Introduction

We have taken up BSc in programming and data sciences and let us pick the word programming from there and ask this question, what is programming all about. Why do programming? Programming is all about, let us say you go to a place, new country, namely Germany, and to talk to people in Germany you may want to know German, go to France, you got to know French to speak to people in France. You come to India, there are so many languages spoken here, but then you can manage with Hindi and English to a very good extent. So, if you want to talk to a computer you should know a language and just like the languages of our country India, there are umpteen number of languages, there are so many languages in computer science and we chose to teach you Python particularly for a simple reason, that this language called Python is today very widely spoken, number 1. Number 2, it is very easy to learn and number 3, it is very powerful. Today we are in 2021, where programming with time should intuitively become difficult to learn, back in 1970s and today computers have advanced tremendously but if you observe the effort involved in learning a programming language has diminished tremendously in the sense that it has, it is very easy for one to start learning how to code. I would say back then in the 80s or 90s, coding was not very easy, Pascal, COBOL, C, C plus plus, it would take a few weeks if not months to start writing code and write some not so straightforward code would take at least a few weeks of practice but today with Python you can learn programming in less than a few hours according to me. In fact, we will teach you, you will be seeing that you will be coding from the very first minute of this session, I mean as we progress the classes. So, why programming? You need to learn a language to communicate with your computer. And why do you choose Python? We chose Python because Python is powerful, easy to understand and very much spoken. So, coming to the word programming once again, programming we think is fun and you will see that readily as we progress with the classes. Programming is fun because you get to talk to the computer to do complex things really fast. There are three things here you may want to note, one is you talk to the computer, you make a computer understand to do seemingly complex things and really fast, by really fast, I mean, really-really fast. You will see that things that humans do with a lot of effort and time is done in a fraction of second, sometimes even at the speed of sound or even light for that matter. So, we will go very slowly, we will go step by step, we will assume nothing from you, except maybe basic English understanding. We will not assume any computer science knowledge from your end, we will start diving into our first program in our very first session that is coming ahead. Assume you want to learn how to drive a car, what would you do today? You would probably open YouTube videos and start watching on the basics of driving a car or maybe you will rent a book, buy a book and start reading it, or even browse an article on how to drive? As you know none of these will be of great help. So, to learn a car, learn how to drive a car you probably should hold the steering and keep your feet on the pedal and start driving, probably slowly though, but you should start driving. The point is you cannot be reading books and trying to theoretically understand how to drive a car but never drive a car. It does not help. Of course, theoretical knowledge helps to some extent to begin with but then your real learning starts when you hold the steering. On very analogously speaking,

programming also starts when you start touching the keyboard and start coding than keep reading what is programming all about. So, we urge, we push, we want to request you all to start coding alongside with us as we open the terminal and then teach you how to program. The whole of this course will mostly be with the terminal and with me and Omkar, trying to code and tell you what is what and we will take you step by step incrementally and we suggest that you keep your terminal open and code with us. So, during the first week of the course, we are going to keep it bare minimum and very simple. All we will be doing is code not beyond, let us say half a page size, so I will be explaining what I am typing and I will be telling you what the output is and also I will be showing you the output and we will be coding very-very slowly. In fact, if you know programming you may even find it a little repetitive and even maybe boring but then please bear with us, with time you will see the pace pickup. But a word of advice, we would like to see you also open a terminal and code alongside as we are coding.

Introduction to Replit

So, let us dive in and write our first program. Programming was not as easy as it is in 2021, thanks to the internet we do not have to install anything before starting to code. So, what do you see? I have now opened my browser, let me just zoom in just so that it is clear to you all and I am typing Replit.com, R-E-P-L-I-T.com. And then you will see this. The moment you see this click on Start coding and then it will ask for a user name and password, if you have a Gmail ID, just click on this G and log into your Gmail ID and you will be getting a window like, a screen like this. Go to this plus symbol, click on it, what is the language that you would like to code in, Python obviously, click on Python and then here you will get a name that is machine given.

It is generally a combination of words, random combination of words. So, what I will do is I will remove it and then give it a name first code and then create on create R-E-P-L. So, I click on it, this is a repository that gets created. Do not break your head about what is a repository, it is just a place where you can come and code and as you can see, you can in fact, see some examples by clicking here. Let us do that. Here are some example codes, but it is a little confusing to see it on your first day of programming. So, let us try coding something from our hands. Let me say print hello, just hello and once I type something here, you will see things are self-explanatory, I will not explain, come here and click on run and you will see hello here. Now, your question would be what just happened?

You say something here and the same word came here. I mean, what is the big deal? What are we into? So, the point is this is the place where you will type your, you can call it commands, and this is the place where your computer will execute them and show you the result of what you have typed here. Just in case you typed, instead of hello you typed Namaste, and then click on run, it will show you Namaste. If you say Namaste India and then click on run, it will show Namaste India , so the nutshell all you do is you type your commands here and you get the output here. You can type more print commands, this is the

second line and you will indeed get Namaste India in the first line and then this is the second line . You can type whatever you want, put dots here or put some special symbols and you will see them here.

Let us do a small piece of code that is not simply printing some English statements. Let me delete this and simply type one star, see what happens, let me not explain what I am doing, it again should be self-explanatory. So, what am I doing? I type two stars in the second line, three stars in the third line, as I run this I will get this.

We keep doing this assuming I have a lot of time this evening, I can keep typing, this five lines and then sixth line is six stars, seventh line is seven stars. Then run, you will get precisely what you have typed. It is sort of has a staircase effect as you can see, correct? And let me continue this 8, 9 and 10. Looking at the pattern you see, I should have included 9 stars here, but I have included 2 stars more, making it 10 stars.

If I run look at this the computer will obviously not know that it should display the 9 stars here and not 10, simply because you have specified 10 stars here. The point to note is that your computer does precisely what you ask your computer to do. If you ask it to display a hash at the end, it will simply display a hash at the end as you can see. Print I don't know what to type , then run it, it will simply say, I don't know what to type as simple as that.

So, this is a gentle introduction, a very quick introduction to Python and the simplest way to start typing your code and the simplest code is, of course, your print statement. I just now showed you how you can type something here and execute the same by clicking on run command and the executed code appears on the screen right here. This should be enough to begin with, you can go ahead and try exploring whatever you want to explore. Before I conclude let me try to display this stars in the reverse order. What do I mean by that? three stars, let us say space space star, give me a minute, you will understand what I am trying to do, space star star and star star star. What do you think this will display? Exactly the same thing but in the reverse order. Wonderful! Now, what if I want to display four lines? Understand a small issue here.

