a business professional, It's also becoming more expected that you know how to analyze the data in a way that will bring insight to the decision context.

Examples of Business Analytics

Ron Guymon: I've got a beautiful view of Bryce Canyon behind me. In this lesson, we'll review examples of how business analytics has influenced a variety of companies. Hopefully, these examples shed light on how you can apply analytics in your current responsibilities. Much like how the light adds more detail to the beautiful scenery and Bryce Canyon behind me.

Please notice how the creative methods for gathering and/or analyzing data led to improved business processes. The first example is from a personal experience with a growing solar company. This company suspected that it was often being overcharged for electrical components. However, there were a couple of factors that made it hard for them to verify that that was really happening.

First, the company had a variety of people buying a variety of electrical components, which made it difficult for the buyers to mentally keep track of the costs. Second, even though the invoices were emailed to a single email account, the files were scanned PDFs, which is basically a photograph inserted into a PDF file.

Thus, it wasn't straightforward to convert the image to a tabular format that could then be used to summarize the variation in prices. Using a coding language and a computer scheduling tool, we set up a script to regularly access the email account to which the invoices were sent. Download the PDF file, use optical character recognition to convert the text image into tabular data and then combine all the data into one comma separated value file.

They then use that file to quickly identify when they were overcharged and recoup what they were paid. They ended up recouping about four times what they paid to set up the script. This is an example of how a small company used technology to gather data and didn't even rely on sophisticated data processing to improve its business.

We're now going to hear from Dr. Unnati Narang who will share examples of how data analytics is used in marketing.

Unnati Narang: In marketing, analytics has become a game changer and a necessity for firms to stay competitive. Let me share two examples from retailing and digital platforms such as social media.

These are areas in which I have over a decade of experience and expertise, both through my roles in industry as well as through academic research. I was also an entrepreneur in the business of retailing prior to getting a PhD in marketing, and I saw firsthand how analytics is transforming both in store and online experiences across channels.

Imagine a retail company trying to optimize its promotions to boost sales while minimizing wasteful advertising expenditure. Historically, businesses would rely on generic promotions or simply gut instinct to target customers. Now, by using analytics tools, companies can dive deeper into customers purchase behavior. For example analyzing point of sale data combined with loyalty program from the retailer's mobile app can reveal new trends, such as which products are commonly bought together, which promotions resonate best with specific customer demographics, and so on.

Using this type of data, companies can design targeted campaigns, personalize their offers, and even optimize inventory to reduce overstocking or understocking scenarios. In some of my research with a large scale consumer electronics retailer with over 4,000 stores in the US, I have shown just how powerful retail app analytics can be in increasing consumer spending as well as loyalty.

Let's switch to another example, in platforms like social media websites, analytics tools enable businesses to measure engagement metrics like click through rates shares as well as sentiment analysis from customer comments. This goes beyond just tracking numbers. It helps marketers understand what types of content would work best on each platform and at what time their audience would be most active.

For example, an outdoor gear company might discover through social listening and through sentiment analysis that their posts showing adventure stories generate higher

engagement among customers than traditional product ads. Armed with this insight, they can refine their strategy to focus much more on storytelling, resulting in a stronger brand connection as well as higher conversations among customers.

They can even integrate generative AI tools. Both of my examples highlight how marketing analytics transforms data into actionable insights, enabling companies to make smarter, faster and more impactful decisions. Creative approaches to data analysis can directly improve customer relationships and drive growth.

Ron Guymon: Sports are a big business, so let's consider analytic examples from that domain.

Baseball is a sport that easily lends itself to statistical analyses because each play is a discrete sequence of events that pretty much starts the same way. For instance, a right handed batter is going up against a right handed pitcher and we pretty much know where all the fielders will be standing.

Moreover, it's pretty easy to keep track of how many pitches a pitcher has thrown during that game. In contrast, basketball is a much more fluid game in which plays blend into each other and players can be in many different places at the beginning of a play. So enterprising companies have created video software that takes 25 pictures per second and tracks the coordinates of a player and the ball.

This allows new metrics to be considered for identifying undervalued players, such as those who are able to make baskets when defenders are really close. It also allows analysts to use sophisticated statistical analyses to evaluate the defensive value of players. Wearable technology also helps teams monitor players, ensuring that they're rested and ready for key moments.

Much like how scientists monitor environmental conditions to preserve this canyon, these examples highlight the innovative use of data to enhance performance. This example illustrates creative uses of data gathering and analysis to improve business. Consider the fast paced world of stock trading. It's like trying to react to fleeting changes in the light on canyon walls.

To compete, traders use automated systems that track news and social media, interpret information, and execute trades in fractions of a second, much faster than any human could. However, you can assume that a bot was set up to track news feeds or social media posts, interpret those bits of news and then place trades fast.

In fact, the bot did it much faster than a human like within three seconds of the news coming out. This example illustrates the use of data gathering and analysis to improve business. I think the advances in technology like data analytic tools, video capturing software technology and AI are something to be loved and not feared.

Some people are concerned that machines are taking over many jobs and I can see where they're coming from. However, those jobs are often quite repetitive and boring. Thus, many mundane tasks can be delegated to machines so that humans can do more interesting things.

FACT Framework

Ron Guymon: In this lesson, I'm going to review a simple framework for making data-driven decisions, the FACT Framework. Before we get to that framework though, let's hear from someone else. I interviewed Luis Guilamo, Director of Analytics and Application Development for the Buffalo Bulls, an NFL American football team. Towards the end of the interview, I asked him what other advice he has for using data analytics to make decisions. Here's what he said.

Luis Guilamo: Really, following a structured process to come to your findings, always follow a very similar approach of breaking down your problem, going through it methodically, is really important. Accuracy. That process leads to accuracy, that accuracy leads to trust. The trust in the data and the trust in the results is what will continue to drive analytics forward. I think whether it's football, or medical information, or retail, it doesn't really matter. If you provide inaccurate results, then there won't be trust in the data and it'll be difficult to continue.

Ron Guymon: As Luis mentioned, the process is really important. Let's now talk about the FACT framework.

The first step in this framework is to frame the question. Identify a problem that needs to be solved. Consider how you can frame the problem in a way that will encourage people to use data to find the root cause and that will lead to some insight on which you can act. At some point, you'll also want to consider framing the question in a way that can be answered using a particular analysis.

Second, assemble the data. Financial data is often important because it helps establish the impact of something on profit. However, other internally available non-financial data, as well as external data, should also be considered. You may even want to gather your own data. Once you identify data, you'll need to find a way to extract, to transform, and load the data, which is often referred to as the ETL process.

Finally, there's a fair amount of data wrangling that will need to take place so that the data is structured for the calculations that you'll perform on it.

Third, calculate the results. Start by making sure you have a good feel for the basic summary statistics of your data. What is the distribution of the factors in your data? How many values are missing? How correlated are the factors with each other? After you have a feel for the data, then you can start performing advanced analytics on the data. These advanced algorithms can provide insight by surfacing patterns in the data. There are a variety of algorithms to choose from. Understanding the strengths and weaknesses of each one can help you know when they should be applied. You should also know how to evaluate the results of the models that are created by the algorithms.

Fourth and finally, tell others about the results. Use technology and methods of communication that are accessible to your audience. Take effort to make sure you balance the main ideas with the nuance. Too many nuances and you will confuse and lose your audience. Too few nuances and your audience may over-generalize your results. As you communicate the results of your analysis to your audience. Make sure to observe their reaction. Be open to receiving feedback and acknowledge the limitations of your analysis. As you open yourself up to other opinions and perspectives, you'll most likely be able to improve your analysis and gain additional insight as a result. Almost always, you'll find additional questions to pursue as a result of digging deeper into the underlying cause of the problem that you're addressing. That's the framework.

But I want to emphasize that each part of the framework is important and useful only in conjunction with the other steps. For instance, what if you don't frame a question?

Consider what could happen if you just start assembling data without having first framed a question to answer, or without having any idea of how it could be useful, or how you're going to make calculations with it. Even though information isn't an asset on your balance sheet, it has similarities with inventory. The more data you have, the more you have to invest in storing it, keep track of it, update it, protect it, and granting access to it. Thus, deliberately framing a question and then gathering data to answer that question is ideal. It can also serve as a compass, a good reference point to go back to if you get sidetracked on a tangential analysis.

