Code with Mosh — Sectioned Lessons

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Your First Python Program 6:08 – 8:08

Hello World folder and go to new Python file, call this file app.py. So by convention, all Python files should have this py extension. Alright, let's go ahead, now let's collapse this project panel by clicking here and write our first python program. Simply type print, pr int, all in lowercase, open and close parenthesis, and inside these parenthesis, add quotations we can either use single quotes or double quotes, now in between the codes write your name. I am Mosh Hamedani, so here, so this is your first Python programming. With these piece of code you can print your name on the screen. Now to run this, go on the top, under the run menu, click run, also note that there is a shortcut associated with this command, when I program I always use shortcuts because that increases my productivity. So here on a Mac computer, the shortcut is control, option, and R. On Windows it's different. So let's do that, now it's asking where you want to run this, click on app, down below, you should see this little terminal window, this is a little window into our program, so here you can see our results for output for our program. In the future as you learn more Python, you'll be able to build applications that have a graphical user interface, or gooey. That is a little bit complicated, so for now we're going to use this little terminal window to see the results of our program. So as you can see my name is printed here, now as you print code, this window might get in the way so always resize it or minimize itty putting this icon here, so this was our first python program. Now if you didn't see the result I showed you in this video, use the comment box below and tell me what error you encountered, I'll do my best to help you move forward.

# How Python Code Gets Executed 8:08 – 11:20

In this Python tutorial, we're gong to take this program to the next level and make it more interesting. So I'm going to show you how to draw a dog hair. Now as part of this tutorial you will learn how Python code gets executed and you will also learn about a few programming terms. So press enter, and on line 2 write another print statement, so print, open and close parenthesis, add a quotation, now here you want to draw a dog. So, add an o, that is the head of our dog, followed by 4 hyphens. So this is the body. alright, now one more time another print statement with quotations, now we need to draw the legs. So add a space. Follow it by 4 vertical bars. So, like this, so here's a little imaginary dot. Now let's run this program and see what we get. So on the top right corner you should see this play button, click that, there you go, so we have our name and right below that we have our imaginary dog. Now what you need to understand here, is that our python code gets executed line by line from the top. So earlier I told you about Python interpreter that is the program that knows how to translate or interpret our Python code into instructions that a computer can understand. So when we run this program by clicking this program here, python interpreter starts executing or running our program line by line from the top. So first it executes line 1, then, moves onto line 2, and so on. So this is how python programs get executed. Now let me show you something cool. Let's add another print statement, with quotations now in between the quotations, add a star or an asterisk, like this. Now after the quotation and before the parenthesis, add a space, once again, add an asterisk, space, 10. What is going on here? Well, anywhere we have quotations like here or here we're defining a string, a string is a programming term which means a series of characters so here we have a string, we also have a string on line 3, as well as line 2 and line 1. Now here, we're multiplying the string by number 10. So this is the multiplication operator, just like the multiplication operator we have in math. So with this piece of code we can draw 10 asterisks on the terminal, let me show you. So let's run this program one more time, there you go. So we have 10 asterisks. Now what we have here, this piece of code here, is called an expression. An expression is a piece of code that produces a value. So when Python interpreter tries to execute line 4, first it will evaluate the code that we put in between parenthesis, so we could evaluate our expressions Our expression will produce 10 asterisks and then those asterisks will be printed on the terminal. Now as an exercise you can use these print statements to draw another shape, you can draw a heart, a ball, whatever you like.

# Variables 12:56 – 18:16

In this Python tutorial, you're going to learn about variables, which are one of the most fundamental concepts in programming, they're not specific to Python, they exist in pretty much every programming language out there. We use variables to temporarily store data in a computer's memory. Here's an example, let's type price = 10, when Python interpreter executes this code, it will allocate some memory, then it will store the number 10 in that memory, and finally it will attach this price label in that memory location. As a metahor imagine we have a box. In that box, we have number 10 and price is the label that we put on the box. Now we can use this label anywhere in our program to access the value that we have in that box. This is a very simplified explanation. So now, let's print price on the terminal. Print, now this time we're not going to add quotations, because if we put quotations here, we will see the text price on the terminal. Now the value of the price variable. So, put it in quotations, and type price, now, let's run this program one more time, there you go. So we see 10, on the terminal. So this is how we define variables, we start with an identifier which is the name of our variable, then, an equal sign and finally a value. Now more accurately, when this number 10 is about to be stored in the memory, first it will get converted to this binary for presentation. So this number 10, is in the decimal system which has all the digits from 0 to 9. Computers don't understand all these digits, they only understand 0s and 1s. So when we store the number 10 in the computer's memory first it will get converted to it's binary representation which will be a bunch of 0's and 1's, like 001, 001, whatever I don't know. Then it will get stored in the computer's memory. So, let's take this program to the next level. On the second line we can update the value of this price variable, so we can reset it to a new value like 20, now when we run our program, we should see 20, because as I told you before Python interpreter executes our code line by line from the top. So first we set the price to 10, then we reset it to 20, and finally we print it on the terminal, let's run the terminal, there you go, so, we see 20 here, okay? Now these numbers that we have here are whole numbers without a decimal point. In programming, we refer to these numbers as integers. But integer is a number without a decimal point. We can also use numbers with a decimal point for example on line 2, we can define another variable called rating and set it to number 4.9. Now in programming, we refer to this kind of number as a floating point number of float for short. So we have integers and floats. We can also define a variable and set it to a string, for example, name equals Mosh, we also have another kind of value which is called boolean, which can be true or false. They are line yes and no in English. Here is an example, I'm going to define a variable, is underline published so we use an underscore to separate multiple words in our variables name. We set this to true, or false. These are boolean values. now note that Python is a case sensitive language, which means it's sensitive to lower case and upper case letters. So when defining variables we should always use lowercase letters, but here false and true are special keywords in the language, so if we spell it with a lowercase f, Python doesn't understand it. You can see we have a red underline here, which indicates an error. Make sure to spell this with a capital F, or if you want to set this to true, make sure the T is capital, so in this program, you're storing simple values in our computer's memory. Simple values can be numbers, which can be integers or floats or they can be strings or booleans. But in Python we can also store complex values like lists and values. And that's what I'm going to show you in the future. So before going any further, I want you to do a little exercise. Imagine we're going to write a program for a hospital. So we check on a patient named John Smith. He's 20 years old and is a new patient. I want you to define 3 variables here, for his name, his age, and another variable for if this is a new or an existing patient. So pause the video and spend one minute on this exercise. When you're done, come back, continue and see my solution. Alright, so here we need three variables, the first one is the patient's name, we set that to John Smith. We can also call this full name, these are both valid names for our variables. The second variable is for the age of our patient. So age is 20, and finally we need a variable to tell if this is a new or existing patient. That's where we can use a boolean value. So, we define a variable, is new and we set it to true.

# Receiving Input 18:16 – 22:08

So, you have learned how to print messages on the terminal window. In this tutorial, I'm going to show you how to receive input from the user. So we're going to write a small program that asks the user's name and then we'll print a greeting message customized for that user. So instead of print we're going to use input. Now both these input and print are functions that are built into Python. As a metaphor think of the remote control of the TV. On this remote control we have a bunch of buttons, these are the functions built into your tv, you can turn it on, turn it off, change the volume and so on. In Python we also have functions for common tasks such as printing messages, receiving input, and so on. So we're going to use the input function, now whenever we have these parenthesis, we're going to say we're calling or executing that function, it's like pressing a button on a remote control. So we're going to call the input function and in between parenthesis we want to add a string to print something on the terminal, what is your name? With a question mark followed by a space. You will see why in a second. So this input function will print this message on the terminal, and then it will wait for the user to enter a value. Whatever the user enters this input function will return. So now we can get that value and store it in the memory using a variable. So we get the result and put it in a variable called name. Okay? Now on the second line we want to print a message like Hi John or Hi Mosh or whatever, so, print, quotations Hi with a space, now after the quotation we want to dynamically print what we have in the name variable. So we had a plus sign and then name. So here we have Hi which is a string, we're concatenating or combining the string with another string, that is what we have in the name variable. So here's another example of an expression. Remember what is an expression? It's a piece of code that uses a value. So this expression concatenates or combines 2 strings. Let's run this program and see what happens. So run okay, here is a question, what is your name? Mosh, now note that earlier we added a space after the question mark, we did this, so here in the terminal window the cursor is separated from the question mark, otherwise it would be so close. So let's type whatever here, plus enter, now we get this message, Hi, Mosh. Now here's a little exercise for you. I want you to extend this program and ask two questions. First all the person's name and then their favorite color. And then print a message like Mosh likes blue. So pause the video, do this exercise and then come back and continue watching. Alright, so here's the first question right after that, all the input functions one more time, this time we're going to ask a different question. What is your favorite color? Now, we get the new value and store it in the variable called color, or you could call it favorite underline. color. Either works. And finally we're going to change what we pass to the print function, so first we print the name then we concatenate this with a string, here we're going to type likes, we also put one space before and after likes, and once again we concatenate this. With the favorite color. So, favorite color, now let's run this program, so what is your name Mosh enter, favorite color, blue, enter, we get this message, Mosh likes blue.

# Type Conversion 22:40 – 29:28

In this Python tutorial, we're going to write a program that will ask the year that we were born in, and then it will calculate our age and print it on the terminal. So, let's start with our input function, input let's print birth here, followed by a colon, and a space. Now let's get the return value and store it in a variable, called birth\_year. So as I told you before, we use an underscore to separate multiple words Next, we need to calculate the age, so we define another variable called age, and here we do some basic math, currently we are in 2019, so let's write an expression like this. 2019 - birth year. Now finally let's print age on the terminal. Let's run our program and see what happens. So, my birth year is 1982, enter, oops, we got an error, what is going on here? So whenever you see this message, that means there is something wrong with your program. With the information here, we can find exactly where the error occurred. So, next to the file you can see the file that generated this error, in this case that is app.py. So currently our program only has a single file, but real complex programs often have hundreds or even thousands of files. So in this file, on line 2, this is where we got this error, and right below that we can see the piece of code that generated this error. So that is where we're calculating the age, and right below that you can see the type of error. In this case, we have a type error, and here's the message. Unsupported operand types for subtraction. Int and str. So int is short for integer and that represents a whole number and str is short for string, so here we're subtracting a string from an integer, and Python doesn't know what to do with it. Let me explain. So I'm going to close the terminal window. So after the first line we executed we have this birth year variable set to a string, so whatever we type in the terminal is always treated as a string, even if you type the number, in other words, when we run this program, this birth year variable will be set to a string, with four characters. 1982. This string is different from the actual number 1982. One is an integer and the other is a string. Right? So, back to line 2, where this error occurred. At run time, which means when we run our program this expression on the right side of the assignment operator is going to look like this. 2019 - string 1982. Python doesn't know how to interpret or how to evaluate this expression. To fix this problem we need to convert this 1982 into an integer and then we'll be able to subtract it from 2019 and that is easy. So far you have learned about two built in functions one is print the other is input. We have a few other functions for converting values into different types. So we have int for converting a string into an integer, we also have float for converting a string into a float, or a number with a decimal point. And we also have bool for converting a string into a boolean value. So to fix this problem, we need to go back on line 2, and pass this birth year variable to the int function like this. int parenthesis, like this so we pass this string to the int function, int will convert it into an itneger and then Python interpreter will be able to evaluate this expression. Now let's run this program one more time, so birth year is 1982 enter so I am 37 years old. In Python we have a useful function for getting the type of variables, for example, let's print the type of birth year, so right after line 1, let's print, now here we're going to call another built in function, called type, and now let's pass birth year, okay, now similarly after line 3, let's also print the type of age, so print type of age. Okay? So let's run our program, so birth year one more time, 1982, okay, here's the result so the type of birth year as you can see is a class of str or strings, we look at classes in the future so for now don't worry about them, and also below them you cans ee the type of the age variable is int or integer. So here's what you need to take away. Whenever you use the input function, you always get a string, so if you're expecting a numerical value you should always convert that string into an integer or a float. So here's a little exercise for you. I want you to write a program ask the user their weight and then convert it to kilograms and print it on a terminal. So pause the video, do the exercise and when you're ready come back to watch it. Alright so let's use our input function and ask for the weight in pounds here we get the weight in lbs or pounds now we need to convert this into kilograms, it's very easy so we defined another variable weight\_kg we set this to weight\_lbs times 0.45. And finally let's print weight underline kg. Let's run this Python program and see what happens. So my weight is 160, alright once again we got an error, can't multiply sequence by non int of type float. So as I told you before, this input function returns a string, so we cannot multiply a string by a float. Python doesn't know what to do with it. So in this case, we should convert this number by an integer or float and then multiply by 0.45. So let's call the int function. And pass weight underline lbs. And run our program one more time, 160 okay, so I am 72 kg's.

# Strings 29:28 – 37:28

In this tutorial, you're going to learn more about Python strings. So I've defined this course variable and set it to Python for beginners now earlier I told you you could use both single and double quotes to define a string, but there are times you have to use a specific form, otherwise you're going to run into issues. Here's an example. Imagine you wanted to change this string into Pythons course for Beginners. So we want to add an apostrophe, like this, Course for Beginners. You can immediately say this is going crazy, because our string starts here and then terminates here, all these characters that we have here after the second apostrophe Python interpreter doesn't know what they are. So to solve this problem we need to use double quotes to define our string so we can have a single quote in the middle of the string. So let's change this to double quotes, now you can see it adds another double quote to close it, you have to manually remove this, and also one more time at the beginning of the string, we need to add another double quote. Now you can see error is gone, so if you print course we see Python course for beginners. Beautiful. Now let's say we don't want this apostrophe here, so we have Python for Beginners, but we want to put Beginners in double quotes. Once again, if you add a double quote here Python interpreter gets confused because it assumes the second double quote indicates that end of the string, so it doesn't know what these characters are, so to solve this we need to change our double quotes to single quotes like this. And then we can add double quotes in the middle of the string. Now let's run this program, there you go. So we get Python for Beginners. So these are the cases for using single or double quotes. Now in all the examples I've shown you so far we only deal with short strings, but what if you wanted to define a string that is multiple lengths? For example, what if you wanted to define a string for the message that we send in an email. In that case we need to use triple quotes. So. We delete this. Now we add three quotes, so 1, 2, 3, there you go, So, we have three quotes to start our string and three to terminate it. Again these quotes can be single or double quotes. Okay? Now, with this we can define a string that spans multiple lines. For example, we can say Hi Jon here is our first email to you. Thank you, The Support Team. Like that. Now, let's run this program and here's the result. So, we get this beautiful multi line string. Now let's change this back to something simple so, we can look at other characteristics of strings and Python. So I'm going to use single quotes and set the course name to Python for Beginners. Now here we're going to use square brackets to get a character and a given index in this string. Let me show you. So to get the first character we use square brackets and type 0. So the index of the first character in the string is 0. In other words, this is how Python strings are indexed. 0, 1, 2, 3, 4, etc. So the index of the first character is 0, the second character is 1, and so on. So let me delete this and run this program we get p. We can also use a negative index here. And this is one of the features that we don't have in other programming languages as far as I know. So we have negative index we can get the characters started from the end. So if I pass negative 1 here, Assuming that 0 is the index of the first character negative 1 is the index of the last character. So when we run this program we should see s. Let's run it, there you go, we get s, if we pass negative 2, this will return the second character from the end. Let's run it one more time, now we get R because that is the second character from the end. Okay? So place close attention to this square brackets syntax because quite often it's the topic for online Python tests or university exams, so if you're preparing for a python test, make sure to watch this tutorial one more time and understand exactly how this square brackets syntax works, we can also use a similar syntax to extract a few characters instead of 1 character. For example, if we type 0, colon 3, Python interpreter will return all the characters starting with this index all the way to this second index, but it does not return the character at this index. In other words, back to these indexes so you have 0, 1, 2, 3, and so on. When you run this program. Python interpreter will return the characters starting from the index 0 all the way to index 3, but excludes the character and index 3, so when we run this Python program we're going to see pint (?). Let me show you, so we're going to delete this line, run this program, there you go. We get pint. Now here we also have default values for the start and end index. So if we don't supply the end index, Python will return all the characters to the end of the string. Let's take a look. So run this program, there you go, Python for Beginners. But if you change the start index to 1, this will exclude the first character so when we run this program, we see ython so p is removed. Okay? Now similarly we have a default value for the start index, so if we don't supply the start index but add an end index like 5. Python interpreter will assume 0 as the start index, so, let's run this program, there you go, we get pytho. Now what if we leave both the start and end index? Well, I told you? Now in this case 0 will be assumed as the start index, and the length of the string will assume as the end index. So with this syntax, you can basically copy or clone a string. In other words, if I define another variable here, let's call it another and set it to course square brackets with just a colon, now this expression will return all the characters in the course variable so variable will be copy of our first variable. Let's take a look, so, let's print another, and load our program there you go, we get Python for beginners. So once again the square bracket syntax is pretty important if you're preparing for online python tests, or college exams, make sure to watch this tutorial again. Now here's a little exercise for you. I'm going to delete all this code define a variable, called name, and set it to Jennifer. Now when we print name of 1: negative 1 what do you think we're going to see on the terminal? I want you to use your knowledge to tell what we're going to see on the terminal, we're now running this program. So pause the video, think about it for a few seconds, then come back and continue watching. So this expression will return the characters starting from index 1 which is the second character all the way to the first character from the end, but excluding the character at this index. In this case, the first character from the in is r, so r will be excluded, in other words we're going to see all the characters starting from e all the way to the second e. Let's take a look. So I'm going to run this program there you go. This is what we get,

# Formatted Strings 37:28 – 40:48

I hope you guessed it right. In this tutorial, we're going to look at formatted strings in Python programming language. Formatted strings are particularly useful in situations where you dynamically generate some text with your variables. Let me show you. Let's say we have two variables first name and last name. So first we set this to John, and last we set this to Smith. Now it's better to call these variables first name and last name, because they're more descriptive. But here I'm using shorter names because I want you to see the entire code on display screen. So let's say with these two variables, we want to generate some text like this. Jon, in square brackets Smith is a coder. Let's say we want to print this on the terminal. How do we do this? Well, we define another variable like message, now here we add the first name, now we need to concatenate this, with a string that contains a space and a square bracket next we need to add a last name, then we need to add a string that contains the closing square brackets followed by is a coder okay? So, then, if you print message and run this program to see John Smith is a coder, right? Now, while this approach perfectly works, it's not ideal because as our text gets more complicated it becomes harder to visualize the output. So someone else reading this code, they have to visualize all the string concatenations in their head. This is where we use formatted strings, they make it easier for us to visualize the output. So, I'm going to define another variable, let's say msg short for message, and set this to a formatted string. A formatted string is one that is prefixed with an f. So f, quotes. Now in between the quotes, first we want to add the value of the first name variable, so, we add curly braces and here we type first. Next we add a space, we add our square brackets, in between the square brackets, we want to display the last name so once again we add curly braces, and type last, and finally here we type is a coder. So this is what we call the formatted string. With these curly braces, we're defining place holders or holes in our string, and when we run our program these holes will be filled with the value of our variables. So here we have two place holders or two holes in our string. One is for the value of our first name variable and the other is for the value of the last name variable. But compare this formatted string with string concatenation. With this formatted string we can easily visualize what the output looks like, right? Now let's print this other terminal to make sure we get the same exact output. So, let's print message there you go. So Jon Smith is a coder. So to define formatted strings, prefix your strings with an F and then use curly braces to dynamically insert values into your strings.

# String Methods 40:48 – 48:32

In this Python tutorial, I'm going to show you some really cool things you can do with Python strings. So let's start by defining a variable, course and we set that to Python for beginners. Now to calculate the number of characters in this string, you can use a built in function called len. So len we give it this course variable, and then, we can print the result. Let's run this program, so as you can see we have 20 characters in this string, this is particularly useful when you receive input from the user. For example you have noticed that when you fill out a form online, each input field often has a limit. For example, you might have 50 characters for your name, so using this len function we can enforce a limit on the number of characters in an input field. If the user types in more characters than we allow, we can display an error, now, this, len function is another function built into Python, it's a general purpose function, so it's not limited to counting the number of characters in a string, in the future when we look at lists, I want to show you that we can use this function to count the number of items in a list. So it's a general purpose function. Now we also have functions specifically for strings for example we have functions for converting all these characters to upper case or lower case. To access these functions we use the dot operator. let me show you. So first we type course, then dot look these are all the functions that are specific to strings. Now in more accurate terms, you refer to these function as methods, this is a term in object oriented programming that we want to look at in the future, but for now, what I want you to take away, is that when a function belongs to something else, or is specific to some kind of object, we refer to that function as a method. For example, here we have this function, upper, for converting the string into upper case, now more accurately because this function is specific to a string, we refer to this as a method. In contrast len and print are general purpose functions, they don't belong to strings or numbers or other kinds of objects. So this is the difference between functions and methods. Now let's take look at this upper method. So, let's print the results and we run our program, there you go, you get all these characters displayed in uppercase. Now note this method does not change or modify our original string, in fact it creates a new string and returns it. So, if we print our course variable right after we call the upper method, we can see that our course variable still has it's original form, so let's run this program one more time, there you go. Look, here is our original course variable, it's not modified. Now similar to the upper method we have another method for converting a string into lower case. So let me show you. Print, course.lower. Now, let's run the program, so on the second line you can see, all characters are in lower case. Now there are times that you want to find a character or a sequence of characters in a string. In those situations you can use the find method. So let me delete these few lines. Call course. find here we pass a character, let's say p, and this will return the index of the first occurrence of that character, let me show you. So let's print the result we get 0, because the index of the first capital p in the string is 0. As another example, if we pass a lower case o here, let's see what we get, we get 4 because the index of this o here is 4. Now note that the find method is case sensitive, so it's sensitive to lower case and and upper case characters. As an example if you pass an upper case here and run this program, we get negative 1 because we don't have an upper case o anywhere in this string, okay? We can also pass a sequence of characters, for example, we can pass beginners with a capital B let's run this program, we get 11 because beginners starts with index 11, now we also have method for replacing a character or a sequence of characters and that is called replace. So let's change find to replace let's say we want to replace beginners with absolute beginners, so we add a comma to pass a second value to this function, or more accurately this method. We add a string, here I'm going to pass absolute beginners. Okay, now let's run this program, so, we get python for absolute beginners. Again, this method like the find method is case sensitive so if you pass beginners all in lowercase, this method is not going to find this exact word in our string, so it's not going to place it with absolute beginners. Let's take a look. So I'm going to run the program One more time, look, we still get python for beginners. We can also replace a single character, for example we can replace capital p with let's say capital j. Now when we run this program we get jython for beginners. So these are the find and replace methods and one last things I want to show you in this tutorial. There are times that you want to check the existence of a character or sequence of characters in your string. In those situations you use the in operator, so let's say you want to know if this string contains the word python. We can write an expression like this. String python space in space course. So we're checking to see if python is in course variable. And this is an expression that produces a boolean value, and I get true or false, so we refer to this expression as a boolean expression, now if we print this on the terminal, we should get true, and by the way I'm going to delete the second line, we don't need it anymore, so run the program we get true, but if I change this capitol p to a lower case p and run the program we get false because we don't have is exact sequence of characters in our strings. Now note that the difference between the in operator and the find method is that our find method returns the index of character or sequence of characters but the in operator produces a boolean value. Do we have this or not? So that's the difference. Now let's recap all the cool things you learned to do with strings in this tutorial. We can use the len function to count the number of characters in a string, this is the general purpose function built into python, we also have specific functions for strings which we refer to as methods, these include upper for converting a string into uppercase you also have lower and title methods, you learn about the find method which returns the index of a character or sequence of characters, we have the replace method for replacing characters and words in a string and finally you learned about the in operator. So some characters in a string.

