c) Ray RLLib

d) Keras

Which of the following models is supported by Scikit-learn?

a) Convolutional Neural Networks

b) Decision Trees

c) Recurrent Neural Networks

d) Generative Adversarial Networks

4. Python in Neural Networks

What is the purpose of convolutional layers in a Convolutional Neural Network (CNN)?

MODULE2

welcome to the complete python Mastery course in this course you're going to learn everything about python from Basics to more advanced concepts so by the end of the course you'll be able to confidently use Python for AI machine learning web development and automation if you have been looking for a comprehensive easy to follow well organized and practical course that takes you from Zero to Hero this is the right python course for you you don't need any prior knowledge of python to get started I will explain everything step by step in simple terms so you can build a solid foundation I'm m hamadani a software engineer with over 20 years of experience and I've taught Millions how to code and become professional software Engineers through my YouTube channel and online school quote withm mar.com if you're new here make sure to subscribe as I upload new videos all the time now let's jump in and get started in this course you're going to learn everything you need to get started with python just be aware that I've designed this course for beginners so if you have some programming experience check out my other python course for developers you can see the link on the top right corner of this video so python is the world's fastest growing and most popular programming language not just amongst software developers but also amongst mathematicians data analysts scientists accountants Network engineers and even kids kids in fact it's the ideal programming language to learn first but what makes python so special here are six reasons with python you can solve complex problems in less time with fewer lines of code than many other languages that's why huge companies like Google Spotify Dropbox and Facebook have embraced this beautiful and Powerful language here is an example let's say we want to extract the first three characters of the text hello work this is the code we would have to write in C this is how we would do this in JavaScript and here's how we would do it in Python see how clean and simple the language is and that's just the beginning python is a multi-purpose language and you can use it for a wide range of jobs such as data analysis Ai and machine learning writing automation scripts building web mobile and desktop applications as well as software testing or even hacking so if you want a high-paying long lasting career in any of these areas especially Ai and machine learning python is the language to put those opportunities at your fingertips in fact according to indeed.com the average salary of a python developer in the US was over $115,000 in March 2018 and here are four more reasons that make python the most desirable language python is a highle language so you don't have to worry about complex tasks such as memory management as you do in C++ it's crossplatform which means we can build and run python apps on Windows Mac and Linux it has a huge community so whenever you get stuck there is someone out there to help and it has a large ecosystem of libraries Frameworks and tools whatever you want to do it is likely that someone else has done it before because because python has been around for over 20 years there are two versions of python out there python 2 which is the Legacy version of python and is going to be supported until year 2020 and Python 3 which is python for the future in this course you're going to Learn Python 3 hi my name is msh hamadani and I'm going to be your instructor in this course I'm a software engineer with 18 years of experience and I've taught way over a million people how to codee or how to become top professional software Engineers to learn more about me and my courses head over to Cod with.com all right now let's get started all right the first thing I want you to do is open your browser and head over to python.org on this page under downloads you can download the latest version of python at the time of this video the latest version is python 3.13 chances are in the future when you're watching this video there is a new newer version of python available don't worry what I'm going to show you in this tutorial will apply to Future versions of python as well so go ahead and download the latest version now if you're on windows before you click install make sure to check this little box here that says add python to path this step is very important and it will save you a lot of headaches later so check this box and follow the installation now to verify that python is successfully installed click this magnifier and and here in this search bar type terminal now here in the terminal window type python D- version this verifies that we have successfully installed python 3.13 now if you're on Mac press command and space to bring up the spotlight search here type terminal now to verify that we have installed python correctly on Mac we should type Python 3 space-- version so as you can see I've successfully installed python 3.13 on this machine so this environment you see here is what we call python interpreter which is basically a program that executes python code we can type our python code in a file and give it to this interpreter or we can type our code directly here in this interactive shell so here we can write an expression like two + 2 in programming an expression is a piece of code that produces a value so here when we add 2 + 2 we get a value that is why we refer to this piece of code as an expression so enter we get four let's try a different kind of expression let's see if two is greater than one we get true which is an example of a Boolean value you're going to learn about these Boolean values in the next section now what if we type two is greater than five enter we get false so in programming we have true and false which are similar to yes and no in English now what if we type two is greater than but we don't add a second value here just press enter we get a syntax error in programming syntax means grammar so just like we have the concept of grammar in the languages that we speak we have the exact same concept in programming if we write a sentence that is not grammatically correct chances are some people may not understand that sentence so in this example we have this expression which is incomplete it doesn't have the right grammar or syntax that is why python interpreter is complaining by returning an error so this interactive shell is a great way to quickly experiment with a bit of python code but that's not how we build real world applications to do that we need a code editor and that's what I'm going to show you in the next lecture when it comes to typing python code you have two options you can use a code editor or an IDE which is short for integrated development environment an IDE is basically a code editor with some fancy features like autoc completion which means as you type code this feature helps you complete your code so you don't have to type every character by hand it's a per activity boosting feature it also gives you additional features like debugging which means finding and fixing bugs in your programs testing and so on for both code editors and Ides there's so many options out there the most popular code editors are vs code atom and Sublime you can use the code editor that you prefer in terms of the Ides again there are so many options out there the most popular one is pie charm in this course I'm going to use vs code or Visual Studio code because that's my favorite code editor later in the course I will show you how to install a Plugin or an extension that will convert vs code to a powerful ID so before going any further head over to code. visual studio.com and download the latest version of vs code now with vs code open on the top from the file menu go to to open and somewhere on your disk create a new folder let's call this folder hello world and then open it beautiful now click this icon on the top this opens up the Explorer panel in this panel you can see all the files and folders in your project so let's add a new file and call that app.py so all our python files should have the p my extension press enter now let's close this and type a bit of python code in this lecture we're going to use one of the built-in functions in Python called print so in Python we have a lot of buil-in functions for performing various kinds of tasks for example as a metaphor think of the remote control of your TV on this remote control you have a bunch of functions like turn on turn off change the channel change the volume and so on these are the build buin functions in your TV we have the same concept in Python and many other programming languages so one of these built-in functions that comes with python is print and we can use this to print something on the screen now whenever you want to use a function you should open and close parentheses in programming we say we're calling the print function calling a function means executing it now let's display the hello world message on the screen whenever you want to work with text you should put your text in between quotes either double quotes or single quote now I'm going to go with double quote and add hello world and then put a happy Persian cat here beautiful save the changes with command and s on Mac or control and S on Windows now to execute this code we need to go back to command prompt on Windows or terminal on Mac but the good news is that we don't have to switch programs here in vs code we have an integrated terminal so press control and back tick that is the key before number one on your keyboard that is just below the escape button so this is our integrated terminal now if you're on Windows type python if you're on Mac or Linux type Python 3 and next to that add the name of our file that is app p and here's our hello word message in the terminal beautiful now let's take this to the next level and make it a little bit more interesting let's close this terminal window by pressing control and back tick and add a second line of code so one more time print this time let's add quotes with a star in between them now let's say you want to repeat this star 10 times so here we can multiply this is by 10 save the changes open up the terminal and run our program and you can see this star is repeated 10 times so as you see the instructions in our program are executed from top to bottom in order in the next lecture I'm going to show you how to convert this vs code to a powerful IDE for building python applications in this lecture I'm going to show you how to convert vs code to a powerful IDE by using an extension called python with this extension or plug-in we get a number of features such as linting which basically means analyzing our code for potential errors we also get debugging which involves finding and fixing errors we'll look at this later in the course we also get autoc completion which basically helps us write code faster so we don't have to type every character we get code formatting which is all about making our code clean and readable just like how we format our articles newspapers books to make them clean and readable we get unit testing which involves writing a bunch of tests for our code we can run these tests in an automated fashion to make sure our code is behaving correctly and finally we get code Snippets which are reusable code blocks that we can quickly generate so we don't have to type them all by hand now don't worry about memorizing any of these as we go through the course you're going to learn about these features so back to vs code on the left side click this icon this opens the extensions panel where we can install additional extensions to enhance vs code up here in the search bar search for python all right look we have an official extension for python from Microsoft so go ahead and install this now you might see a box