Now, what if you don't assemble data? Framing a question is a good starting point. But if you don't have any idea of what data is available to use, or if you don't know what can be done with it, then you'll have a hard time framing the question in an effective way. Also, it's often the perception that sophisticated algorithms can find patterns in any data, especially if there's a lot of it. There may be some truth to that. But more often than not,

you'll make better use of your time and resources by assembling the right data. Thus, it's important to assemble a relevant set of data.

While assembling data is super important, what if you don't make calculations with the data? Sophisticated algorithms are really useful because they can identify complex patterns in the data that could take humans a much longer time to identify. For instance, identifying the optimal ticket price often requires a consideration of many factors, such as time of the year, day of the week, holidays, weather conditions, and macroeconomic conditions like oil prices, to name a few. It would take a human a long time to evaluate all of the combinations of those factors to arrive at the optimal price.Thus, using powerful calculations is important because it can arrive at that optimal price much quicker.

Finally, what if you don't effectively tell others about the results? Even when the question is framed well, a nice dataset is assembled and powerful calculations are made. It's important to communicate the right information to the right people in the right way so that it is acted upon. Failure to accurately and succinctly communicate results is a common reason why companies fail to become more data-driven. Thus, being able to effectively tell others about the data is crucial.

The FACT Framework provides guidance to convert data to improved business performance. Doug Laney, the author of *Infonomics*, has a great graphic called the Gartner Analytics Ascendancy Model, which also highlights that data analytic analyses should focus on action. I like this model because it highlights how insight can be turned into foresight. Identifying relationships in the data can help you move from descriptive analytics to prescriptive analytics. In other words, historical data helps identify relationships which can then be used to predict what will happen. If that prediction is unfavorable or unprofitable, then you're better prepared to recommend a course of action that will prevent the unprofitable from occurring.

In conclusion, as you start on your journey to use data and advanced analytics to answer business questions. Remember that all steps in the FACT Framework are important. You can go from simply descriptive analytics to prescriptive analytics.

Why Python for Business Analytics

Ron Guymon: Here I am on a really windy part of the Navajo Trail in Bryce Canyon. You might wonder why I'm here and what this has to do with the course, hopefully I'll make a few linkages that will be helpful for you. First of all, this windy trail reminds me of a snake and the key tool that we're going to use is Python, the coding language.

Now you might ask, why are we using Python instead of low code tools like Excel and Power Bi and Tableau? Well, the main reason is because Python allows you to perform every step of the FACT framework, whereas the low code tools only allow you to perform one or two steps.

It's kind of like being able to just go on the main parts of the trail here and not being able to explore the off-beaten path parts of the trail. Python allows you to do that, another benefit of using Python is that it lends itself really well to AI help.

So tools like ChatGPT and Gemini, they provide tremendous help in generating code that you can use for business analytics. It's kind of like having an expert guide that can take you up and all around these hoodoos here. So I hope you're as excited as I am to use Python and explore business data, just as I'm as excited to explore the other trails here in Bryce Canyon.

Python Accoutrements

Ron Guymon: In this video, we'll talk about how to use Python once you've installed it. While you could use Python by itself in a simple text editor type of environment like command prompt or Terminal, most of the time it doesn't make sense to use it that way. Instead, most of the time you will use Python along with other accoutrements or accessories that make it much easier to create, read and run Python code.

The three main accoutrements are one a computer for running the code, two programming modules so that you can use functions rather than recreate commonly used code, and three integrated development environments or IDEs to help you remember the syntax and the functions for running the code. Now, let's consider an autoshop analogy that will hopefully put each accoutrement into perspective.

When you need to repair a car, many of you will take it to a mechanic. This mechanic is like a computer that can perform certain tasks much faster than we can perform them. The mechanic communicates with other mechanics using a specific language. The language that they use to communicate is Python in our case.

Mechanics have a variety of tools that they use to perform certain tasks. The mechanic buys the tools once and will then only bring the set of tools to the car that are relevant to the problem that they are fixing. For instance, ratchets and screwdrivers would be used for almost every task, so they may keep those in a general-purpose toolbox.

They would have a different toolbox for diagnosing problems which might contain tools like multimeters and scan tools that they would bring to the car if a check engine light is on. They would have yet a different toolbox that would include a jack, jack stands and a torque wrench which they would bring to the car if they were fixing a flat tire.

The various toolboxes are like programming modules which contain a set of functions that make it much easier to perform specific tasks. Just as a mechanic does not bring all tools to the car every time they work on a car, only specific programming modules are imported based on the needs of the task.

The programming modules need to be installed only once, but they must be imported to the environment every time you use functions from the module. Finally, the mechanic most likely works in a garage that helps them keep track of their tools, has quick reference posters on the walls, and has ways to adjust the lighting and maybe can even run the car to test out if the repair worked.

This garage is like the integrated development environment that is used when working with Python. Now, integrated development environments include ways to quickly reference code rather than recall it exactly from memory. They also help to make sure that your code syntax is correct, and they can even help you run small parts of your code and evaluate the output.

I hope that analogy is helpful. The important thing at this point is to recognize the distinction between Python and the accoutrements, so that one you have a better understanding of how Python is used, and two you know that you will not have to memorize all the code. I think it's particularly important for those who are new to Python to recognize that they will not have to commit to memory the code and processes, because that can be overwhelming.

Now, as we go through the other videos and lessons, you'll gain more experience with using Python and its accoutrements.

Python and Integrated Development Environments

Ron Guymon: In this lesson, we want to distinguish between Python code and the integrated development environments or IDEs that are used for helping to create the Python code. Now, in a previous course, I introduced very briefly the Visual Basic Editor, which is an integrated development environment for running the Visual Basic for Application language, which is used in Excel for creating macros. Now, what you see on the screen here is the Visual Basic Editor. In this particular box right here, this window,

is the Visual Basic for Application code as well as comments that performs k-mean clustering. Now, this environment is really helpful. Let me just quickly illustrate why. In this code, if I want to see how it's working, I can set a break point, for instance, and then I can click the "Play" button. Now, you see in this locals window the different variables that I have created and the values that are assigned to them. If I click the "Play" button again, it will go through and if there are any values that get updated, then I can scroll through and see what those are in here. This is really helpful as I walk through the code and make sure it's doing what I think it's doing.

Another great thing about using the Visual Basic Editor in Excel is that they're in different window. If I have two screens or I want to split my screen, I can actually step through the code and see what is happening with the data itself, or maybe the charts that are created in the Excel worksheets. Notice when I click the "Play" button, a new cluster assignment is made. I can even remove this break point and run through the rest of the macro, and it tells me it's done.

You can see that it has gone through this cluster, this k-mean's cluster analysis, and assign each observation to a cluster. It's really helpful to be able to see the output as you run the code, especially when you're developing that code. Now, this idea of having software for creating and troubleshooting code and stepping through it is so important because it is very unlikely that you're ever going to create code that does not have any type of error in it. It's also very unlikely that you'll be able to remember all the code itself. It's really important to have tools to help remind us of what we need to run and to be able to see what we're doing.

For the rest of this video, I want to introduce you to four IDEs that are used with Python. We will talk about how to install these IDEs and the strengths and weaknesses of them in later videos. At this point, I just want you to see that there are different environments so that you can really distinguish between Python itself and the code. These first two IDEs are made by Jupyter. This first one is called Jupyter Notebook. You can see it's a browser-based tool. This looks really nice in here. You can format text in here so that it's easy to read. Then you can have other code chunks or cells that contain the Python code. This first chunk in here, this first cell, is really just text. It's not code. But that helps me present the code and the output of the code so that it's easy to read. This is the Python code itself.