# Arithmetic Operations 48:32 – 51:28

So, you have learned that in Python programming language you have 2 types of numbers, integers which are whole numbers like 10, they don't have a decimal point, and floating point numbers or floats. Which are numbers with a decimal point. Now in this tutorial you're going to look at the arithmetic operations supported in python language these are the same arithmetic operations that we have in math, we can add numbers, multiply them and so on. So let's look at a few examples, we can print, 10 plus 3, so this is the addition operator, we also have subtraction, we have multiplication, we have two kinds of division, here's one with a forward slash, let's run this program and see what we get. we get a floating point number. But we also have another division operator for getting an integer. So if we add another slash here and run this program we get an integer. We have another operator called modulis (?) which is a percent sign. And this returns the remainder of the division. So when we run this program we should get 1, there you go. And one last operator we have here is exponent which is the power. So, that is indicated with 2 asterisks and this will return 10 to the power of 3. So let's run this program we get 1000 so these are the arithmetic operators in python programming language. Now for all these operators that you learned we have an augmented assignment operator. That is very useful, let me show you. So let's say we have a variable called x we set it to 10, now we want to increment this by 3, we'll have to write code like this. X we set this to x plus 3. So Python interpreter will add 10 to 3, the result is 13, and then it gets stored into x again. So when we print x we should see 13, there you go. So this is how you can increment a number, right? Now augmented assignment operator is a way to write the same code but in a shorter form. This is how it works. We type x plus equals 3. What we have on line 3 is exactly like what we have on line 2. So this is what we call the augmented assignment operator we have augmented or enhanced the assignment operator. Now in this particular case we are incremented a number using the augmented assignment operator, but we can also subtract or multiply a number by a given value for example, let's delete what we have on line 2, we can type subtract equals 3. So here we subtracted 3 from x. When we run this program we should see 7, there you go.

# Operator Precedence 51:28 – 54:56

Now let me ask you a question, I'm going to clear all this code here to define x and set it to 10 plus 3 times 2. What do you think is the result of this expression? this is a very basic math question that unfortunately a lot of people fail to answer. The answer is 16. Because in math we have this concept called operator precedence which means the order of operations. So the multiplication operator has a higher precendence which means it's applied first which means 3 x 2 is executed first, the result is 6 and then its added to 10, that's why x showed up as 16 after we run this code, let's verify that. So, print x run the program, x is 16. So this is what we call operator precedence, it's just a basic math concept. It's not about python programming language. So all the other programming languages behave the same way, so here's the order, first we have the exponentiation which is the power, like 2 to the power of 3, then we have multiplication or division and finally we have addition or subtraction. This is the order of operations. Let me show you another example. Here I'm going to add the exponentiation operator, so, 2 to the power of 2. Once again, what do you think is the result of this expression? Pause the video and think about it for a few seconds. The answer is 22. Because the exponentiation operator takes precedence, so first 2 to the power of 2 is executed, the result is 4, then 4 is multiplied by 3, that is 12, and finally 12 is added to 10. So x should be 22. So let's run this program and verify this. So I'm going to delete these lines here. Run the program, there you go. X is 22. Now let me bring back these rules here. We can also use parenthesis to change the order of operations so if we have parenthesis we always takes priority. In this case we can add parenthesis around 10 + 3, so this piece of 3 will be executed first, the result is 13, then the exponentiation operator will be executed, so 2 to the power of 2 is 4, and finally 4 is multiplied by 13. Now here is a little exercise for you. I'm going to set x to parenthesis 2 + 3 x 10 minus 3. What is the result of this? Pause the video and think about it for a few seconds. So you learned that parenthesis always overrides the order, so this piece of code is executed first. The result of these 5. Then, between the multiplication and subtraction, you know that multiplication takes precedence. So next, 5 will be multiplied by 10, the result is 50 and finally we have subtraction. So 50 minus 3 will be 47. Let's verify this, print x, run the program there you go, I hope you guessed it right. So this is all bout operator precedence, it's a very important topic and I see it quite often in Python tests. So if you're preparing for a Python test

# Math Functions 54:56 – 58:16

make sure to watch his tutorial one more time. In this tutorial, we're going to look at a few useful functions for working with numbers let's start by defining a variable like x and set it to 2.9. Now to round this number we can use the built in round function, so we call the round function, give it x, and then print the result. Let's run this program so, we get 3, we have another useful built in function called abs which is short for absolute, and this is the absolute function we have in math, we give it a value and it always returns the positive representation of this value, even if the value is negative. Here's an example. Let's call the abs function and give it negative 2.9 When we run this program we're gong to see 2.9 on the terminal. So let's go ahead there you go. So absolute always returns a positive number. But technically in Python we have a handful of built in functions for performing mathematical operations, if you want to write a program that involves complex mathematical calculations, you need to import the math module. A module in Python is a separate file with some reusable code. We use these modules to organize our code into different files. As a metaphor think of a super market. When you go to a super market you see different sections for fruits and vegetables, cleaning products, junk food and so on. Each section in the super market is like a module in Python. So in Python we have this math module which contains a bunch of rustable functions for performing mathematical calculations. So, let me show you how to use this module. On the top we type import, math, all in lowercase with this we can import the math module. now math is an object like a string, so we can access it's functions or more accurately it's methods using the dot operator. So if you type math. look these are all the mathematical functions available in this module. For example you can call the seal method to get the sealing of a number. So if you pass 2.9 here and then print the result we should see 3. Let me delete all this other code here. Alright, let's run this program there you go. So we get 3. Another useful method is the floor method, so let's give that a try, floor of 2. 9. What do you think we're going to get? We get 2. Now there are so many functions built in this module and we don't really have time to go through all of them. But let me show you how we can learn about them on your own. Open up your browser and search for Python 3 math module. Make sure to add the version python 3. Because the math module in python 2 is slightly different from the math module in python 3. So python 3 math module, now here you can see the documentation of this kind of module let's go let's have a look, if you scroll down, we can see the list of all the functions and their explanation. So as an exercise I encourage you to have a quick look at this documentation. See what functions are there for you in case you need them.

# If Statements 58:16 – 1:06:24

In this tutorial, I'm going to talk to you about if statements in Python if statements are extremely important in programming and they allow us to build programs that can make decisions based on some condition. So if some conditions are true we're going to do certain things, otherwise we're going to do other things. Here's an example. Over here I've got this text file with a bunch of rules for our program. If it's a hot day, perhaps we want to tell the user it's a hot day, so make sure to drink plenty of water. Otherwise, if it's cold, so here's another condition if this condition is true we're going to tell the user it's a cold day so where warm clothes. And otherwise if it's either hot or cold, we want to tell the user it's a lovely day. So let me show you how to write a program that simulates these rules. So, back to our program here, we start by defining a boolean variable is underline hot. We set this to true. Next, we add an if statement, so if, here we need to add a condition, in this case we're going to use our boolean variable. So, is underline hot. So if this values to true. Then we're going to do certain things. In this case, we want to tell the user hey it's a hot day, drink plenty of water. So, back to our program after our condition we add a colon, now, note that when I press enter pycharm automatically indents our cursor. Now any code that we write here will be executed if this condition is true, otherwise it will be ignored. Here is an example. Let's write a print statement, here I'm going to use double quotes because I want to use an apostrophe in our string. So, it's a hot day. Now let's press enter you can see the cursor is still indented, that means we can write more code that we executed if this condition is true, in this case let's say we don't want to write any extra code, so to terminate this block we need to press shift and tab, now the cursor is at the beginning of the line so lets write a print statement with a message like enjoy your day. Now when we run this program, because this condition is true, you're doing to say this message followed by this second message, take a look, so run, there you go, it's a hot day enjoy your day. But if I go over here and change this boolean value to false and run the program again, our first message disappears and we only see the second message, enjoy your day. So this is how if statements work. Now back here we can add another print statement let's say drink plenty of water. Now because this spirant statement is also indented it will be executed if this condition is true. So I'm going to revert is hot to true and run the program one more time. There you go, so it's a hot day, drink plenty of water, and enjoy your day. Alright now let's add a second rule here, if it's hot we're going to execute these two lines, otherwise if it's not hot we want to print a different message. So here we are moving an indentation and typing els colon. Now when we press enter once again our cursor indented so the code that we write here will be executed if this condition is not true. So here we can print it's a cold day, print, where warm clothes. Now let's run our program one more time. So we get the message about a hot day followed by enjoy your day. You don't see any message about a cold day. Now if we go back here on the top. And change this boolean value to false and run our program we see different set of messages. It's a cold day wear warm clothes and enjoy your day. But there's a problem with our program. If it's not hot it doesn't necessarily mean that it's cold, it means it's a lovely day. So the absence of heat doesn't mean its cold. Back to our conditions, here on line 4 we have this rule that says if it's a cold day, then print these messages otherwise if it's neither hot or cold say it's a lovely day. So, to implement this rule we need to go back and put py and define another variable. So let' say is underline cold we set this to true. Now here we need to add a second condition. So after our first if statement, we can use an el if statement to define a second condition. So here's how it works. So, el if which is short for els if or otherwise if, now here we add another condition, so, is cold So if t's cold you want to execute these few lines. So let's cut these from here, and move them under our second condition and finally if none of these conditions are true, you want to print a different message. It's a lovely day. So, right now, is hot is false, is cold is true, so when we run this program, python interpreter is going to execute the first if statement, in this case because our condition is false, these two lines will be ignored, then Python interpreter will look at line 7, it will evaluate this condition, in this case is cold is true so we're going to see these two messages on the terminal. Now, in this case, because one of these conditions was true, this els statement will be ignored, so we are not going to see this message, and finally as before, we are always going too see this message. So let's run our program, there you go. it's a cold day, where warm clothes and and enjoy your day. Now, back to the top, if we change is cold to false it's neither hot nor cold so it's going to be a lovely day. Let's run the program, and here you go, it's a lovely day enjoy your day. So these are the basics of using if statements. As you can see they are very useful in programming and with these we can build all kinds of rules into our programs. Okay here's an exercise or you. Imagine the price of a house is 1 million dollars. Now if the buyer has good credit, they will need to put down 10 percent of the price of this property otherwise they need to put down 20 percent write a program with these rules an display the down payment card for a buyer with good credit. You will see my solution next. Alright let's define a variable for the price of this house so price we set this to 1 million so 1 with 6 0s. Next we need a variable to tell if this buyer has good credit so has good credit, and we set this to true, now we need an if statement so if has good credit has true colon, here we need to calculate a down payment so the down\_payment should be equal to 0.1 x the price. That is 10 percent of the price of this property. Otherwise colon the down payment should be 0.2 times price. Now finally we remove the indentation and print here we can use a form of valid string, first we add a label, down payment colon and right after that we add a placeholder or a hole for our down payment variable. So curly braces down payment. Let's run this program, so down payment for a buyer with good credit is 100,000 dollars. Now let's improve this by adding a dollar sign before this number. So back to our formatted string, just before the curly brace I'm going to add a dollar sign let's

# Logical Operators 1:06:24 – 1:11:20

run this one more time, that is better. In this tutorial I'm going to talk to you about the logical operators within Python. We use these operators in situations where we have multiple conditions. Here is an example. Let' say we're building an application for processing loans. If an applicant has high income and good credit, then they're eligible for a loan. So in this example we have two conditions, one is having high income and the other is having good credit. So if both these conditions are true, then the applicant is eligible for a loan. So this is where we use the logical and operator. We use this operator to combine two conditions, and by the way this is not specific to python programming language, pretty much any programming language that supports if statements also supports the logical operators. So, back to our program, let's define two variables, has high income, we set this to true. And another one has good credit, we also set this to true, now our if statement if has high income has true, and has good credit is also true, then we're going to print eligible for null. So this is where we're using the and operator. So if both these conditions are true then this message will be printed. If one of them is false, we're not going to see this message. Let's try this out. So I'm going to run this program so we see it eligible for loan, but if we change either of these conditions to false, and run the program again look, the message disappears. So this is the logical and operator. We also have the logical or, and we want to use that in situations where we want to do certain things at least one of the conditions is true, for example let's change the rule for this program, such that if the applicant has high income, or good credit, then they're eligible for a loan, so if either or both these conditions are true then the candidate is eligible. Now back to our program we can implement this rule by using the logical or operator. So we simply replace and with or, now when we run this program we're going to see this message because at least one of our conditions is true, let's take a look. So the applicant is eligible for a loan for a loan because they have good credit. If you change this to false but set the other condition to true, we still see the same result, but if both these conditions are false then we're not going to see this message anymore. So this is the difference between these operators. With the logical and operator both conditions should be true, with the logical or operator at least one condition should be true we also have another logical operator called not and that basically inverses any boolean value we give it, if we give it, we give it a true boolean value it converts it to false. For example let's make up a new room, if applicant has good credit and doesn't have a criminal record then they're eligible for a loan. Let me show you how to implement this. So, we go back to our program, in this example we don't need a first variable for let's delete that. Let's set this variable to true we also define another variable like has criminal record. We set this to false. Now, we want to check to see if this applicant has good credit and not a criminal record. This is where we use the not operator. So, if they have good credit, and not criminal record. So, in this example, has criminal record is set to false, when we use the not operator this basically gets changed to true, so we have two conditions that are true. Here's ones and here's another one. So our applicant is eligible for a loan. And when we run this program we see this familiar message. However if an applicant has a criminal record, so let's change this to true, now when we run this program we can see our applicant is not eligible because when we apply then operator on this variable, we'll get false. So true changes to false. And we'll end up with two conditions, one that's true and the other is false. And that's why this message is not printed. So this is all about the logical operators in

# Comparison Operators 1:11:20 – 1:16:16

python. In this tutorial I'm going to talk to you guys about comparison operators in Python. We use comparison operators in situations where we want to compare a variable with a value, for example, if temperature is greater than 30, then we want to print it's a hot day. Otherwise, if it's less then 10, it's a cold day, otherwise it it's neither hot nor cold. And by the way I'm taking about celsius, not farenheit. So, to build these rules into our program, we need to use comparison operators. Back to app.py, I define this temperature value, let's write an if statement, if temperature now we want to check to see if this is greater than 30, so we use the greater than operator. If this is greater than 30, we want to print it's a hot day otherwise, let's just print it's not a hot day. Now, when we run this program, we're going to see this second message because 30 is not greater than 30. So our first condition a value is to false. let's verify that. So run, it's not a hot day. Now if you change the temperature to 35 and run this again, we're going to see a different message, it's a hot day, so this is where we use comparison operators. Now what we have here as you know is an expression because it's a piece of code that produces a value. So more accurately this is a boolean expression. So this is the greater than operator, we also have greater than or equal to, we have less then, less then or equal to, here's the equality operator, so if the temperature equals to 30, then you can say it's a hot day. Note that this is different from the assignment operator that has only one equals sign. You can see that if we use only one equal sign here we immediately get this red underline because this is simply an assignment statement. We're changing the value of the temperature. you are setting the value of something else. So we don't have a boolean expression, you are not producing a boolean value. Okay? So, our equality operator has two equal signs and finally we have not equal which is an exclamation followed by an equal sign. Now here's an exercise for you. You have probably seen that when you fill out a form online, sometimes the input fields have validation messages, for example, let's say we have an input field for the user to enter their name. Now if the name is less then 3 characters wrong we want to display a validation error, like name must be at least three characters, otherwise, if the name is more then 50 characters long then we want to display a different validation error like name can be a maximum of 50 characters. Otherwise if the name is between 3 and 50 characters then we just want to tell the user that name looks good. So go ahead, and write a plan to implement these rules. Alright let's define a variable called name and set it to let's say j. So we're assuming this is what the user types into an input field. Now, we want to get the number of characters in this string. So we use the len function, right? Len of name. When we print this we get 1, right you have seen this before. Now here we want to use an if statement so if len of name is less then 3, then we want to print name must be at least 3 characters now here we need a second condition to check the upper limit. So el if len of name is greater than 50, then we want to print a different message, name, must be a maximum of 50 characters. Okay? And otherwise if else none of these conditions are true that means the name looks good. So, print, name looks good. Let's run our program. So in this case we get this message because our name is too short. Now if you go back here and type something really really long. And then we run our program we're going to see a different message name must be a maximum of 50 characters and finally if we type a proper name here like John Smith and run our program we get name looks good.

# Weight Converter Program 1:16:16 – 1:20:40

Here's another good exercise that combines many of the materials you have learned so far, so earlier you built a program to convert someone's weight from pounds to kilograms. Now we want to extend this program and allow the user to enter their weight in either kilograms or pounds and then we will convert it to the other unit. Here's how our program is going to work. So I enter my weight in pounds so 100 and 60 now it's telling me if it's in pounds or kilograms. So here I'm adding l to lbs or k for kilograms. And by the way, this program is not case sensitive so when I enter a capital l or lowercase l it takes it as pounds. Now it tells me ur set it to kilos. Let's run this program one more time, this time I'm going to enter my weight in kilo's, so send it to is the weight and the unit is kilograms so k, and it says you are 160 pounds. So go ahead and spend a few minutes on this exercise, you will see my solution next. Alright first let's ask the user their weight. So we use the input function, weight colon we get the return value and store it in the variable called weight. Now the second question, so one more time we use the input function el for pounds. Or k for kilograms. So, let's get that too and store it in a variable called unit now we need an if statement. So if unit equals l then we need to convert this weight into kilograms. However, with this implementation we are only allowing the user to enter a capitol l, if they enter a lowercase l this code is not going to work. So this is where we use the upper method of string objects so this unit is a string because as I told you before, the input function always returns a string. So, we can use the dot operator to access all it's methods or functions, here we call the upper method, this will convert whatever the user enters to upper case and then we'll convert it to a capital l. Now, if this condition is true, then we need to get the weight and multiply it by 0,.45 However, as you know this weight is a string object, and we cannot multiply a string by a floating point number, we talked about this earlier in this course. So first we need to convert this weight to a numerical value. So right here, when we call the input function, we can get the return value and pass it to the int function. So, we call the int function and give it the return value of the input function. Now, the in function will return an integer so we can store it in this weight variable. So here's the converted weight, let's store it in a variable called converted, then we print here we can use a formatted string, so we prefix this string with f ur we add curly braces to dynamically insert the value of converted variable. And finally we add kilo. Otherwise, if the unit is kilograms. We need to divide the weight by 0.45. q So, weight divided by 0.45 and just to refresh your memory, this division operator returns a floating point number but if we use double slashes we'll get an integer. In this case, we want to get a floating point number, finally let's print a formatted string, ur curly braces, converted pounds. Okay? Now let's run this program and see what happens. So weight is 160 and lbs in and that equals to 72 kilos, perfect, if we run it one more time, and enter 72 kilos we get 160 pounds.