here saying reload if you see that make sure to click it to reload vs code now with this extension installed we have a ton of new functionality in vs code for writing python code the first one I'm going to show you in this lesson is the ability to run our code so back to app.py look with this extension installed now we have this play icon on the top for running our code so if we click it we can see the output of our program in the terminal window in this lecture I'm going to show you linting and action so let's start by writing some invalid code like this print space with no parenthesis and then hello world earlier I told you that print is a built-in function and whenever you want to use or call a function you should always use parenthesis now to be more precise this is actually valid python 2 code but because we're using Python 3 here this is invalid code from python 3's point of view so now when I save the changes you can see this red underline here let's hover our Mouse over this underline you can see this tool tip it's coming from Pilot and here's the error message missing parenthesis in call to print did you mean print with parenthesis so this is the benefit of lenting as you're writing code you can see potential problems in your code you don't have to wait to run your program to see these errors so now if we put parenthesis here and save the changes you can see that red underline is gone let's look at another error let's type two plus and then save the changes earlier we run this code in Python interpreter's interactive shell there we got a syntax or grammar error so if you hover your mouse here one more time you can see pilent is telling us that this is invalid syntax or invalid grammar it's like an incomplete sentence so this is linting an action now let me show you a couple useful shortcuts here on the top look at the view menu here we have this problems menu look at the shortcut on Mac it's shift command and M on Windows it's probably shift control M so as you're working with vs code try to memorize these shortcuts because they really help you write code faster now let's take a look at this problems panel so this problems panel lists all the issues in your code in one place so if you have an application with multiple files this is really useful because some of those files may not currently be open so this linter pilent will analyze all your files and if it finds any issues it will list them here in the problems panel now you can also put this on the right side of the screen so let's put it here so so as you write code these problems will appear here now let's fix this issue so I'm going to add three here save the changes and you can see the problem disappeared and one last thing before we finish this lecture once again on the top let's go to the view menu the first item is command pallet this is a very important feature in vs code once again look at the shortcut that is shift command and P on Mac or shift control p on Windows with this command pallet you can execute various commands in vsod if you type lint here you can see all commands related to linting as you can see all these commands are prefixed with python because these commands come with the python extension that we installed earlier so these are additional features available to us in vs code the First Command here is Select linter in this list you can see various linters available for p pent so as you're reading tutorials or talking to other people you will hear about linters such as flake 8 my pie pep 8 and so on different developers prefer different linters I personally prefer py lint that is the most popular one and that is the default linter set and the python extension of vs code if you're adventurous you can try using other linters on your own the difference between these linters is in how they find and Report errors some error messages are more meaningful or more friendly the others are more ambiguous so that's all about linting in the next lecture we'll talk about formatting code in Python Community we have a bunch of documents called python enhancement proposals or peps here on Google if you search for python peps you can see the list of all these PS under python.org sdev peps let's have a quick look here so here are the peps you can see each pep has a number and a title the one that is very popular amongst python developers is Pep 8 which is a style guide for python code a style guide is basically a document that defines a bunch of rules for formatting and styling our code if you follow these conventions the code that you right will end up being consistent with other people's code now if you have time you can go ahead and read this pep eight documentation but if not don't worry because throughout this course I'm going to explain the key things in pep 8 in this lecture I'm going to show you a tool that helps you automatically format your code according to pep 8 so back in vs code let's write some python code x equal 1 here I'm declaring a variable and setting it to one if if you're not familiar with variables don't worry in the next section you're going to learn about them so according to pep 8 this code is considered ugly because by convention we should add a space around this equal sign or the assignment operator now since you're starting out with python you probably don't know these rules so let me show you a tool that helps you automatically format your code let's revert this back to its original state now we need to go back to the command pallet remember so it's right here under View and the shortcut is shift command and P here if you search for format you can see this command format document the first time you execute this command you're going to see this message here formatter autopep8 is not installed so there are a bunch of tools for formatting python code the most popular one is Auto Pep 8 and this is the tool that this python extension we installed uses to format our our code now if you don't see this you can install autopep8 using the extensions panel so once again on the left side click this icon and search for Auto Pep 8 there it is let's install it good so let's go ahead and install this good now one more time let's open up the command pallet and execute format document see this tool automatically formats our code beautiful let's take a look at another example I'm going to declare another variable Y and set it to two and a variable with a long name like unit underline price and we set this to three now some developers have this habit of formatting their variable declarations like this so they put all these equal signs in the same column according to pep 8 this is considered ugly so once again let's format our code that is better beautiful now let me show you a trick opening up this command palet and searching for format document every time is a little bit timec consuming so I'm going to show you how to have your file automatically formatted as soon as you save the changes on the top let's go to the code menu preferences and settings here in the search box search for for format on save so we have this option editor format on Save take this now back to app.py I'm going to change the formatting of these lines make them really ugly now as soon as I save the changes you can see my code is reformatted beautiful all right now let's talk about a few different ways to run python code as I told you before one way to run python code is by opening the terminal window if you're on Windows type python if you're on Mac type Python 3 followed by the name of the file this approach is useful in situations where you don't have access to a code editor okay now with the python extension in vs code there is a simpler way to run python code we get this play button on the top when we click it we see the output in the terminal but clicking this button every time we change our code is a little bit tedious so let me show you how to associate a shortcut to this button first we close this next we bring up the command pallet the shortcut on Mac is shift command and P on Windows is shift control P here we search for open keyboard shortcuts look we have this command up here now on this window we can see all the commands in vs code and the shortcuts associated with them here in the search bar search for run python file okay so this is the command that is associated with the play button as you can see we don't currently have any key bindings or shortcuts here so double click in this column now here you can press any key combination for creating a shortcut I'm going to press controll and R okay now we press enter with this in place we can go back to app.py and press controlr and here we see the out beautiful when we talk about python we mean two separate things that are closely related Python language and a particular implementation python as a language is just a specification that defines a set of rules and grammar for writing python code a python implementation is basically a program that understands those rules and can execute python code earlier in the course we downloaded python from py python.org this is the default implementation of python called cpython It's a program written in C that's why it's called C python so here in terminal when we run python we get this C python this is the default implementation of python there are a few other implementations out there such as jython written in Java iron python written in C and piie written in a subset of python itself as new features are added to the the Python language they are first supported by cpython because that's the default implementation and then they will gradually come to the other implementations in theory if we give some python code to any of these implementations we should get the same result but in practice that's not always the case certain features may be available in one implementation but not another or they may just behave a little bit differently in a particular implementation now you might ask what is the point of this why do we have several implementations of python wouldn't C python be enough well it's for the same reason that we have multiple operating systems or multiple browsers or multiple programming languages after all these years we programmers haven't agreed on a single programming language and that's the same story with python implementations however there is one technical reason behind these implementations that you should be aware of since jyon is implemented in Java it allows you to reuse some existing Java code in a Python program so if you're a Java developer and you want to import some Java code into a Python program you should use jython instead of cpython similarly iron python is written in C so if you're a c developer and want to bring some C code into a Python program you will have to use iron python next we'll look at how exactly cpython executes python code the programming languages we use like C C Java python these are all Simple Text based languages that we humans understand computers don't understand them they only understand machine code so if we have some code written in C we should convert it to machine code and that's the job of a c compiler so a c compiler is a program that knows how to convert or compile C code into machine code however this machine code is specific to the type of CPU of a computer so if we compile a c program on a Windows machine we can't execute it on a Mac because Windows and Mac have different machine code just like how people from different countries speak different languages Java came to solve this problem Java compiler doesn't compile Java code into machine code instead it compiles it into a portable language called jav Java bite code which is not specific to a hardware platform like Windows or Mac now we still need to convert Java bite code to machine code so Java also comes with a program called Java virtual machine or jvm for doing this when we run a Java program jvm kicks in it loads our Java bite code and then at runtime it will convert each instruction to machine code with this model we can run Java bite code on any platforms that have