I cannot simply put four here. I should put one space extra here, one extra space here and one extra space here, do you see, do you see that I clicked on run and I got the output, a beautiful staircase effect, but in the staircase effect, but in the reverse direction. But do you observe something? It is not as easy as the previous program where you simply typed one star in the first line, two in the second line, three in the third line, fourth in the fourth line, it is a little more involved than that.

To make your life even more difficult, rather I am making my life difficult here, not yours. I am sure when you type start typing your piece of code your life will become difficult as well. You see for five I should again go and push these things by one space run and here I am. And assuming I did that for about 10 lines and the 11th line will involve me going up and then

putting space to all the lines above, oh, it is going to be some hard work if you are going to do this for like 20 lines. I stop here with a question - Is there any way we can automate this? Of course, we are a couple of weeks ahead with this question, rather I am asking this question couple of weeks ahead, but let us wait and watch if there is a nice way to display this by coding something really nice here, in just three to four lines, so let us wait for the forthcoming weeks where we take you on our journey of automating things using programming.

More on Replit, print and Common Mistakes

Hello Python students. In this lecture we will see some more features of Replit and print command. Also we will see some common mistakes one may do while writing a Python program. Let us start with the different features of Replit. If you observe the left side panel, you will see various icons available over here. Each icon represent a different feature of Replit. Let us first focus on this first icon, which is files . Inside files you can see there are two different options at the top called add file and add folder . These two options will allow you to manage your programs in more systematic way. For example, we can add a new file. Let us name it new program dot py . You can always go back to the previous file or you can write in your newly created file. Similarly, we can also create one folder. Let us call it week 1.

Also we can organize these files by simply dragging them inside week 1 folder. So, inside week 1, we have new program dot py as you can see over here at the top. Similarly, we can keep adding such files in a folder or outside the folder. This particular feature of Replit allows us to create multiple programs and organize them in a systematic manner.

Next it has many features like version control, packages, home secrets, settings, databases, unit tests. But for now we will explore only one part of it, which is settings . So, if you go to settings the first is layout, the default layout is side by side, but we can always change it to stacked. It will bring the place where we write the program at the top whereas the console where we see the output at the bottom. Also it has two different themes, which is light or it can be dark. Similarly, we can change the font size and so on. There is one more interesting feature which you all must explore, which is this last option code intelligence . Let us see the difference between disabled and enabled code intelligence feature. Let me type print command currently it is not showing any information regarding this particular command.

Let us enable this code intelligence feature and retype this print command. If you observe, we are getting various different options and it also provides the detailed information of this particular print command. We will explore all these options in detail as we go on, but for now you can read this information, which will help you to understand this particular command. These many features of Replit are sufficient for now. Let us move to Python program. If you remember in last lecture you have studied a print command where you print something like this hello India and executed the program. Now, I have one question for all of

you - Is it necessary to print only one string at a time? What if I want to print hello India as well as hello world in a single print statement? Can we do that? Definitely, it is possible, we can write two different strings or two different messages in the same print command separated by comma, hello world . Let us execute this, as you can see it is possible to print different messages using a simple print statement separated by comma. Similarly, we can add one more command as well, hi Python students . As you can see it is printing all three messages one after the another in that same order, which means the order is very important. The first message gets printed first, the second message appears next to it and then the third message and so on. So, print is pretty flexible with respect to this where we can print one message at a time or it also allows us to print multiple messages. Let us ask one more question - Is it necessary to print only messages over here or can we print some numbers as well? What if I write something like print 10, what will happen if I write print 10? It prints the number 10 as it is.

Which indicates that it is not necessary that we only have to type messages using letters from a to z, we can also print numbers using print. Now, you all must be thinking is it necessary to print only whole numbers or can we print fractional numbers as well. Let us try it 20.5. Let us execute, yes, we can we can print strings, we can print whole numbers or we can print fractional numbers using print. Now the next obvious question arises is can we combine all these three things together? Let us try that, let us replace this with 10 and this with 20.5. Let us remove this. Let us try to execute, yes, we can hello india , 10, 20.5. So, we can conclude that print command can be used to print multiple messages or multiple values at the same time. Also it allows us to print string whole numbers as well as fractional numbers together.

Now, let us look into the second part of this particular lecture, which is some common mistakes which you may do while writing the python program even with as simple statement as print. Let us look into those common mistakes. Print, first the spelling of word print might be wrong, which is kind of obvious, but the next part is more critical, you may think all brackets are same, but when it comes to programming each bracket has a specific meaning.

Usually after print, we use a round bracket. Now, is it necessary to use round bracket or can we use some other type of a bracket. Let us try it square bracket, hi, it shows an error message, which means square bracket is not allowed. Let us try a curly bracket, still an error message. Let us try angular brackets, again error message, which means in Python, along with print we must use round brackets.

We always type whatever message we want to print in these single quotes. Is it necessary to have these quotes? Let us try, error message, which means the quotes are required. Now, the next question is is it necessary to have single quotes or can we use double quotes as well, like this? What do you think whether the program will execute or it will not?

Let us try, it executes, which means Python is very strict when it comes to brackets, but it is very flexible when it comes to these quotes. We can use double quotes or we can use single quotes both works correctly, only requirement is if you open double quotes, then you must close with double quotes and if you open single quotes then you must close with single quotes. A combination of single and double quotes will not work. All these various constraints of a language is generally referred as syntax, which defines how a particular code has to be written in a specific language. Thank you for watching this lecture. Happy learning!

A Quick Introduction to Variables

Let us get back to our print command and type this statement, as you can see I say print 10, when I execute this I see that 10 is being printed, displayed. Now, look at the small modification that I do. I remove this 10 here, just above the print statement, I say a equals 10, and then I include print a and what does this do? In mathematics we have observed this, we always declare a variable, we say let us call a variable; let us assign the value 10 to it.

A computer also sees this as the letter a is assigned the value 10 and you are printing a, which means whatever is assigned to this variable, will be printed here, as simple as that. So, when I execute this I continue to get 10, here as you can see. Now, if I were to say b equals 20 and then when I say print b, let me execute this.