# While Loops 1:20:40 – 1:41:44

In this tutorial I'm going to show you how to use y loops in python. We use y loops to execute a block of code multiple times and there are often useful in building interactive programs and games. In a future tutorial I'm going to show you how to build a simple game using a y loop. So, let's get started with the basics, we write a y statement and right after that, we type a condition followed by a colon as long as this condition is true the quote that we write in this block will be repeatedly executed. Here is an example. We can define a variable like i, as in short for index and set it to 1. Now we set our condition to i less than or equal to 5, so as long as I is less then or equal to 5, we can print i, on the terminal. And then we need to increment i, by 1. So we set i to i plus 1. The reason we do this is that if we don't do this I will be 1 forever so we'll end up with an infinite loop. Because this condition will always be true. One is always less then 5, so in every iteration of this loop, we increment i by 1, so at some point, i is gonna be six and then that is when this condition will be false and then we'll jump out of this loop, okay? Now to demonstrate how everything works after this loop I'm going to add a print statement say done. So note that these two lines are indented so they are part of the y block. Okay, now let's go ahead and run this program and see what happens. So, take a look, we get the numbers 1-5 followed by done. So heres how this program gets executed first we set i to 1 now python interpreter executes line 2, this condition is true because i is less then 5, so i is printed on the terminal and then incremented by 1. Then the control moves back to the beginning of the y loop. So it doesn't go to the next statement. So, we come back here and now we are in the second iteration. In the second iteration i is 2, and because 2 is less then 5, our condition is still true, so i will be printed on the terminal, and once again it will be incremented by 1, so at some point i is going to be 6, and that's when this condition will be false so our look will be terminated and this done message will be printed on the terminal. So this is the basics of y loops. Now let's make this program a little bit more interesting. Here we can write an expression like this. We add a string, and in this string we add an asterisk and then we multiply this string by i. So with this expression, we can repeat a string, when we multiply a string by a number, that string will be repeated. For example, if i is two, this expression will produce a string with two asterisks. Now let's run the program and see what we get. So we see this little triangle shape here. Because in the first iteration i is 1, so, 1 times an asterisk produces 1 asterisk. In the second iteration i is 2, so when we multiply 2 Building a Guessing Game by 1 asterisks, we'll get 2 asterisks. In this tutorial I'm going to show you how to use a y loop to build a guessing game like this. So we have this secret number which is currently set to 9. Now the computer is asking me to make a guess. So, let's say 1 is not right because the secret number is 9, okay, try again, 2, no it's not right, let's try again, so I only have three chances to make a guess. If I can't guess the number the program tells me that I failed. let's run the program one more time, this time I'm going to guess the number, it's 9, there you go, it says you in. So let's go ahead and build this program using a y loop. Alright, let's start by defining a variable to store our secret number so, we call the secret underline number and set it to 9. Now we need a while loop to repeatedly ask a user to make a guess. So while condition colon What is our condition here? Well we want to give our user a maximum of three guesses. So similar to the last tutorial, we can define a variable like I, set it to 0, and assume this represents the number of guesses the user has made. And then we write our condition as i less then 3. Note that here I'm not using less then or equal to operator, because with this condition our loop will be executed 4 times, while i is 0, one, two, and three, so here we should use the less then operator. Now if we give this code to someone else it's unclear what does i represent here, it's only in our head that i represents the number of guesses the user has made. So as a best practice, always use meaningful and descriptive names for your variables. So it's better to rename this variable to guess, count. Let me show you how to rename. So right click on i variable, and then go to refactor and rename it. Look at the shortcut. On a Mac computer it's shift and f 6. Now in this dialogue box we can easily rename your variable and pycharm will update all the references to that variable so we don't have to manually update each instance, okay? Let's change this to guess\_count enter, there you go, now that is better, also it's better to store 3 in a separate variable to make our code more readable, because it's not quite clear what does 3 represent here. So, let's define a variable called guess limit say to 3, and then we can change 3 to guess underline limit, now our code is more readable while guess count is less then guess limit, see it reads like a story this is how you should write code. Okay, so while this condition is true, we want toast the user to make a guess. So here we use our input function, guess Now whatever the user enters comes out as a string so we need to convert it to an integer. So right here, we pass the result to the end function and then get it and store it in a separate variable called guess. So at this point the user made a guess, now we need to increment guess count so guess count we set it to plus equal 1 or okay, now we need to check to see if the user will make the right guess. So here we need an if statement. If what the user guessed equals our secret number, again see our code is so readable. It's like a story you can read it like plain English. So if this condition is true we want to tell the user they won. So print you won now lets go ahead and run our program up to this point. So okay it's asking me to make a guess, I'm going to make the wrong guess so one it asked me again, 2 one more time, 3, okay, what is missing in this implementation is the message that tells me that I failed. We're going to take care of it momentarily, but let's run the program one more time and make the right guess. So, 9 okay it says you won, but it's still asking me to make a guess, because our while loop is going to get executed 3 times. Look 1 and 2. So we need to change our program such that if the user makes the right guess, we need to terminate our while loop, we need to jump out of it. How do we do that? So, over here if the user makes the right guess, after we print this message we can use the brick statement to terminate terminate a loop, when python interpreter sees this, it's going to immediately terminate our loop, it's not going to evaluate this condition again. Now let's run our program and see what happens. So, I'm going to guess the right number, you won and look, you are now asked to make two more guesses, beautiful. Now the last thing we need to add here is the message that tells the user that they failed if they could not guess the right number. How do we do that? Well in Python our while loops can optionally have an else part. similar to the if statements. So earlier you learned that our if statements can optionally happen else part. Here, so if this condition is true, do this, otherwise do something else. In this case our if statement doesn't have an else part. Now, similar to the if statements Our while loops, our while statements can also have an else part. So, right at this level we can add an else block, so else colon. And the code that we write here will get executed if this while loop completes successfully without an immediate break. In other words. If the user guesses the right number, you break this loop, you jump out of it so the code that we write in the else block will not get executed. But if the user cannot guess this number, you're never going to break out of this loop, so this loop will be executed to completion untill this condition become false. In that case, the code that we write in the else block will get executed, and this is the perfect opportunity for us to tell the user hey, you made three guesses but none of them were right. So, print, sorry you failed. Now, let's test the program one more time. So, guess 1, 2, 3, sorry you failed, let's run it one more time. This time I'm going to make a wrong guess, and then the right guess, we won and our loop terminated immediately. Building the Car Game Alright, now it's time for you to practice what you have learned so far. So once again we're going to build a game this game is a simulation card game. Now our game doesn't have a graphical user interface or gooey and it doesn't really matter for now, our focus is entirely on building the engine for this game. So let's see how this works. When we run this, we get this little symbol here, and our program is waiting for us to enter a command. If you type help either a lower case or upper case we get the list of commands that our program or our game currently supports. So we can type the start command to start our car, we can type stop command to stop our car, and quit to terminate the game. Any other commands that we type our program is going to tell us hey I don't understand that. For example, if I type asd here, it's going to say I don't understand that.If you type start, we get this message, car started, ready to go, if you type stop it says car stopped, and finally if we hit quit our program terminates, this is a fantastic exercise for you to practice wha you have learned, so pause the video and spend 5-10 minutes to build this program. Alright, we're going to start with a while loop with a condition What is our condition here? We want to run this loop until the user types quit. So we can define a variable or a story to command what the user enters, and then we can run this loop as long as the command does not equal to quit. So right before the loop, we define a variable, command, and initially we set it to an empty string. An empty string is a string that has no characters in it. We only have the quotes. So then we type out our condition as while command does not equal to quit then do something. Now immediately we have a problem here because we're assuming that the user types the command in lower case, so if they type this in upper case they our program is not going to behave properly, so to fix this problem, you need to call the lower method of the string object and then compare the results. With this quit. You could also call this upper and then type quit in upper case. It's about our personal preference in this demo I'm going to use lower case characters. So, okay, now in this loop we need to ask the user to enter a command. So once again we're going to use our input function, we're going to add a greater then symbol followed by a space, whatever the user enters, we get it and store it in our command variable. Now apart from quit command, there are three other commands that we need to support. Start, stop, and help. So here we need an if statement to compare what the user enters with one of the supported commands. So, if command.lower equals start then you want to print the message like the car started. So print the car started. Ready to go, it doesn't matter, now the second condition. What if its not start, maybe it's stop? So, el if command.lower equals stop, there you go, then we print a different message car stopped. Now look at our code. We have repeated this lower lower lower multiple times. This is bad, in programming we have a term called dry, which is short for don't repeat yourself. So whenever you have duplicate your code that means you're doing something wrong. So how can we solve this problem. Well, instead of calling the lower method in each condition, we can call it right here when we get the input from the user, so this input function as you know returns a string, we can immediately call the lower method on this string, and with this command will always be in lowercase, so we don't need to call this method in every condition. Look, we remove the duplication and also our conditions are shorter and easier to read. There is also one more place we need to modify so, it's right here. That is better. Now, the third command. We need one more el if. If the command equals help, then, we want to show the commands that we support. So, here we're going to print a multi line string. So we use triple quotes like this, and give the user a guideline like this. So start to start the car stop to stop the car, and quit to quit. Now finally we need an else part, so if what the user enters is none of these commands, we're going to tell them, hey we don't understand these, else, colon print sorry I don't understand that. And by the way note that here becomes I'm using double quotes, I can easily use a single quote as an apostrophe, okay? So let's run our program up to this point and see what happens. Alright, let's type start car is started, beautiful, stop, car is stopped, help, we get this guideline, but there's so much indentation before our commands, we'll fix that in a second. And finally let's test the quit command, oops, our program didn't work properly. Here is the reason. With these if statements, we're comparing the command with start stop, and help. Anything else will end up here, so that's why our program says it doesn't understand that. So that's why our program says it doesn't understand that command. However, after this el statement the control will be moved to the beginning of the loop. At this point our command is quit, so our loop will complete and the program terminates. In other words when we run this program and type quit, our program actually quits but we still see this message which shouldn't appear here. How can we solve this problem? Well, we can come back here and just before the else block, add another el if, something like this. el if command equals quit then you can immediately break. This will solve our problem, but note that we have kind of repeated this expression in two places. The reality is that we don't really need this condition on the top, because with these if statements well more accurately with this el if we can jump out of this loop and terminate our program. So, we can simplify our condition to something like this. True. So while true means this block of code is going to get executed repeatedly, until we explicitly break out of it, okay? Now let's test our program one more time. So, quit now our program terminates and we don't see that message beautiful. So let's fix the last problem. You saw that when we typed help, these guidelines appeared with so much in indentation, and here's the reason, look, right here in our code, they are already indented. So when we use triple quotes, what we type here will be printed exactly as is. So, because we have an indentation here, this indentation will also be printed on the terminal. So, let's delete these okay, run the program one more time, type help, the indentation is gone. Beautiful. Now here's a challenge for you. I want you to take this program to the next level. So right now if we type start we get this message car started. And if we type start again we get the same message. It would be better if we got a message like car is already started so it doesn't make sense to start a car twice. Similarly, if we type stop it says car stopped, if we type it again we get the exact same message, it doesn't make sense to stop the car twice. So here's what I need you to do if the car is stopped and the user tries to stop it again, the program should say hey, the car is already stopped, what are you doing? And similarly if the car is already started and the user tries to start it again, the program should yell at the user. So go ahead and make the necessary changes to implement this scenario. Alright to add this to our program, we need to know if the car is started or not. So there is one more piece of information we need to store in the memory. What is the kind of data we need to store here? A boolean. Is the car started or not, it's a matter of yes or no. True or false. So on the top, here we can define another variable like started and initially we set it to false. So the car is not started, right? Now when the user types the start command, here we need to check to see if the car is already started. If not the we'll start it or otherwise we'll yell at the user. So in this block we'll write another if statement, if it's already started and we print car is already started. Otherwise, so if you add an el statement here. And at this point, you set started to true. So we start the car and we print this message, okay? Now we need to make a similar change for the stop command. So if the car is already stopped we need to print a different message. If not started, so here we're using the not operator to see if the car is stopped. So if it's not started that means it's stopped, okay? So if it stopped we print car is already stopped with double p's, otherwise so else we need to stop the car, how do we do that? We set started to false. And then we'll print this message. As easy as that. Let's go ahead and run our program. So, initially our car is stopped. So I'm going to type stop, it says the car is already stopped, so lets start it, okay, now our car is started, let's start it one more time. The program is yelling at us. So we can not start the car twice. Beautiful. Now let's stop it it says the car is stopped, let's stop it one more time, we get this message

# For Loops 1:41:44 – 1:47:44

again. In this tutorial, I'm going to talk to you guys about for loops in python. In the last tutorial, you learned about while loops you learned that we use while loops to execute a block of code multiple times. In python we have another kind of loop, that is a for loop, and we use that and we use that to iterate over items of a collection, such as a string. Because a string is a sequence of characters, so it looks like a collection so we can use a for loop to iterate over each character in a string and then do something with it. Here's an example. We type out for then we define a variable, this is what we call a loop variable. In each iteration, this variable will hold one item. So, let's call it item, in here we type out a string like Python, and then colon. So with this for loop we can iterate over a string and in each iteration this item variable will hold one character at a time in the first iteration it will be set to p then in the second iteration it will be set to y, and in the third iteration it will be set to t and so on. So here we are inside our for block, because of the indentation here, so whatever we type here will be executed in each iteration, for now we can simply print this item now let's run this program and see what happens. So you can see each character in this string is printed on a new line. Let's look at another example. In Python we can define lists using square brackets, so let me remove this string from here, and define a list using square brackets, a list is simply a list of items, a list of numbers, a list of customers, a list of emails, products, blog posts, whatever. So here we can define a list of names like Mosh, Jon, Sarah, and then go ahead and run our program so we can see in each iteration we get one name and print it on a new line. We can also loop over a list of numbers, for example, 1, 2, 3, 4, let's run it, again we see each number on a new line, but what if we want a list of numbers? We don't to explicitly type out a list with let's say 100 or 1000 numbers, we don't want to type, 5, 6 7 all the way to 100. That is when we use the range function. So, let me delete this, In Python we have a built in function called range, for creating range of numbers. So, we give it a number, let's say 10, let's run this program, now we can see here on the terminal we have 0 all the way to 9. So 10 is not included. So basically when we call the range function, this range creates an object, it's not a list, it's a special kind of object we can iterate over, in each iteration this object will spit out a new number. We can also work with a range of numbers here, let's say you want to start from 5, and go all the way to 10. So, let's run our program, now we have the numbers 5, 6, 7, and 9. Also, this range function can optionally take a step, so we can pass two asa step to this function, and when we run our program, we can see our first number is 5, now we go two steps forward to get 7, once again we go to two steps forward we get 9 and that is the end of our range. So this is the basics of using for loops in python. Now here's an exercise for you. I want you to write a program to calculate the total cost of all the items in a shopping cart. So let's say we have a list of prices like 10, 20, and 30, I want you to use a for loop, to calculate the total cost of all the items in our imaginary shopping cart, so calculate that, and then print it on the terminal. That's pretty easy. And you should do it in a couple minutes. So as you learned we use for loops to iterate over all the items in a collection, a collection can be a string, it can be a list, it can be a range object that is returned from the range function. Anything, any kind of object that has multiple items. So in this example we're going to use a for loop to iterate over the list of prices. So for item in prices, colon, and by the way this loop variable we don't have to call this item, can call this anything. For example, in this case we can rename it to price. So for price, in prices. Now in each iteration this price will hold one value. In the first iteration it's going to hold 10, then it's gooing to be 20, and then it's going to be 30. So we need to define another variable to calculate the total so we define that outside of our for loop, let's call it total and initially we set it to 0. Now in each iteration we get the current price and add it to the total so we write total equals total plus price or as you learned earlier we can use the augmented assignment operator to simplify this code. So, after our for loop. This total variable has a total of all the prices we can simply print it here, or we can use a formatted string, so a string prefix with f, the other label, like total, curly braces to dynamically include some value in our string, in this case our total variable. So let's go ahead and run this program, there you go, so the total cost of all the items in our imaginary shopping cart is 60.

# Nested Loops 1:47:44 – 1:55:44

In this tutorial I'm going to talk to you guys about nested loops in python. Using a nested loop basically means adding one loop inside of another loop, and with this technique we can do some amazing things for example we can easily generate a list of coordinates. So, a acquired (?) as you know is a combination of x and y value. Let's say 0 and 0. Now let's say you wan to generate a list of coordinates like this. So we have 0 and 0, then we'll have 0 and 1, then 0 and 2, next we're going to change x, so we're going to use 1 for x, and once again we're going to use these 3 values for the y coordinates so 1 and 0 then 1 and 1 and 1 and 2, you got the point. We can easily generate these coordinates using nested loops. Let me show you. So, we start with one loop, let's say for x in range 4. With this loop, we can generate value for the x coordinate. Let's print this on the terminal, Okay, so, we get the values, 0 to 3. Now for each x, like 0, we should generate a few y values. So that is where we use a nested loop. So inside of this loop we're going to add another loop, so instead of just printing x first we want to add another loop, for y in range let's say 3, now we can print x and y together so, print, here, we use a formatted string. to display coordinates like this. So we add parenthesis inside of this parenthesis first we need to add x so curly braces x then a comma followed by another set of curly braces, and y, let's run this program and see what we get. There you go. So, we have these coordinates 0 and 0, 0 and 1, 0 and 2, then you have 1 and 0 1 and 1 1 and 2 and so on. So let me explain exactly how this program gets executed. So in the first iteration, of our outer loop, x is 0. Now we are on line 2, here we have a new loop which we call an inner loop. In this inner loop, in the first iteration y is going to be 0, so we print 0 and 0 on the terminal. Now the control goes back to line 2 or our inner loop. In this second iteration y will be set to 1, but we are still in the first iteration of our outer loop. So x is still 0, but now y is incremented to 1. So that is why we see 0 and 1 on the terminal. Once again, the control goes back to line 2, we are in the third iteration of our inner loop, so this will continue until our inner loop completes. That is when y reaches 2 because this range function generates number 0 to 3 but not including 3. So we'll have 0 1 and 2. After this inner loop completes, then the control goes back to line 1, and at this point we're going to be in the second iteration of our outer loop. So x will be 1, then the control will be moved to line 2, or our inner loop, at this point, this range function is going to generate the numbers 0 to 3 one more time. So this inner loop will be executed 3 times. And then we'll go back to our outer loop. So this is how nested loops get executed. Okay, here's an exercise for you, but this one is a little bit more challenging then the exercises you have done so far. So I really don't expect you to do it, but if you do it, wow, I will be so proud of you, so see what I've done here? Using nested loops, I've written some code to draw this f shape, can you see that? So I let me give you a hint, first of all we have this list, called numbers, in this list we have these values, 5, 2, 5, 2, 5, 2. These values determine the number of x's we have in each line. So, for example, the first item in this list, this tells us that we should have 5 x's on the first line. There you go, so 1, 2, 3, 4, 5, on the second line we're going to have 2 x's, on the third line we're going to have 5 x's like this. So I've written code to convert a simple list of numbers into a shape like this. Now here's a tip for you. Using your for loop you need to iterate over this list. In each iteration you get one number, this determines the number of x's to be displayed on that particular line. So if you want to cheat, you can get this number and multiply by a string that contains x, so if you multiply x by 5, we'll get 5 x's, that's not what I want you to do. I want you to use an inner loop here to generate a string that contains 5 x's. So imagine in Python we cannot multiply a string by a number so to solve this problem we need a nested loop. So go ahead and spend five minutes on this exercise. And by the way, do your best to solve this. It is a little bit challenging, but it's not extremely difficult. It just requires a little bit of focus. You'll see my solution next. Alright so first we need to iterate over all the items in this list. So for item in numbers, or, you could rename this variable to x\_count. That is the number of x's on each line. Okay? Now I told you that if you want to cheat you can write code like this. Print x times x underline count. If you run this program we get the same output. So this is the beauty of Python, with Python we can write expressions like this, we can multiply a string by a number to repeat it. A lot of other programming languages don't support this feature. But for this exercise, I wanted you to imagine that we don't have this feature in Python so you will have to use an inner loop to solve this problem. Here's how it works. In the first iteration, x count is going to be 5. So we need to generate 5 x's. How can we do that? Well, let's say we define a variable called output and initially set it to an empty string. Now we need to add 5 x's to this string. So, we can use another loop for count in range, of x underline count. So we're using the range function to generate a sequence of numbers from 0 up to x count. So in our first iteration x count is going to be 5, so range of 5 would generate the numbers 1, 2, 3, 4. So this inner loop will be executed 5 times. That is exactly what this count represents. So now in each iteration we simply need to append an x to our output variable. So we set output, plus, equals x, and then after this inner loop we simply print the output. With his we'll print 5 x's on the first row. Then we go to the second iteration of our outer loop. At this point x count is going to be 2, now on line 3, we're going to reset our output variable to an empty string. So we start over. Then we go to our inner loop, this loop will be executed 2 times, so we'll append to x's to the output variable and then print it, as simple as that. So see, it wasn't really that difficult, but it was slightly more difficult then the previous exercises. So let's run this program, there you go. Now if you're adventurous, I want you to modify the values that we have in our numbers list,

# Lists 1:55:44 – 2:01:44

to print and l here. Now in this tutorial we're going to take a closer look at lists. So, I'm going to define a list of names, let's send them to John Bob Mosh Sarah and Mary. So, if you print this list here, what we see on the terminal look exactly like how we define our lists we have square brackets, and in between these square brackets we have our item our items, so we have 5 strings in this list. So we can also access an individual element using an index just like how we can access an individual character in a string using an index. So, here we type out square brackets, and specify an index. The index of the first item in this list is 0. So, let's run this program, there you go, we get Jon now if you want to print out the third element in this list it's index is 2, so names of 2, returns Mosh, now we can also pass a negative index here, so negative 1, refers to the last item in this list, that is Mary, let's run the program, there you go, we see Mary, if you pass negative 2, this returns the second item from the end of the list. So let's run the program, there you go, we get sarah, so this is exactly like accessing individual characters in a string. We can also use a colon to select a range of items, for example if you pass 2 colon. This will get all items starting from the index of 2, that is Mosh here all the way to the end of the string. So let's run this program there you go. We get this list with 3 items, Mosh, Sarah, and Mary. We can also specify an end index, let's say 4, so this will return all the items up to this index, but it doesn't include the item at this index. So when we run this program we only see Mosh and Sarah, the item I've indexed 4which is the 5th element or 5th item in this list is not returned. So Mary is not returned. And also here we have default values, so if you leave out the end index this expression is going to return all the items starting from the index of 2, to the end of the list or if you leave out the start index, this expression asumes 0 as the default index, so it will return all the items from the beginning to the end of the list. And by the way, just like strings, these square brackets here, don't modify our original list, they simply return a new list. For example, if we pass 2 here, you can see this returns a new list with 3 items. So if you go back here and print our original list of names right after you can see it's not affected. So here we want to use square brackets with a colon to select a range of items, we get a new list, and by the way we can also modify any of the elements in this list. For example let's say we made a mistake and the first item shouldn't be john with an h, so we want to remove the h, that is very easy, so we access it, using an index that is names of 0, and we set it to a new value like this. Now, let's print our list so, you can see the first item in this list is now data. So this is the basics of lists. And here's a exercise for you. I want you to write a program to find the largest number in your list. This is a fantastic exercise for beginners, so go ahead and spend a few minutes on this, then come back and continue watching. Alright, let's define a list of numbers, numbers, with a bunch of random numbers 3, 6, 2, 8, 4 and 10. Now to find the largest number in this list. We need to define another variable let's call it max, this variable will hold the largest number, now initially you want to assume the first item in this list is the largest number. So we set max to numbers of 0. We're only assuming that the first item is the largest number. Chances are our assumption is wrong. So we need to iterate over this list, we need to loop through it, get each item and compare it with max. If it's greater than Max, then we need to reset max to that number. So in the first iteration we get 3, and max is also 3. Is 3 greater than 3. No, so we move on, we get the second number, is 6, greater than 3? It is, so we need to reset max to 6. Once again, we continue, we get 2, these two greater than 6, no it's not, so we move on, then we get 8, is 8 greater than 6? It is, so we should reset max to 8. That is pretty easy. So here we need a for loop, for number in numbers colon now we need too heck to see if this number is greater than max, so, if number is greater than max, max colon then we need to reset max, so max to this new number, that's all we had to do. So, let's print max, and run our program we can see the largest number in this list is 10. It doesn't matter whether this number is at the end of the list or the beginning. So if I move 10 and put it right at the beginning we should still see the same result. Let's run our program, we still see 10, if I put this somewhere in the middle, our program should still work. Let's put it right after 2. Run the program, we still get 10.