a jvm we have jvm implementations for Windows Mac and so on so the jvm implementation on Windows knows how to convert Java bite code into machine code that a Windows machine can understand C and python have also taken the same route so they are platform independent when we run a Python program using cpython first it will compile our python code into python bite code then it will pass that bite code to python virtual machine which will in turn convert it into machine code and execute it this is how cpython works in the last lecture we talked about various python implementations I told you that if you want to reuse some Java code in a Python program you should use jython now let's see how jython makes this possible when you use jython to run a Python program instead of compiling your python code into python by code it will compile it to Java by code so we can take this Java bite code and run it using Java virtual machine and that's why you can import some Java code into a Python program when using jython because the end result is Java bite code which will eventually be executed by Java virtual machine so I've got a few questions for you cuz I want to see if you have been really paying attention to this video or not you better have so here's the first question for for each question I want you to pause the video think about the answer for a few seconds when you're ready continue watching so here's the first question what is an expression an expression is a piece of code that produces a value here's an example of an expression what do you think is the value of this expression well here we have this string we're multiplying this by three so the result will be a string of three asterisk like this here's another question what is a syntax error a syntax error is a kind of error that is due to bad syntax or bad grammar in the code and finally the last question what does a linter do a linter is a tool that checks our code for potential errors mostly in the category of syntactical Errors so if you have grammatical issues in our code the linter will tell us before running our program okay okay that's it for now if you like more quizzes and programming exercises look at the link below this video and if you have enjoyed this video I hope you have please support me by giving a thumbs up please like this video and share it with others in the next section we're going to look at the fundamentals of python hey guys I just wanted to let you know that this tutorial is actually the first two hours of my complete python Mastery course if you're finding this helpful and want to dive even deeper the the full course covers everything from beginner Basics to advanced concepts like machine learning web development and automation you'll also get Hands-On projects to build your skills step by step I put the link in the description box if you're ready to take your python knowledge to the next level now let's continue let's start this section by a discussion of variables which are one of the Core Concepts in programming we use variables to store data in computer's memory here are a few examples I'm going to Define a variable called students underline count and setting it to a th000 when we run this program python interpreter will allocate some memory and store this number thousand in that memory space then it will have this variable reference that memory location so this variable is just like a label for that memory location we can use this variable or this label anywhere in our program program to get access to that memory location and the data stored there so now if we print students count and run our program we will get the number thousand so this is the basic of variables now what kind of data can we store in computer's memory well we have several different kinds of data in this section we're going to look at the built-in primitive types in Python primitive types can be numbers booleans and strings let me show you so here we have a whole number we refer to this as an integer in programming we can also have numbers with a decimal point let's take a look so rating we set this to 4.99 this is what we call a float or a floating Point number and this terminology is not specific to python in the future when you learn a new programming language you're going to hear these terms again now let's take a look at an example of a Boolean is published we set this to true or false these are examples of Boolean values in programming so Boolean values can either be true or false and these are exactly like yes and no in English later in the course you will learn that we'll use these Boolean values to make decisions in our programs for example if the user is an admin user perhaps we want to give them extra permissions so these are the Boolean values now take into account that python is a case sensitive language which means lowercase and uppercase characters have different meanings so Boolean values should always start with a capital letter like what you see here if we type false or false these are not accepted Boolean values in Python only what you see here is a valid Boolean value so false or true and finally let's take a look at an example of a string so of course underline name we set this to a string like Python Programming so a string as I told you before is like text whenever you want to work with text in your programs you need to surround your text with quotes so these are the basics of variables so these are the variables from the last lecture now I've got a question for you there are four things that I've consistently used in this program can you spot them if you want you can pause the video think about this for a few seconds and then continue watching so here are those four things the first thing is that all my variable names are descriptive and meaningful so students count represents the number of students for a course or course name clearly explains that this variable holds the name of a course one of the issues that I see a lot amongst beginner programmers is that they use mystical names for their variables something like this CN as in short for course name when someone else reads this code they have no idea what CN stands for or they use variable names like C1 when I look at that code I wonder where is C2 and what is the difference between C1 and C2 so these variable names are very mystical that's a bad practice make sure your variable names are always descriptive and meaningful because this makes your code more maintainable now there are times that you can use short variable names like x y z if you're dealing with things like coordinates so that's an exception now the second thing that I have consistently used in this code is that I have used lowercase letters to name my variables so here we don't have course name all in capital or in title case all letter are lowercase right let's delete this the third thing that I've consistently used here is that I have used an underscore to separate multiple words and I've done this to make my variable names more readable because in Python we cannot have a space in variable names so we cannot have course name and if you put these two words together it's a little bit hard to read that's why we use an underscore and the fourth thing that I have used consistently here is that I have put a space around this equal sign again that's one of the issues I see a lot amongst beginners they write code like this this is a little bit ugly this is what we call Dirty code dirty stinky smelly you should write code that is clean and beautiful so other people can read it like a story like a newspaper article it should be formatted properly and that's why we have pep 8 in Python now the good thing is if you forget these rules when you save the changes autopep8 kicks in and it automatically reformats your code but that aside you should always give yourself the habit of writing clean code without relying too much on the tooling so these are all the best practices about naming your variables next we're going to look at strings in more detail so here we have this course variable set to Python Programming as I told you before whenever you work with text you should surround your text with quotes you can either use double quotes or single quotes that's more of a personal preference but quite often we use double quotes we also have triple quotes and we use them to format a long string for example if you have let's say a variable message that is the message we want to include in the body of an email you can use triple quotes to format it like this hi John this is msh from code with m.com blah blah blah so that's when we use triple codes now we don't need this in this lecture so delete let me show you a few useful things you can do with strings first of all we have this built-in function in Python for getting the length of strings what is a function a function is basically Bally a reusable piece of code that carries out a task as a metaphor think of the remote control of your TV on this remote control you have buttons for different functions like turn on turn off change the channel and so on these are the built-in functions in your TV in Python and many other programming languages we have the exact same concept so we have functions that are built into the language on the platform you can reuse these functions to perform various tasks so here we can use the built-in Len function to get the length of a string which means the number of characters in that string now whenever you want to use a function you should use parenthesis now we say we're calling this function which basically means we're using this function now some functions take additional data which we refer to as arguments these arguments are inputs to these functions so this Len function takes an input or an argument here we pass our course variable and this will return the number of characters in this string so let's print that and see what we get run the program we get 18 because we have 18 characters here let's look at another example if you want to get access to a specific character in this string you use the square bracket notation so here we add course square bracket brackets to get the first character you use the index zero so in Python like many other languages strings are zero index which means the index of the first character or the first element is zero so now when we print this we'll get P okay now you can also use a negative index like minus1 what does that mean well if zero represents the first character here what do you think negative 1 represents that takes us back to the end of the string so that Returns the first character from the end of the string let's draun this program you will see we'll get G there you go using a similar syntax you can slice strings let me show you so I'm going to duplicate this line and remove ne1 now let's say we want to extract the first three characters in this string so here we need two indexes the start index colon the end index so this will return a new string that contains the first three characters in this course variable that would be P Y and T so the index of these characters are zero 1 and two so that means the character at the end index is not included okay let's run the program and make sure we get the right result there you go PYT now what if we don't include the end index what do you think we're going to get it's Common Sense we start from index zero and go all the way to the end of the string so this will return a new string that is exactly the same as the original string let's take a look so we get Python programming now what if we don't include the start index but include the end index what do you think we're going to get once again it's common sense so by default python will put zero here so it will start from the beginning of the string so when I run this program we should get PYT one more time there you go and finally as the last example if we don't include the start and the end Index this will return a copy of of the original string let's look at this so we get Python Programming now you don't have to memorize any of these just remember