I get print a, a gets printed here, print b, b gets printed here, which is 20. So, that is it, pretty self-explanatory. What is the big deal about it? Now, look at this. I will say print a plus b, so here is a plus b and then I say print a into b, I get a into b, which is 200, 10 times 200. Now, let me delete all these things start afresh. I will say a equals 10 as before and say print a, look at this, this is slightly counter intuitive. I say a equals a plus 1. In mathematics what does it mean? This means you can cancel a and a and 0 becomes equal to 1, which means this does not make sense in mathematics or the conventional way in which we have learnt algebra, this statement really does not make sense, but in computer science this is the most frequently used programming statement. This simply means for the value of a, you take the existing value of a and add 1 to it, as simple as that, the best way to see how something works is to execute the program and then see it, so I have typed this program, it will of course, print the value of a here and then I do something here and then print, let see what that something results in. It tells you 11, which means the value of a right now is 11. Initially it was 10, now it is 11. Now, if you print a once more, it will continue to stay 11, you see a is 11. Print a once again it says 11. So, let us do a small modification to this, in between can you guess what is going to happen if I do this, you have guessed it right, it is a very simple program which displays 10 increments and then displace increments is adding 1 to it, that is called incrementing in computer science.

So, a equals a plus 1 and then even in real life incrementing means going one step up,

correct? So, you have 10, 11, 12 and 13, pretty simple. These are called variables where you instead of typing directly here as 10, you could do this, you could also do, a equals 10 and then say print a, as simple as that. So, now let me try writing a small piece of code as and always, I will write the code and I will let you guess what this code is doing. I say print enter a number what will this display as I execute, it just asks print enter a number , you can see the output this side you get enter a number this side and that is it empty nothing else, so what you should do is you should let your user, I mean, you enter a number here.

How is that done? Just concentrate, it is a very quick command, it goes like this, that is, you may have to remember this, what exactly this does, we will discuss later, but then this will do the required magic look, the cursor is waiting for you to enter a number. When you enter some number here, and then press enter, the program ends.

Now, what I will do here is I will say print n here . Can you guess what this does? Firstly, it says enter a number and then waits for this line to happen, this line in Python stands for please ask the person to input something, convert that to an integer and then assign it to n . When I say 10, it assigns the value 10 to n and then it prints n. Let me say it prints n plus 1, and then prints n plus 2 and print n plus 3 and so on. I execute this remember it says; let me say 100, as you can see it displays 100, 101, 102, and 103. Exactly what is happening here? You may be a tad bit confused on what is this int here. I will come there soon. It will be very clear to you in a minute s time.

Variables and Input Statement

Try observing this program. I say print hello, type in your name . Remember in the previous case I said type a number , enter a number and now I am saying type in your name . What exactly was the previous code? I said n equals int input and so on. That was because it was a number. We told the computer that the number that is being input the, whatever the user is trying to input it is actually an integer.

If you were to type in your name it will be a string. So, you should put str and then input and then go to the next line, this may sound a little confusing, I must warn you, but do not worry as we keep using it more and more, it will get very, you will get very used to it. And then I will say print n and so let us execute this and see what happens.

Hello, type in your name, Sudarshan and then you see the first one was what I typed, second one was what it displayed because of the print statement here. I can go ahead and say hello comma n, to execute this type Sudarshan, it says hello Sudharshan . You can of course, continue and say and it will indeed say, hello, how are you? Look at this Sudarshan, hello Sudarshan how are you? Wonderful! So, you are slowly getting a hang of what is programming and how we can, so what is unique about this program? What is unique about this program is that whatever you enter, it says hello to that string that is being entered. And I am trying to make it really interactive although this may sound really dumb, what is so

special about a program saying, hello Sudarshan how are you. But then you can think of the complexities that one can add to this. I will just try to include just one complexity to this. I will say, type in your name, and take a string, then you say which place are you in? delete this, which means I am supposed to enter the place here, it will again be the same p equals str input. Please note, why did I say p? I say p here because you have n here, which stands for your name, if you put n here assume you put n here, then the name that you typed in which gets stored in n, will be lost, because this is the bucket, in which you stored your name and then your place also is being stored there. You probably are wondering why cannot it store both? That is not how the computer reads it. So, without any further explanation, let us see what exactly this does? Hello type in your name, Sudarshan, which place are you in? I am now in Mysore. It does not display anything simply because I have not displayed anything here. So, look at what I would want this program to tell me, it should say hello, name, and then how is the weather in p?

I make it one level more interactive here if you compare this with the previous code. Let me execute it, it says, hello, type in your name . I say Sudarshan, it says, which place are you in, I say Mysore, there seems to be a problem somewhere. Let us figure out what is the problem? p is not defined, you see I made a mistake here, n should be p, without declaring something if you simply use it this is the kind of error that you will encounter. In fact, we have not edited any of these errors that we get while we are coding that is because the mistakes that we make you may also make, so we do not want to appear very perfect, so any mistake that happens here will be corrected here and the video will not be edited, you will be seeing why the error came and you will be careful enough to reverse the error in case you encounter a similar error.

Now let us go ahead redo this, Sudarshan is my name, what place are you in, Mysore, it says Hello Sudarshan, how is the weather in Mysore? Perfect, very nice! Now, if I were to rerun this program and instead of my name I type Amitabh, as an Amitabh Bachchan, which place is Amitabh right now? I am hoping he is not, given the Covid situation he probably is at home. So, let me type what am I typing, let me type Mumbai. Hello Amitabh, how is the weather in Mumbai. So, it is not grammatically correct, how is the weather in p, I should probably put a question mark after this, so hello, type in your name, I am the, let us say it is the Prime Minister of India. Which place are you in? New Delhi. Hello, PM, how is the weather in New Delhi?

So, you can make it even more complex here. You can make it display the right way, you see there is a small problem here, there is a space in between New Delhi and question mark, you probably are wondering how to remove that space, you can think of using something called format specifier, which is coming next. But I am just giving it as a puzzle try to look up online how this can be removed and then try to remove it. Most of these things are available on Google, just type it and you will have a question on some forum, which addresses exactly this question and you get an answer as well. That is with variables, we will go ahead and study more variables right now. You understood that n was a string, p was a string, I will go

ahead and say what is your age? You see this will be an integer. Let me say age equals integer input.

We execute this. Hello, type in your name, Sudarshan, how is the weather in, which place are you in, Mysore, hello, how is the weather in Mysore, by the way what is your age? I will say my age is 90. That is it. It is not saying anything. So, I will say good to know you are, age, years old. So, you can guess what is going to happen now.

Sudarshan, which place are you in, Mysore, what is your age, let me call myself a teenager and feel good about it, I say 15. Good to know you are 15 years old, says the computer. Point to note here, you can use print statement and include variables there instead of telling you what exactly is variables, explaining it in detail, I simply used it in a program and made you feel comfortable about how variables are being used.