# 2D Lists 2:01:44 – 2:05:04

In this tutorial I'm going to talk to you guys about two dimensional lists in python. Two dimensional lists are extremely powerfully and they have a lot if applications in data science and machine learning. Here's an example, In math we have a concept called matrix, which is like a rectangular array of numbers, let me show you. So we have 1, 2, 3, 4, 5, 6, 7, 8, 9. So we have a rectangular array of numbers. You have rows, and columns. So this is a 3 x 3 matrix in math. Now we can model this in python using a 2 dimensional list. A 2 dimensional list is a list where each item in that list is another list. So, you want to define a matrix. We set it to a list, each item in this list, is going to be another list, and that list represents the items in each row. So, the first item in our list is going to be another list, and in this other list we're going to have the values 1, 2, and 3. Now the second item in our matrix list, once again, we have a list, this list represents the items in the second row. So, 4, 5, and 6, and finally 7, 8 9. So as you can see we have a 2 dimensional list Each item in our outer list is another list. Okay? So, this is how we can implement a matrix in python, now to access an individual item in our matrix, once again we use square brackets, let me delete this stuff, alright, so how do we access 1 here, well, you start with our list then we add square brackets first we need to go and get the first item in this list. Right? So we pass 0, now this expression returns another list. That is the inner list. In this list, let's say you want to access the second item, so, once again we add square brackets, and we pass 1, that is the index of 2 in this list, alright. So if we print this on the terminal we get 2. Okay? So using 2 square brackets we can access individual items in our matrix, and also we can modify these values using this syntax, so, before printing this, let's change this to 20, so matrix of 0, and 1, let's change it to 20, and then print it, there you go, it's modified, now here you can also use nested loops to iterate over all the items in this matrix. Let me show you. So, we start with a rows for the rows in matrix, so with this loop, we are iterating over our Matrix list, in each iteration row will contain 1 list, 1 item, okay? Now, here we need to use an inner loop. So we need to loop over this row which is a list of items. We can type out 4 item in row colon and print item. Okay? So, let's run this program,

# List Methods 2:05:52 – 2:13:20

In this tutorial, I'm going to talk to you guys about the list methods, or list functions. These are the operations that we can perform in a list. So let's define a list of numbers, here we pass a bunch of random numbers, like 5, 2, 1, 7, and 4. Now there are a number of things we can do with this list, we can add new items to it, we can remove existing items, we can check for the existence of an item these are the operations that we can perform on a list. So, the we type numbers, we can see all these functions or more accurately the methods that are available in our list objects. So we can call the append method to add a new item to this list. Let's say 13. Actually, no it's not a good number. I'm joking I'm not superstitious, so let's add 20, it doesn't really matter and then print our list when we run the program we can see 20 is added at the end of this list but what if you want to add a number somewhere in the middle, or at the beginning of our list? For that we use a different method, that is called insert. So, insert, now this method takes 2 values, let me show you. So when we open parenthesis look at this little tool tip above the insert method. You see the first value that we need to pass here is an index, so this is the index at which we want to insert this new item. Let's say we want to add an item at the beginning of our list, so we passed our index position of 0, and then the second value is the actual object we want to add to this list. Let's say we want to add the number 10, now when we print this list you can see that the number 10 is placed at the beginning of the list and all the other items are pushed to the right. We can also remove an item so we call remove and pass the item that we want to remove 5. Now we print our list so we see 5 is gone, and we have 2, 1, 7, 4. If you want to remove all the items in the list, you can call the clear method, so, clear this method doesn't take any values, so, we simply call it and it empties our list, all the items are removed. We also have another useful method called hop, and with this we can remove the last item in a list. Let me show you. So, we run our program, you can see the number 4 is removed from the end of our list. Now if you want to check for the existence of an item in our list, you can call the index method. So, we call index and pass a value here, like 5 and this returns the index of the first occurrence of this item. So let's print this on the terminal. We don't need this line anymore. So the index of 5 is 0. What if we pass a number that doesn't exist in this list? Let's say 50? Run the program, we get an error. We get a value error. 50 is not in the list. There is also another way to check for the existence of an item, we can use the in operator. So, let me show you we type our 15 in numbers, earlier we used the in operator with a string, we check for the existence of a character or a sequence of a character in a string, now here we're checking for the existence of 50 in the list of numbers, so let's print this, we get a boolean value, false, so unlike the index method, this expression, doesn't generate an error, so it's safer to use this. We also have another method for counting the occurances of an item, let's say we have another 5 over here. Now we can call numbers.count and pass 5 and this should return 2 because we have 2 5s in this list. Take a look. There you go. That is pretty useful now if you want to sort your lists you can call sort method. So, we call the sort method here, this method doesn't take any values so, look at he return value, that is none, none is an object in python that represents the absence of a value. So this sort method doesn't really return any values it simply sorts this list, in place, so instead of printing the return value oft his method, we simply call it to sort our list and then print our list. Take a look. Now, all the items are assorted in ascending order. We can also sort the items in descending order, so after we sort the list we can call the reverse method. We can simply reverse our list. Now let's go ahead and run our program, take a look. Our numbers are sorted in descending order. And one last method I want to show you here that is pretty useful is the copy method. So copy, with this method you can get a copy of our list. So let's define another variable called numbers 2, now numbers 2, is a copy of our original list. So if you make any changes to our original list, if you add new items to it, if you remove existing items, these operations are not going to impact our second list. Let me show you. So after we show you a copy of our numbers list let's add a new item to this list, so numbers.apphend ten. So the first list is updated, so now we have a new item in our first list, then lets print the second list. Take a look, we don't have the number 10 here, because these are 2 independent lists. So these are all the operations that we can perform on lists. We can add new items to a list, we can remove exxisitn items, we can check for the existence of an item, we can sort our list, and copy them. Now here's an exercise for this tutorial, I want you to write a program, to remove the duplicates on our list. Again, this is a fantastic exercise. So spend a few minutes on this and then come back and continue watching. Alright let's say we have a list of numbers with a bunch of duplicates. So 2, 2, 4, 6, 6, 3, 4, 6, 1. We want to remove the duplicates. So we need to define another list let's call that new list. Initially we set it to an empty list. Then we need to iterate over our first list, get each item, and if we don't have that number in this unix list, then we'll add it to this second list. As simple as that. So, for number in numbers, now we need to check to see if we have this number in the second list. So we use the in operator. If number not in units so if we don't have an operator in this units list, then we'll need to add it, so units. apphend number, that's all we have to do. So, let's go ahead and print the units list there you go. So you have 2, 4,

# Tuples 2:13:20 – 2:15:28

6, 3,and 1. The duplicates are removed. In this tutorial I'm going to talk to you guys about another important structure. in Python called tuple. Tuples are similar to lists so we can use them to store a list of items. But unlike lists we can not modify them, we cannot add new items, we cannot remove existing items, we tuples are immutable. We cannot mutate or change them. So let me show you. So I'm going to start by defining a list of numbers, 1, 2, 3, 4, so we use square brackets to define lists and parenthesis to define tuples, so if we change this to parenthesis, 1, 2, 3. Now we have a tuple. So if we type numbers. look here we don't have the append or insert methods, so we cannot add new items to this tuple. We also don't have remove clear and pop, we cannot remove any of these items here. We only have two methods, count, and index. We use count to count the number of occurrences in an item, and index, to find the index of the first occurrence, of an item. So we can only get information about a tuple, we can't change it. And by the way, these other methods that you see here, they start with two underscores, we refer to these as magic methods, they're more of an advanced topic, so they go beyond the scope of this tutorial. If you're interested to learn bout them. You can get my python course, I've covered them in detail. So similar to lists you can address individual items using squre bracketts,s o we can get the first item like this and then print another terminal, there you go, the first item is 1, but if you try to change the first item we'll get an error, so, numbers of 0, we set it to 10 and run our program there you go. We get this type error, because the tuple object does not support item assignment. So we cannot mutate or change tuples, they are immutable. Now practically speaking, most of the time you'll be using lists, but tuples are also useful. If you want to create a list of items and make sure

# Unpacking 2:15:28 – 2:18:16

no where in your program you accidentally modify that list, then it's better to use a tuple, In this tutorial I'm going to show you a powerful feature we have in python called unpacking. So lets find a tuple called coordinates and here we pass 3 values, 1, 2, 3. So you imagine these are the coordinates for x y and z. Now let's say we want to get these values and use them in a few expressions, a few complex expressions in our program. Maybe we want to include them as part of a large complex formula. So together we'll have to write code like this, coordinates of 0, then let's say we want to multiply this by coordinates of 1, and then multiply it by coordinates of 2, our code is getting a little bit too long, this is just a very simple example. But let's say we want to use these values in quite a few places in our program, a better approach is to get these values and store them in separate variables like we can get coordinates of 0, and store it in x, then you can get coordinates of 1 and store it in y. And similarly we get coordinates of 2, and then store it in z. Now we started repeating coordinates of 2 or coordinates of 0 multiple times, we can simply work with these variables, x times y times z, that is better. Right? So nothing new so far. But in Python we have a powerful feature called unpacking and with that we can achieve the same result with far less code. So we can define our variables x, y, and z. And set them to our tuple. What we have on line 6 is exactly identical to what we have on lines 2-4. So this is a shorthand to achieve the same result. So let me delete this and explain how this code works. When Python interpreter sees this statement, it will get the first item in this tuple and assign it to the variable. Then it will get the second item in this tuple, then similarly we get the third item in this tuple and assign it to the third variable we have here, so we are unpacking this tuple into 3 variables. Now if we print x you can see x is 1, similarly y is 2, there you go. So this is unpacking. And by the way this is not limited to tuples, we can use this feature for lists as well. So, if I change parenthesis to square brackets, now coordinates is a list, so we can unpack our list into 3 variables

# Dictionaries 2:18:16 – 2:26:16

now we run our program we can see y is 2. In this tutorial I'm going to talk to you about dictionary in python. We use dictionaries in situations where we want to store information that comes as key value pairs. Here's an example. Think of a customer. A customer has a bunch of attributes like name, email, phone number, address and so on. Now each of these attributes has a value. For example the name can be jon smith the email can be jon@ gmail.com the phone can be whatevr so what we have ere is a bunch of key value pairs. So in this example, our keys are name, email, and phone, and each key is associated with a value. So this is where we use a dictionary. With a dictionary, we can store a bunch of key value pairs. So let me show you how to define a dictionary in python. Back to our program, I'm going to define a variable, customer, and here we set it to curly braces. With these curly braces we can define dictionary. In this example we have an empty dictionary that doesn't have any key value pairs. Now we can add one or more key value pairs in between the braces. So let's add a key value pair here, I'm going to set the key to a name, and the value to John Smith, then we add a comma, to add another key value pair. So let's set age to 30 let's add another key value pair is underline verified and we set this to a boolean. Now what matters here is that these keys should be unique. So if I add another key value pair here, set age to 40, now look pycharm has highlighted the age key, because we have duplicated that and that's not allowed. So each key should be unique in a dictionary, just like, the dictionaries we have in the real world, in a real dictionary we have a bunch of words and they're definition each word is only listed once in a dictionary. We don't have the word book twice. So let's delete the second duplicate key value pair, so the keys should be unique, and in this example, I'm using strings, but they can also be numbers, we're going to look at that later, but the value can be anything, it can be a string, a number, a boolean, a list, literally anything. Now we can access each item in this dictionary using square brackets. So, we type customer, square brackets and then specify key like name and this will return the value associated with the name key. Let's print it on the terminal, have a look, there you go, so, the name is John Smith, now what if we pass a key that doesn't exist, lets say first date. You run the program, you get a key error because we don't have a key called birth date. Also, if we spell name with let's say a capital N, we get the same error because we don't have a key with the exact same sequence of characters in this dictionary. Now to get around this we can use the get method, so instead of using the square brackets we call the get method and specify the key Now if you use a key that doesn't exist here, it doesn't yell at us. For example, if you pass birth date it simply returns the non value. Earlier I told you that none is an object that represents the absence of a value. So instead of getting a key error we get none and we can also option supply a default value, for example, if this dictionary doesn't have this key, we can supply the default value, let's say January 1st 1980. Let's run the program now instead of getting none we get this default value. So, this is how we can access the value associated with the key in a dictionary. We can also update these values for example before a print statement we can write code like this, customer of name, let's update the name to jack smith, now this little warning is telling us here we can put jac smith here instead of defining it once, and then update it. Don't worry about them, it doesn't really matter. Now with this line if we print the name of this customer, we should see jack smith, let me show you. So, I'm going to use the square bracket notation again, let's print the name of the customer you can see that is updated here, we can also add a new key here, let's set the birthdate to some value like january first 1980. And then we can print it here, so as you see we can easily add new key value pairs to a dictionary. So this is the basics of using dictionaries in python. They're extremely important and they have a lot of applications in the real world. Okay, here's an exercise for you. So here we have this program that asks our phone number. Let's type 1 234. We type it in digits and then this will translate it, to words, take a look. Enter, it prints, 1, 2, 3, 4. That's a pretty cool program, so go ahead and spend a few minutes on this exercise, it's pretty easy, I will see you next. Alright, so first we need to get the users phone number, we call the input function with the label phone, we get the result and store it in this variable, now let's say the user enters 123 4. So we need to look through this string, get each character and translate it to a world, so what we need to implement this scenario. is a dictionary, because a dictionary is a structure that allows us to a key to a value. So we can have a dictionary with keys like 1234. And we map each of these keys to a word. So we cannot put digit 1 to the word 1 we can map 2 to two you get the point. So let's define a dictionary, you can call it digits, underline mapping now this dictionary I'm going to add a few key value pairs. One, we map it to 1, 2, we to 2, 3, to 3, and finally, 4 to 4. Now technically we should add all the digits from 0 to 9, but I don't want to waste all your time typing repetitive things here, you got the point. So lets move on, now we need to look through the phone string. So 4 character in phone, we get each character and then use it to access a key value pair in dictionary. So, digits underline mapping we can use square brackets or call the get method, I would prefer to use the get method, so in case the user enters some character that is not part of our dictionary, our program is not going to yell at them. So, we call the gt method and pass this character a the key, and if we don't have this key, we can supply a default value like exclamation mark, so with this we get a word now we need to add this word to an output string, so we can define an output string, initially we set it to an empty string, in each iteration we get this and add it to our output string. So, we type output, plus equals. This and we should also add a space at get. So the words are not close to each other, okay? That's all we have to do. Now finally let's print this output, run our program, so I want to type 1, 3,4, 5, lets

# Emoji Converter 2:26:16 – 2:30:24

see what we get. We get 1, 3, 4 with an exlamantion mark. In this tutorial I'm going to show you something really cool that you can do with dictionaries, so here in this program you can type a message like good morning followed by a smiley face, when we press enter, we get this beautiful smiley face, or we can type i am sad, with a sad smiley face and it gets translated to this beautiful emoji. So this is another application if using dictionaries, we have a dictionary that maps these characters into smiley faces. So lets go ahead and build this program together, we start by calling the input function, here we pass a greater than symbol as an indicator for the user to type a message, we give that message, store it here, as you know that is a string, now we need to split this string by a space, so if the user types good morning with a smiley face, we want to break this down into 3 words, good morning, and smiley face. To do that we can call the split method. So we call message.split and pass a string with one space as a separator. What this method does is basically it goes through this string and anywhere it finds this character, in this case a space, it uses it as a boundary to separate this string into multiple words. And then it will return a list. Let me show you. So let's store that here, words, and then Then print it on the terminal, let's go ahead and run our program. So if you type good morning sunshine, enter, we get a list with three items each item is a string, okay? Now, back to our program, we need to define a dictionary for mapping special characters like these 2 into a smiley face, so, we can call that emojis set it to curly braces, here we add a key value pair, a smiley face, and we map that to a string, in this string we want to add an emoji, if you're on a mac keyboard you can press control command and space and this should bring up this emoji box, if you're on Windows honestly I really don't know how you can do this, but there must be applications who do this for you. So lets go ahead and pick this happy smiley face, good, now lets add another key value pair, or sad face we map this to, this face here, good. So, we have a list of words now we need to look through this list, get each word and potentially map it to an emoji. So we can write a for loop for word in words, now we want to go to our dictionary, and see if you have an item with this word as the key. If you have an item with that key, we get it's value, otherwise we want to use the same word. So if a user types good morning, we don't want to translate those words, we just want to return those. So that is the case, where we use the get method to supply a default value, so we wan to get an item with a key word, and if we don't have an item with that key, we simply use that word as the default value. Now similar to the exercise that we did in the last tutorial we need an output variable. So let's define that here, output set it to an empty string, now back to line 9, we get the return value of the get method, and add it to our output variable. So output plus equals whatever we get from the get method, and finally we need to append a space at the end. That's it, we're done. So let's print the output and run our program and I'm going to type good morning sunshine with a smiley face, we get that beautiful, let's try the sad face as well, I am sad.