we use the Len function to get the length of a string we use bracket notation to get access to a specific element or a specific character and we use this notation to slice a string so we have this string here python Pro programming now let's say we want to put a double quote in the middle of this string there is a problem here python interpreter sees this second string as the end of the string so the rest of the code is meaningless and invalid how do we solve this problem well there are two ways one way is to use single Cotes for our string and then we can use a double code in the middle of the string but what if for whatever reason perhaps for being consistent in our code we decided to use double quotes how can we add another double code in the middle of this string well we can prefix this with a backs slash backslash in Python strings is a special character we have a jargon for that called Escape character we use it to escape the character after let me show you what I mean so let's let's print this course and run this program what's going on here we don't have the backs slash because we use that to escape this double code and basically display it here so backs slash is an escape character and back SL double quote is an escape sequence in Python strings we have a few other Escape sequences that you should be aware of let me show you so in Python we use a hash sign to indicate a comment a comment is like additional note that we add to our program it's not executed by python interpreter okay so here are the Escape sequences you have seen back SL double quote we also have back SL single code so we can use that to add a single code here let's run the program here it is beautiful we also have double backs slash so if you want to include a backslash in your strings you should prefix it with another backslash let me show you so when we run this we get python one back slash programming and finally we have back sln which is short for new line so now if I add a back slash n here see what we get we get a new line after python so programming will end up on the second line so these are the Escape sequences in Python here we have two variables first and last let's say we want to print my full name on the console so we can Define another variable full set it to first then concatenate it with a space and one more time concatenate it with last now when we print full we get my full name on the console beautiful now this approach of using concatenation to build a string is okay but there is a better and newer approach we can use formatted strings so here we can set full to this string and prefix it with an F which can be lowercase or uppercase this formatted string doesn't have a constant value like these two strings here It's actually an expression that will be evaluated at runtime so here we want to add our first name we use curly braces to print the value of the first variable after that we add a space and then we add curly braces one more time to print the last name so at run time this expression will be evaluated what we have in between curly braces will be replaced at runtime now let's run this program one more time we get the exact same result just be aware that you can put any valid expressions in between curly braces so earlier you learned about the built-in Len function we can call Len here to get the length of this string let's run this program one more time so we get four we can also replace last with an expression like this 2 + 2 let's run this program we get four and four so when using formatted strings you can put any valid expressions in between curly braces in this lecture we're going to look at a few useful functions available to work with strings so earlier you learned about this builtin Len function this function is general purpose so it's not limited to Strings later I will show you how to use this function with other kind of objects but in Python we have quite a few functions that are specific to a strings let me show you so here if we type course dot see all these are functions available on strings now in precise terms we refer to these functions as methods this is a term in object-oriented programming that you will learn about later in the course for now now what I want you to take away is that everything in Python is an object and objects have functions we call methods that we can access using the dot notation so here course is an object we use the dot notation to access its functions or more accurately methods let's take a look at a few of these methods we have upper to convert a string to uppercase now let's print this and run the program here's what we get beautiful now note that the methods that you call here return a new string so the original string is not affected let me show you so print course run the program one more time look this is our original string right so course. upper returns a new string a new value we can store it in a variable like course underline Capital like this now to keep this demo simple and consistent I'm going to revert this back and use a print statement we also have the lower method to convert a string to lowercase we also have title which will capitalize the first letter of every word so if our string was like this when we call the title method we get Python Programming as you see here okay another useful method is strip and we use it to trim any white space at the beginning or end of a string this is particularly useful when we receive input from the user let me show you so let's imagine the user entered a couple of white spaces at the beginning of this string when we call course. strip those white spaces will be removed take a look so note that in the first three examples we have the those white spaces but in the last one it is removed so a strip removes the white space from both the beginning and end of a string we also have l strip which is short for left strip and R strip which is short for Right strip so it will remove the white space from the end of a string if you want to get the index of a character or a sequence of characters in your string you should use the find method and me show you so of course find so as an argument here we pass another string we can pass a character or a series of characters let's find the index of Pro run the program so the index of pro is nine so if we start from zero here all the way to nine this is the index of pro okay now as I told you before python is a case sensitive language so if I pass a capital P here obviously we don't have these exact characters in our string so let's see what we get we get -1 that means this string was not found in the original string another useful method is replace so we call replace with this we can replace a character or a sequence of characters with something else so let's say we want to replace all lowercase P's with J with this we get jython durog gramming whatever that means and finally if you want to check for the existence of a character or a sequence of characters in your string you can use the in Operator Let Me Show You So print we write an expression like this Pro in course so this is an expression as I told you before an expression is a piece of code that produces a value so this expression checks to see if we have Pro in course the difference between this expression and calling the fine method is that the fine method Returns the index of these characters in our string but as this expression returns a Boolean so it's a true or false let me show you so run the program we get the Boolean true and finally we have the not operator and we use that to see if our string does not contain a character or a sequence of characters so let's change this to Swift not in course when this expression is evaluated what do you think we're going to get well we don't have Swift in this string so not in will return true let's take a look there you go go so these are the useful string Methods next we'll look at numbers in Python we have three types of numbers two of these you have already seen before they are integers and floats we also have complex numbers so complex numbers in math are in the form a plus b i where I is the imaginary number number now if you're not familiar with this concept don't worry this is something that is used a lot in mathematics and electrical engineering if you want to use Python to build web applications you're never going to use complex numbers but let me quickly show you the Syntax for representing complex numbers instead of I we use J so here is an example 1 + 2 J so X now is a complex number and by the way as I told you before this is just a common or an additional note in our program when we run this program anything after this H sign will be ignored so these are the three types of numbers we have in Python for all these types of numbers we have the standard arithmetic operations that we have in math let me show you so we have addition subtraction multiplication division but we actually have two different types of divisions let me show you first let's run this program so with this division operator which is a slash we get a floating Point number if you want an integer you use double slashes let me show you so double slash run the program we get three okay we also have modulus which is the remainder of a division and finally exponent Which is less left to the power of right so 10 to the power of 3 will be a th000 these are the standard arithmetic operators now for all these operators we have a special operator called augmented assignment operator let me show you so let's imagine we have X set to 10 we want to increment X by let's say three we can write an expression like this x = x + 3 or we can use an augmented assignment operator that is a little bit shorter so we write X+ equal three these two statements are exactly the same now here I'm using addition as an example you can use any of these operators here next I'm going to show you some useful functions to work with numbers in this lecture we're going to look at a few few useful functions to work with numbers so we have this built-in function round for rounding a number so if we pass 2.9 here and print the result we will get three we have another useful built-in function called ABS which Returns the absolute value of a number so if we pass -2.9 here we'll get positive 2.9 now teic we have only a handful of built-in functions to work with numbers if you want to write a program that involves complex mathematical calculations you need to use the math module a module is like a separate file with some python code so in Python we have this math module which includes lots of mathematical functions for working with numbers but we need to import this module so we can use it on the top we type import math now math in this program is an object so we can use the dot notation to see all the functions or more accurately all the methods available in this object as an example we have math. seal for getting the ceiling of a number so if we pass 2.2 here and run this program we get three now in this math module we have lots of functions let me show you how to find the complete list here on Google search for Python 3 make sure to add the version number math module on this page you can see all the functions in the math module so in this lecture we looked at math. seal we also have math. copy sign Fabs and so on as an exercise I encourage you to play with a couple of functions in this module all right now let's take a look at another useful built-in function in Python we use the input function to get input from the user as an argument we pass a string this will be a label that will be displayed in the terminal you'll see that in a second so let's add X colon now this function returns a string so we can store it in this variable now let's imagine that y should be x + 1 save the changes now don't run this program using the code Runner extension because code Runner by default runs your program in the output window which is read only so you won't be able to enter a value so open up the terminal using control and backspace once again if you're on Windows type python if you're