Similarly, here's some more Python code. This particular set of code reads and stock data and plots it. I can just scroll briefly down through here and see a multi-line plot of Microsoft and Nvidia stock prices over time, as well as what's called a pair plot. This is the Jupyter Notebook and you'll see this a lot. It's really simple and easy to use. The

second IDE is also created by Jupyter and it's known as JupyterLab. Also a browser-based tool. It actually looks very similar to Jupyter Notebook, although this has a black background. We can set it so that it's white just like the Jupyter Notebook. But it has more tools that are integrated into it. It has windows within this environment itself as well as a file explorer. This is one that we hope that you become familiar with as well.

The benefit of this is that it looks a little more complex. But as you get more experienced with Python, you will really like having some more tools that are available to you. Now, these other two tools that I want to introduce you to are more traditional development environments that resemble the Visual Basic Editor. This IDE is called Spyder.

Here is the Python code in this window here, then down here there's a console so that you can see the output of running that code. Then we've got the plots that we can see over here, and then the File Explorer here. There are several other tools in here for reading about help. I should just pause here and say knowing how to find help in an IDE is critical. That's one of the key things to know when you're coding. A key secret is that you don't have to memorize everything. You have to learn a lot and remember it, but knowing how to use the built-in help is the most important thing I think you can do at this point.

This is the Spyder Integrated Development Environment. The fourth one I show you is actually very similar to Spyder, although this one has a white theme to it. This IDE is RStudio. This one is useful for those of you who want to learn how to use R and Python together. This is the code window over here, the editor, this actually contains the Python code. Then we've got a console window down here that allows us to see the output of running the code. We've got the File Explorer here as well as the plots. We can go through and see the plots and then the environment where we can see the variables that are created. We can even open them and explore them as we would in an Excel worksheet. These are the four IDEs that we wanted to introduce you to.

Like I said, these Jupyter IDEs are the key ones for this course, but you will be exposed to these other IDEs. We will explore the strengths and weaknesses of all of these IDEs in later videos.

Installing Python Using JupyterLab Desktop

Ron Guymon: In this video I'm excited to show you how you can get started using Python using one of the easiest approaches possible, and that is by downloading and installing the JupyterLab desktop application. One of the great benefits of this approach

is that it's nearly identical regardless of whether you're using a Windows or a Mac operating system.

So let's jump over to a web browser and talk about how to do this. The first thing that you could do is search for JupyterLab desktop and make sure that you spell JupyterLab correctly. Notice that it's all one word and Jupyter is spelled J-U-P-Y-T-E-R. And that name is paying homage to the fact that Jupyter works with various data analytic languages, including Julia, Python, and R.

Now, one of the links you should see is a link to a GitHub repository and it'll say JupyterLab desktop application, based on Electron. So if you click on that link, it'll take you to the GitHub repository. Now, GitHub is a website where developers share the code if they want to make it freely available.

And jupyterlab-desktop is an open-source free application. So once you get here, you can scroll down and there's a little bit of information about what this application is and a lot of other information, but we're gonna just stop this installation section. Now, in here there's a table and we've got three columns.

One that pertains to the three main operating systems, so Windows, Mac and Linux. If you're a Linux user, you probably know what to do from here on out. If you're a Windows user, then you'll want to click the link for the x64 Installer and that will download the installer onto your computer.

If you're using a Mac like what I'm using in this video, then there are two links and one is for the newer version of Mac, Apple Silicon. The other is for the older version that uses an Intel chip. If you're not sure what version of Mac you have, you can go up to the Apple icon and click about this Mac.

And where it says Chip, if it says Apple M something, then you're using a newer version. These are the Apple Silicon versions here, okay? M1, M2, M3, or M4 or something else. Okay, so since I'm using a newer Mac, I'm gonna click on that link. I already did that and it downloaded the installer.

Now, this installer was 300 megabytes. So depending on your Internet connection, you may have to wait a little while, but once you've downloaded the installer, then go ahead and double click on it. And for Mac users, this is kinda the standard. You just take this icon and drag it to your applications and it doesn't take very long at all for it to install on your computer.

Once you've done that, you can X out of here. You can delete the installer if you want, and then you can open up the JupyterLab desktop application just like any other

application. So I'm gonna go to my application section of my finder here and I can see that I've got JupyterLab.

So I'm gonna go ahead and double click on that. And if this is your first time opening it, then Mac requires you to indicate that you're okay opening this. And I am, so I click the open button. Okay, now this brings us to the welcome page. And in the welcome page here, one of the key things is on the left-hand side where it says Start and there are three links that are grayed out and then a fourth link that you can connect.

We want to make these grayed out links available for us to use. So right now we only have the jupyterlab desktop application. We still need to install Python. So at the very bottom we've got this warning that says Python environment not found, and it gives us a couple of links down here.

And the easiest thing is to click this link that says install using the bundled installer. So I'll click that and then this may take a little bit of time, not too long, but it's installing Python as well as some commonly used Python modules for data analysis. So we'll just wait for that to complete.

Okay, it said the installation has succeeded and okay, in just a few seconds then these grayed out links now become clickable and you could just create a new notebook. But if you do that, it'll create it in your root directory, which is probably like your users/your name folder.

And so what I'm going to do instead is click on this New session, that's what I usually do. And this will open up the JupyterLab environment. You can decide whether or not you want to get notified about official Jupyter news. I'm going to click no. Okay, now if your environment doesn't look exactly like this, one thing you could do is go to Settings, actually go to View, and then go to Appearance.

And you can see what I have selected and unselected here. So I have unselected the Simple Interface and the Presentation Mode. I think I also just clicked on the Reset Default Layout, it made it look like this. So this may be the easiest thing is just click on this Reset Default Layout and looks like it brings us to here.

Okay, now this is all ready to go. You can navigate to different folders on your computer here and I've got this Test Folder. Once you get into here, meaning the folder that you want. You can create new files by clicking this plus button and we'll talk about how to use JupyterLab in another video.

But anyway, this is how you can download and install JupyterLab and Python in a really easy to use way. So I highly recommend this approach if you're just getting started using a programming language.

Installing and Running Python Using the Anaconda Distribution

Ron Guymon: In this video, we want to help you get started using Python on your own machine. So, we're going to show you how to install Python as well as those four IDEs Jupiter Notebook, JupyterLab, Spider, and RStudio, all at one time using the Anaconda distribution.

Now Anaconda is a software, open-source software, meaning it's free, and it allows you to download all of those in one package so it makes it really easy. So, to do this, go ahead and navigate to anaconda.com in your browser, and that will take you to the homepage for Anaconda. Then go ahead and hover over the products menu item and select Individual Edition, Open Source Distribution. And from here it should detect what type of machine you're using and prompt you to download the right version of Anaconda whether you're using a Mac or Windows operating system or something else.

All right, so I'm going to go ahead and click on this download button, and you can see that it started downloading. It also prompts me to sign up for some other help which you can if you want but I'm not going to do that. This is 515 MB, so I'll wait a few minutes and then get started once it is finished downloading.

All right, now that it has finished downloading, I will go ahead and open the installer. And then I will follow the instructions to just download it in the default way. [SOUND].

All right, now that Anaconda is installed, I can go ahead and open it as I would any other application. All right, so this is called the Anaconda Navigator, and you can see that this contains those four IDEs that we discussed as well as several other applications. So here's JupyterLab, Jupyter Notebook, Spider, and RStudio. And if I want to run a Jupyter Notebook IDE, I can click launch, and in my browser, it will bring up this dashboard page and we'll talk more about how to use Jupyter Notebook in a later video. But that's how you would start Jupyter Notebook.

JupyterLab, I could click launch and it will bring up the JupyterLab in a window on my browser as well. You can see it's the localhost, so it's running on this machine here. All right, and in another video, we'll talk about how to use JupyterLab.

And you can launch Spider from here. All right, so here's Spider, and I can also install RStudio and then launch that if I wanted to, it's not necessary for this course though. And then when you're done using these IDEs, you can simply go ahead and close the Anaconda Navigator as you would any other program. So, I'll just go up to menu item and select Quick Anaconda Navigator.

All right, and there you have it, so that's how you can get started using Python and those IDEs on your own machine. And the Anaconda Navigator is really convenient. However, it does have a few drawbacks because it has those IDEs and other applications as well. It's a little bit bloated and so sometimes it will run slower and installing additional applications is different than if you just install Jupyter Notebook on your machine. We can talk about how to do that in another video.