# Functions 2:30:24 – 2:35:20

Too bad. Beautiful! So as you can see dictionaries have a lot of use cases in real applications. Here's a solution from the last tutorial where we created an emoji converter. Now so far we have been writing all our code right here, in app,py, but as our programs grow we need a better way to organize our code. We need to break up our code into smaller, more manageable and more maintainable chunks which we call functions. So function is a container for a few lines of code that perform a specific task for example you have learned about a few of the built in functions in python like print, and input, each of these functions have a purpose, they know how to perform a specific task. So when we build large complex programs, we should break up our code into smaller reusable chunks which we call functions to better organize our code. And that's what I'm going to show you in this tutorial. So let's write this simple program, for printing a greeting message. So we're going to do a print, hi there, and lets add another message welcome aboard. So here we have a simple program with only two lines of code. Now let's say these two lines potentially we're going to need them in other programs so we can put them in a function that we can reuse. Let me show you how to do that, so, we start typing out death, that is a reserve keyword in Python, and it's short for define. When Python interpreter sees this it knows that we're defining a function, next we need to give our function a name, lets say greet\_user. So all the best practices you learned about naming your variables also apply here, which means we should name our functions with lower case characters if there are multiple words, we need to separate them using an underscore and you should always, always, always use meaningful descriptive names for your functions. So, here we define a function called greet user, after that we'll need to add parenthesis followed by a colon. Now you know that whenever we add a colon at the end of a line we're telling python that we're defining a block of code. So when we press enter, the next line is indented. Now any code that we write here will belong to this function. So I'm going to move these two lines these two print statements inside of our function. You can simply press tab to indent them, beautiful, also lets remove this extra line break, that is better, so these two lines now are part of our greet user function, so whenever we call this function, these two lines will be executed. Okay? Now, let's remove indentation so here we are outside of our function, let's do a print statement and print start after that let's call our function, so greet user. And finally let's do another create statement and print finish. Now we have this little warning here, this underline, if you hover your mouse over this line, this tool was telling you that we should add two blank lines after a function definition. This is coming from pep8 which is a document that defines the best practices for formatting our code. We'll look at that in the future. So for now to make pycharm happy, lets add an extra line break to this function so whenever we define a function we need to add two lines breaks after. Okay? so, now let's run our program and see what happens. Alright, so we have 4 messages on the terminal first we get the start message then we get the messages coming from our create function, so either and welcome aboard, and finally we see the finished message. So let me explain the flow of our program when python runs this code it doesn't actually print these two messages on the terminal because these two lines of code are inside of this function, so they would only get executed if we call this function. If we don't call this function, these lines don't get executed. Okay? So the execution of our program actually starts here, the first message that we see on the terminal is the start message. After that, Python sees that we are calling the greet function, so it will jump over here and then execute these two lines. Then, it will jump out of this function and continue the normal execution of our program so it will print the finish message on the terminal. So this is how functions work. Also note that the order of this code matters, so here we're calling the greet function after after we have defined it. If you try and call it before you define it you will see an error. For example, if we call the greet user function right here, see we immediately get this red underline which says unresolved reference, greet user which basically means python doesn't know what is this greet user, it's not defined anywhere. So we always define our functions first, and then

# Parameters 2:35:20 – 2:39:20

call them. Here's a function we created in the last tutorial now I got a question for you, what is the difference between calling this functions nd the print function of python? The difference is that the print function takes some information, in this case the message that we want to print, but our greet function doesn't take any information. Wouldn't it be nicer if you could pass the name of the user here and then instead of saying hi there, you could print the name of the user. So, in this tutorial, I'm going to show you how to pass information to your functions, and that is pretty easy, so back to the definition of our greet user function, inside of these parenthesis, we can add parameters these parameters are pale holders for receiving information. For example, you can add a name parameter, and we can pass the name of the user when calling this function. So let's pass Jon here, now when we call this function and pass this value this name parameter will be set to Jon. So it will act like a local variable that we defined inside of this function. Just imagine, we didn't have this parameter, and instead we had a variable like name that was set to jump. Now here we could simply print the name of this user, so we could use a formatted string, and print the value of the name variable, right? Now, we don't really have this local variable defined in this function we have a parameter, it's exactly the same thing. So with these parameters we can receive information in our functions. Now let's go ahead and run this program, so take a look. We get Hi John welcome aboard, we can take this to the next level. Now we can call this function one more time. So greet user, and this time supply a different name. Let's say Mary. So we are reusing our greet function and passing it a different value. Imagine if we didn't have this function you would have to repeat this line twice, once for hi jon and another time for hi mary. So this is the benefit of defining functions, we can take a few lines of code that have a specific purpose and put them inside of a function, so now let's run our program one more time there you go. So you get hi job, welcome aboard, then you get hi mary welcome aboard. Now note that when a function has a parameter we are obligated to pass a value for that parameter. So if you remove jon from here and run our program, we get this type error, greet user, missing one required positional argument. Argument in programming is the value that we supply to a function, so back to this example, marrying this case is an argument that we pass to the name parameter. And lot of programmers don't know the difference between parameters and arguments and think they are the same. But they are not, parameters are the holes or placeholders that we define in our function for receiving information, arguments are the actual pieces of information that we supply to these functions. Okay? So, let's bring back this jon string, now we can also define multiple parameters, for example, let's rename this parameter to first name as I told you before, you can right click here, go to refactor and then rename, or preferably you can use a short cut. So the shortcut for this on Mac is shift and f6. And this brings up the renamed dial up box, here we can change this to first underline name. Now we can add a second parameter so comma followed by last underline name so back to our formatted string, let's also print the last name, beautiful, so here when calling this function, then we can supply another argument so jon smith and lets delete this line we don't really need it anymore

# Keyword Arguments 2:39:20 – 2:44:40

run our program, take a look, hi jon smith, so in the last tutorial you learned that whenever we define parameters for our functions we should always supply values otherwise we'll get an error. Let's look at that error one more time. So In going to delete this second argument and only pass jon. Run our program, we get this message, greet user missing one required, position argument. So these arguments that you have here like jon and smith you refer to these as positional arguments. Which means their position or argument matters. In other words, the first argument we add here is the value or the first parameter and the second argument is the value for the second parameter. So if you change their position or order, we'll get different results. Let's take a look. Let's pass smith first and then jon. Okay, good, delete, now let's go with this, so instead of getting Hi Jon Smith, we get hi smith jon. So that is the reason we refer to these arguments as positional arguments, they're positioned or order matters. But in python we also have keyword arguments, and with those the position doesn't matter. Let me show you how they work. So, let's say here you want to pass Jon as the value for the first name parameter. We can simply specify that here. So first name equals jon. Now what we have here is a keyword argument. the key word argument. So this combination of having the parameter name, followed by it's value is what we call a keyword argument, and what we get with this keyword argument the position doesn't really matter. So here we're supplying Jon as an argument fort he first parameter. Now similarly, we can use a keyword argument for smith, so last name equals smith, so that we have 2 keyword arguments. So with these keyword arguments we don't have to worry about the order of the parameters. Now that doesn't mean we should always use keyword arguments. Not really. Most of the time we use positional arguments, but in certain situations these keyword arguments helming us improve the readability of our code. So on line 7 it's quite clear that we are supplying smith as the last name and jon as the first name. Now in this particular example, even if we didn't have the keyword arguments, just passing two positional arguments like jon and smith, would be sufficient, because it's easy to guess what is the purpose of these arguments. But sometimes you're dealing with functions and you're passing numerical values to them, in those situations it may not be quite clear what those values represent. Let me show you an example. Let's say we have a function for calculating the total cost of an order, so we still call that calculate, underline cost, so when calling this function, you need to supply three pieces of information. One is the total cost of all the order items lets say 50 dollars. The other is the shipping cost, let's say 5 dollars, and the last piece is discount, let's say 0,.1. Someone reading this code, may not be sure what these three values represent, in this case, we can improve the readability of this code, by using keyword arguments. So we can prefix each value with the name of parameter they target, for example, we can set total to 50 and then shipping to 5, and then discount to 0.1. Now if you give this code to someone else, they can immediately tell what these values represent. So that is the power of keyword arguments. So here's what I want you to take away. For the most part, use positional arguments, but if you're dealing with functions that take numerical values, see if you can improve the readability of your code by using keyword arguments. You simply prefix the arguments that you pass with the name of their parameters and this will increase the readability of your code. Now there is one more thing you need to know about keyword arguments. These keyword arguments should always come after positional arguments. Let me show you what I mean. So, back to our simple greet user function, let's use a keyword argument for the first thing. So, let's set first name to jon and see what happens. So we immediately get this error and positional argument after keyword argument. So when calling this function we're supplying one keyword argument and one positional argument, and python doesn't like this. So, if you're mixing positional and keyword arguments, you should always use and then the keyword arguments. In other words, lets remove this keyword argument here, so jon is now a positional argument, is targeting the first parameter. Now after this positional argument we can supply any number of keyword arguments, so if you set last name to smith look, python doesn't have any problem with this because we are adding a keyword argument after a positional argument. So let's quickly recap, for the most part use positional arguments if you're dealing with functions that take multiple numerical values

# Return Statement 2:44:40 – 2:53:36

and it's not quite clear what those values represent, use keyword arguments to improve the readability of your code, and finally if your passing both positional and keyword arguments, use the keyword arguments after the positional arguments. So far you have learned how to create functions and send them information using their parameters in this tutorial I'm going to show you how to create functions that return values, this is particularly useful if you're doing some kind of calculation your function and you want to return the result to whoever is using your function. let me show you. So let's start by defining a function, called square, this function simply calculates the square of a number. So we need to add one parameter here, that is is our number, now here we want to calculate number times number, that is the square of a number. Now to return this color outside of this function we simply use the return statement. So return. Now, our square function is very simple, it only had one line of code. But in a real program, you might have a function that has ten lines of code to calculate something. So, this return statement is very valuable. So now we can return values to the caller of our function. So, with this, when we call the square function, let's say we give it 3, now this function returns a value, just like an input function of python, this input function waits for the user to type something and press enter and then it will return the user's input as a string so we can get it and store it in a variable like this, right? We have the same story with our story, with our square function. So our square function returns the value we can simply get it and store it in a variable. Let's call that result. Now, we can print result on the terminal, there you go, so result is 9, we could also pass this function called directly inside of the print function without defining a separate variable. Sometimes this is useful for writing shorter code, so in this case. So, in this case, we don't really need this result variable. We simply want to print the square of 3. So, we could simply pass this function call right here, as an argument for the print function. So let's delete this variable now our code is shorter so with this we're calling the square function, this returns a value and we'll pass that value as an argument to the print function. Okay? When we run this program, we get the exact same result. There you go. Now what happens if we don't use a return statement in our function. So in the last example we simply used a print statement. Let's temporarily remove this return statement and print the result on the terminal. Like this. So let's run our program one more time and see what we get. We see two things, the number 9, and none. What is happening here? Well when Python interpreter executes this code, first it will call the square function. So the control moves to the square function here we calculate the square of this given number and then print it on the terminal. That is the reason we see 9 right here, now, by default, all functions return the value none. So if we don't have a return statement here, by default python returns none. None is an object that represents the absence of a value, it's like nothing or null i c, c++ java and Javascript. So, in this example, we didn't have a return statement, and by default python returned none from this function. So, after this square function is executed, the value none is returned and passed as an argument to the print function, that is the reason we see none on the second line. So here are the two things you need to take away from this tutorial. By default all functions in Python Creating a Reusable Function return none. You can change that, so if you have a function that calculates something, you can return the result using the return statement. Here's the code for our emoji converter program that we wrote earlier. If you missed that tutorial, make sure to go back and watch it, as I told you before, there is a table of content right below this video in the description. So we can simply click on the emoji converter tutorial to watch it. Now, your exercise is to reorganize this code using a function. So I want you to extract a function in this code, because this algorithm for converting these smiley faces into emoji's is something that we probably want to use in a couple different applications. We can use it in a chat application in an email application and so on. We don't want to write all this code every time we want to convert these characters into smiley faces. So, out of all these lines of code that we have from line 1 to line 10, the lines 10-2 belong to our algorithm, all these lines together implement this feature. So, you should put these lines in a separate function, Now, why don't we have the input in this function? Because the input will can come in different forms, currently we're using the import function to receive the input from the terminal, but in other applications we might receive the input from a graphical user interface, so the input can come in different forms. We don't want to put that in our function, because this line will not be reuseable. You want to give our function the actual message, as a string. We don't care how we got that message. In one program you might get it from the terminal, in another program you might get it from a graphical user interface. So that is the reason this line up code should not be in that function. By the same function, you shouldn't include the last line in that function because what we do with the output is different from one program to another. In this program, we are printing the output in another program, you might send an output as an email, or as a response in a chat application, so as a general rule of thumb you're function should not worry about receiving input and printing it. These lines of code should not belong to your functions. So go ahead and reorganize this code by extracting a function. Alright, on the top, first we need to define our function. So first we type def, give our function a name like emoji\_ converter, so the name of this function clearly tells me what it does. Each function should be responsible for one and only one task, and that task should be clearly explained in the name of the function. So, our emoji converter function should take a parameter, we call that message, that is a string, right? Colon now we need to move the lines 5 all the way to the end of our for block into the function, so let's select all these lines, you can cut them from here, by pressing command and x on Mac, or control and x on Windows, then we paste it here, as you can see these lines are automatically indented so, they are part of the emoji converter function, if this didn't happen on your computer, make sure to select these lines and use the tab to indent them. Alright, so here's our emoji converter function beautiful. Now after that in the main flow of our program, first we get the input from the user we store it in this variable message and then we call emoji converter and pass this message, this function should return a value, that is what we use to calculate the end of this function, so right after this for block we need to add a return statement, return output, now that our function returns a value, we can get it and store it in a separate variable, let's call that result and then we can simply print that here, or we can make this code shorter, we don't really need this code here so we can simply call the emoji converter, and print the result. Like this. Now here we have an underline because as I told you before we need to blank lines after our function definitions. So currently we have only one blank line we need to add another black line, good, let's go ahead, and run this program, and make sure everything works as before so I'm going to type a message like good morning with a smiley face good, let's run it one more time, times that, to add okay beautiful, so, here's the final result, we defined this function called

# Exceptions 2:53:36 – 2:59:12

emoji converter, this function takes a parameter called message. And finally returns the output. In this tutorial I'm going to teach you how to handle errors in your python programs, so let's start by writing a small program to get the user age from the terminal. So we call the input variable input, and a label here, like age, this returns a string, so we need to pass it to the end funciton and store the result in a variable called age, now let's print age to make sure you have implemented everything properly we run our program, we type 20, 20 is printed here, beautiful, now look at the exit code of this program, exit code 0 means our program terminated successfully, there were no errors, but what if we run this program one more time and instead of entering a numerical value, we enter something like asd. We get a value error with his message invalid literal for int with base 10. Basically what this message is telling us is that this string asd does not contain a valid home number that can be converted to an integer. Now look at the exit code of this program, exit code 1 means our exit code crashed, so 0 always means success and anything but 0 means crash now as a good python programmer you should anticipate this situation. You don't want to let your entire program crash just because the user entered an invalid value. So instead of letting the program crash you should handle the situation and print a proper error message, and that's what I'm going to show you in this tutorial, now before we get any further look at the type of error we get here, value error, so remember this, we're going to get back to this shortly now how can we handle these errors, in python we have a construct called try accept we use that to handle errors. Let me show you how it works. So on the top we type try: now as you know we are defining a code block so, I'm going to move these two lines, inside of our try block so press tab here. And remove this black line, now these two lines are part of our try block, right? So after try we type accept, and then we add the type of the error that our program may encounter, that is value error okay? Then, we add a colon and in this block we can define what should happen if our program encounters an error of type value error. In this case we want to print a proper error message, so print, invalid value. So with this try and accept we are telling python hey, go ahead and try running one or these two lines of code, if you encounter, an error of type. Value error then instead of crashing the program, try this error message on the terminal so this is how this works. Now more accurately in programming we reefer to this kind of error as an exception. So an exception is a kind of error that crashes our program. So our user enters an invalid value, this line of code in line 2 will raise an exception, and on line 4 will catch it and print this error message, so our program will no longer crash. Let me show you. So let's go ahead and run this age 20, everything works just like before and the exit code is 0, let's rerun this program, and enter asd, so we get this proper error message invalid error menu and once again the exit code is 0, so that means our program completed successfully. Didn't crash, now, we can also handle different kinds of errors or exceptions, for example right when we get the age let's define a variable income and set it to 20,000. And then define another variable called risk and set it to income divided by age. Let's run this program and enter 0 as the game to see what happened. So 0 is obviously a numerical value so when we enter this value, technically we shouldn't get any exceptions. Take a look. Our program crashed, we did a 0 division error, because we cannot divide a number by 0. Once again look at the exit code, so we couldn't catch this kind of error or this kind of exception with this accept block. Accept block is only cashing exceptions of type value error. And this happens when we try to convert a nonnumerical value, to an integer. So in a situation like this. We should handle different kinds of exceptions, we can add another except statement for an exception of type 0 division error. Now in this block we can print a different kind of message, like age cannot be 0. Let's run our program one more time. Now, I'm going to enter 0 as the age so we get this error message age cannot be 0 and look at the exit code. It's 0, so our program didn't crash, so to recap, we `used try accept blocks to handle exceptions that are raised in our programs, as a good programmer you should always anticipate

# Comments 2:59:12 – 3:01:44

these kind of exceptions and handle them properly. In this tutorial I'm going to talk to you guys about comments in python. We use comments to add notes or comments to our programs. For example, here I'm going to make a print statement, sky is blue. Now, before that, I can add a comment using a hi sign and then whatever we type here is going to get ignored it's not going to get executed. Let me show you so let's run this program, look we only see this message. Sky is blue. So with this comment we can explain something about our code, we can use it as reminders to fix things or clear things up, or we can use it to communicate things with other developers reading our code, why we have written this code in a certain way. These are good use cases for using comments, we can also have comments over multiple lines, so we can simply add more comments, each line should start with a new hi sign. Now one thing you need to avoid when using comments is explaining what the code does. So here's an example of a bad comment. Let me just delete these two lines, and with this comment I'm going to say print sky is blue. Why is this a bad comment, because it's telling me the obvious, it's telling me this slide is going to print sky is blue. Now the problem with this comment is that apart from being verbose and repetitive, if you come here, and change sky to ocean, now this comment gets outdated. So overtime we change this code, then we have to come back and modify the corresponding comment. So I've seen some developers add comments in front of their functions, for example, they define a function let's say a square that takes a number and simply returns number times number. But it is obvious that this function calculates and returns the square of the given number. So there is really no need to add a comment and explain that. Calculates and returns the square of a number. This is very repetitive and it creates noiser code. Someone else reading your code they get distracted with all these unnecessary comments, so use comments to explain whys and hows. Not why's if you have made certain assumptions, we can use comments to explain those assumptions. Or we can use them to add notes to remind yourself or other developers to do something in the code.

# Classes 3:01:44 – 3:07:44

So comments are good, but too much of a good thing is a bad thing. In this tutorial I'm going to talk to you guys about classes in python. Classes are extremely important in programming and they're not specific to python, in fact, a lot of other programming languages do support the notion of classes. We use classes to define new types. For example, so far you have learned about the basic types in Python like numbers, strings, and booleans, these are the basic or simple types in python, you also learn about a couple of complex types like lists and dictionaries. While these types are extremely useful, they cannot always be used to model complex concepts, for example, think about the concept of a point, or a shopping cart. A shopping cart is not a boolean, it's not a list, it's not a dictionary, it's a different kind of thing. So we can use classes to define new types to model real concepts. Now in this new model I'm going to show you how to define a new type called point, and this new type is going to have methods for working with points. For example, back to our main program. Let's create a list, numbers, 1, 2, 3, you know that here, when we type that dot, functions or methods available in list objects, now similar to this, we're going to create a new type called point, this point is going to have metas like move, draw, get, distance from another point, these are the operations that we can perform on points. So let me show you how to do this. We start by defining a class by using the class keyword, and right after that we give our class a name. Now look at the naming convention I've used here. I've capitalized the first letter here, this is what we call a pascal naming convention. So the naming convention we used for naming our classes is different from the convention we use for naming our variables and functions. For variables and functions we always use lower case letters and we separate multiple words using an underscore. But when naming classes, we don't use an underscore to separate multiple words, instead we capitalize the first letter of every word. So in this case our class has only a single word, if you had multiple words, let's say email client look I've capitalized the first letter of every word. Once again, this is called pascal naming convention and it comes from the old pascal language that you might have heard of. So, back to our point class. Here we add a colon which means we're going to define a block, now in this block we can define all the functions or methods that belong to points, for example, we can define a function called move for moving a port, now note that as soon as we type open parenthesis, pycharm automatically adds self here, this is a special keyword and we're going to get back to this shortly. So, for now, in this method, let's print move on the terminal, now let's define another method like draw, so define draw and in this method, let's print draw on the terminal, now let's remove the indentation, we're done with the definition of our point class, so we need to add two line breaks okay, so with this class we defined a new type with this new type we can create new objects, so an object is an instance of a class, a class simply defines the blueprint or the template for creating objects, and objects are the actual instances based on that blue print. So we can have tens of hundreds of points on the screen, these are the objects or the instances. So to create an object, we type out the name of our class and then call it like a function this creates a new object and then returns it. So then we can store that object in a variable, let's call that point 1, there you go, now when we use the dot operator. Look, we have these two methods that we defined, draw and move, we also have a bunch of other methods that we start with two underscores, these are called magic methods we'll look at them later in the course. Now, let's call the draw method of our point object, and run our program so you can see draw is printed on the terminal. Now apart from methods these objects can also have attributes and these attributes are like variables that belong to a particular object. For example, here, after we've defined.1. We can set.1.x to 10. And.1.1 to 20. Now we can easily print out these attributes, let's print.1.x and run our program one more time take a look, so the x coordinate of.1 is 10. Now we can create another object, let's call that.2 so.2 once again we type out the name of our class and then call it like a function. Now this object is completely different then the first object. So if you print.2.x look we get this attribute error, because this point object doesn't have an attribute called, so each object is a different instance of our points class. Now here we can assign a value to.2.x. Let's send that to 1, and then run our program so we can see the x coordinates of.2 is 1. So to recap we use classes to define new types, these types can have methods that we define in the body of the class and they can also

# Constructors 3:07:44 – 3:14:40

have attributes that we can set anywhere in our programs. You have learned how to create new types using classes now therein a tiny problem in this implementation. We can create a pooint object without an x or y coordinate let me show you. So point we create this object here, and before we set point.x let's print point.x run our program we get this attribute error which you saw in the last tutorial, so point object has no attribute called x This is the problem we have here, it is possible, to have a point object without an x or y coordinates. And that doesn't really make sense, because whenever we talk about a point, we ned to know where that point is located. To solve this problem, we use a constructor. A constructor is a function that gets called at the time of creating an object. So here on line 9, when creating this point object, we want to pass values for x or y coordinates, let's say 10 and 20, and with this, this point object that we get here, will have it's x and y coordinates initialized. So how do we do this? We need to add a special method in this class called constructor. So, on the top, I define a new function, look at the name of this function, double underscore init, double under score. So init is short for initialized, and this is the function or the method that gets called when we create a new point object. Now, press enter, so this automatically adds, self in parenthesis, we're going to come back to this shortly. Now, right after self, we want to add 2 extra parameters, x and y, so let's type out x and y, and then in the body of this method, we should read the values past here, and use them to initialize our object, so when we pass 10 and 20, these arguments will be used for x and y parameters. Now to initialize our object, we type our code like this. Self.x equals x. And then self.y equals y. What is going on here, this self that you see here is a reference to the current object. So, back to line 13, when we create a new point object, self references that object in memory, the same object, that we're referencing using this cariable. So earlier we typed point.x equals 10, with this code we set the x attribute of this point object. Now what we have in the constructor is exactly the same, we use the self to reference the current object and then we set the x attribute to the x argument passed to this function. So, using this init method, we can initialize our objects and we refer to this method as a constructor, this method is used to construct or create an object, okay? so, Now, let's remove line 14, so we created point object with these values and then print point.x let's run our program there you go, x is 10, we can also change these values later, so just before printing point.x we can set point.x to 11 and then run our program so x is updated, beautiful, alright, here's a exercise for you. I want you to define a new type called person these person objects should have a name attribute as well as a talk method. This is a very easy exercise so, you can tackle it in probably a couple of minutes. Alright, so first we start with a class keyboard, we define this person class once again look at the naming convention I have used here, the first letter of the first word in this case the only word is uppercase. Now, in this class we define all the methods we need, currently we only have method that is the talk method, so define talk, pycharm automatically adds self, we come back to this shortly, for now let's just print talk. So this is the body of our person class. We remove the indentation and add two line breaks after defining our classes. Now we can create the personal object, let's call that jon, we set it to person, we can call jon. talk, let's run the program so we get this message, beautiful, but our person objects should also have a name attribute, it doesn't make sense to have a person without the name. So that is when we use the constructor function. So on the top, we add a new method, at the name of this method is double underscore, init double underscore. So we call this method a constructor. Now in this method we need to add another parameter which is name, so we simply set self.name to name. Once again, self references the current object, we're setting the name attribute of the current object, to the name argument passed to this method. With this we can go back to line 9, and when creating a person object we pass the name, let's see jon smith now let's print jon.name and run our program there you go. So jon Smith is printed on the terminal, beautiful, so this is all you had to do as part of this exercise. But let's make this program a little more interesting. Instead of printing this boring message here, let's print hi, I am Jon Smith. So, we can use a formatted string with a message like Hi, I am, now we want to dynamically add the name of this person. How do we do this? Well, look at the parameter of this talk method. As I told you before, every method in our class should have this parameter, self. And they should be the very first parameter of each method, okay? So, with this self we can get reference to the current object, so we can add curly braces and type out self.name. Thsi returns the name attribute of the current person object. With this we can remove line 10, where we print jon.name and simply call jon.talk. Lets run our program. Hi, I am Jon Smith. There we go, beautiful. Now we can build another person object, let's say Bob, is a new person called Bob Smith, when we say Bob.talk. And run our program we get a different message, hi I'm Bob Smith, so