on Mac or Linux Linux type Python 3 and then app.py so here's our label let's enter a value like one we got an error type error what is going on here well when we receive input from the user this input always comes as a string so this expression at runtime will look like this string 1 + 1 note that the number one is different from string one because these are two different types now when python sees this expression it doesn't know what to do because two objects can be concatenated if they are of the same type so here we need to convert this string one to a number in Python we have a few built-in functions for type conversion we have int for converting a number to an integer we have float we have bull and stir or string now in this case we don't need to convert X to a string because X is already a string if you don't believe me let me show you so I'm going to comment out these few lines now let's print type of X so type is another built-in function we pass an object as an argument and it returns its type also I'm going to comment out this line because that's the bad boy we don't want to execute this save the changes back in the terminal let's run this program one more time enter one look this is what the type function returns now don't worry about the class we'll talk about classes later in the course so the type of X is a stir or string so let's delete this line to fix this problem we need to convert X to an integer and then we can print both X and Y using a format of string remember so we add an F quotes right here we add a label like X then we'll add a field so here we want to print the value of x variable after that we add some more text and finally we want to print the value of y let's run this program one more time so here in the terminal let's enter one and here's the result X is one and Y is two beautiful now all these built-in functions are self-explanatory the only tricky one is bull because in Python we have this concept of truthy and falsey values these are values that are not exactly a Boolean true or false but they can be interpreted as a Boolean true or false so here are the falsy values in Python M2 strings are considered falsy so they're interpreted as a Boolean false numbers Z is also falsy we have an object called non which represents the absence of a value we'll look at this later in the course so whenever we use these values in a boan context we get false anything else will be true let me show you a few examples so in this interactive shell in Python let's convert number zero to Bull that's falsy so we get false what about bull of one we get true if we pass a negative number we also get true if we pass a number larger than one like five we still get true so we only get false when we try to convert zero to aoia now with strings I told you that an empty string is falsey so here we'll get false anything else is true so even if I have a string that is false we'll get true because the this is not an empty string it's a string with a few characters that's why it's evaluated as true all right once again it's time for another quiz let's see if you have been really paying attention to this tutorial so here's the first question what are the built-in primitive types in Python we have strings numbers and booleans numbers can can be integers Floats or complex numbers here's the second question you have this variable fruit set to Apple what do you think we will see on the terminal when we print fruit of one well using screw brackets we can access individual characters the index of the first character is zero so this expression Returns the second character which is p what if we add a colon and negative one here well using the syntax we can slice a string our start index is one and our end index is negative -1 which refers to the first character from the end of the string now when slicing a string the character at the end index or1 is not included so with this expression we'll get all the characters starting from the second character which is p all the way until we get to e so the result of this expression is PPL L here's another question what is the result of this expression well this is what we call the modulus operator and it Returns the remainder of a division which is in this case one and finally the last question what do you think we will see when we print bull of false well earlier I told you about falsy values in Python so number zero an mty string and the non-ob these are all falsy values anything that is not falsey is considered truthy here we have a string that has five characters it doesn't matter what those characters are this is not an empty string so it's not falsey it's truy so when we convert it using the bull function we'll get the Boolean true and this brings us to the end of the section in the next section you're going to learn the fundamentals of computer programming I hope you have enjoyed this section and thank you for watching we're going to start this section by exploring comparison operators we use comparison operators to compare values here are a few examples so 10 is greater than three we get true so what we have here is a Boolean expression because when this expression is evaluated we'll get a Boolean value that is true or false here is another example 10 is greater than or equal to three once again we get true we also have less than so 10 is less than 20 we have less than or equal to here's the equality operator so 10 is equal to 10 what about this expression what do you think we're going to get we get false because this values have different types and they're stored differently in the computer's memory and finally we have the not equal operator so now with this expression we should get true beautiful we can also use these comparison operators with strings let me show you so we can check to see if bag is greater than and apple we get true because when we sort these two words bag comes after so it's considered greater now what about this one bag equals Capital bag we get false here's the reason every character you see here has a numeric representation in programming let me show you so we have this built-in function called or don't worry about memorizing this because you're probably never going to use this in the future but let me show you the numeric representation of the letter B so that is 98 in contrast capital B is represented as 66 that is the reason these two strings are not equal so these are the comparison operators in Python next we'll look at conditional statements in almost every program there are times you need to make decisions and that's when you use use an if statement here's an example let's say we have a variable called temperature we set it to 35 now if temperature is greater than 30 perhaps we want to display a message to the user so we use an if statement if after if we add a condition which is basically a Boolean expression an expression that produces a Boolean value so if temperature is greater than 30 here we have a Boolean expression if this expression evaluates to true the following statements will be executed let me show you now here's the important part that a lot of beginners miss when you use an if statement you should always terminate your statement with a colon now let's see what happens when I press enter our cursor is indented so here we have two white spaces this is very important because using these indentations python interpreter will know what statements should be executed if this condition is true here we want to print a message like it's warm we can print another message as well drink water so we can have as many statements as we want here as long as they are indented they belong to this if block now when we finish here we should remove indentation to indicate the end of this if block so here we can add a print statement with a message like Don this statement will always be executed whether this condition is true or not now note that when I save the changes this indentation you see here is going to be doubled up take a look save there you go so when we save the changes autop pep 8 reformats our code and uses four white spaces for indentation so one 2 3 4 it uses four white spaces because that's what pep 8 recommends all right now let's run this program so because temperature is greater than 30 we see the first two messages and we see the dawn message regardless so if I change the temperature to let's say 15 and run the program one more time look this Dawn message is executed whether our condition is true or not so pay great attention to these indentations that's one of the issues I see in beginner's code let's say they want both these print statements to be executed if the condition is true accidentally they remove the indentation on the fourth line and that's why their program doesn't work as they expect so be careful about this now what if you want to have multiple conditions we use an L if statement so L if that is short for L's if here we can add another condition another expression so temperature is greater than 20 one once again colon enter now by default here vs code is using two white spaces so don't worry about this as soon as you save the changes those two white spaces will be converted to four white spaces so let's print a different message it's nice save the changes now look all these lines are indented consistently you can have as many l statements as you want and optionally you can also have an else statement so if none of the previous conditions are true then what you have in the else block will be executed once again we add the colon annotation print here we can add a message like it's called save the changes in this case temperature is 15 so none of these two conditions will be true and we will see it's called let's run the program there you go in this lecture I'm going to show you a technique for writing cleaner code so let's say we're building an application for University and we want to check to see if the person who's applying for this University program is eligible or not so we start by defining a variable called age set it to 22 now if age is greater than or equal to 18 colon print eligible remove the initation else colon print not eligible let's run the program make sure it works beautiful now there is nothing wrong in this piece of code but I want to show you a cleaner way to achieve the same result instead of having a print statement here we can define a variable like message and set it to this string that is the first step so message equals this string and then we will print this message now when you have an if L statement with this structure where you're basically assigning a value to a variable you can rewrite this in a simpler way so this is how it works all we want to do over these VI lines is to assign a value to this message variable right so with start with message we set it to eligible if age is greater than or equal to 18 else we set it to not eligible this statement is almost like plain English so what we have on line seven is exactly equivalent to these four lines of code delete save the changes run the program you can see this person is eligible if I change the age to 12 12 and run the program we get not eligible so what we have here is called Turner operator in Python we have three logical operators and we use these operators to model more complex conditions so these operators are and or and not let's see a real word example of using these operators so imagine we're building an application for processing loans so we need two variables High income we can set this to true and good underlined credit we set it to true now here's the condition we want to implement if the applicant has high income and good credit score then they are eligible for the loan so if High income and good credit we add the colon and print eligible now note that here I have not compared the value of this variable with true that is one of the issues I see in a lot of beginners code this is redundant and unprofessional because High income is a Boolean so it's either true or false we don't need to compare true with true so if this condition is true and