Not only did I use the variable here, I used the variable and assigned a value to it by using a, I mean, making use of the input statement, whenever you say input, it will ask for the input, it will convert that to, the type that you have specified, int type and then assign that to the variable.

Variables and Literals

Hello Python students. In this lecture we will see variables and literals. In order to demonstrate these concepts, we will use a Python code which is familiar to you. This is a similar code which we saw in the previous lecture. Type your name, Sudarshan, type your location, Mysore, then we will print message, Hello Sudarshan, how is the weather in Mysore? Then the next question is what is your age. Let us say 40. Good to know that you are 40 years old. Before moving to variables and literals let us see is it necessary to have print and input as two separate commands or can we merge them together? To answer that question first we have to understand what both these commands are doing. A print command is only printing a message, whereas this particular input command is taking the input from the user and storing it in n. This printing a message and taking a input can be combined together. The same message we can place inside the brackets of input command and we can remove this print. Let us execute, type your name, Sudarshan, type your location, Mysore, Hello, Sudarshan, how is the weather in Mysore. What is your age? You say 40. Good to know that you are 40 years old. Still we are getting the same output. This is how we can merge print and input together where we are displaying the message along with the input statement itself. Now the next question is is it necessary to have the name as Sudarshan, location as Mysore and age as 40 or can we change the values. Let us try it. What is your name? Omkar. Type your location, Pune. Hello, Omkar, how is the weather in Pune? That works. Let us see whether next statement works. What is your age? Let us say 30. Good to know that you are 30 years old. Which means the same program executes perfectly with same messages even though the values are changed. Now that brings the next question How that happens? There is something called as variable, for example, this n or this p or this age,

all these three are variables; which means their values can be changed, they are nothing but a container where we can store different values. Let us try to understand it with a different example. Let us say name is a variable. Now as per first program the name was Sudarshan, but now as I have mentioned name is just a variable, we can change the value which is stored in variable name from Sudarshan to Omkar. The name time we can also change it back to Sudarshan, which means variable we have used over here can store a different value as we go on. Similarly, age can be 40, it can be changed to 30 and so on. So, in this case the name or the age are referred as variables whereas these names Sudarshan, Omkar or the values 40, 30 are referred as literals. Literals are the actual values which are stored inside a variable. Variable can store different literal values and they can be modified as per the requirement. The same age can be modified like this, age is equal to age plus 1. This particular thing is possible but if we try to execute something like this, then it will not work. Because as per this particular line number 6, what we are saying, take the latest value of age which is 30, add 1 into it and store the updated values in variable age. Which means at this particular point the value is 30, then we did 30 plus 1, and now the updated value is 31. But if you come here we cannot say 30 is equal to 30 plus 1. Hence, you can conclude the literals can be used only on the right hand side of the equal to sign, whereas variable can be used on either sides of the symbol equal to. This particular difference between variable and literal leads us to a new question when to use variables and when to use literals? Let us write one Python program which will demonstrate the use of variables and literals. You observe and tell me what happens in this particular code. Let us see what we have done over here, r is equal to int of input of enter the radius of the circle, area is equal to 3.14 multiplied by r multiplied by r. Print area of the circle with radius r is area. First let us execute this particular program, then we will try to answer our previous question. Enter the radius of the circle, let us say 5, area of the circle with radius 5 is 78.5, we are getting the output but still the question is when to use variable and when to use literal. The answer of this question is in this second line over here. The answer is we use r as a variable, area as a variable but the value of pi as literal, because value of pi will never change, whereas radius of a circle might change, and with respect to that area will also change, which means we can execute this program again. And instead of 5 now we can enter 15, which will give us a new area value which is 706.5, irrespective of the radius of the area the value of pi is always 3.14. We will use variables only when there is a possibility that the value which is being stored make change and when we are sure that the value is not going to change we will literal. Similarly, in previous program we used name as a variable, age as variable because those two values may change from person to person, but over here value for pi will always remain same. Thank you for watching this lecture. Happy learning!

Data Types 1

Let us say we declare n equals 10 a variable n and then print the value of n. The output will be as expected as you see the number 10. Assuming I take another number, let us say r equals 6.3 and then print r, obviously it will print first 10 and then will print r. Now, the point is what if I said s equals, please note sudarshan, and then I will print s as well. So, you

see what is happening here?

Here is a number without any decimal value, here is a number with a decimal part. So, it is not just an integer, it is a decimal, it is a fraction, 6.3, and then here this is neither an integer nor is it a decimal number like this. This is a string, a name. I say print s and sudarshan gets printed. Now the point is what we may want to note here is let me explain that with an illustration.

When I say print type of n, you will see what this type does, you see it says class int, do not worry about the, I would say it is a very ugly looking output, so with less than symbol, greater then symbol, the word class and things like that, but all that it says is it is an integer type number in n. What if we said print type r, executed, it says class float and then, so let me just write this for clarity.

I will say the type, n is of type, let us say n is of type, then I say type r, make sense, r is of type r. So, mind you the letter n here is not a variability, it is just, I am asking the computer to print the whatever is inside the code, this n is not the value 10 here, this is simply the letter n, which for reference let me just run this and show you, it will say n is of type class and r is of type class float.

And then I say print s is of type, type s and I will execute this, you see what happens, s is of type str, we have seen this somewhere before. So, str is string. What exactly does it mean? What is int here, what is float here, what is str here? You never specified that anywhere here, you went ahead and said n equals 10, r equals 6.3, s equals string sudarshan.

But the computer automatically called this as int type, integer, called this float, the word float simply means something that is more than an integer, 10.171, now that is a floating point, it is called a floating point number. So, in a first course in computing you will study all these things, I am sure you did if not you can always look up, it is a very easy thing to understand. Floating point representation of a number is the way a computer stores a number that is not an integer, it stores in a very, it is a little too technical to get in right now, but just assume from my side that this is not an integer and if it is not an integer, if it is beyond an integer it is called a float for a computer, it is automatically calling it a float and this is called a string. A computer automatically recognizes the type of data that is being stored.

Why would anyone want something like this? The answer is very simple if you were to create a vessel where you will put your, let us say rice, look at the shape of the vessel that you use, a jar rather, what kind of a jar do you use for rice, what kind of a jar do you use for, let us say to store some liquid, they appear differently. It is not just for the showy, aesthetics reasons. It is it is simply because the container that holds rice may have to ensure that it is free from moisture whereas a container that that stores water you need not to worry about moisture. So, in a very-very similar way the kind of data that you store, your computer does

some work internally in allocating some memory location for it to be stored.