# Inheritance 3:14:40 – 3:19:28

each object is a different instance of a person class. In this tutorial Im going to talk to you about inheritance in Python, inheritance is a mechanism for using code and once again it's no limited to python, most languages that support classes also support inherence. So let ms how you how this works. Let's say we have this dog class with a simple walk method, in this method we're simply printing the walk message on the terminal, now let's say in real program, instead of just one line of code here we're going to have 10 lines of code. Now, what if in the future we want to define a another class let's say cat and we want to add this walk method there as well. Sow e'll have to repeat all that code, in this new class, walk. This is bad. because you have repeated or duplicated our code. In programming we have a principle called dry which is short for don't repeat yourself. Let's say sometime in the future, we discover a problem with our walk method. If you have repeated or duplicated this method in many other places, then we'll have to come back and fix that problem in every single place we have duplicated this code. So that's why in programming we should not define something twice. So how can we solve this problem. There are different approaches, one approach that is easier to understand for beginners is inherence Now here's how it works. We define a new class called maml and move the walk method right there. Then we'll have the dog and cat classes inherit that method from their parent just like how humans inherit certain attributes or behaviors form their parents. That' exactly the same, so on the top, we define a new class, let's call it mammal then we need to move the walk method there. So, we select it, cut it, using command and x on Mac, or control x on Windows and then paste it here, beautiful, now we want our dog class to inherit this class from the mammal class so right after the class name we add parenthesis and then type out the name of the parent class. In this case, mammal. That's all we have to do. So with his the dog class will inherit all he methods defined in the mammal class. Now there is a tiny problem here, if you look at line 9, there is a red underline here. The reason this is happening is because python doesn't like an empty class, so right now our dog class is empty, it doesn't have anything, we either need to add methods specific to dogs or just to make Python happy, we can use the path statement, and that basically means nothing. We're telling python interpreter, hey, pass this line, don't worry about it. But with this, we don't have an empty class, okay? Now as I told you before after each class I should have two line breaks so right now we only have one line break and that is the reason we have this little warning here, so let's add another black line, beautiful, so our dog class is inheriting the walk method from the mammal, you need to apply the same change to the cat class. So we'll have cat inherit from mammal, and then, use the path statement here. With these changes, now we can create either a dog or cat object, let's say dog, one, we set it to a dog object, and when we type dot look we have the walk method that is defined in the mamall class. We can easily call it, let's run our program there you go, you see the walk message, so both the dog and cat classes are inheriting all the classes methods defined in their parent class. Now here we can add methods specific to dogs, for example, we can define a method called bark, and then print bark. Now, we don't need the path statement because we already defined one method in our dog class. So now when we type out dog1. look we have two methods one is the walk method that we inherited and the bark method that we just defined in the dog class. Similarly, we can add a method specific to a cat, for example, we can define a method called be annoying, I hate cats, so let's print annoying, now when we create the cat object, let's remove that, cat1 and type out cat1. you have these two methods dog and be annoying. So these are the basics of inheritance in python Now there is more to this that really goes beyond the scope of this beginners tutorial.

# Modules 3:19:28 – 3:30:08

If you're interested to learn more about inheritance and other mechanisms for reusing code, I encourage you to get my complete Python course. In this tutorial, I'm going to talk to you about modules in python. A module in python is basically a file with some python code. And we use modules to organize our code into files. Just like sections in a super market. When you go to a super market there are different sections for fruits, vegetables, cleaning products, and so on. We don't have one section with all the products in the super market. We have the same concept in programming. So instead of writing all our code, instead of writing all the functions and classes in app.py we want to break up our code into multiple files. We refer to each file as a module. With this, not only is our code better organized and structured, but we'll also have the ability to reuse our code. For example, over here in app.py, I've defined two functions for converting weight from pounds to kilograms and vice versa. Now we can take these two functions and put them in a separate module called converters, and then we can import that module into any program that needs these converter functions. Let me show you how to do this. So, let's open up our project panel and right click the project, go to new, and add the new file, we're going to call this file, converters.py. Then, we go back to app.py select all this code, cut it and then paste it into our new module. converters.py. With this we have cleaned up our app module it's not bloated with different functions, so as we write more functions with classes, we put them in their corresponding modules, a module should contain all the related functions and classes, okay? Now you want to import the converters module into our app module, and that is really easy, on the top we write import converters without extension, so we don't add.py we only type out the name of the file. Let me close that project panel. Now, this converter is an object so we can use the dot operator to access it's members. So currently we have defined these two functions inside of our module, kilogram to pound, and pounds to kilograms. So we can easily call one of these functions let's pass 70 kilos get the result and print it out. Run our program, so, that is under 55 pounds. Now there is also another syntax for importing modules, let me show you, so instead of importing the entire module we can import specific functions from that module. Here's how it works. We type out from then we add the name of our module, so converters import, now here if you press control and space we can see the list of all the functions defined in this module. So here we can grab one of these functions let's say kilogram to pound, and with this we can directly call this function just like the function defined in this file. App.py. Compare what we have on line 4 to what we have on line 6. On line 6 we'll have to prefix this function with the name of an object converters. So we have to type converters. and then we'll be able to access this function. In contrast. When we import a specific function from our module, then we can easily call that function without prefixing it with the module name. So here's what you need to take away. We use modules to better organize our code. Instead of writing all the code, instead of writing all the functions in one file, you break up our code across multiple files. Each file is called a module and it should contain all the related functions and classes then we can import a module into another module, in this case, we are importing the converters module into our app module. And as you saw, there are two ways to import this module, we can import the entire module, and then we'll get an object with the same name as the module so we can type converters. and then we'll be able to access all the functions and classes defined in this module. The other appraoch is to impot a specific function or class from a module using the from import statement. Now here is your exercise. Earlier in the course you had an exercise for finding the largest number in the list. So here on the top we have a list of numbers, initially we set max to the first number in this list then, we look through this list, get each number, if the current number is grater than max, then we reset max. And finally when we are done with our loop we print max on the terminal. Now here's what I want you to do. This code currently has no organization. We don't have any functions, we don't have any modules, we wrote all the code in app.py, that is okay for small programs, but as your programs grow, you need to properly organize your code into classes and modules. So, as part of this exercise, I want you to write a function, call it find max. This function should take a list, and return the largest number in that list. Now after you do this, go ahead and put this function in a separate module. So extract it from here, and put it in a module, called utils. So in this module we're going to have a bunch of utility functions, oaky? Then import the utility module into the current module and call this function. Finally, get the result and print it on the terminal, this is a fantastic exercise to teach you how to properly organize your code. So, go ahead and spend a couple minutes on this exercise. So the first step is to define a function called find max. So on the top, I'm going to define function, find max this function should take a list of numbers, so we add a parameter called numbers. Now inside of this function you want to have these few lines, this is the algorithm for finding the largest number in a list, we don't want to bring the first line, because this is where we create a list. So somewhere in our program we're going to create this list, and tehen pass it as an argument to pass max function. Okay? So, let's go ahead and grab these few lines, cut them, and put them inside of find max function. Alright, so we find the largest number in this list, finally we need to return it, so, return max. So here's our function. Now we need to move this into a separate module. So let's go to the panel right click the project and add a new file, we can also add a new python file that's the same, let's call that utils so if you add a new python file pycharm automatically adds.py extension, but if you go to new file we'll have to explicitly type out.py. So make sure to get the extension right. Now, back to app.py, let's cut this code and paste it into this new module, okay beautiful. Now I'm going o go back to app.py and import our new module. We can either import the entire module, so import utils and then access individual functions so we can call utils.find max or we can explicitly import this function from the utils module. It doesn't really make a difference. So, we could type from utils import find max. And then we don't need to prefix this with the name of it's object, so we simply call find max. Now we need to call this after we initialize our list. So first we need to define the list of numbers. Then, we pass that list as an argument to find max we get the result which is max and finally we print it on the terminal. let's go ahead and run our program, so the largest number is 10, beautiful, now one last thing before we finish this tutorial, here we have this little warning under max and the tool tip says shadows built in name max. The reason we see this is because we actually have a function called max. And it does the exact same job as our find max function. So this is actually built into python. But I ask you to do it as an exercise because it's a great exercise for beginners. Now, back to the warning, this warning is basically telling us that we are overwriting the built in max function in python so we are changing the meaning of this function, and this is considered a bad practice, let me show you what I mean. So let's comment out what we have on line 4, by putting a hash sign here, we can also use, a short cut command and slash on Mac, or control on slash on Windows. So now this line is commented out. Look at the color of max here, it's purple, it's the same color as print. And this is the color used to identify the built in functions. So now we can call this function and give it a list of numbers run our program we get the same exact result however, if we bring this line back, once again we can use the same shortcut that is command and slash on Mac or control slash on Windows. Now, we are redefining max, so up to this line, max used to be a function, but after line 4 is executed max is set to an integer, that is the largest number in this list. So, we'll no longer be able to call it on line 5, let's run our program and see what happens. So there you go, we get this type error, int object is not callable, because max is now an integer, it's not a function, so we cannot call it. So to solve this problem, we can rename this variable to something else. So right click here. Go to refactor and rename, and rename this to maximum. So you can see the warning is gone. As you can see we have the same warning in our utils module. So here we're redefining max and as I told you before this is a bad practice. We're changing the meaning of built in functions in python. So, let's rename these as well, I'm going to use the short cut that is shift and

# Packages 3:30:08 – 3:36:16

f6 there you go and here we can change this to maximum. Enter, done. In this tutorial we're going to talk about packages in python, packages are basically another way to organize our code, so currently we have only 3 files for modules in our project but a real project can contain hundreds or even thousands of modules. We don't want to add all those modules here, because over time this directory will get bloated with so many files, so a better approach is to organize related modules, inside of a package, so a package is a container for multiple modules In file system terms a package is a directory or folder. So in our project we can add a new directory and in that directory we can add all the directed modules, as a metaphor think of a mall. When you go to a mall or a shopping center there are different sections for mens women and kids clothing, so that is ilke a package, now when you go to the mens section, there are different areas fior different kinds of products, we have a section for shoes, t shirts, jackets, and so on, so each of these is like a module, so, in this tutorial, I'm going to show you how to create and use packages in Python, here in our project panel, right click the panel and add a new directory. Let's call this directory e commerce. So we're going to create a package called e-commerce and in this package we're going to have all the modules related for an ecommerce application. For example, we can have modules for sales, shipping, customer service and so on. So let's go ahead Now here we have an empty directory, in order to convert this to a package we need to add a special file in it. So, right click this directory and add a new python file, call that file double underscore in it, double underscore this is a special convention in Python. When Python interpreter sees a file with this name and name in a directory, it treats this directory as a package, we also have a shortcut for that in pycharm so let me delete this directory and start again. So delete, once again right click the project, and go to new but instead of creating a new directory we can create a new python package, let's call our package ecommerce as you can see pycharm automatically creates this file for us so we don't have to manually create it. Okay? Now, in this package, let's add a new module, so a new python file, let's call it shipping okay? Now in this module we want to have a function for calculating the shipping costs. So define calculate, underline, shipping. Now don't worry about parameters or implementation, we simply print calculate underline shipping. Now lets say you want to import this shipping module into our app module, with this new structure this shipping module is now part of the ecommerce package, so we cannot import a direct you have to start from the ecommerce package. So, bak to app,py once again there are two ways to import this module. We can import the entire module or we can import one of it's functions or classes. So let me show you both approaches. First, we're going to import the entire module. Now what is the name of our package? it is ecommerce, right? So ecommerce. shipping. So instead of typing import shipping we type import ecommerce.shipping. we have to prefix it with the name of it's package. Now to access any of the functions or classes in this module we'll have to type out ecommerce.shipping. and there you go. Now we can access the calculate shipping function. So let's go ahead and run our program, we see this message printed on the terminal, beautiful. However with this approach, every time you want to call one of the functions to this module, we'll have to prefix it with ecommerce. shipping. that is very verbose. So when working with packages we often use the second approach, using the from statement. So we can type from ecommerce.shipping import calculate shipping. Now we don't have to prefix this function with ecommerce.shipping, and we can call it multiple times in this module, so our code is a little bit shorter. But what if you want to use multiple functions in this shipping module. We can either import them here, so calculate shipping, comma, let's say calculate\_ tax. Or we can import the entire module and then access all he functions or classes in that module. So, let me revert this back. To import the entire module we type from ecommerce so we remove.shipping so from this package, import the shipping module. Now the shipping module, is an object so we can access all the functions and classes defined here, using the dot operator, so shipping. there you go, calculate shipping. So to recap, when using the from statement we can either start from the package and import a specific module, or we can start from package.module and then import a specific function. Like calculate shipping. So, this was the basics of using packages in python. As I told you before, packages are extremely important, especially when you want to work with a framework like django, we use django for building web applications with python, and I'm going to show you that later in this course. So in Django we have quite a few packages and