this second condition is true then we will print eligible in the terminal so save the changes and run the program obviously this person is eligible however if one of these conditions is false we will not see eligible in the terminal so let's add an lse statement here and print not eligible run the program we see not eligible so this is how the and operator works with and operator if both conditions are true the result will be true in contrast with the or operator as long as at least one of the conditions is true the result will be true so if I replace and with or here we should see eligible in the terminal let's run it one more time there you go so these are the and and or operators now let's take a look at an example of the not operator so I'm going to Define another variable student set it to True temporarily I'm going to remove this expression and simplify it we'll come back to this later so let's say if the person is eligible if they are not a student the not operator basically inverses the value of a Boolean so in this case student is true when we apply the not operator the result will be false so in this case our condition will be false and that's why this print statement will not not be executed let me show you so save run the program they're not eligible if student was false when we apply the not operator will get true so our condition will be true and we'll see eligible let's run it one more time there you go with these operators we can model even more complex conditions here's an example a person can be eligible if they have either High income or good CR credit and they should not be a student let me show you how to implement this condition so if High income or good credit we want at least one of these conditions to be true so we put these in parenthesis we want to separate these from the other condition which is not a student now the result of this should be true which means at least one of these conditions should be true after that will add and not student and finally call so with these operators you can model all kinds of real word scenarios so here's the example from the last lecture a person is eligible for a loan if they have high income and good credit and they're not a student now one thing you need to know about this Boolean operator is that they are short circuit what do I mean by that well when python interpreter wants to evaluate this expression it starts from the first argument if this is true it continues the evaluation to see if the second argument is also true so it continues the evaluation all the way to the end of this expression however as soon as one of these arguments is false the evaluation stops let me show you what I mean so if I change High income to false when python interpreter sees this expression it starts here it knows that high income is false so it doesn't matter what comes after the result of this entire expression will always be false because at least one of the arguments or one of the operant is false this is what we call short circuiting just like the short circuit concept we have in electronics so the evaluation stops as as soon as one of these arguments evaluates to false we have the same concept with the or operator so if I change these and operators to or let's see what happens with the or operator we know that at least one of the arguments should be true so the evaluation stops as soon as we find an argument that evaluates to true in this case when python interpreter evaluates this expression it sees that high income is false so so it continues the evaluation hoping that the next argument will be true here good credit is true so evaluation stops and the result of this entire expression will be true so in Python logical operators are short circuit in this lecture I'm going to show you how to chain comparison operators this is a very powerful technique for writing clean code here's an example let's say we want to implement a rule that says age should be between 18 and 65 here's how we can implement it so we Define a variable like AG set it to 22 now if age is greater than or equal to 18 and age is less than 65 then we print eligible now here's a question for you how do we write this rule in math we can write it like this well more accurately we should have an equal sign here so age should be between 18 and 65 this is how we write this rule in math now I've got some good news for you we can write the exact same expression in Python so I'm going to move this up put an if statement here line four and line three are exactly equivalent but as you can see line four is cleaner and easier to read so let's get rid of line three this is what we call chaining comparison operators all right here's a little quiz for you I want you to pause the video and think about this quiz for 10 to 20 seconds what do you think we'll see on the terminal when we run this program so pause the video figure out the answer when you're ready come back continue watching all right let's see what happens when we run this program first we get this if statement in this case we're comparing two different objects for equality and these objects have different types we have a number compared with a string so number 10 and string 10 are not equal that is why a will not be printed on the terminal so the control moves to the L If part here we have two Boolean Expressions here's the first one here's the second one and they are combined using the logical end so if both these expressions are evaluated to true then this entire expression will be true and we will see be on the terminal let's see if both these expressions are evaluated to True here's the first part bag is greater than Apple that is true because when we sort these words bag comes after Apple but look at the second part part this expression is evaluated to false because bag is not greater than cat so when we apply The Logical end between true and false the result will be false that is why this statement will not be executed so to control moves to the lse part and when we run this program the letter c will be printed on the terminal there are times that we may want to repeat a task a number of times for example let's say we send a message to a user if that message cannot be delivered perhaps we want to retry three times now for Simplicity let's imagine this print statement is equivalent to sending a message in a real world program to send a message to a user we have to write five to 10 lines of code now if you want to retry three times we don't want to repeat all that code that is ugly that's when we use a loop we use Loops to create repetition so here is how it works we start with four number in we have a built-in function called range now how many times we want to repeat this task let's say three times so we call range and pass three as an argument now similar to our if statements we need to terminate this line with a colon enter we get indentation so in this block we can write all this statements that should be repeated three times let's do a print a message like attempt save the changes run the program so we have attempt printed three times beautiful now what is this number let's take a look it's a variable of type integer so let's pass it as the second argument to the print function number run the program this is what we get 012 so here we have a for Loop this for Loop is executed three times in each iteration number will have a different value initially it will be zero in the second iteration it will be one and finally in the last iteration it will be two now here we can do something fun we can add one to this around the program and now the messages that we print are kind of more meaningful or more user friendly like attempting number one attempting number two and so on we can take this to the next level so we can pass another argument here I'm going to add an expression one more time number + one so we'll get 1 2 3 now I want to put this expression in parenthesis so let's select this put it in parenthesis and then multiply it by a DOT so here we have a string that is multiplied by a number the result will be that string repeated that number of times let's take a look so run the program see that's pretty cool isn't it now let me show you one more thing before we finish this lecture as you saw this range function generates numbers starting from zero all the way up to this number here but it doesn't include this number here we can pass another argument say start from one and finish before four with this change we don't need to add one to number every time because in the first ation this number variable will be set to one so we can simplify our code and make it cleaner let's run it one more time we get the exact same result we can also pass a third argument as a step so I'm going to change the second argument to 10 and pass two as a step look at the result these are the numbers we get 1 3 5 and so on so pretty useful you're going to use this function a lot in real world application continuing with the example from the last lecture let's imagine the scenario where after the first attempt we can successfully send the message in that case we want to jump out of this Loop we don't want to repeat this task of sending a message three times let me show you how to implement this so in this demo I'm going to simulate the scenario where we can successfully send a message so we Define a variable successful and set it to true now here after this print statement we'll have an if statement if successful colon then perhaps we can print successful now here we want to jump out of this Loop for that we use the Breck statement let's trun this program and see what happens so there you go after the first attempt you're successful and there are no more attempts so once again I want you to pay great attention to the indentation here because that's one of the common issues amongst beginners so here's our for Loop these two lines are indented with four spaces and they belong to our for Loop in every iteration these two lines will be executed now when we get to line four if this condition is true then these two lines will be executed because both these lines are indented below this if statement now let's take this program to the next level what if we attempt three times and we still cannot send an email perhaps we want to display a different message to the user we say hey we Tred three times but it didn't work so I'm going to change successful to false now at the end here we can add an L statement this is what we call a for l statement what we put under this L statement will only be executed if this Loop completes without an early termination so if we never break out of this Loop then the L statement will be executed so here we can print a message like attempted three times and failed so run the program see what we get three attempts followed by this message attempted three times and failed in contrast if we change successful to true because we terminate this Loop using this break statement what we have in the else block will not be executed take a look R the program we have one attempt successful done in programming we have this concept called nested Loops so we can put one Loop inside of another loop and with this we can get some interesting results let me show you so I'm going to start with this Loop for X in range five colon now inside of this Loop I'm going to add another loop so for y in range three colon and then in our second Loop I'm going to add a print statement here we can use formatted strings to display coordinates remember formatted strings so we have F followed by quotes now here we add parentheses for our coordinates first we want to display X and then comma followed by y let's run this program and see what happens there you go pretty cool isn't it so we get zero and zero 0o and one zero and two then we get one and zero one and one one and two and so on now let me explain how exactly python interpreter executes this code so here we have two Loops this is what we call the outer loop and this is the inner loop so the execution of our program starts here in the first iteration of