Now it cannot simply keep the same location for all types of data that you may want to store, be it string or be it floating point or be it integer. It is not the same mantra for everything. The computer recognizes what kind of data you are storing and it declares the data type, this is called data type and it stores the values there and says this is the type integer, I will store it in this way only, details aside with time you will understand it, but given that you are new to programming all that you need to know is Python has different data types.

So, with this I will end with a small data type which will be discussed in detail later though, but right now I will give you a quick glimpse of what this data type is all about. Let me refresh the screen. So, what if I said what if I said, remove this, l is equal to let us say 10 comma 20 comma 30. Please note the format here, I give a variable name here and say is equal to start a big bracket and put some numbers separated by comma and close the bracket that is it and I say print l. You can guess the obvious, you will see this same values being printed here. And the best part is when you say l of 0 it displays 10, if you say print l of 1, 0 and 1, it displays the 0th value which is 10, the first value that is 20. And let us say print of l of 2 will be 30.

Now, what are we trying to say here? A computer always starts naming, assume you go to a classroom and you start counting people, you start with here, this fellow is the first person, he is the second person, third person, fourth person, we humans start with 1, 2, 3, but a computer generally starts with 0, 1, 2. The reason is a little technical we will get there sometime, but then as of now you please assume that a computer always starts with 0, does not start with 1. The first element here is called the zeroth element, then the first element, and then the second element and so on. So, if I were to have more numbers here, let us say 68, 720, 732, you can put whatever you want, and then you say print of l of 3, you will get 10, 20, 30. And then the l of 3, which is the fourth element here, which is 68 and so on, this is called a list.

So, let us see print type of l. What kind of data is this? Is this the jar that contains rice or is this a jar that contains a juice or a jar that contains sugar? The computer stores this as what? Class list. It calls it a list. It has a very different mechanism, very different way in which it stores, but that is not important for you right now. All you need to know is Python has different data types. By data types we mean it has a knack to store what kind of data in what form.

Let us revise whatever we discussed so far, if you say n equals 10 this becomes an integer, if you say r equals some 6.9, it becomes float, as I said, let us try printing these things, print n, print r and then we discussed another thing, the last thing was a string, s equals let us say India and when we say print s, this prints the values, the first one was type, but what we wanted was type, so I will change this to type here, type, and then let us say type. And you know the obvious will happen. It is throwing an error simply because I did not close the

brackets. Let us try closing the brackets, now it says the first one is a list, second is an int, third is a float, fourth is a string. So, let me try showing you something and leave it as an exercise for you all to figure out what is happening. What will happen if I were to type, type of l of 2? So, let us say print l of 2 and there is a print type l of 2, and I delete the rest, I remove this also. Guess what will happen if I execute this. I will execute and show you and I leave it to you to realize what is happening here. So, with this we end this topic of data types. I hope it is very clear. Do not break your head much. All that you need to know is computer has a different way to handle different types of data that you may want to use in your programming show. Thank you!

Data Types 2

Hello Python students. In last lecture we saw something called as data type. Each data element has a specific data type, which represents which category of data that particular value belongs to. For example, 10 over here is of type integer, 5.6 is of float, whereas value India is of type string. A data type of any variable can be checked using a command called type. Let us print type of i, type of f, and type of s. As expected it prints int, float and string. Now in this particular we will introduce one more data type called Boolean. Let us add one more variable B1. Now this particular data type has only 2 different values, one it is True, other is False. Let us say B1 is equal to True and B2 is equal to False. Let us print B1 and B2. Let us execute this particular program. As expected the data type for variables B1 and B2 is Boolean. This is a different kind of data type where any variable created using this particular data type can hold only two different values, which is True or False. And notice carefully, the letter T in True and letter F in False has to be capital otherwise it will not be considered as Boolean. Now as we have studied four different data types let us try something different which is data conversion from one data type to other. First let us start with integers. Let us look at this particular code, value 5.7 is of type float, but we are explicitly telling the computer to convert this 5.7 to integer and then store it in variable a. Similarly, over here, 10 is a integer but it is currently enclosed in single quotes, which means it is string and we are asking computer to convert this string into integer and then store it in variable b. Computer will convert 5.7 into integer, string 10 into an integer and ultimately variables a and b should be integers. Let us print the values of a and b, also the type of a and type of b, and let us see how it works. As you can see value of variable a is 5 and type is integer, whereas value for variable b is 10 and type is integer. In the case of conversion from float to integer, computer will only take 5 and store it in a, whereas this 0.7 part will be ignored. This is how we can convert a float or a string into an integer. Now, equal to let us look at similar type of conversion from integer and string to float. Let us look at this particular float, where 9 is integer and 5.3 is a string, but we are telling the computer to convert both these values from integer to float and then from string to float. Let us execute the code. As expected the original value 9 got converted to 9.0 and type if float, a string representation of 5.3 got converted to its float representation, which is again 5.3, but the type had changed to float. This is how we can convert values from integer or from string to its equivalent float representation. Now, let us see how to convert values of type integer and float to its

equivalent string representation. Once again this 9 is a integer and 5.3 is a float but we are telling the computer to convert this 9 to a string, also 5.3 to a string. Let us execute the code. As expected we got the output as 9 and 5.3, but the type has been converted from integer to string and then from float to string. This data type conversion is generally referred as type conversion or type casting as in conversion from one data type to other data type. If you have noticed so far we are trying convert values between integers, floats, and strings. But we have not discussed any type conversion related to Boolean data type. Let us see type conversion can be done from integers, float and strings to Boolean data type. Let us look at this particular code where we are trying to convert three different integer values to its equivalent Boolean values, Boolean of 10 and 0 minus 10. Let us first execute this code and then we will see how particular output is coming. Class or type of variables a, b, and c is Boolean but if you observe the values then you will notice there is some difference. It says True, False and True. Value for variable b is coming out to be False, whereas values for variable a and c are True. This is happening because whenever computer converts an integer to a Boolean, all values except 0 are consider as True whereas 0 is the only value which will give us Boolean representation which is False. That is the reason 10 as well as minus 10 are getting converted to Boolean value True, whereas 0 is converted to Boolean value False. Now, let us try the similar thing with a floating point value. Now let us look at this particular code where we are converting values from float data type to Boolean data type. Let us execute, once again a data type has been converted from float to Boolean in all three cases whereas values for variables a, and c are True but for variable b it is still False, because 0.0 is nothing but 0 only. And as we have mentioned earlier Boolean representation of 0 is always False. Other than that for all other positive as well as negative numbers irrespective of integer or float, we will get a Boolean representation as True. Now, let us see what happens when we try to convert string to Boolean. Let us look at this particular code block. India, 10, minus 10.4 as well as 0 all are strings. We are trying to convert all these string values to its equivalent Boolean representation. Let us execute the code and see what kind of output we are getting. All data types are Booleans and all values are True. Now even in the case of 0 it is given value True, because in this case 0 is neither an integer nor a float, here 0 is enclosed in this single quotes which makes it a string and string representation of Boolean is always True except for one condition. Let us see what that condition is. e is equal to Boolean of empty string, here we are simply giving single quotes and inside those single quotes we are not writing anything, which means it is an empty string and a Boolean conversion of empty string is False. As you can see over here in last line of the output, the class is Boolean but the value is False. All strings are converted to Boolean with value True except an empty string. So, in order to summarize this particular lecture, we saw one more new data type called Boolean and also we studied type conversion within integer, float and string and at the end we converted all different types of integers, floats and strings to its equivalent Boolean representation. Thank you for watching this lecture. Happy learning!