# Generating Random Values 3:36:16 – 4:58:32

each package contains several modules. So make sure to watch the tutorials, about modules and packages one more time, otherwise you're going to get confused later when we get to Django. In this tutorial I'm going to talk to you guys about the built in modules in python, so python comes with a standard library that contains several modules for common tasks such as sending emails, working with date and time, generating random values and passwords and so on. There are so many modules already built into Python and that means there is already lots of functionality that we can reuse. Let me show you where you can find this standard library. So open up this browser and search for python 3, module index, make sure to add the version number, because the modules in python 2 are different from modules in python3. So here we search for python 3 module index, and the first page here, you can see is python module index, or python 3, let's go ahead, so these are all the modules built into python, for example we have modules for working with date and time, we have modules for sending emails for encoding values, there are so many modules here. Now the explanation of all these is beyond the scope of this course, that really requires it's own course. Because we have so many modules here, but don't let this list intimidate you. This is not something that you need to learn right now, as you learn python as you build more applications, gradually you will get familiar with more modules here, in fact nobody knows all the modules in python standard library. Everybody knows some modules based on the applications they have built. So in this tutorial, I'm going to show you how to use one of the built in modules, for generating random values. So on the top, we import the random module, now because random is a built in module, we don't need to have a file here called random.py. So python interpreter knows that random is a built in module as long as it knows where to find it. Now if you're curious where this random module is stored on the disk, let me show you, so here in the project panel, expand external libraries, then expand python 3, and then one more time python 3.7 library route. So this is a folder somewhere on your computer, in this folder, we have all the built in modules in python. So, if you scroll down, you should be able to find. The random module. There you go. It's right here. So random. py. You can open this and this shows you the source code of this module. there you go. Now in this tutorial, we don't want to worry about hat, so let's go back to our app.py, and use this module to generate random values. So when we import this module, now we have this object, random, so we can use the dot operator to acces it's methods. One method that is useful here is the random method, and every time we call it it generates a random value between 0 and 1. So here we can do a for loop, let's say for I in, range 3. So just to refresh your memory we are using the range function to create a range object. We can look through this range object and in each iteration this object will spit out a value. So with this for loop, we can execute this code 3 times. Now, let's indent this. Okay, and finally, let's print the result, There you go. So we can see that in each iteration we get a new random value between 0 and 1. Now what if we want a random value for a particular age? Let's say random values between 10 and 20, well, there is another method here, rand int, we use 2 arguments here to specify our range, let's say 10 and 20. Run our program one more time. Now we have the numbers 20 13 and 18. If we run our program one more time, We get different values, so 20, 19, 11, one more time. Look, you also have another powerful method for randomly picking an item from a list. Let's say we have a list of team members and you want to randomly pick someone as the leader. Let me show you how to do that. So, we define the list of team members, we set it to jon, mary, bob, and mosh. Now we can call random.choice and pass our members list this method randomly picks a method from this list and returns it. SO, we can store the result in leader and then print it on the terminal. let's run our program, so now the leader is mary, if you run our program one more time this line is mosh now it's mosh again one more time, it really likes mosh, you get the point. So the random module is very powerful, and it has a lot of applications. Now here's a very cool exercise. I want you to write this program to roll a dice. So every time we run this program we get a different value. Now we have 3 in 1, next time we get 3 and 6, one more time 5 and 6, 4 and 3, double one, that's a very cool exercise. So before we get started, I want you to pay attention to a few things. I want you to define a class called dice in this class we're going to have a method called rope so every time we call this method, we get a tuple, a tuple remember is a list of values but values cannot be changed, you cannot add a new item to it, we cannot remove it. So it's like a read only list. So every time, we call the row method, we should get the tuple of two random values. So go ahead, and do this exercise, You'll see my solution next. Alright, we start by importing the random module on the top, then we define a class called dice: this class we want to have a method called row. So define row parenthesis, this automatically adds self, now in this method, you want to generate two random values between 1 to 6. So we call random.randint. And pass 1 and 6. So this generates the first number, we can ut that here, now let's call this one more time, so I'm going to copy what we have on line 5, put it here, and change this variable to second. So we have two variables first and second. With two random values, finally we need to return them, in a tuple, so return parenthesis instead of square brackets, first and second, now in python when you want to return a python from a function, you don't have to add this parenthesis, so you can simplify your code and python will automatically interpret this as a tuple. So we are done with a definition of our class, we remove the indentation and add two line breaks. Now we create an object of this type so you set dice to dice next we roll the dice, so dice.roll and print the result on the terminal, that's all you have to do. Let's run this program, so now we get 2 and 5, lets run it one more time, 3 and 2, 1 and 1, beautiful. Now we have this little warning here, that says expected 2 black lines found 1. Once again this is coming from pep 8, pep is short for python and proposal. We have so many python enhancement proposals or peps. Pep 8 is a very famous 1, and in this document we have all the best practices for formatting our code. So pep 8 says, that after we define our classes we should have 2 lines breaks, we've already done that so the reason this is complaining is because we had 1 line break before the class, so let's add one more line break, now pycharm is happy. So basically pycharm is following all the best practices defined in pep8. Now you don't have to memorize Working with Directories any of these best practices, as you code in pycharm, pycharm gives you a little more warning, and gradually you will learn how to properly format your code, so it's readable by other people. In this tutorial, I'm going to show you how to work with directories in python. So here in python3, module index, you can see we have a module called path lib which provides an object oriented file system pact. That basically means it provides classes that we can use to create objects to work with directories and files. So, if you click here, you can see how this module works, we can see all kinds of examples. For example, here on there basic useage, you can see this is how we import the path class from the path lib module. Let me zooming so you can see clearly. Okay, so look, from path lib, that's our module, import path, note the naming convention. P is capitalized, that means path is a class, so we need to create an instance of this class as you can see, right here, now there's so much documentation here, you can read this on your own, but let me quickly show you a few examples to help you get started. So after pycharm on the top, from the path led module, that's import the path class. Now we need to create, a path object to reference a file or directory on our computer. So there are basically to ways to do this we can use an absolute path, or a relative path which basically means a path starting from the current directory. For example, if you wanna reference this ecommerce directory in our project, we can use the relative path. So we start from the current directory and then go somewhere else. With absolute paths we start from the root of our hard disk. For example, if you're on Windows you might have an absolute path like this, c drive, backslash, program, files, backslash microsoft, so these are directories in c drive. On Windows we use a backslash to build a path. If you're on Mac or Linux, your paths will look a little bit different, so instead of backslash we have a forward slash lets say user/local /bin. So these are examples of absolute paths. In this tutorial we're going to work with he relative path to work with this ecommerce directory that we have in our project. Now if you didn't create this directory earlier, simply right click on the project and then go to new directory. oaky? So, let's close this panel, we create a path object, now, if you don't pas an argument here, this will reference the current directory alternatively, we can pass string, in this string we can add a file or a directory. let's say the ecommerce directory. Now this runs a path object, so we store it here, now this path object has a few interesting methods for example, we can check to see if a path exists by calling the exist method. This simply returns a boolean so let's print it on the terminal, there you go. So this path exists, what if it changes to ecommerce 1, and run our program and get false, we can also create a new directory for example, let's change our path to email, now when we run this program, obviously we don't have this directory but we can create it by calling the mk dir method so make directory. Let's run our program alright, this method returns none which basically means it doesn't return any values. Now if you look at the project panel you can see here we have a new directory called emails, we can also delete this directory so, instead of mk dir we call rmdir which is short for remove directory, but let's go ahead, now we can see that directory is gone. We can also see all the files and directories in a given path, that is, very useful if you want to write a little program to automate something. For example, you can iterate over all the spreadsheets in a directory one them and process them. Let me show you. First we change the path to the current directory and then we call the glob method. With his method we cans each for files and directories in the current path. So as the first argument we need to pass a string that defines a search pattern. We can type an asterisk and that means everything, all files and all directories. We can optionally add an extension, so to get all the files, you need the files you use star.star. With this pattern, we'll only get the files in the current directory. But not the directories. We can also search for all the py files, or all the excel spreadsheets. Anything. So let's search for all the py files in the current directory, now when we run our program we get this generator object. Generator object are kind of an advanced topic, and they are beyond the scope of this beginners python course. But for now all you need to know is we can iterate or loop through these generator objects. So instead of printing this generator object, let's just iterate over it using a for loop, so, for, file in, the generator object that is returned from the glob method. q Let's print file. We run our program, there you go. So these are all the py files in my current directory, this could be different on your machine if you didn't follow the exercises I gave you earlier. For example, this utils.py, we created this as part of one of the exercises earlier in this course. So don't worry if the files you see here are different on your machine. So, you can use the glob method to search for files using a pattern, we can also get all the files and directories in the current path, so, we just use one asterisk, then run our program, take a look, so we have this Pypi and Pip ecommerce directory, we have notes.txt, this is a file I added earlier just to write notes as part of my presentation so you're not going to have this file. We also have a bunch of directories and files. Even though python's standard library is comprehensive and has so many modules for common tasks, it's not complete by any means. That's why we have this directory called python package index. Or pypi. And in this directory you can find hundreds of packages for doing awesome things. These are packages that people like you and I have built as part of our projects, and then they have published it as part of this directory. For other people to use. So that's why we have a fantastic community in python, there are so many python developers out there and there are so many packages right in front of us to reuse in our programs, let's say you want to send text messages in your programs, you don't have to build this functionality from scratch, we can simply come to pypi.org and search for sms aas you can see there are so many projects or apckages that give you this functionality. Of course not every package, not every project is complete or bug free, some of them are still in development or they have bugs, but if you look around I'm pretty sure for any kind of function you can imagine there is a good reliable and well documented package for you. In my complete python course, I will show you some of those most useful packages on pypi. For example you will learn how to access the information on yelp.com in your programs. So you can write a program and search for businesses registered on Yelp. Let's say you want to search for the best barber in New York City. I will show you how to do that. You will also learn how to use a technique called web scraping, which means you can build an engine and have that engine browse a website, and extract information from HDML files. This is the same technique that Google does to index various websites. So they have several engines, or web crawlers, these web crawlers are continuously looking at various websites and extracting information from their pages. When you publish a blog post Google's web crawler finds the title of your post, its keywords and so on. So, I'm going to show you how to do that with Python, we'll also look at browser automation, and this is extremely powerful, because you can automate testing of your web applications. So you might get a job as a test engineer, and your daily job is to browse a website, click here and there, fill out this form, that form, and ensure that this website is functioning properly. You can automate all that with a powerful python package called selenium, as I have shown in my complete, as I show in my complete python course. Now in this tutorial, I'm going to show you how to install packages, from pypi.org. So, here on pypi.org let's search for open py xl. This is a package that we use for working with excel spreadsheets. So let's search for this package, there you go, so currently it's at version 2.5. On the top you can see, this instruction pip install open pyexcel. this is the command that we need to type in the terminal window to install this package into our program, so back to pycharm, here on the status bar. You can open up a terminal window for executing various commands. So let's open it, now if you're on Windows you're probably going to see something different instead of this dollar sign, don't worry about that just execute this command I'm going to show you. So pip install open py xl Make sure to spell everything properly. So pip is the tool, that comes with a python installation, and we use it to install or uninstall packages registered on pypi.org. So here, we're using pip to install this package. open py xl. Enter, let's go ahead with that, it's going to take a few seconds, oaky done, now we can import this package and it's modules just like the built in modules in python, or the modules that we have in our project. Now if you're curious where this package is stored on disc let me show you. So open up the project panel, and then expand external libraries, next expand python 3, and then expand sight packages. This is where all these packages that we installed with pip are stored. So here we have open py xl, you can open this, in this folder we have another folder called cell, now Now, look what we have here you have this init type, earlier I told you that whenever we add this file to a folder, python treats that folder as a package, so sell is a package, and this package we have is modules. Sell, interface, read only and text. So as you can see, this package openpyxl has several Project 1: Automation with Python subpackages like cell, start, chart sheet and so on. And in each package we have several modules, in the next tutorial I will show you how to work with excel files. So we have covered all the core concepts in Python, I hope you have learned a lot. please support my hard work by liking this video and sharing it with others. And be sure to subscribe to my channel for more tutorials like this. Next we're going to work on our first python project that involves automation. You're going to write a python program that can process thousands of spreadsheets in under a second. Are you excited? Let's get started. In this tutorial I'm going to show you something really awesome you can do with python. You're going to learn how to process spreadsheets, and this is extremely valuable in situation where you have hundreds or thousands of spreadsheets that need to be updated. If you want to do this by hand it may take hours or days weeks, or even months, you can simply build a python program in less then half an hour. And have that python program, automatically process thousands of spreadsheets in under one second. So, here we're going to work on this excel spreadsheet. You can download this below this video. In this spreadsheet we have three columns, transaction ID, product ID, and press. Now don't worry about the fourth column yet. So in this spreadsheet we're going to record all kinds of transactions, but let's say due to an error, it could be a human error or a system error. The price that we have here is wrong. Let's say we need to decrease this by 10 percent. If you want to do this manually, we'll have to come back here and add a formula, so we typed out equal sign, let's say you want to recalculate say you want to recalculate the value of this cell. have to get that and then multiply it by 0.9 and that is like 90% of the original price, we are reducing the original reducing the price by 10%. Now when you hit enter you can see the updated price. Next we'll have to come back here, select this cell and apply the same formula to all the other rows. Now what if you have thousands of rows here? You'll have to scroll up and down, this is going to be very tedious. So we're going to write a python program that will do this for us. It's going to auotmate this process and not only that it will also add this beautiful chart right here, once again if you want to open up hundreds or thousands of spreadsheets and add a chart to each one, this is going to take at least a week or two Our python program is going to do all of this in a matter of seconds. So, let's get started. Alright the first thing I want you to do is to download this transactions file and add it to your project. I put the link below this video for you to download. So if you're wondering how to put this in this project, you can simply right click the project, and if you're on Mac you have this item where reveal in finder, if you're on Windows you should see something like reveal in explorer, or file explorer, or whatever it's called when you click this, this opens up the direct on your machine where this is stored. So simply copy paste the transactions file right here. Now, here on the top, first we need to import the open py excel package. So import open py excel, we can give it a an alias to make our code shorter. So asxl. This is not required but it just makes our code a little bit cleaner. Let me show you. So now instead of typing open pyxl. we type xl. okay so that's an alias for this package. Now in this package we have a function for loading an xl workbook. So let's call load workbook, and as the argument we pass transactions..xlsx. So this we load our excel workbook and return a workbook object. Okay? Now if you open this workbook, you can see you add only 1 sheet, and that is called sheet 1, so to access that you use square brackets, and specify the name of the sheet, that is sheet one, make sure to spell this with a capital S. Because this is case sensitive, if you don't spell it properly you're going to get an error, so this returns a sheet, now in this sheet, we have various cells. So next you need to learn how to access a particular cell, and that is very easy. back to py charm, with this sheet object, you can use square brackets and give the coordinate of a cell. Coordinate is the combination of the column and the row. For example, this cell it's coordinate is a1. So back to pycharm, we can type a1, and this returns our first cell. Also, there is another approach to getting a cell, instead of using square brackets with a string, we can use the cell method of the sheet object, so, sheet.cell, you pass the row and the column, so row and column 1 these return the exact same cell. Now let's print the value of this cell, cell.value and run our program so the value is transaction id, beautiful, so now we need to iterate over all these rows and for each row we need to get the value in the third column. That is the price. We'll get that and then multiply by 0.9. So first we need to know how many rows we have in this spreadsheet. We can get that from the sheet object so sheet has an attribute called mx\_ row. Let's print this, so in this sheet, we have 4 row, and you can verify that here. So we have a total of 4 rows. So we need to add a for loop. that would generate the numbers 1-4. So, let's delete this print statement, nd add a for loop for row in were going to use the range function unction to generate a range of numbers, starting from 1, all the way to sheet.max\_grow. Plus 1. The reason we're adding 1 to this, is because as I told you before this range function will generate numbers, starting from this value all the way to this value, but it will not include the second value. So currently max row returns 4, if you use range of 1-4 this will generate the numbers 1, 2, and 3, but not 4. So to include 4, we need to add 1 to it. Let's go over here, so we tied sheet.max underline row plus 1. Now before going any further let's just print row and make sure our program is working up to this point. So let's run this, we get the numbers 1-4 beautiful. So now we can easily get the cells in the third column. However we don't really want this first cell, that's the heading. so technically you want to ignore the first row. Back here, you want to change our range function, and start from 2. Now, we're going to use sheet.cell to get access to the cell at this row. And the column should be 3. We get a cell object, let's print it's value and make sure you're on the right track, so run the program these are the values we get. 5.95 6.95 and 7.95 these are the values of the third column. Next we need to multiply each value by 0.9 so instead of printing cell.value we multiplied by 0.9 and this is the corrected underline price, so this is the mistake we're going to make with our program, now we need to add a new cell to our worksheet, so we want to add all the corrected prices, ia new column, it could also overwrite the values in this column, but in this tutorial I'm going to show you how to add a new column. So using the same method, now we need to get a reference to the cell, to the given row, but in the fourth column. So, we call sheet.cell, and pass row and four as the arguments. Once again, this returns a cell object, let's store that in a separate variable and call it price\_cell. So note that my variable names are very descriptive. There is no guess work. You have corrected price which holds actual value and we have corrected price, which is a cell object in this spreadsheet. Next we have to set the value in this cell. So we set corrected price cell. value to this corrected price. With these few lines we can easily update our spreadsheet, but first we need to save it. So, after our for loop, we call workbook. save. Now let's save this in a new file because we don't want to accidentally overwrite the original file in case our program has a bug so, I'm going to call that transactions 2.sms ex. And finally let's run our program, so, you can see that happened in a split of a second, now here we have a new file tranactions 2. When we open this we get something like this. So you can see we have this 4th column with the updated prices. Beautiful. So half of the problem is solved, now we need to add a chart here. To add a chart, we need to add a couple classes on the top, so, after this import statement let's type out from open py excel.chart, import, bar chart, comma, reference. So lets see what's going on here. In this package you have a module chart, and from this module we're importing two classes, bar chart, and reference. Again, look at the naming convention used here. The first letter of e very word is capitalized. Also it's better to add a line break after our import statements to make our code cleaner. So, before we save our workbook, we need to add achart to the current sheet. First we need to select a range of values. For this exercise, I'm going to select the values in the fourth column/. So all the values in rows 2-4 these are the values I'm going to use in our chart. Now in this part 2 example it doesn't really make sense to add a chart per transaction, but that doesn't really matter we just want a bunch of numbers to create a chart, so after our for loop, we're going to use the reference class to select a range of values. Now the first argument in the constructor is the sheet. So we pass our sheet here, then we add 4 keyword arguments, the first one is min\_row, we set this to 2. The second one is max\_row we set this to the maximum row in this sheet. That is sheet,.max\_row, so we want to select the cells in row 2-4. kay? Now to make this code a little bit cleaner I'm going to put this on a new line, so we can see clearly, now this is going to select all the cells in all the columns in these rows. That's not what we want. we only want the values in the fourth column, so back here we need to set a couple more key word arguments men\_call or column, we set that to for, and also max underline call, we set that to 4 as well. So we are limiting the range of cells we're selecting to the fourth column. Now, we're creating an instance of the reference class, let's store the result in a variable called values, so this values object will have all these values in the fourth column, now we are ready to create a chart, so we create an instance of the bar chart class and store it in this object. Next, we call chart.add\_date and pass our values. That is pretty straight forward finally we need to add this chart to our sheet. So, we call sheet,.add\_chart and pass this chart object We should also specify where we want to add this chart. let's say we want to add it here, on row 2, after the fourth column. So the coordinate of this cell is e2, and this is going to be, the top left corner of our chart. So as the coordinate I'm going to pass e2 that's all we had to do, done. Let's run our program one more time, beautiful, let's open up the update workbook. And here's the end result, so right after the fourth column, we have this beautiful chart. Now we could take this to the next level, we could add a legend here, we could change the color of these bars, we could even use a different kind of chart, it doesn't have to be a bar chart. So to learn more about this we can read the documentation for openpyxl. So our program is complete, for our code is dirty, it's not clean, it's not organized. We don't have any functions here. We have written all the code in app.py. So let's goo ahead and organize this code like a professional software developer. So, on line 4, we are loading our workbook then we're getting a reference to the first sheet, but we don't really need these two lines here this is purely for demonstration to show you how to access a cell. So, let's delete these unnecessary lines, this is a very important practice as you're coding always look at your code, always review it, see if you have some code that is not used. Always delete those. Okay, next we are iterating over all the rows, fixing the prices, and then we select the values to add a chart, and finally we save the workbook. Now, if you were going to use this to automate the process of updating thousands of spreadsheets, this program wouldn't work, because it's only relying on this file. So we want o reorganize this code, and move it inside of a function. This function should take the name of a function So let's define a function. Let's call it process\_workbook. It gets a file name. Now, we move all the code inside of this function, so let's select everything, and press tab, beautiful, now back to our function, instead of loading this transactions file we're going to load this file name, that's better, Also, now that our program is working properly, there is really no need to store the result in a separate file. So here we can simply overwrite the same file. So let's pass file name, that is the end of our function, so, now, we have this reusable function, we can simply reuse this to process thousands of spreadsheets. Earlier in this python course you learned Project 2: Machine Learning with Python how to find all the files in a directory, so we can get each file in a directory and pass the name of the file to this function, this function will go ahead and update this spreadsheet in just a second or less. So, that was just one example of using python to automate repetitive boring task that waste your time. But automation is not just about processing excel spreadsheets, there are so many things we can automate. Here's a question for you, what do you want to automate, how do you want to use python to make your life easier? Use the comment box below and let me know. I love to hear about your ideas. Next we're going to work on project which involves machine learning, which is a subset of artificial intelligence. Are you excited? Let's get started. In this section, you're going to learn about machine learning, which is a subset of AI or artificial intelligence. It's one of the trending topics in the world, these days, and it's going to have a lot of applications in the future. Here's an example. Imagine I ask you to write a program to scan an image, and tell if it's a cat or a dog. If you want to build this program using traditional programming techniques, your program is going to get overly complex You will have to come up with lots of rules to come up with specific curves edges and colors in an image to tell if it's a cat or a dog. But if I give you a black and white photo, your rules may not work, they may break. Then you would have to rewrite them. Or I may give you a picture of a cat or a dog from a different angle that we did not predict before. So solving this problem using traditional programming techniques is going to get overly complex or sometimes impossible. Now to make the matter worse. What if I ask you to extend this program such that it supports 3 kinds of animals. Cats, dogs, and horses, once again you will have to rewrite all those rules. That's not going to work. So machine learning is a technique to solve these kinds of problems, and this is how it works. We build a model or an engine and give it lots and lots of data. For example, we give it thousands or tens of thousands of pictures of cats and dogs. Our model will then find and learn patterns and the input data, so we can give it a new picture of a cat that we haven't seen before. And ask it, is it a cat or a dog or a horse and it will tell us with a certain level of accuracy, the more input data we give it, the more accurate our model is going to be. So that was a very basic example, but machine learning has other applications in self driving cars, robotics, language processing, vision processing, forecasting things like stock market trends and the weather, games and so on. So that's the basic idea about machine learning. Next we'll look at machine learning in action. A machine learning project involves a number of steps, the first step is to import our data which often comes in the form of a csv file. You might have a database with lots of data, we can simply export that data and store it in a csv file for the purpose of our machine learning project. So we import our data, next, we need to clean it. And this involves tasks such as duplicated data. If you have duplicates in the data, we don't want to feed this to the model, because otherwise our model will learn bad patterns in our data and will produce the wrong result, so we should make sure that our input data is in a good, and clean shape. If there is data that is irrelevant we should remove them, if there are duplicates or incomplete we can remove or modify them, if our data is text based, like the name of countries, or genres of music, or cats and dogs, we need to convert them to numerical values. So this step really depends on the kinds of data we're working with, every project is different. Now that we have a clean data set, we need to split it into two segments. One for training our model, and the other for testing it, to make sure our model produces the right result. For example, if you have 1,000 pictures of cats and dogs, we can reserve 80% for training, and another 20% for testing. The next step is to create a model, and this involves selecting an algorithm to analyze the data. There are so many machine learning algorithms out there, such as decision trees, neural networks and so on. Each algorithm have pros and cons in terms of accuracy and performance so the algorithm you use, depends on the kind of problem you're trying to solve and your input data. Now the good news isnthat we don't have to explicitly program an algorithm, there are libraries out there that provide these algorithms, one of the most popular ones, which we are going to look at in this tutorial, is side kick learn. So we build a model using an algorithm, next we need to train our model. So we feed it our training data. Our model will then look for the patterns in the data, so enxt we can ask it to make predictions. Back to our examples of cats and dogs, we can ask our model is this a cat or is this a dog and our model will make a prediction, now the prediction is not always accurate in fact when you start out, it's very likely your predictions are inaccurate. So we need to envalue the predictions and measure their accuracy. Then we need to get back to our model and either select a different algorithm that is going to produce a more accurate result for the kind of problem we're trying to solve. Or fine tune the parameters of our model. So each algorithm has parameters that we can modify to optimize the accuracy. So these are the high level steps that you follow in a machine learning project. Next we'll look at the libraries and tools for machine learning. In this lecture we're going too look at the popular python library that we use in machine learning projects. The first one is numpy which provides a multidimensional array. A very, very, popular library. The seconnd one is pandas, which is a data analysis library that provides a concept called data frame. Data frame is a two dimensional data structure similar to an excel spreadsheet. So we have