Installing Python Using Homebrew and Pyenv (Mac)

Ron Guymon: In this video, I'll demonstrate how to install Python on a Mac operating system using Homebrew, which is a free and open source package manager that makes it easier to install and uninstall software on a Mac. So the first thing we'll want to do is make sure that we don't already have a version of Python 3 installed.

To do that, let's go ahead and open the Terminal application. I'm going to use a Spotlight search by pressing Command in the spacebar and then type out Terminal, then hit Enter. And then to see if we have Python installed, type out Python 3 version and hit Enter. And if it's installed, then it will return a version of Python, like in this case, 3.12.0.

Okay, if it's not installed, then let's keep going through here. If it is installed, then you may not need to do anything else. All right, so since we're gonna use Homebrew to install it, let's see if we have Homebrew already installed. So in the terminal I'm going to type out which brew, and if you get a path to where it's installed at, then it means it's already installed.

If it's not already installed, then you can navigate to Brew sh or just do a search for it and it'll probably take you to brew.sh. And in here it gives us the Terminal command for installing Homebrew. So all you have to do is click on this clipboard icon and then you can go back over to Terminal, paste that, that big long command and hit Enter.

Now you may be asked to enter your password, I'll go ahead and do that. And then it tells you what will be installed. And it will install a variety of things, and it's okay to install all those things. I guess you can decide if it's okay, but I'm gonna go ahead and decide it's okay and hit Enter.

Now, I've already got it installed, so it should go really quickly for me. If it's not already installed on your computer, then you're gonna be prompted to install some additional software. And if you're prompted to install xcode, then go ahead and do that, that is Mac's IDE for creating software for Mac operating systems.

Now, as you're going through the installation process, you will be prompted to install Homebrew to your path so that you can easily access Homebrew. So make sure and copy and paste the commands that it gives to you and run those in the Terminal window as well. All right, and then once you've gone through everything and it says installation is complete then you can check if Homebrew is installed again by typing out which brew.

So which and brew and you should see a path for it. Now, at this point, you could install Python from here. All right, now, don't do it quite yet I'm gonna tell you another way. But you could use Homebrew to install Python by just typing out brew install Python3.

So this would work, but I encourage you to install Python using another package called Pyenv for PyEnvironment. So to see if you have Pyenv installed, then you can type out Pyenv -- version, and if it's installed, then it'll tell you a version number like this. If it's not installed, then you can go ahead and type out brew install pyenv and hit Enter.

And it will go through the installation process with you. You can see that because I have it installed it says it's already installed and up to date. Now, at some point along the installation process for pyenv, you will be prompted to copy and paste some commands that will make it easier for you to use pyenv.

And so it's okay to go ahead and copy and paste those, and then as you hit Enter and accept everything else, it should finish the installation process for you. Okay, now, at this point, you can install Python using pyenv through Homebrew. It's kind of complicated, but it ends up making it a whole lot easier.

So the benefit of using pyenv or Python environments is that it allows you to quickly install different versions of Python and switch between those versions of Python. And that can be really helpful because not all modules that you'll want to use work with all versions of Python, so you may need to use an older version of Python for some projects.

All right, so let's see what versions of Python are available for us to install through pyenv. If we type out pyenv and then install, then -l for List, it will give us a list of all the different versions of Python we can install. You can see there's a ton of them in here.

We want the ones up at the very top that are just numbers here. Then you can go down and you'd probably be okay getting the most recent stable release that doesn't have any letters after it. So 3.13.0, but they're often updates, so it may be different when you watch this video, but you could probably go ahead and install that one.

So let's say I want to install Python 3.13.0, then I can just start typing out pyenv install 3.13.0 and it will go ahead and install all the software that we need for that version of Python. Okay, once it has finished installing Python, then you can type out pyenv versions, no double hyphens this time, and hit Enter.

And it tells you the different versions that you have installed and it'll have an asterisk next to the version that is being used by your computer. All right, so you can see I've got version 3.12.0 and 3.13.0. If you want to switch versions that your computer uses, then all you have to do is type out pyenv global and then the version that you want.

So 3.13.0 and hit Enter and now you can see if I rerun pyen versions 3.13.0 has a star next to it. So I know it's more complicated to install Python this way on a Mac, but ultimately it makes it really easy because you can switch among the different versions of Python and it's also really easy to uninstall a version that you don't need anymore.

In another video, we'll talk about how to install an integrated development environment like Visual Studio Code or Jupyter Lab to make it easier to use Python rather than just using it in the terminal like this.

Installing Python from Python.org (Windows)

Ron Guymon: There are a variety of ways to install Python on your computer. The approach that I will demonstrate in this video is to install python directly from python.org. Now, for those of you using a computer with a Mac operating system, I encourage you to watch another video in which I talk about how to install Python through homebrew, and I talk about the benefits of that approach and how to do it in that other video.

That being said, you can install Python through python.org on both Windows and Mac operating systems. The first thing you should do is, make sure you don't already have Python installed on your computer. The way you should do that is to open up either the command prompt if you're on a Windows machine, or the terminal if you're on a Mac machine.

So I'm going to use a Spotlight search, by pressing command spacebar and then type out terminal and hit enter. And that'll bring up this basic window here. And then once

I've opened this, I can type out Python3 --version. And if a version of Python3 is installed, then you'll get a message that tells you Python and the version.

So you can see the version that I'm using is Python 3.12.0. If you don't get anything, then you need to install it. Alright, so to install Python, navigate to python.org and then go to the download section. And your browser should detect what type of operating system you are using.

And you can just click this button here that will encourage you to download the latest version of Python. So I'll go ahead and click this button here, Python 3.13.2 and it's downloading the installer and into the downloads folder on my computer. Okay, once it has been downloaded, I can go to my file navigator and I can double click on the installer, and this will walk me through the steps for installing Python.

So you'll need to go ahead and read through various pieces of information, what's being installed, where it's being installed, the license agreement, and so on. So I'll let you go ahead and decide how much of that you want to read. And for those using a Mac, you can go ahead and accept the default settings.

If you're installing this on a Windows computer, you'll get to a point where you can either click Install now or a custom installation. And before you click on Customize Installation, make sure to click on Add Python.exe to PATH, because that will allow you to start up Python in an easier way.

All right, and then you can decide if you want to use admin privileges, when installing the executable file, and then click on Customize Installation. And that should take you to another Window, in which it tells you all the additional features that you can install along with Python. And you could go ahead and click all of them, but the key thing here is to make sure that pip is checked.

And pip is a package manager, which is very helpful for installing additional modules that will allow you to extend Python for a lot of different purposes. All right, so once you click that, then go ahead and click next, and for both Mac and Windows users, you can accept the remaining settings and finish installing Python.

Now, once you've gone through the installation process, you should be able to go back to either the command prompt or the terminal, and type out Python3 --version, and verify that you have the latest version of Python installed. You can even use Python from this shell script right here, either the command prompt or terminal.

So if you just type python3 and hit enter, it takes you into what's called the REPL with these three greater than signs, and you can do some basic Python calculations in here. So five plus six enter is 11, all right? And then to get out of it, you can type quit, open, close parentheses, and then you're out of Python.

You now have Python installed. However, make sure to follow along in other videos where we demonstrate how to install an integrated development environment like Jupyter Lab or Visual Studio code, so that you can use Python more effectively than using it just in the shell script.

An Example Workflow with JupyterLab

Ron Guymon: We like to teach by example. So in this video, I'm going to show you a workflow for reading data in from an Excel file and we'll read it into a Python environment in JupyterLab. And then I'll explore the data a little bit, manipulate it, create a visualization, and save it as a comma-separated value file on my computer.

Now, the purpose of this video is not to explain every single detail that I do, it's just to help you get a feel for how data analytics works within the JupyterLab environment. In later videos, we'll go into more detail about how the code works and how to use JupyterLab.

Once I've started JupyterLab, I'm going to go to the file browser and I'm going to navigate to where the data is located on my computer. And in my desktop, I've got this folder right here. Okay, so here's the data file sba_disaster_loan_data_fy22.xslx. Since I'm now in the right place, I'm going to create a new notebook file and I'm going to save this new untitled.ipynb file with a better name.