Operators and Expressions 1

So as anyone would expect it is only obvious that when you say n equals 3 plus 2 and then

print n, it prints 5. If you were to say 3 times 2, it prints 6. Then when I say 3 times 2.6, it prints the actual number 7.8. Fine, perfect. So, what about let us say a equals 1, b equals 2 and then I print n equals a plus b. This again behaves naturally as we expect.

If we include floating point numbers again behaves as we expect, but then if you were to simply say a equals sudarshan, b equals India and say n equals a times b, let us see what gets stored in it. It throws an error obviously because it does not know how to multiply these two strings, that is when we may want to recollect our discussion on data types.

This is of the type string, this is another type string, it does not understand, your computer does not understand what is into of two strings, but if it is a number it readily does the multiplication or let us say even division for that matter, correct? But then if I say a plus b, what is your guess? Do you think it will throw error?

Let us check, not at all. What it does? It simply mixes these two strings, mixes as in puts it one next to the other, this is called concatenation in English and more so in computer science this is very often used, they call it concatenate, put two words together, one next to the other. So, whenever you say plus of two strings, the computer seems to understand that.

Now you can ask me this question why is it that the computer understands plus and not minus or subtraction, I mean, subtraction or multiplication? It is programmed that way, it is for convenience that they use plus for concatenation of two strings.

So, let us go ahead and do something more here, let us say a equals 1, 2, 3, let us explore, what if b becomes 7, 9 and 15, what will a plus b be? a plus b simply gets, this is called union in mathematics. What is union? Union is simply, you take this list and this list and put them together. In fact, it is not even union, it is simply put them together, make them, make it together. For example, if you had another 2 here, it will include that 2 too here as you can see. Union means there should not be a repetition, that is the idea of sets that you would have studied in your high school days, anyway let us not get there, it is not required here. All that this does is simply puts them one next to the other and creates a new list. So, now you understand a data type matters when we say an operator like plus, when we put what is the data type of a and b matters for the computer to give the right output? So, with this let us go ahead to the next idea, which is let us say a equals 11, b equals 15, I say n equals a by b, I say print n. Again as expected it prints the floating point, the float number. Float is something that is not integer as I keep saying, so it is 0.7333. Fine, so far so good. Now, what if we were to simply say n equals, let us say 10 plus 13, again does the obvious here, but then what if I said 10 plus 13 times 2. What is it that you would expect here? I would expect this to be 10 plus 13, let us say it goes from left to right, first it does 10 plus 13, this becomes 23, and then multiplies 23 into 2, which is 46, this is my guess. You comment using hash here, my guess is n will be how much, 10 plus 13 is 23 times 2 is 46. Let us see if this is the answer. No, this is not the answer, the answer seems 36. Why? That is because your computer does not do it the way you expect it to do, it does the way it was programmed to do. Internally it was told

to calculate such expressions, arithmetic expressions in a particular way.

And that particular way is that whenever you have a plus and a multiplication, give priority to multiplication, which means come immediately to the place where there is into, the star and multiply these two numbers, you get 26, and then add 10 to it, you get 36, and that is precisely what is being displayed here, not the typical first guess that I did, 10 plus 13 is 23 times 2 is 46.

No, it first comes to into and then does plus. This concept is called operator precedence, let me just comment that too, the expected answer was incorrect, the correct answer turns out to be 36 that is due to what is called the operator precedence. It is a very complicated term for something as simple as some operators, by operator we mean into, plus they all are called operators. Some operators are precedence over the other. They are executed first over the other. This is an important concept, in all programming classes they teach this, although in my personal opinion this is not very important. For a simple reason that we can always do what it takes for us to simply put braces, brackets and then tell what you want. Here the moment you put your brackets, 10 plus 13 will get executed first, 23, into 2 is 46, it has to be that.

Now the computer cannot do into first and then do plus next. It did that when no brackets were there like this, but the moment you put brackets the computer understands what you expect it to do, it will indeed show 46 as you can see here. So, now you understand what are operators, your plus, your star, your minus, your division, slash, all of them are called operators and when you put them in tandem, when you put them together the way we did before, how was that.

Let me put that, when you put them continuously like this and confuse the computer, the computer indeed does not get confused, it does what is called operator precedence rule with which it first does the multiplication and then does the addition. As I repeat do not break your head so much about this concept, it is just that you need to know what happens when you give an arithmetic expression like this. What is more important is trying to code and get the output that is expected, rest are all just details. So, let us see more about these operators and operator precedence and arithmetic expressions in our next video.

Operators and Expressions 2

Introduction Hello Python students. In last lecture we saw operators and expressions. We will continue from there and see some more different types of operators. In Python operators are divided into three major categories, first, arithmetic operators; second, relational operators and third, logical operators. Let us start with arithmetic operators. As you can see Arithmetic Operators we have these many arithmetic operators in Python.

Addition, subtraction, multiplication, division, these four operators are familiar to all of us.

First, let us see these four operators and then we will move to remaining three operators. Addition 2 plus 3, subtraction 9, so on, multiplication 5 multiplied 4, division 7 divide by 3. Let us execute the code as expected we are getting the output as 5, 8, 20, and 2.333 for the division. Now we will focus on remaining three operators.