rows and columns we can select columns in a row or a column or a range of rows and columns, again, very very popular in machine learning and data science projects. The third library is mad plot lib which is a two dimensional plotting library for creating graphs on plots. The next library is sidekick learn, which is one of the most popular machine learning libraries that provides all these common algorithms like decision trees, neural networks and so on. Now, when working with machine learning projects, we use an environment called jupiter for writing our code, technically we can still use vs code or any other code editor, but these editors are not ideal for machine learning projects, because we need to frequently inspect the data, and that is really hard in environments like vs code and terminal. If you're working with a table of 10 or 20 columns visualizing this data in a terminal window is really, really difficult and messy, so thats why we use jupiter. So that's why we use jupiter, it makes it really easy to inspect our data. Now to install jupiter, we are going to use platform called anaconda. So, head over to Anaconda.com/download. On this page you can download Anaconda distribution for your operating system, so, we have distributions for Windows Mac and Linux. So let's go ahead and install Anaconda for python 3.7. Download, Alright, so here's Anaconda, download it on my machine, let's double click this. Alright, first it's going to run a program to determine if the software can be installed. So, let's continue and once again continue, pretty easy, continue one more time, I agree with the license agreement, we can use the default you ca use the default installation location so don't worry about that, just click install, give it a few second, now the beautiful thing about Anaconda is it will install jupiter, as well as all those data popular science libraries like numpy, Pandas and so on. So we don't have to manually install this using pip. Alright now as part of the next step Anaconda is suggesting to install Microsoft vs code. We already have this on our machine so we don't have to install it we can go with continue and close the installation, now finally we can move the this to trash because we don't need this installer in the future. Alright, now open upa terminal window and type jupiter, with a y, space, notebook. This will start the notebook server on your machine. So enter there you go. This will start the notebook server on your machine, we can see these default messages here, don't worry about them, now it automatically opens a browser window, pointing to local host port 888. This is what we call jupiter dashboard. On this dashboard we have a few tabs, the first tab is the files tab, and by default, this points to your home directory. So every user on your machine has a home directory, this its my home directory on Mac, you can see here we have a desktop folder as well as documents, downloads and so on. On your machine you're going to see different folders, so somewhere so somewhere on your machine you need to create a jupiter notebook. I'm going to go to desktop here's my desktop I don't have anything here, and then click new, I want to create a notebook for python 3. In this notebook we can write python code, and execute it line by line. We can easily visualize our data as you can see over the next few videos. So, let's go ahead with this, alright, here's our first notebook, you can see by default it's called untitled, let's change that to Hello World so this is going to be the hello world of our machine learning project. Let's rename this now if you look at your desktop you can see this file hello world.ipynb. This is a jupiter notebook. It's kind of similar to our py files where we write our python code, but it includes additional data that jupiter uses to execute our code. So back to our notebook, let's do a print hello world. And then, click this run button here and here's the result printed in jupiter, so we don't have to navigate back and forth between the terminal window we can see all the result right here. Next I'm going to show you how to load a data set from a csv file in jupiter. Alright, in this lecture we're going to download a data set from a very popular website called caggle.com. Caggle is basically a place to do data science projects. So the first thing you need to do is to create an account, you can sign up with Facebook, Google, or using a custom email and password, once you sign up then come back here, on caggle.com, Here one the search bar search for video game sales. This is the name of a very popular data set that we're going to use in this lecture. So, here in this list you can see the first item with this kind of reddish icon, so, let's go with that, as you can see this data set includes the sales data for more then 16,000 videos games. On this page you can see the description of various columns in this data set, we have rank, name, platform, year, and so on, so here's our data source, it's a csv file called vg sales.csv, as you can see there are over 16,000 rows, and 11 columns in this data set. Right below that you can see the first few records of this data set. So, here's our first record, the ranking for this game is 1, it's the wi sport game for wii as the platform and it was released in the year 2006, now, what I want you to do is go ahead and download the data set, and as I told you before you need to sign in before you can download this. So this will give you a zip file as you can see here, here's our csv file, now I want you to put this right next to your jupiter notebook, on my machine that is on my desktop so I'm going to drag and drop this onto the desktop folder. Now, if you look at the desktop you can see here's my jupiter, hello world notebook, and right next to that we have vgsales.csv. With that, we go back to our jupiter notebook, let's remove the first line and instead import pandas as pd. With his we're importing pandas module and renaming it to pd, so we don't have to type pandas. several times in this code. Now let's type pd. read\_csv. And pass the name of our csv file. That is vgsales. csv. Now because this csv file, is in the current folder right next to our jupiter notebook, we can easily load it, otherwise we have to supply the full path to this file. So, this returns a data frame object which is like an excel spreadsheet. Let me show you. So we store it here, and then we can simply type df to inspect it. So one more time let's run this program here's our data frame with these rows and these columns so we have frank name platform and so on. Now this data frame object has lots of attributes and methods that we're not going to cover in this tutorial, that's really beyond the scope of what we're going to do, so I'm going to leave it up to you to read pandas documentation or follow other tutorials to find out about pandas data frames. But in this lecture, I'm going to show you some of the most useful methods and attributes. The first one is shape, so shape let's run this one more time, so here's the shape of this data set, we have over 16,000 records and 11 columns. Technically this is a 2 dimensional array of 16,000 and 11, okay? Now, we can see here we have another segment for writing code. SO we don't have to write all the code in the first segment so here in the second segment we can call one of the methods of the dataframe, that is df.describe. Now when we run this program, we can see the output for each segment right next to it. So here's our first segment, here we have these three lines, and this is the output of the last line. Below that we have our second segment, here we're calling the describe method, and right below that we have the output of this segment So this is the beauty of jupiter, you can easily visualize our datsa, doing this with vscode in windows is really tedious and clunky. So what is the describe method returning? Basically it's returning some basic information about each column in this data set, So, as you saw earlier we have columns like rank, year and so on. These are the columns with numerical values. Now for each column we have the count which is the number of records in that column, you can se our rank column has 16,5 -98 records whereas the year column has 16,3- -27 records.. So this shows that some of our records don't have the value for the year column. We have null values. So in a real data science or machine learning project we'll have to use some techniques to clean up our data set. One option is to remove the records that don't have a value for the year column. Or we can assign them a default value. That really depends on the project. Now another attribute for each column is mean so this is the average of all the values, now in the case of the rank column, this rank doesn't really matter, but look at the year. So the average year year for all these video games in our dataset is 2006. And this might be important in the problem we're trying to solve. We also have standard deviation, which is a measure to quanitfy the amount of variation in our set of values, below that we have min, as an example the minimum value for the year column is 1980. So quite often when we work with a new data set, we call the describe method to get some basic statistics about our data. Let me show you another useful attribute. So, in the next segment, let's type df.values. Let's run this, as you can see this returns a two dimensional array, this square bracket indicates the outer array and a second one represents an inner array. So the first element inn our outer array, is an array itself, these are the values in this array, which basically represent the first row in our data set. So the video game we ranking 1, which is called wii sports. So this was a basic overview of pandas data frames, in the next lecture I'm going to show you some of the useful shortcuts of jupiter. In this lecture I'm going to show to you some of the most useful shortcuts in Jupiter, Now the first thing I want you to pay attention to is this green bar on the left. This indicates that this cell is currently in the edit mode, so we can write code here. Now, if we press the escape key, green turns to blue, and that means this cell is currently in the command mode. So basically the activated cell can either be in the edit mode or command mode. Depending on the mode, we have different shortcuts, so here we're in the command mode. If we press h, we can see the list of all the keybord shortcuts, right above this list we can see, Mac OS modifier keys, these are the extra keys on a Mac keyboard. If you're a Windows user you're not gong to see these. So as an example, here is the shape of the command key, this is control, this is option, and so on. With this guideline you can easily understand the shortcut associated with each command. Let me show you. So here we have all the commmands when a cell is in the command mode. For example, we have this command, open the command palette. This is exactly like the command palette that we have in vs code. Here is a short cut to execute this command. That is command shift and F. Okay, so here we have lots of shortcuts, of course you're not going to use all of them all the time, but it's good to have a quick look here to see what is available for you. Tis these shortcuts you can write code much faster. So let me show you somme of the most useful ones. I'm going to close this, now with our first cell in the command mode I'm going to press b, and this inserts, a new cell below this cell. We can also go back to our first cell, press escape, now the cell is in the command mode. We can insert an empty cell above this cell by pressing a. So either a or b. A for above or b for below. Now if you don't want this cell, you can press d twice to delete it. Like this. Now in the cell I'm going to print a hello world message. So print hello world. Now, to run the code in this cell, we can click on the run button here, so, here's our print function, and right below that you can see the output of this function. But note that when you run a cell, this will only execute the code in that cell. In other words, the code in other cells will not be executed. Let me show you want I mean, so in the cell below the cell, I'm going to delete the call to a describe method. Instead I'm going to print ocean. Now, I'm going to put the cursor back in this cell where we print this hello world message and run this cell. So we can see hello world is displayed here, but the cell below is still displaying the describe table, so we don't see the changes here. Now, to solve this problem, we can go to the cell menu on the top, and run all cells together. This can work for a small project, but sometimes you're working with a large data set, so if you want to run all these cells together it's going to take a lot of time. That is the reason jupiter saves the output of each cell, so we don't have to rerun that code if it hasn't changed. So this notebook file that we have here, includes our source code organized in cells as well as the output for each cell. That is why it's different from a regular py file where we only have the source code. Here we also have autocompletion and intellisence, so in this cell, let's call df data frame. now if you press tab you can see all the attributes and methods in this object. So let's call describe, now with the cursor on the name of the method we can press shift and tab, to see this tool tip that describes what this method does and what parameter it takes. So here in front of so here in front of signature you can see the describe method, these are the paramter and their default value, and right below that you can see the description of what that method does. In this case, it describes generates descriptive statistics, that summarize the central tendency and so on. Similar to vs code, we can also convert a line to comment by pressing command and slash on mac, or control slash on windows. Like this. Now that line is a comment, we can press the same shortcut one more time to remove the comment, so these are some of the most useful shortcuts in jupiter. Now over the next few lectures we're going to work on a real machine learning project, but before we get there, let's delete all the cells here, so we start with only a single empty cell, so here, in this cell, first I'm going to press the escape button, now the cell is blue, so we're in the command mode, and we can delete the cell by pressing d twice. There you go. Now, the next cell is activated and is in the command mode. So, let's delete this as well, so we have two more cells to delete there you go, and the last one, like this, so now we have an empty notebook with a single cell. Over the next few lectures, we're going to work on a real machine learning project, imagine we have an online music store, when our users sign up, we asked our age and gender, and based on their profile, you recommend various music albums their likely to buy. So in this project, you want to use machine learning to increase sales. So, we want to build a model, we feed this model with some sample data, based on the existing users. Our model will learn the patterns in our data, so we can ask it to make predictions. When a new user signs up, we tell our model, hey, we have a new user with this profile, what is the kind of music, that this user is interested in, our model will say jazz, or hip hop, or whatever, and baed on that we can make suggestions to the user so, this is the problem we're going to solve, now back to the list of steps in the machine learning projects, first we need to import our data, then, we should prepare or clean it, next we select a machine learning algorithm to build a model, we treat our model and ask it to make predictions. And finally, we evaluate our algorithm to see it's accuracy. if it's not accurate we either fine tune our model or select a different algorithm. So let's focus on the first step. Head over to bit.ly/music.csv this is a very basic csv that I've created for this project, it's just some random made up data it's not real. So we have a table with 3 columns, age, gender, and genre. Gender can either be one which represents a male, or a 0, which represents a female, here I'm making a few assumptions. I'm assuming that men between 20 and 25 like hip hop men between 26 and 30 like jazz, and after the age of 30, they like classical music. For women I'm assuming that if they are between 20 and 25 they like dance music, if they are between 26 and 30 they like acoustic music, and just like men, after the age of 30 they like classical music. Once again this is a made up pattern, it's not the representation of the reality, so, let's go ahead and download this csv. Click on this... icon here and download this file. In my downloads folder, here we have this music.csv. I'm going to drag and drop this onto the desktop because that's where I've stored this hello world notebook. So I want you to put this csv file right next to your jupiter notebook. Now, back to our notebook, you need to read the csv file, so just like before first we need to import the pandas module, so import panda as pd. And then we'll call pd,.read\_csv and the name of our file is music.csv. As you saw earlier this returns a data frame which is a two dimensional array similar to an xl spreadsheet, so let's call that music\_ data. Now let's inspect this music\_data to make sure we loaded everything properly so run, so here's our data frame, beautiful, next we need to prepare or clean the data. And that's the topic for the next lecture. next lecture. The second step in a machine learning project, is cleaning, or preparing the data, and that involves tasks such as removing duplicate null values and so on. Now on this particular data set we don't have to do anything kind of cleaning, because we don't have any duplicates, and as you can see As you can see, all rows have values for all columns, so we don't have null values. But there is one thing we need to do, we should split this data set into two separate data sets, one into the first two columns, which we refer to as the input set, and the other with the last column which we refer to as the output set. So when we train a model, so when we train a model we give it two separate data sets. The input set and the output set. The output set which is in this case the genre column contains the predictions so we're telling our model that if we have a user who's 20 years old and a male they like hip hop. Once we train our model then we give it a new input set. For example, we have a new user who is 21 years old and is a male, what is the genre of the music that this user probably likes. As you can see in our input set, we don't have a sample for a 21 year old male, so we're going to ask our model to predict that. That is the reason we need to split this data set into two separate sets. Input and output So back to our code, this data frame object has a method called drop. Now if you put the cursor on the method name and press shift and tab, you can see this tool tip, so this is the signature of this drop method, these are the parameters you can pass here. the parameter we're going to use in this lecture is set to none by default. Witht his parter we can specify the columns we want to drop, so in this case we set columns to an array with one string, genre now this method doesn't actually modify the original data set. In fact it will create a new data set but without thisq call. So by convention we use a capital X to represent that data set, so capital x equals this expression. Now, let's inspect x, so as you can see our input set or x includes these two columns, age and gender, it doesn't have the output or predictions. Next we need to create our output set so, once again we start with our data frame using data, using square brackets we can get all the values in a given column. In this case, genre. Once again this returns a new data set. By convention we use a lower case y to represent that. So that is our output data. Let's inspect that as well. So, in this data set we only have the predictions or the answers. So we have prepared our data, next we want to create a model using an algorithm. The next step is to build a model using a machine learning algorithm. There are so many algorithms out there, each algorithm has its pros and cons, in terms of the performance and accuracy. In this lecture we're going to use a very simple algorithm called decision tree. Now the good news is we don't have to explicitly program these algorithms, they're already implemented in a library called side kick learn. So, here on the top, from sklearn.tree let's import the decision tree classifier. So sklearn is the package that comes with sidekick learn library, this is the most popular machine learning library in python. In this package, we have a module called tree. And in this module we have a class called decision tree classifier. This class implements the decision tree operator, okay? So, now we need to create a new instance of this class. So, at the end, let's create an object called models nd set it to anew instance of decision, tree classifier. Like this. So now we have a model, next we need to train it so it learns patterns in the data, and that is pretty easy. You call model,.fit this method takes 2 data set. The input set and the output set. So they are capital x, y. Now finally, we need to ask our model to make a prediction, so we can ask it, what is the kind of music a 24 year old male likes? Now before we do that, let's temporarily inspect our initial data set. That is music data. So, look what we got here. As I told you earlier, I've assumed that men between 20 and 25 like hip hop music but here we only have 3 samples for men age 20, 23,and 25. We don't have a sample for a 21 year old male. So if you ask our model to predict the kind of music that a 21 year old male likes, we expect it to say hip hop. Similarly, I've assumed that women between 20 and 25 like dance music, but we don' have a sample for a 22 year old female. So once again if you ask our model to predict the kind of music that a 22 year old woman likes, we expect it to say dance. So, with these assumptions let's go ahead and ask our model to make predictions. So let's remove the last line, and instead we're going to call model.predict. This method takes a 2 dimensional array. So here's the outer array. In this array, each element is an array. So I'm going to pass another array here, and in this array I'm going to pass a new input set. a 21 year old male. So 21, 1 that is like a new record in this table. Okay? So, this is one input set, let's pass another input for a 22 year old female. So here's another array, here, we add 22 comma 0, so we're asking our model to make 2 predictions at the same time. We get the result and store it in a variable called predictions, and finally let's inspect that in our notebook. Run, look what we got, our model is saying that a 21 year old male likes hip hop, and a 22 year old female likes dance music. So our model successfully will make predictions here. But wait a minute, building a model that makes predictions accurately is not always that easy. After we build a model we need to measure it's accuracy. And if it's not accurate enough, we should either fine tune it or build a model using a different algorithm. So next lecture I'm going to show you how to measure the accuracy of a model. In this lecture I'm going to show you how to measure the accuracy of your models. Now in order to do so, first we need to split our data set into two sets. One for training and the other for testing, because right now, we are passing the entire data set, for training the model and we're using 2 samples for making predictions. That is not enough to calculate the accuracy of a model. A general rule of thumb is to annotate 70-80% of our data to training, and another20-30% for testing, then instead of passing only two samples for making predictions, we can pass the data set for testing we'll get the predictions and then compare the predictions with the actual values. In the test set. Based on that, we can calculate the accuracy. That's really easy, all we have to do is import a couple functions and call them in this code. Let me show you. So first on this top from sklearn.model underline selection module we import a function called train test split. With this function we can easily split our data set into two sets. So training and testing, now, right here, after we define x and y, sets, we call this funciton, so train test split. We give it 3 arguments, x, y, ad a keyword argument that specifies the size of our test dataset. So test \_size we set it to 0.2 So we're allocating 20% of our data for testing. Now this functions returns a tuple, so we can unpack it into 4 variables right here. x\_train x\_test y\_train and y \_test. so the first two variables are the input sets for training and testing. And the other are the output sets for training and testing, now, when training our model, instead of passing the entire data set we want to pass only the training data set. So, x\_train, and y\_train. Also, when making predictions instead of passing these two samples, we pass x\_test. So thats the datas set. That contains input values for testing. Now we get the predictions. To calculate the accuracy we simply have to compare these predictions with the actual values we have in the output set for testing. That is very easy. First on the top, we need to import a funciton, so, from sklearn.metric.metrics import accuracy \_score. Now at the end. We call this function, so accuracy score and give it two arguments y\_test. Which contains the expected values and predictions which contains the actual value. Now this functions returns an accuracy score between 0 to 1. So we can store it here, and simply display it on the console. So let's go ahead and run this program. So, the accuracy score is 1 or a 100% but if we run this one more time, we're going to see a different result because every time we split our data set into training a test sets. We'll have different data sets, because this function randomly picks data for training and testing. Let me show you, so put the cursor in the cell, now we can see this is activated note that if you click this button here, it will run the cell and also insert a new cell below this cell. Let me show you. So if I go to this second cell press the skip button, now we are in the command mode, press d twice, okay now it's deleted, if we click the run button, we can see, this code was executed, and now we have a new cell, so if you want to run our cell multiple times every time you have to click this and then run it, and then click again, and run it, it's a little bit tedious. So I'll show you a shortcut. Activate the first cell, and press control and enter this runs the current cell without adding a new cell below it. So, back here, lets run it multiple times okay, now look the accuracy dropped to 0.75, it's still good, so the cuuracey scored here, is somewhere between 75% to a hundred %. But let me show you something, if I change the test size, from 0.2 to 0. 8, so essentially we're using only 20 percent of our data, for training this model. And you're using the other 80% for testing. Now let's see what happens when we run this cell multiple times. SO, control and enter, look, the accuracy immediately dropped to 0.4 One more time, now 46% 40%, 26% it's really, really bad. The reason this is happening is because you are using very little data for training this model. This is one of the key concepts in machine learning, the more data we give to our model and the cleaner the data is, we get the better result, so if you have duplicates, irrelevant data, or incomplete values, our model will learn back patterns in our data. That's why it's really important to clean before training our model. Now let's change this back to 0. 2, run this one more time, okay, now the accuracy is 1, 75%, now we drop to 50% again, the reason this is happening, is because we don't have enough data. Some machine learning problems, require, thousands or even millions of samples, to train model, the more complex a problem is, the more data we need. For example, here we're only dealing with a table of three columns, but if you want to build a model to tell if a picture of a cat or a dog or a horse, or a lion, we will need millions of pictures, the more animal it will support the more pictures we need. In the next lecture, we're going to talk about model persistence. So this is a very basic implementation of building a training a model to make a prediction. Now to simplify things I have removed all the code that you wrote in the last lecture for calculating the accuracy, because in this lecture, you're going to focus on a different topic, So basically we import our data set. Create a model, train it, and then ask it to make predictions, now this piece of code that you see here, is not what we want to run everytime we have a new user or every time we want to make recommendations to an existing user sometimes it's really time consuming. In this example, we're dealing with a very small data set that only has 20 records. But in a real application, you might have a data set with thousands or trillions of examples, training for that might take seconds or minutes or even hours. And that is wy model persistence is important, once in a while, you build and train our model and then we save it to a file. Now, next time we want to make predictions, we simply load the model from the file and ask it to make predictions. That model is already trained you don't need to retrain it, it's like an intelligent person. So let me show you how to do this. it's very very easy. On the top, from sklearn.externals module we import joblib. This job lib object has methods for saving and loading modules. So, after we train our model we'll simply call job lib.dump and give it two arguments. Our model and the name of the file in which we want to store this model. Let's call this music -recommender.job lib That's all we have to do. Now temporarily I'm going to comment out this line, we don't want to make any predictions, we just want to store train model in a file. So let's run this cell with control, and slash, okay, look, in the output we have an array that contains the name of our model file. So this is the return value of the dump method. Now back to our desktop, right next to my notebook we can see our job lib file, this is where our model is stored. It's simply a binary file. Now back to our jupiter notebook. As I told you before in a real application we don't want to train model every time. So let's comment out these few lines, on Mac, we can press command and slash and on windows control slash. Okay, these lines are commented out, now this time, instead of dumping our model, we're going to load it, so we call the load method, we don't have the model, we simply pass the name of our model file. This returns our trained model. Now with this two lines we can simply make predictions, so, earlier, we assumed that men between 20-25 like hip hop music. let's print predictions and see if our model is behaving so control so this is how we persist and load models. Earlier in this section, I told you that decision trees are the easiest to understand. And that's why we started machine learning with decision trees. In this lecture, we're going to export our model in visual format, so you will see how this model makes predictions, that is really, really cool. Let me show you. So once again I've simplified this code so we simply import our data set, create input and output sets, create a model, and train it, that's all we are doing, now I want you to follow along with me, type everything exactly as I show you in this lecture. Don't worry about what everything means we'll come back to it shortly. So on the top, from sklearnimport tree, this object has a method for exporting our decision tree in a graphical format. So after we train our model, let's call tree \_graph vis. Now here are a few arguments we need to pass. The first argument is our model, the second is the name oft he output file. So here we're going to use keyword arguments, because this method takes so many parameters, and we want to selectively pass keyword arguments without worrying about their order. So, the parameter we're going to set it out\_file let's set this to music-recommend.dot. This is the dot format, which is the graph format, which is a graph description language, you will see that shortly. Now the other parameter you want to set is feature \_names. We set this to an array of two strings, age and gender. These are the features or the columns of our data set. So they are properties or features of our data. Okay? The other parameter is class names. So class\_names we should set this to the list of classes or labels we have in our output data set, like hip hop, jazz, classical and so on. So, this y data set includes all the genre or all the classes of our data, but they are repeated a few times in this data set. So, here we call y.unit this returns the unique list of classes, now we should sort this alphabetically. So, we call the sorted function, and pass the result a y.unique. The next parameter is labeled, we set this to a string, all, once again don't worry, about the details of these parameters, we're going to come back to these shortly. So, set label to all, then round it, to true, and finally field to true. So, so this is the end result, now let's run this sound using control and enter, okay, here we have a nw file music recommender.. that's a little bit funny. So we want to open this file with vs code. So drag and drop this into a vs code window. Okay, here's a dot format, it's a textural language for describing graphs Now to visualize this graph we need to install an extension in vs code. So on the left side, click extensions panel and search for dot, dot. Look at this second extension here. graphvis or.language by stephon vs. Go ahead and install this extension and then reload vs code. Once you do that, you can visualize this dot file. So let me close this tab. Alright, look at this... on the right side. Click this, you should have a new menu, open preview to the side. So like that alright, here's the visualization of our decision tree, let's close the dot file, there you go. This is exactly how our model makes predictions. So we have this binary tree, which means every node can have a maximum of two children. On top of each note we have a condition if this condition is true we go to the child node on the left side. Otherwise we go to the child node on the right side. So let's see what's happening here, the first condition is age less then or equal to 30.5. If this condition is false, that means that user is 30 years or older, so the genre of the music that their interested in is classical. So here we're classifying people based on their profile. That is the reason we have the word class here, so a user who is 30 years or older, belongs to the class of classical. Or people who like classical music. Now what if this condition is true. That means that user is younger then 30, so, now we check the gender, if it's less then 0.5, which basically means it equals to 0, then we're dealing with a female. So we go to the child node here, now once again we have another condition, so we are dealing with a female who is younger than 30. Once again we ned to check their age so is the age less then 25.5? If that's the case then that user likes dance music, otherwise they like acoustic music. So this is a decision tree that our model uses to make predictions Now if you're wondering why we have these floating point numbers like 25.5 These are basically the rules that our model generates, based on the patterns in our dataset. As we give our model more data, these rules will change so they are not always the same. So they are not always the same, also the more columns or the more features we have, our decision tree is going to get more complex. We only have two features, age, and gender. Now back to our code let me quickly explain the meaning of all these parameters, we set field to true so each box or each node is filled with a color. We set rounded to true, so they have rounded corners. We set label to all so every node has labels that we can read.