So I'll click Save Notebook, so Loan Disaster Data Exploration, Rename. And now I can see that it's been renamed there and I've got this new ipynb file in that same folder on my computer. All right, I'm gonna close that. Let's go ahead and give this a title. I'm gonna change it to a markdown file, then type out a title, run that, and there's my title.

All right, let's go ahead and start doing the analysis. First I'm gonna import functions from the Pandas module. And now I want to read in the data and save it as a data frame. I'm gonna create a new data frame object df and set it equal to the pd.read_excel() function, and in here I'll type out the name of that file.

So I'm gonna go back into here and just right click on it, and copy path, and paste it in there. Get rid of the whole path name, I just need the file name. All right, now, to help you get a feel for what this file looks like, here it is in Excel.

And you can see I've got multiple worksheets. I wanna read in data from this FY22 Business worksheet. So in addition to the file name, I need to indicate the sheet that I want to read in. All right, and then once I read it in, let's make sure it works by displaying the first five rows of the data frame.

Okay, so there it is. Now I can see right off the bat that I've got some column names that don't make sense. The column names that I want are down here. And so I want to skip rows 1, 2, 3, and 4. So there's another argument in here, skiprows, set that =4.

To make this easier to read, I'm gonna put these on new rows like that. All right, we run this. Okay, now I've got those column names that are much easier to understand now. All right, so let's look at this data a little bit. Just by exploring it visually, I can see I've got different observations from losses that have happened as a result of disasters for some small businesses and the location.

All right, the key columns are the total verified loss and then the total approved loan amount. Let's say we want to evaluate the total approved loan amount as a proportion of the total verified loss. So we wanna create a new column in here that's a percentage. So I'm gonna create a new code cell.

I'm gonna create a new column name. All right, so the new column will be loan_pct_of_loss, and that's going to be equal to the Total Approved Loan Amount divided by the values in the Total Verified Loss column. And then to make sure it worked, let's go ahead and display the first five rows of that data frame and scroll to the far right, and there it is, that new column.

Okay, it looks like it worked, although I've got some observations that don't have a number in it because there's no total approved loan amount. Let's just calculate the average value of that column there. So I'm gonna create another code cell and let's calculate the average. And it looks like it's infinity.

Okay, so again, I've got some observations that don't have any loan amount there. So let's go ahead and filter this so that it's less than infinity. So I'm gonna create another data set df2, and this will be equal to the data in the df data frame, but we're gonna reduce the observations only to those for which the loan percent of loss is less than, let's say 1,000, bretty big.

All right, and then df2., all right, so calculate the average again. Okay, so we've got an average of 2.3, but let's look at how the data is distributed in more detail. So another code cell, let's create code to create a histogram. Okay, I'll run that and here's my histogram.

Looks like I've got tons of observations between 0 and about 20, but I must have some outliers that are way up here. They're really big. So let's change this so that we're only looking at observations for which the loan as a percentage of the loss is less than 5.

Rerun that and create the plot again. Okay, this is much more informative now, but I wanna make it even more informative. I'm gonna create some more bins instead of just the default of 10 bins here. So let's go into here and say bins=100, and let's give it a title.

And let's put the title on a new row so it's not so wide. And I'm also gonna add a semicolon at the end so I get rid of this text right here. I'll rerun this. Okay, this is a much more informative histogram now, I've got a title up here, and so I can see a bunch of the loans are at this one, which is 100%.

So the loan amount is the same as the total verified loss and then most of the others fall below that but we do have some that are above that for some reason that we don't know. Okay, now, perfect. Let's assume we're done here, but we wanna save the data for future analysis.

So I'm gonna take this data in df2, I'm gonna save it as a CSV file. I'll give it a name of smaller_loans.csv. And we wanna set the index_label=False. Okay, then I'll run that, and if I go back to my file browser. In a few seconds, I'll see that file show up here if I ran it correctly.

Okay, there it is, smaller_loans.csv. I can double click on this and explore it, much like I would in an Excel file. Anyway, it looks like it worked fine. Let's go ahead and summarize what we see here. So I'll create another code cell, change it to markdown, and let's tell others about the takeaways.

I'm gonna go back in here and I'll create a bulleted list here of the takeaways. Okay, so there are some loans given that are worth more than the loss. Maybe I'll add one other takeaway. Okay, there seems to be a lot of loans, given, that are worth the same amount as the loss, and I'll run that.

Okay, fantastic. Let's go ahead and save this again. And now I want to export this so that it's easier to share with others. So I'm gonna go to Save and Export Notebook As

HTML, and in my downloads folder, I now have this HTML file. It's not interactive, but it's HTML so others can open it and view it, okay?

And you can see I've got this formatted text, my code, the output of the code, including the visualization and the takeaways right here. Let's say I wanna share with others as a PDF document. It takes some extra work, but if you install some additional software, you can save it as PDF like this.

And now it's a nice looking PDF document, although the data is displayed in a way that's less helpful. But everything else looks really nice. Got nice looking fonts here and everything. Another thing I could do is I could print this HTML file as a PDF document and then it'll convert it into a PDF.

So this is a workaround to get to PDF document. Anyway, there's an example workflow. I hope that gives you an idea for how to perform data analytics within JupyterLab. In later videos, we'll go into more details about the code as well as the keystrokes that I use to work with JupyterLab.

Tour of JupyterLab

Ron Guymon: In this video, I want to give you a tour of JupyterLab. Now, this video would be really long if this were a comprehensive tour. So I'm going to highlight some key features of JupyterLab to help you get started. And then as you use JupyterLab more, you'll become familiar with all of the nuances.

So, as you can see, JupyterLab works in a browser, and I have a ipynb file interactive Python notebook file here in the main area. I'll talk about interactive Python notebook files in a separate video. But let's talk about all these other menus and everything in JupyterLab. Let's go ahead and maybe just start with the Launcher window.

If you go to file and go to new launcher, and so this launcher window allows you to create new files. Most often we'll create new notebook files, IPY and B files. Okay, Python 3 and IPY kernel, this is an interactive Python kernel. You can also create console windows that won't save your code, it's just for running temporary code.

You can also create a terminal window, I'll click on this. And this allows you to interact with the command line interface of your computer, which is like Terminal on a Mac or command prompt on a Windows machine. And this is just a shortcut to access the terminal. You can see, I can also create a new launcher window just by clicking this plus button up here.

You can also create text files, markdown files, Python files, so .py files and show contextual help. And we'll talk about this in later videos as well. Anyway, this is how you create new files, so I'll X out of that. Let's go ahead and talk about some key things up here in this top menu.

So in this file menu, we've got lots of different options that you would expect to have if you're familiar with, like Microsoft Word or Excel. So you can create a new file from here as well. I can save the notebook file manually by default, it'll save your file like every minute or something like that.

But if you wanna manually save it, you can save it with a new name. And then of course, we've got the Save and Export the notebook. So you can export it into a variety of different formats. Typically, HTML is the one that we'll use most often. PDF is also very helpful, that's a format that many people use.

However, it requires some additional installation. So you're gonna wanna figure out on your own machine how to do that. We won't require you to do that for this course. There's also Web PDF and then there's also Reveal JS slides. So you can export this as a slideshow and then share it with others.

And then you can shut down your JupyterLab environment by using Shutdown right here. And of course there's other things, but those are some highlights in the file menu in this edit menu. This gives you a lot of different ways for interacting with the cells in here. So you can move them, split them, merge them, cut them and paste them.

The main thing in this view window right here is that I wanna show you right now at least is the show line numbers. So if I were to look at my code cells in here, I don't have any lines on there. And sometimes you wanna communicate with others like, hey, let's look at your middle row, right, or your second row and you kind of have to count down.

It gets especially hard when you've got like 25 or 30 rows in a cell. So you can go to view and show line numbers. And now you can easily refer to what's on separate lines of your code there. All right, you can turn that on and off by using the shortcut keys right here, so shift L.

These shortcut keys will become more useful for you as you start using JupyterLab more. But to show how it works, I can just push shift L and turn those line numbers on and off. All right, I'm gonna skip the run one for now, but in kernel we've got some important commands right here.