Out of that the first one is called floor division. This particular operator looks similar to division operator, but instead of one forward slash, we have two forward slash in floor division operator. Let us first execute the code, then we will see what is the difference between division and floor division.

As you can see over here the output for division operation was 2.333, which is a fractional value whereas for floor division it is an integer value and that is the difference between division operator and floor division operator. Whenever we divide 7 by 3, we get some fractional value, but when we divide 7 by 3 using a floor division operator it provides us only the integer part of that division, which is 2 in this case.

The next operator is called modulus operator. This operator is also related to division operation but it is different than division or floor division. Let us see how it works? This particular operator is represented using this percent symbol, 7 modulo 3. Let us execute first, output is 1. We are getting this output as 1 because whenever we divide 7 by 3, the remainder is 1, division operator gives us the output in fractional value.

Floor division divides the numbers and provides only the integer quotient part of it whereas modulus operator provides the remainder after the division. Now let us move to last operator which is exponential operator which is represented using this star star symbol. Exponential 6 raised to 2. As expected the output is 36, that is because 6 raised to 2 as in 6 power 2 is 36. All these different set of operators which you can see on your screen are called as arithmetic operators.

Relational Operators Now let us move to second set of operators. These operators are called relational operators, greater than, less than, greater than equal to, less than equal to, double equal to, and not equal to. If you remember you have studied all these operators in your previous course Computational Thinking. Let us go through these operators one by one to demonstrate how they are used in Python. Let us say print 5 greater than 10, output is false, because 5 is not greater than 10, if we invert these two values and make it 10 greater than 5, then the output should be true, as you can see this greater than operator is only giving output in terms of Boolean value which can be either false or true. Let us execute the same code by changing the operator from greater than to less than, now 5 is less than 10.

So, this statement should be true whereas 10 is not less than 5, so this should be false. Let us see, true and false, which is what was expected. Now, the next operator is greater than equal to, now let us see what is the difference between greater than and greater than equal to. Let us make it 5 greater than 5 and 5 greater than equal to 5.

Let us execute and see the difference. In first case 5 is not greater than 5 because they are equal, that is why the output is false whereas in second case 5 is greater than equal to 5, in this case the greater than part may not be true but this equal part is true because of which output we are getting as true. Similar is the case for less than equal to symbol. Let us try it less than less than equal to. Let us execute, same output, because once again 5 is not less than 5, but 5 is less than equal to 5. Once again this equal to part is true that is why we are getting output as false, followed by true. Next symbol is double equal to, let us use that, 5 double equal to 50, 5 double equal to 5, let us execute and see the difference. Now the output which we are getting is false followed by true. The first we are getting false because 5 is not equal to 50. The number 5 is not equal to the number 50.

At the same time the number 5 is equal to number 5 that is why we are getting true in second case and false in first case. This double equal to operator compares first operand with second operand and if they are equal, then it prints the value true and if they are not it will print the value false. The last operator is not equal to operator, which is exactly opposite of this double equal to operator. Over here if we use not equal to 50 and not equal to 5, then let us see what happens to the output. Now we are getting true when we are saying 5 not equal to 50, which means this not equal to operator will compare operand one which is 5 with operand two which is 50, and if they are not equal then it will print true, whereas if they are equal then it will print false. As you must have noticed with respect to relational operators the output is always Boolean value, which means output of any expression using relational operator is always going to be a Boolean value.

Next set of operators are called as logical operators. These are the logical operators and, or Logical Operators and not. Once again we have seen these operators before, but let us explore them using Python. First let us see the and operator print true and true, true and false, false and true, false and false. Over here we are using this and operator with all four possible combinations of true and false. Let us check the output first.

True, false, false, false, we are getting this output because true and true evaluates to be true, in all other cases the output is false, based on this we can conclude that when the and operator gets both its operand as true then and then only it gives the output as true. In all other cases the output is false. Now, let us replace the and operator with or and see the difference in the output.

Let us execute first true, true, true, and last is false. Now, in case of or operator expects at least one of its operand to be true, if that is the case the output is true otherwise the output will be false. Hence, in first case both operands are true, so output is true, the first operand is true, in third case the second operand is true, that is why in both these cases as well the output is true, whereas for last case both the operands are false as we cannot find even a single true on either side of our operator the output is false.

Now, let us see the last operator which is not operator. Let us execute this code, output is false followed by true, not of true is false, whereas not of false is true, which means this not operator simply inverts the Boolean value from true to false and from false to true. But you must have observed one difference in these two print statements.

In first print statement I have used not followed by a bracket inside which I have used a Boolean value true whereas in second print statement I have used not operators followed by false directly without using any parentheses, still we are getting the expected output which means the not operator can be used either with brackets or without brackets, still we get the expected output. Thank you for watching this lecture. Happy learning!

Introduction to Strings

We did see more on strings, but let us revisit strings now and start with declaring a variable s as let us say coffee and then t will be, what goes good with coffee, maybe cake or better even bread, I want something long word. So, you all know what happens when I print s, will be printing coffee, when I print t I will be printing bread, correct, coffee and bread and as we discussed when we print s plus t it will concatenate, that is probably a new word for you all.

It will concatenate, which means it will place one next to the other and display coffee bread.Right,fine now let me take you all to one level deep into strings, so the next level would be to understand what will happen if I were to simply say s of 0, even this we discussed, it will show you the first letter of s, s of 1 second letter of s, if I were to say s of 2, third letter of s and so on. But then what I did not tell you people is that if you specify s of 1 here and then 3 here it does something else.

Let us see what it does? It is showing 'of'. What exactly is it doing? It is taking, it is starting from the first number 1 s of 1, it is starting and going up to 2, it does not go up to 3, it goes one less, s of 0 is c, s of 1 is o right, s of 1 is o, and s of 2 is f, so it starts from s of 1, and displays this, what if I said 5 here, it will start from 1, s of 1, 2, 3, 4, it should display so much.

You may be wondering why does not it go up to 5 that only makes sense for it to go up to 5, for some reason Python is created this way. There will be many such questions in your mind. Why did Python do it this way? The answer invariably is always, it was their choice; there was probably some reason that we may not know today or maybe you can Google it up and you may know why exactly they have done it like this. But by the way there are very nice reasons why Python starts with 0. So, does your C language. Anything before c with the best of my knowledge would always start with 1, the list would start with 1, the string would start with 1, you would refer it as s of 1 for c, but in Python you start with s of 0 for the first letter.