this Loop X is zero now we get to this statement which is a child of this four statement because it's indented four times this statement itself is a loop so what we have inside of this Loop will be executed three times in the first iteration X is zero because we're still in the first iteration of the outer loop and Y is also zero because we are in the first iteration of the inner loop that is why we get zero and zero now we go to the second iteration of this Inner Loop in this iteration y will be one whereas X is still zero that is why we get 0 and one and similarly in the third iteration of our inner loop we'll get zero and two in the terminal now we're done with the execution of the inner loop so the control moves back to our outer loop here will be in the second iteration so X will be one and then we start here again so we have to execute this inner loop three times in the first iteration y will be zero and X is one so here we have one and zero then we'll get one and one and one and two you got the point so this is all about nested Loops so you have learned how to use for Loops to repeat one or more statements in your programs now let's dive deeper and see what this range function returns so earlier you learned about the built-in type function with this function we can get the type of an object so if I pass Five here and run this program this is what we get so the type of this number or this object is int or integer now let's look at the type of the value that we get from the range function so as an argument we pass range of a number let's run this program so this range function returns an object of type range so in Python we have primitive types like numbers strings and booleans but we also have complex types range is an example of one of those complex types throughout this course you're going to learn about a lot of other complex types now what is interesting about this range object is that it's iterable which means we can iterate over it or use it in a for Loop that is why we can write code like this so This range function returns a range object which is iterable which means we can iterate over it in each iteration X will have a different value now range objects are not the only iterable objects in Python strings are also iterable so here we can add a string like python now in each iteration X will hold one character in this string let me show you so print X and I'm going to delete these two lines here let's run this program so in each iteration we'll get one character and print it we have another complex type called list which we use to store a list of objects so we add square brackets this indicates a list now we can add a list of numbers or a list of strings like a list of names you will learn about lists later in the course so let's run this one more time as you can see we can iterate over lists in in each iteration will get one object in this list now later in the course I will show you how to create your own custom objects that are iterable for example you will learn how to write code like this for item in shopping cart print item so shopping cart is going to be a custom object that you will create it's not going to be an integer or string or Boolean it's a custom object it has a different structure and we'll make it iterable so we can use it in a for Loop and in each iteration we can get one item in the shopping cart and printed on terminal so you have learned that we use four Loops to iterate over iterable objects in Python we have another kind of loop that is a while loop and we use that to repeat something as long as a condition is true here's an example so let's define a variable number and set it to a 100 now we use while and here we add a condition as long as number is greater than zero we add a colon once again we have indentation so we can repeat one or more statements we can print this number and then we can divide it by half so number equals number use the integer division to divide it by two or we can use the augmented assignment operator to shorten this code like this now let's run this program so here's what we get initially our number is 100 we divide it by half we get 50 then 25 and so on so as you can see in this example we are not iterating over an itable like a range object or a string or a list we are evaluating a condition and repeating a task let me show you a real word example of a y Loop in this interactive shell python is waiting for an input we can type something like 2 + 2 it will evaluate it and ask for the next input we can add another expression like 10 is greater than two so these steps will continue until we press contrl D so behind the scene we have a y Loop that continues execution until we press contr D that is the condition that causes the Y Loop to terminate let me show you how to build something like this in Python so let's Define a variable command and set it to an empty string now here we need a y Loop we want this y Loop to execute as long as command does not equal to quit so command does not equal to quit colon in this Loop we want to continue continuously get input from the user so we use the built-in input function we add a label like this get the result and store it in the command variable now at this point python interactive shell will evaluate this command we're not going to do that in this lecture because that's way too complex for Simplicity we can just Echo back what the user entered so print Echo and as the second argument we as this command so this is our y Loop it will execute until we type quit now as I told you before don't run this program using the code Runner extension because by default it will run your program in the output window which is read only so open up the terminal using control and back tick and run python or Python 3 app.py so here's our Command Prompt let's type 2+ two it echoes back let's type 3 * 2 there you go if we type quit our program terminates now let's try it one more time what if we type quit in uppercase the program doesn't terminate because as you learned before lowercase and uppercase characters have different numeric representations so quit in lowercase is different from quit in uppercase now to solve this problem an amateur programmer may do something like this and command does not equal to Capital quit so while command does not equal quit in lowercase and quit in uppercase continue getting input from the user let's run this program in terminal and see what happens so one more time python app.py we type quit beautiful it works we type quit in uppercase that would work too but what if I type quit with an uppercase q and lowercase U our program doesn't terminate so this is a poor way of checking for the quit command what is a better way let me show you so we don't need this end operator here instead because command is a string we can call the lower method so whatever the user types in first will'll convert it to lowercase and then compare it with quit in lowercase with this change it doesn't matter how the user types the word quit will always terminate the program now the last thing I want to discuss in this section is the concept of infinite Loops an infinite Loop is a loop that runs forever so if I change this condition to true because true is always true this y Loop will will run forever so to jump out of this we need a break statement so after we get the input from the user we can get the command convert it to lowercase and see if it equals to quit if that's the case we want to break now with this change we no longer need to initialize command to an empty string previously we needed this because we had a while statement like this while command does not equal will quit so we had to Define this command variable and that's why we have set it to an empty string without this line when python interpreter tries to execute this code it doesn't know what command is so now that we have an infinite Loop we no longer need to Define command and set it to an empty string so in terms of functionality this program is exactly the same as the program we wrote in the last lecture just be aware of these infinite Loops because they run for ever you should always have a way to jump out of them otherwise your program will run forever and this can sometimes cause issues because if you're executing operations that consume memory at some point your program may run out of memory and crash all right time for an exercise I want you to write a program to display the even numbers between 1 to 10 so when you run this program you should see 2 4 6 and 8 and after these I want you to print this message we have four even numbers now here's a quick hint before you get started you should call the range function with one and 10 do not use the third argument which is called Step so basically I want you to iterate over all the numbers between 1 to 10 check if each number is an even number and then print it on the terminal so pause the video spend 2 minutes on this exercise when you're done come back continue watching so we start with a for Loop for number in range 1 to 10 colon we check to see if the remainder of division of this number by two equal Z so if number modulus 2 equal Z then reprint this number now let's run this program so we get 2 4 6 8 beautiful now to count the even numbers we need a separate variable so let's call that count initially we set it to zero now in this if block every time we find an even number we need to increment count so we said count plus equals 1 and finally after our for Loop we can print a formatted string we have count count even numbers let's run the program and here's the result so that brings us to the end of this section in the next section you're going to learn how to create your own functions I hope you enjoyed the section and thank you for watching so far you have learned how to use some of the built-in functions in Python such as print round and so on in this section you're going to learn how to write your own functions now you might ask but why do we even need to write our own functions well when you build a real program that program is going to consist hundreds or thousands of lines of code you shouldn't write all that code in one file like we have done so far you should break that code into a smaller more maintainable and potentially more reusable chunks you refer to these chunks as functions so let me show you how to create your own custom functions we start with the defa keyword which is short for Define next we need to give our function a name so let's call this greet all the best practices you learn about naming your variables also apply to naming your functions so make sure your function names are meaningful descriptive use lowercase letters to name your functions and an underscore to separate multiple words now after the name we need to add parentheses you will see why shortly and then we'll add a column now what is going to happen you know it we're going to get indentation which means the following statements will belong to this function so here I'm going to add two statements hi there and welcome aboard both these lines belong to this function because they're indented now we're done with this function we need to call it so we remove the indentation and we add two line breaks after this function this is what pep8 recommends to keep our code clean and maintainable now if you forget to add two line braks don't worry as soon as you save the changes autopep8 will automatically add these line braks for you let me show you so I'm going to remove these line braks and call this function great with parenthesis just like how we call the built-in functions now save the changes there you go so we get two line breaks after our function now let's run this program so we get these two messages on the terminal now here's a question for you what is the difference between the GRE and print functions the difference is that this print function takes an input whereas our grd function doesn't take any inputs so let me show you how to pass inputs like first name and last name to this function when defining a function in between parentheses we