You can restart the kernel, which is kind of like turning off Python and turning it on again. So if something is just not working, sometimes you just kind of wanna unplug it, so to speak. And rather than actually doing that, you can just restart the kernel. You can also restart the kernel and clear the outputs of all cells, so I'm go ahead and click that.

And now you can see that I'm left only with the cells in here and not the outputs of them. Then you can also do restart kernel and run all cells. And it's gonna go through and run every single cell. And this is really helpful because if you had an error in your code, let's just say I have that and I'm about to turn in something or share it with others.

Then you'll want to run everything first to see if there's an error in here. And as it goes through and runs, it'll stop when it gets to an error, it didn't keep running this. So anyway, that restart and run all is a really helpful one. Or there's also this restart kernel and run up to a selected cell.

Okay, so rather than going back and manually running everything, you can just do that right there. All right, Tabs, I don't use that too much, so I'm gonna skip over that. Settings is helpful so you can change the theme to like JupyterLab Dark if you like this type of theme.

All right, I'll go back to JupyterLab Lite. I really like this auto close brackets enabled. Because what that allows me to do is when I' typing out like I want to type out a column name, I just type the open bracket and it automatically adds a closing bracket.

Same with parentheses. Okay, so I like enabling that, you can decide if you wanna enable that or not. But I've enabled that. You can change the font size, you can also just push control plus and minus to increase the size. And then finally, the settings editor has a lot of useful things in here.

So I one of the things that I find most useful in this code completion section, I like to enable auto completion and then have this suppress when the inline completer is active. We'll talk later about using the built in help. But anyway, I like setting that and you can go through and change a lot of other things in here.

And then this help is actually really helpful. So in here you can learn about JupyterLab, what version you're running and links to learn more about it. Showing contextual help is something that is really useful that allows you to see help for running the functions that you have or maybe looking at your data.

And then there's also references for markdown in here. Okay, so that is really handy, you can see all this just opens up as another tab in here. Okay, so there's the top menu.

Let's go ahead and look at the left sidebar. So this is the file browser and you can navigate to different areas on your computer by just double clicking on things there.

Okay, and then this next item here allows you to see the tabs that you have open. And the thing that I find most useful is this kernel. Now, a kernel is a portion of your computer. And so if I had another ipynb file open, it would use a separate kernel so that anything I run in this file is not going to affect what's in another file.

And at some point if you have a lot of files open and running, your computer may slow down. So you can come in here and just shut down all of them or individual ones. Okay, just click that X like that or shut down all of them, all right?

And that should get things running faster if you have a lot of them open, okay? And then the next one is this table of contents right here. This is really helpful because when you have longer files it uses the headers as a way to help you navigate to different parts of your file.

So it's really important to format your files with headers. It allows you to navigate super easily. Then you've got the extensions manager and extensions I'll let you explore on your own. I won't use extensions in here because they change from time to time but feel free to add extensions if you want.

And then finally this is AI help and we'll talk more about this later on but this is something you have to add in separately. So there's an introduction to JupyterLab, it's just a foundation. The best way to become familiar with the nuances is to use it.

Using Interactive Python Notebook (IPYNB) Files

Ron Guymon: In this video, I want to introduce you to Interactive Python Notebook files, which are simply known as IPYNB files. Now, these IPYNB files are really popular, and they're popular for at least two reasons. The first reason is because they allow you to combine formatted text with code, as well as the output of the code into a single document.

So you don't have to copy and paste from a data analytics software tool into a word processing tool. The second great characteristic about IPYNB files is that they allow you to modularize your code by combining lines of code into cells that you can run separately from other code cells.

In contrast, in other types of files like PY files, if you want to run a portion of your code, then you have to enable special debugging settings in the IDE that you're using, or you have to change the default settings. So the inherent modularization of IPYNB files allows you to step through your code, which is super useful for data analytics.

And that's because it's really hard to keep track of what's happening to your data. So you'll want to run some code, evaluate the data to make sure the code did what you thought it would do, and then add more code to it. Let's look at an IPYNB file, and I'll highlight some things about it.

So in JupyterLab, I've got this file open, and if I click on my file browser here, I can see that it has a name, but at the end of it, it has this .ipynb file, okay? So again, interactive Python Notebook files. And there are other types of files that you can store a Python code in and run it.

In fact, if you click on the Launcher and go to the bottom, you can create Python files. And you can see that these are .py files, so you can still add code in here for interacting with Python, but it won't put the output of the code right below the code itself.

These py files are also really useful, but we're not gonna use them in JupyterLab. For the most part in JupyterLab, you'll use IPYNB files. A really important thing in IPYNB files are these cells here. So you can see that if you click on a cell, it highlights it on the left-hand side, and if there's output, it will also highlight that output as well.

If you've got a lot of output, you can click on this margin on the left right here, and that makes it so that it collapses it. And then you can scroll through it without having to show all of it and then have to scroll really far down. Okay, so that can be useful.

When you see the blue line, you know what cell is selected. This cell type is really important. There's two main cell types that we're gonna use, which is Markdown and Code. You can see there's also Raw, but we're not gonna worry about that, we don't use it that often.

Markdown cells, like this right here, are cells that allow you to create formatted texts. And I can learn about how to format the text by going to Help and going to this Markdown Reference. And so you can see right here, I've created a Heading 1 by using a single hashtag in a space.

A Heading 2 is two hashtags, and I think you can go out to six headings. Each additional increase in heading level will make it a little bit smaller. So let's create a new markdown

cell, and let's create a third level heading and then run this. And you can see it's definitely smaller than this one here.

Now, these headings are really important to learn because they allow you to use this table of contents in a really useful way. You can navigate to different sections of your document by using the table of contents on the heading. So use those a lot, super helpful. All right, we also have code cells, and these code cells, obviously, are the ones where you will enter the Python code.

Now, importantly, you can change cell types anytime. So if I've got a cell here, I accidentally started with it as a code cell, I wanna change it to markdown, I can just go to the drop-down menu and click Markdown. And now, when I run this cell, it's not gonna perform any Python function, instead, it's just text, okay?

I'll change it back to code cell. All right, now you've seen me running the code cells, this menu up here is something that you'll probably wanna use initially to start with. And you can take any cell and run it by pushing the play button. So if you double-click in a markdown cell, you can edit it, right, it's interactive.

And then to run it, then I can push play. Same with this code cells, I can click the play button, and that actually runs the Python code in there. You can stop a code cell from running if it's taking a long time. And there's a lot of other things, you can cut a code cell, you can copy it, you can add a new code cell or a cell just by pressing the plus button.

By default, it's code, and we could change it to markdown. Okay, we can remove it by using the cut right there. But at some point, pretty soon, you're gonna wanna learn to use the shortcut keys. If you go into the Edit menu right here, you'll see that you can cut cells by simply using X, okay?

So I've got this Heading 3 that I don't want. As long as I don't have the cursor in the cell, okay, so I click outside of the cell, I can just push X and I just cut that cell. Those shortcut keys become really useful, you can copy cells by just pressing C, and then pasting it by pressing V, deleting the cell by pressing D, D.

There's also some shortcut key references in the Help menu right here, and this show keyboard shortcuts, which you can also bring up by pressing Shift+Cmd+H. And when you click on that, it helps remind you of how you can use these IPYNB files a little more efficiently, just keeping your hands on the keyboard.

I don't remember most of these, but there are a few that I use a lot, like run selected cell, that's shift+Enter, show the left sidebar, Cmd+B. So when you press Cmd+B, what that does is it brings up the sidebar right here. And if I don't want it anymore, I can just push Cmd+B and that closes it.

So Cmd+B is a toggle key. Saving the notebook, Cmd+S, the table of contents, I use quite a bit, Shift+Cmd+K. And by the way, you can add the cell above the cell that's selected by simply pressing A, and D twice to delete it. I can add a cell below it by pressing B, and then delete it.

So there's a couple other things that will be helpful for you to know. Sometimes you'll see some cells that start with either one or two percentage signs like this. These are cell magics, they're not Python code, they are helpful processes that JupyterLab makes available by just using the percentages.

Okay, so those are called cell magics. So you can load code into the current frontend by using two percent signs, load, and then the name of the file that you want to load, okay? Another one that can be helpful is timeit. So now if I add that at the beginning of my cell, and then I have my Python code below it and I run it, you'll see that it will print out how long it took to run that code cell.

That becomes helpful later on, at this point, you don't need to know that, but you will see in other's code at this point pretty soon some cell magics like that. Another thing that you'll see sometimes is a exclamation point at the beginning of a cell. And whenever you see that, it means the contents of this cell are not gonna be Python code necessarily, but they will be terminal commands.

All right, so if I want to install a new package, I can run !pip, then install pandas. It shouldn't need to do that, but then I run that code cell and it performs that task right there. So that can be really handy. If you know other terminal commands, you wanna list the files that you have in your working directory or print your working directory, okay, you can run those and see where you're at.

So that's an introduction to IPYNB files and how to use them. It may be a little bit unusual at first to have to deal with all these different cells, but like everything else, the more you use IPYNB files, the easier it will be for you to use them.

In short, IPYNB files are really great for performing data analytic tasks with Python.

Tour of Jupyter Notebook

Ron Guymon: In this video, I am going to introduce you to the Jupyter Notebook integrated development environment, which is a simplified version of the JupyterLab integrated development environment. First of all, how do you access Jupyter notebook? Well, you can access it from the Anaconda Navigator, or you can start it up using the command line interface by typing out Jupyter, a space and then Notebook and then hitting enter.

You can also access it through JupyterLab, which is what I'm going to show you right here.

So, I've got a file open in JupyterLab called Loan Disaster Data Exploration. And you can see in JupyterLab right here, I've got this notebook icon. If I click on that, it takes that ipynb file and opens it up within the Jupyter Notebook environment.

And you may not even notice a difference at this point. But in JupyterLab, I've got these menus on the left and the right, which integrate a lot of additional functionality that is not included in Jupyter Notebook. Okay, so like I said, Jupyter Notebook is like a simplified version of JupyterLab, and that's both an advantage and a disadvantage.

Initially, when you're first using Python, you might like that because JupyterLab might be kind of overwhelming with all these different things that you can access. Okay, so Jupyter Notebook keeps it nice and simple. You run cells and create new cells and delete them the same way that you would as in JupyterLab.

It's just that you don't have all the other accoutrements that you have with JupyterLab. So, the disadvantage is that at some point you're gonna wanna have those other tools readily available to you. You can access the file browser by clicking on this Jupyter icon in the top left.

And in a separate browser tab, you'll see this file browser. Okay, so you can just click on the different folders that you have and files and open them that way. If I wanna open up another ipynb file from this browser window, I can do that. It opens it up as a separate tab in here.

You can shut down those that are running by clicking on that running, and then just exit out of it by closing the tab.

Ron Guymon: So that's the brief introduction to Jupyter Notebook. You'll use it sometimes, but I would say when you have the option between using JupyterLab or Jupyter Notebook, then you should probably go with using JupyterLab.

But at least now you know what it is. And you know how to use it.

Basic Calculations with Python

Ron Guymon: We want to help you get started coding with Python as quickly as possible. So in this video we're going to show you how to perform some basic calculations in Python, most of which you may already know how to use with a calculator or with Excel. So let's go ahead and jump on over to Jupyter Lab.

And I'm going to create a markdown cell and enter a title Mathematical Operations. And let's demonstrate some basic math operations. So if we enter 9+7 and run that code cell, it returns 16. And you can see that next to the cell we've got this 1. So the first input that it received during this session was this right here.

And the first output is the 16 right here, okay? So that's what these numbers in brackets mean. I'm gonna create a new code cell and let's try division. So 9/4, run that. Okay, gives us the answer 2.25. And we've got the second input and second output. Okay, let's try another calculation.

9 * 3 and run that, we get 27. Now what if I want to raise 9 to the third power? Then the way we do that is we use two asterisks, okay? So 9 raised to the third power is 729. And then there's a couple other operators that are not as common, but you may see them at some point.

So let's take 10 and divide it by 3. That gives us 3 and a third. But if we use two divisors, so two forward slashes and run that, it gives us just 3. All right, so this is floor division, where it returns only the integer from this division operation right here.

And complimentary to that, there's a modulo operator. So 10% 3, if I run that, I get 1. Now 10 divided by 3 is 3.333, or 3 and 1/3, 3 with the remainder of 1. Okay, so this one right here is the remainder. All right, so you can kind of break apart that division into the integer component and the decimal component.

Now a couple of these mathematical operators also work with text. So I'm gonna create another code cell and I'm gonna just enter some text. And the way we say we're gonna

create text is we put either single or double quotation marks around it. So I can just say Hello, but then if I take Hello and multiply it by 2, then I get HelloHello.

All right, so if you take text and multiply it by a number, it just repeats that associated with the number that you multiply it by. Let's take a string of text Hello, and then we'll add to it another string of text and I'll just say Hello space there and run that.

You can see that what it does is that it concatenates the two strings together. So now I've got hello there. Okay, so the multiplication and addition operators also work with text. Let's create another cell and let's talk about comparison operators. All right, comparison operators are things like greater than, equal to, not equal to, and so on.

So if we take 10 and say greater than 3 and run that, then Python will evaluate if 10 is greater than 3 and return a boolean outcome. Boolean is just a binary outcome, either true or false. So this is true. What if I try 10 is less than, and let's change up a little and say less than or equal to 3.

Okay, so the way you do less than or equal to or greater than or equal to is just put the equal after the inequality there. So I'll run that and this time it's false, all right? What if I want to test if 10 is equal to 3, then I can say 10 double equal signs and 3.

And when I run that it is false. And I'll talk about why you have to use double equal signs in just a minute here. Alright, one other thing is I want to test if 10 is not equal to 3. So the way I can do that is use an exclamation point followed by an equal sign.

And that is true because 10 is not equal to 3. All right, so those are the basic comparison operators that we will use a lot. All right, let's also talk about variables. I'm gonna say x = 3, and then below that I'll just type x. All right, you can see the result is just a 3 right here.

So what happened is Python first assigned the value of 3 to the variable x, then it printed x. Okay, so if I were to try doing something like 10 equals 3, I'm gonna get an error because I can't change what the value of 10 is. All right, so that single equals is used to make assignments.

And I can't assign 3 to the number 10. So that's why we have to use two equal signs to evaluate if things are equal to each other. All right, let's add in another variable. Let's say y is equal to 5 and let's say I want to print x and y.

I run that well, all I get is y. So you can see that the only thing that is displayed is the last variable. If I want to display both variables, I want to add the print in here. So I'll surround both of these by print and then run it.

And now I can see both x and y that are displayed. And of course I can use variables to create other variables. So I can say z = x*y. And then I'll also include print z here, and run that. And as you would expect, when I print z, we can see that it's equal to 15 because x is 3, y is 5, z is 15.

Now, if you're ever curious about the value stored by a variable, you can always just create a code cell and enter the variable and run it and see what it returns. But you also may wonder, what are the variables I have? So there's a helpful magic. This is not Python code, but a way to explore your variables.

And so if you type out percentage and then whose and run that, this will return a list of the variables that you've created during this session and tell you some information about those variables. I've got x,y,z, they're all integers and it tells me the value. All right, the last concept I want to illustrate is F-strings.

F stands for formatted. So if I were to try and combine something like hello + z, which is 15, I'm gonna get an error. Because it can only concatenate strings, it can't combine numeric values with string values. So F strings allow us to do it in a pretty simple way.

So what we would do is we would type f and then include whatever my string is within quotation marks. But if I want to include the value of z, then within this f string I add open and close curly braces. And within that I can have z like this.

And now when I run it, I get hello. And then the value of z is included in there. All right, so these s strings will come in really handy.

Okay, that's all for this video. The calculations we showed are pretty basic, but you have to start somewhere.

We'll be using these calculations with larger data sets, so it's nice to know how they work with small data sets. Every expert was once a beginner, so keep practicing, stay curious, and you'll be amazed at how quickly you progress over the next few weeks.