list our parameters so here we add two parameters like first underline name and last underline name now when calling this function we need to supply two values for those parameters we refer to them as arguments so m hamedani these are the arguments to the greed function that's one of the terms that a lot of developers out there don't know they don't know the difference between parameters and arguments a parameter is the input that you define for your function whereas an argument is the actual value for a given parameter okay now let's change line two and instead of saying hi there we can greet a person by their full name so we can convert this to a formatted string and pass two Fields here first name as well as last name save the changes run the program and this is what we get in terminal now this function is more useful we can reuse it and call it with different arguments so let's greet John Smith as well run the program so we get Hi msh hamadani and hi John Smith now note that by default all the paramet that you define for a function are required so here our greed function takes two parameters if I exclude one of these arguments and save the changes you can see we have this red underline so pilent is complaining and saying there is no value for argument last name also if we run the program we get this type error greet missing one required positional argument so let's put this back now let ler I will show you how to define optional parameters so this is the simplified version of this great function we created earlier now in programming we have two types of functions functions that perform a task and functions that calculate and return a value here are some examples both the print and GD functions are example of type one they're performing a task which is printing something on the terminal in contrast the round function is an example of a function that calculates and returns a value so the functions that you create fall into these two categories now let me show you how to rewrite this great function but in the second form so instead of printing this string on the terminal we simply return it let me show you so I'm I'm going to delete all this code Define a new function but call it get underline greeting we add the name parameter and simply return this formatted string High name that's all we have to do so we use the return statement to return a value from this function now we can call this function get underlined greeting pass a name like msh because it returns a value we can store that value in a separate variable like message now you might be curious which form of these gting functions is better well with this first implementation we loged to printing something in the terminal in the future if we want to write that message in a file or send it in an email we have to create another function so we cannot reuse this great function in other scenarios in contrast this second form is not tied to printing something on the terminal it simply returns a value now we get this value and we can do whatever we want with it we can print it on the terminal or we can use the built-in open function to write this message to a file so we can create a file like content. txt open it for writing this returns a file object and then we can call file. write message now and don't worry about these two lines later in the course I'm going to talk about working with files but what I want you to take away here is that we have this message variable and we can do whatever we want with it we can print it on the terminal write it to a file send it in an email and so on and one more thing before we finish this lecture so here's our GD function and as you can see we're simply printing a string now if we call GRE give it a name run the program we get this message hi msh but what if we put this inside of a call to the print function let's see what we get we get high M followed by nonan what is this nonan is the return value of the great function so in Python all functions by default return the non value non is an object that represents the absence of a value later in the course you're going to learn more about nonone what matters now is that all functions return non by default unless you specifically return a value so here if we return some string none will no longer be returned now I just want to clarify something earlier I told you that we have two types of functions in programming functions that carry out a task or functions that calculate and return a value so back to the code we previously had so even though this function returns nonone by default it is still classified as a function that carries out a task let's create another function we call it increment we want to use this function to increment a number by a given value so here we simply return number plus by now we can call this function like this and commment two and one this returns a value so we can store it in a variable like result and then print it on the terminal let's run the program we get three beautiful now we can simplify this code we have used this result variable only in a single place that is line six so we don't really need it so on line six we can replace result with a call to increment function like this so when python interpreter executes this code first it will call the increment function it will get the result and temporarily store it in a variable for us we don't see that variable and then it will pass that variable as an argument to the print function now if we run this program we get the exact same result beautiful now we can make this code more readable if someone else looks at line five they may not know exactly what these arguments are four we can use a keyword argument to make this code more readable so this one here is the value of this by parameter we can prefix it with the name of the parameter like this now we can read this code almost like plain English increment to by one so if you're calling a function with multiple arguments and it's not quite clear what these arguments are for you can make your code more read by using keyword arguments so here y equals 1 is a keyword argument earlier I told you that all the parameters that you define for a function are required by default in this lecture I'm going to show you how to make the by parameter optional so let's say we don't want to explicitly pass by equals 1 every time we want to call this incr function we want to use this function to increment a value by one so we remove the second argument now we need to give this parameter a default value so we set it to one now if we call this function and don't Supply the second argument this default value will be used otherwise the value that we specify here will be used let me show you so we run this program the result is three but if we pass the second AR argument here will increment two by five so we will get seven so you can see it's pretty easy to make a parameter optional just be aware that all these optional parameters should come after the required parameters in other words I cannot add another required parameter here let's call that another I cannot add that here if I save the changes you can see we get a red underline here so all the optional parameters should come after the required parameters now obviously in this case we don't need the second parameter so let's delete it there are times that you may want to create a function that takes a variable number of arguments here is an example let's define this function multiply that takes two parameters X and Y and simply returns x * y now we can call this function like this so far so good but what if you want to pass one or two more arguments here that doesn't work because our multiply function takes only two parameters to solve this problem we need to replace these two parameters with a single parameter we use a plural name here to indicate that this is a collection of arguments and then we prefix it with an asterisk this is the magical part let me show you what happens when you use an asterisk here so temporarily let's delete this line and simply print numbers let's see what we get here so run the program you can see all our arguments and they're packed in parenthesis what is this well earlier you learned about lists I briefly mentioned that you can use square bracket to create a list of objects like 2 3 4 5 now later in the course we have a comprehensive section about lists so don't worry about the details of lists and how they work but what I want you to note here is that the only difference between this list and what we have here is in the notation so we use square brackets to create lists and parentheses to create toles some people call it tles or tuples so a topple is similar to a list and that it's a collection of objects the difference is that we cannot modify this collection we cannot add a new object to this toppo once again later in the course we're going to have a comprehensive section about lists top holes and other data structures what matters now is that these topples just like lists are iterable so we can iterate over them which means we can use them in Loops let me show you so let's write for number in numbers colon let's just print one number at a time actually we don't need this line so delete and run the program so we iterate over this top hole and in each iteration we get one number and printed on the terminal so now with a simple change we can calculate the product of all these numbers all we have to do is to Define a variable like total initially we set it to one and then in each each iteration we get total and multiply it by the current number or we can rewrite this statement using an augmented assignment operator so total times equal number line five and four are exactly identical so I'm going to use LINE five because it's shorter and cleaner delete and finally we'll return the total now one of the issues I see often in beginnner code is that they don't use this indentation properly so they put the return statement here and then they wonder why their function doesn't work properly if you put the return statement here it will be part of the for Loop so it will be executed in each iteration in this case after the first iteration because of this return statement will return from this multiply function so the total will not be calculated properly we need to put this at the same level of indentation as other statements in this function so here we have our four statement we Loop over all the numbers we calculate the total and then finally return it so with this implementation we can get the result and printed on the terminal let's run the program and you can see the product of these numbers is 120 hey guys I just wanted to let you know that this tutorial is actually the first two hours of my complete python Mastery course if you're finding this helpful and want to dive even deeper the full course covers everything from beginner Basics to advanced concepts like machine learning web development and automation you'll also get Hands-On projects to build your skills step by step I put the link in the description box if you're ready to take your python knowledge to the next level

Here are 50 multiple-choice questions (MCQs) related to Python programming along with their answers:

1. Which of the following is used to declare a variable in Python?

a) int var = 10

b) var := 10

c) var = 10

d) declare var = 10

Answer: c) var = 10

2. What is the output of print(type(10))?

a) <class 'int'>

b) int

c) type 'int'

d) None

Answer: a) <class 'int'>

3. Which of the following is not a valid data type in Python?

a) str

b) list

c) dict

d) settext

Answer: d) settext

4. What will be the output of the following code?

20 short qsn based on python programming basic
MODULE3
Master the fundamentals of objectoriented programming in Python and transform the way you write code this comprehensive course takes you from basic concepts like classes and objects to Advanced topics like inheritance and polymorphism perfect for developers looking to level up their python skills you'll learn through practical examples and real world