Google Colab - an Online Version of Jupyter Notebook

Ron Guymon: In this video, I'm going to introduce you to Google Colaboratory, which is also just known as Colab. Colab is Google's browser based IDE that is similar to

JupyterLab, one of the great things about it is that you don't have to have Python or an IDE installed on your computer.

There are other great things about Colab, I'll show you more. So the first thing that we wanna do is make sure that we have a Chrome browser window open. And what I wanna do is run the same ipynb file where I analyze disaster loan data in Google Colab.

Now I have a personal account and a work account, I want to switch over to my work account rather than logging out of my personal account. What I can do currently is click on these three dots in the top right hand corner and then hover over my profile, and a new menu comes up and I'm gonna select my other profile Ronald Work.

Okay, now I have a new Chrome browser window, and in here I can navigate to Google Drive by clicking that icon or by typing out drive.google.com if you're not familiar with Google Drive, it's like Dropbox or Box. So it's an online file storage, and software system and you have access to it if you have a Google account.

I want to first go to my Drive, this is like my home folder, in here I can see folders that I've created and other documents. What I'm going to do is click on this new button, and I'm going to create a new folder, and I will call this Video Recordings Without a Space, which makes things easier if there's not spaces in the names.

So I'll click Create, and I now have this new folder right here, so I'm gonna double click on that. So I want to upload data to my Google Drive account so that I can analyze it in Google Colab. Now that I have this folder created, I'll click New again, I'll go to File Upload and I'll find where that data is located on my computer.

So here's the Excel file, now if you want to be able to run an ipynb file that has already been created, you can do that as well, and to make things easier, I'm gonna just hold command down, and select both of those and upload them both. So I'll click Open okay, I now have both of these files in this Video Recordings folder.

Now if I didn't have an IPYNB file, I could just Go to New and go to More and select Google Collaboratory, and it would create a blank IPYNB file and it just titles it Untitled 0. Okay, and then you could start creating code just like you would in Jupyter Lab.

You can also create markdown cells, they call them text cells here. Okay, so if I just say heading with a hashtag in front of it, it gives me a preview of what it'll look like, then I can run these cells, I'll hit shift enter. I can also hit the play button to run the cells, I can delete cells by pressing delete right here.

I hope you get the idea that it works very similar to how things work in Jupyter Lab. Alright, now importantly, I want to use the file that has already been created. So rather than copy and paste things in, I can simply double click on this file that I want to use and it will open it up in Google Colab alright, so I can see I've got everything that I have in here, but I'm operating on a cloud version.

Okay, so this is a great advantage of Colab is that you don't have to have python installed on your computer. Moreover, you can get access to some really powerful processors for a fee. Let's look at a few other things about Google Colab, we've got these menus up at the top which are very similar to what we see in Jupyter Lab.

So I can run Cell, run the selection, run all I can get keyboard shortcuts okay, so a lot of that's the same. A cool feature that isn't included by default in Jupyter Lab is Gemini, so I can click on that and then get access to Google's AI tool Gemini, close out of that, and then on the left hand side I've got some additional tools.

I've got the table of contents that I can use to navigate my file, just like Jupyter Lab, I can find text in this file. I've got a variable explorer which can be really helpful, let me create a new code cell here I'll say X equals 3, just run that.

Okay, the first command always takes a little bit, but I ran it and now I can see that I've got this variable X. So this is really nice, it'll keep track of all the variables in here. I'm not gonna worry about the secrets, but I do have this files and this is really important.

So to highlight the importance of files here, let me just try running some of this these cells in here. So I'm gonna run the import Pandas as pd that worked, if I run this next code cell to read in data, it's not working, but I know I have that data in here.

If I go down here and look at the error, it says it can't find it, you could actually have it explained to you why you can't find it, and it's really helpful. The AI is integrated in here as well, I'll just show you real quickly what you need to do to read data in, you have to mount your drive.

Okay, so I'm going to click on this folder, and you can see that I've got sample data, but I don't see this SBA disaster loan data. So all you have to do is click on this mount drive icon up here, when I click that, it added this code cell and it says run this.

Okay, so I'm gonna go ahead and run this, and it's mounting it says, do you want to grant permission? And I'll have to go in and say yes okay, it mounted the drive so I can go ahead and delete this code cell. Now I should see another folder, okay, there it is, I've got this drive folder, and in here I've got my drive and here's this video recordings.

So you could try running it again, but it's still not working, so what you need to do is hover over the data that you wanna read in, click on the three dots, and then select Copy Path and then paste that path in here for the file name. Now when I paste it, you'll see not only does it have the file name, but the system of folders that you have to go through to find that data.

And now when I run this, just hit the play button here close that. Okay, now it worked, you can see I've got the first five rows right here, okay and I could go ahead and run through the rest of this if I wanted. And it's really cool that it works in the cloud like this, and I could share this with someone else.

You can find others Colab files and run those, what if you wanna export it as an HTML file? So what you would do is you'd create another code cell somewhere and I'm gonna add an exclamation point. So this is going to be a terminal command, okay. And I'm going to run Jupyter NB Convert, this is really what's happening under the hood when you do this in Jupyter Lab, it's just that there's a button for it.

But I'm gonna hit type out jupyternb convert, and remember the exclamation point, and then two hyphens and then two HTML. And then what I will do is I will copy the path of the file that I want to convert. So I'm going to copy path, and since there's a space in it, I need to use quotation marks and then paste that in there.

Okay, so there's the whole path to that Loan Analysis file. Now I will run this, and it's converting it to an HTML document. You can see that it's writing it to the same folder, so if I go into my folder, and refresh this. There it is, loan analysis HTML.

So let's go ahead and download this, and this is in my Downloads folder on my computer, if I click on that, you can see that it converted it to HTML. Okay, that's an introduction to Google Colab, there are a lot of great things about it. For instance, it's really helpful when you want to collaborate with others on code, and it makes it easy to work on the same file together.

Also, you don't need to have Python or an IDE installed on your computer. Another great thing about Google Colab is that once your files are in Google Drive, you can access them and work on them from any computer as long as it's connected to the Internet. The main disadvantage of Google Colab is that you have to mount the data to the file that you're working with.

Also, the default processing power may not be as much as on your personal computer. By the same token, you can also access more processing power. If you want to pay, you

can increase the default processing power and you can even pay for way more processing power than what you could have on a personal computer.

There's a lot more to learn about Google Colab, but that should be enough to help you get started using it.

Module 1 Conclusion

Ron Guymon: At this point, I hope that you have a more informed idea of how analytics for business works. I'm guessing that you're feeling a bit overwhelmed, especially by what you've learned about using Python in this module. That's normal and you're in good company. There's more than any one person can know.

There are currently more than 500,000 programming modules that people have made to help you use Python. So in the second module, we are going to focus on how to frame questions for data analytics in business, as well as when learning how to use Python. A big key is that it's more important to know how to use the built in tools than to memorize code.

There are two other things to keep in mind as you get started with Python. First, one of the best things that you can do is to find a meaningful project that you can try to work on with Python. Your needs will guide what you should learn, and what you learn will last long beyond this course.

If you're also excited about what you've learned, then you may have already identified a project that you can start working on. If you need help, consider a task that you repeat a lot in Excel. Ask others in this course for suggestions. The second thing to keep in mind is to be patient with yourself.

Just like waiting for the sunrise at Bryce Canyon. At first it's dark and you might feel a bit lost and cold. But slowly, almost imperceptibly, you're able to see more. You warm up. Then finally, the sun crests the horizon and the hoodoos erupt in a symphony of color.

All that struggle, all that waiting, culminates in a moment of breathtaking beauty, and you realize that the journey was essential to truly appreciate the view. Similarly, with Python and any spoken language, you'll be able to read it faster than you'll be able to write it, as you may have already experienced.

When you start writing it, you'll get hung up on the smallest details. You see the end goal, but the steps to get there are unclear. Again, you're in good company. I suspect

that everyone goes through that to some extent. All the time you spend trying to figure out why your code won't run, and getting frustrated because of just one missing comma or quotation mark will help you remember it better even though you may not realize it.

With consistent, diligent effort, you'll one day recognize how far you've come and you'll be impressed.