So, what will this output? This should output start from 1 go up to 5 minus 1, which is s of 4 this should be the output. Let us see if my, yes, my prediction is right, it goes up to 5, if you

let us say start from 3 and go up to 5 what will happen, 0, 1, 2, 3, and go up to 5, 5 minus 1 which is 4, 'fe' should be displayed that is indeed displayed here.

So, you sort of, you can slice the strings as you can see. This is called string slicing. So, let me go ahead and then play around a little more with the, with these two words, I said s plus t will give you concatenation as explained before, s minus t will throw up an error because it has no clue what to do with subtraction of two strings.

Now, to put everything in context I will type a nice piece of code and I will let you think on what exactly will this output. What is s? s is a string 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 and what is a, a is s of 0. What am I trying to do here? You probably are confused, you always type something on the right side which is assigned to the left side right, you never put another variable here on the left side, this is indeed okay, you can put whatever you want on the right side, its value gets assigned to the left side. It will take some time for you to get used to it, but commonsensically speaking a is now assigned s of 0 which you would have guessed is this 0, b will be assigned s of 1 which you would have guessed is 1. Now, when I say print a and then print b right, you will see the obvious happen here. It is taking some time, but what do you expect to see. You will see the value of, this is a usual problem with Replit.

I do not want to you know edit this part, but I want you all to take this warning message that it, if it so happens that it says fail to connect, it means there is some problem with your internet, you may have to wait for a second and then it gets connected, you see the output here, 0 and 1. Perfect. If I were to say s of let us say 4 and 7, it will say 4 and 7. Print a s of 4 is 0, 1, 2, 3, is indeed 4 itself, the number 4. So, it will say this. Now can you all guess what will the output of this be, a plus b? Do you expect it to say 11? Let me see, I would expect as a layman, as a person who is new to programming, I would expect this to be 4 plus 7, 11, but if you have watched carefully from what we were discussing a minute before, you will say no it will not be 11, so let us see what the answer is, as I execute it I see it is 47, 4 plus 7 is 47.

But why on earth would it say 47? That is because as you can see a here is actually a string, a one letter string, a character, b is another string, plus understands that it should take this string and this string and place them one next to the other, the concatenation as I have been talking to you people about, it will not see them as numbers, unfortunately and I have been telling you people computer does exactly what you instruct the computer to do.

It does not have the intelligence to add things up if they are numbers. So, what do we do to make these make 4 plus 7 equals 11 here? That is going to be pretty simple, all you need to do is when you say a equals s of 4, you must say convert that to an integer, when you say b equals s of 7, you must say convert that to an integer, a very simple trick. And now finally when you say print, it will say 4 plus 7, simply because the plus here is acting on an integer a and an integer b, unlike before where both of them were strings. If they are strings, plus this addition of strings means place them one next to the other, when you put plus between two numbers, it means do addition, the old school style, whatever you were doing in your school,

7 plus 4 was 11, but if there are strings, 7 plus 4 will be 7 4 placed next to each other. What is the moral of the story? The more the story is the way your computer sees things depends on how you store them and computer stores in its own way unless or otherwise specified.

You may have to specify in what format you want this to be stored just so that you can do the necessary operations on it. I hope things are clear, if not you may have to practice little bit on what I said just now. I will now try to give you a small program and I leave it to you people to figure out what is the answer for this code.

What if I said a equals s of 3 and b equals s of 8 and I will print, let us say, I will say n equals a plus b and then convert this to int, convert this to int, and then I will print n, what will be the output? The output is 38 as expected, correct? But if I were to put int here and an int here what is the output? I will remove the int here. What is the output? The output is 11.

Simply removing the int here will give you some output and retaining the int there and then the output will be something else. So, you can you can redo this code all by yourself and try to understand the motivation for data types.

More on Strings

Hello Python students. In last lecture we saw various concepts related to strings, like concatenation, indexing, and slicing. In this lecture we will see few more concepts related to strings. In the first concept, it is called replication. Let us declare one variable s as a string, value is good, print, so far we have seen what happens when we use a plus operator or a minus operator with string. Now, we will see what happens if we use multiplication operator with string. s star 5. So, what you think? What should be the output of this particular Python program? Will it encounter an error or will it print something? Let us try it. It prints the string 5 times one after the another, which means it is like an concatenation operator where same string is concatenated with itself 5 times. Let us take it one step further, instead of s if we make it s of 0, now what will be the output? Let us see, 5 times the letter g because s of 0 is g. This particular feature of string data type is called as replication where a string or a character from a string replicates itself based on the number with which it is multiplied by. Now the next string property is string comparison. Let us write one small piece of code. x is equal to India, print x equal equal to India and print x equal equal to India but over here this i is small. So the question is what will be the output of this kind of comparison. A letter being capital or small, does that make any difference or still it will consider the string India as same which is like variable x. Let us execute the code, first is true, obviously because we have I' as capital, but in second case it is false, which means a single letter change from capital to small letter is considered as not equal, therefore x equal equal to India with small i is resulting into false. Let us see some more examples where string comparison is happening. Print apple greater than 1 and print four less than 10. What you think, what will be the output of such string comparison? First, even is it possible to

compare strings like this or will it result into an error? Let us figure out there is no error. It shows some output false and true, now let us see why false and true. First when we compare apple greater than one, it says false. If you see a string apple is longer or has more characters than the string one, still it is not greater. Similarly, in second case when we compare 4 less than 10, it says true. But we know 4 is not less than 10. But that is true when we compare 4 and 10 as integers, here the word 4 and 10 are strings. Hence, the comparison operator works in a different manner. Let us see how these comparison operators greater than and equal to works with strings. A computer starts the comparison process character by character as in the zeroth letter of first string which is a in this case will be compared against the zeroth letter of second string, which is o in this case. That means a computer will compare whether a is greater than o. As we know in alphabetical order a comes before o, which means a is not greater than o, because of that it results into false. Similarly, in second case the letter f is less than t because the letter f comes before letter t in an alphabetical order, because of that it says true. Now, you must be thinking what happens if first letters are equal. Let us try to have one example where the first letter is equal and see what happens next. Print ab less than az. In this case the first letter a is equal, but the second letter b and z and are different. Similarly, in second print statement first couple of letters are same, in fact, all the letters from second string are exactly identical with initial letters from first string, except the last one character. Let us see what output this particular string comparison gives us? It says true followed by false, which means ab is less than az. This is happening because when the first character is equal computer moves to the next character which is b and z. As b comes before z in alphabetical order it says true. Now, in case of second print statement the first letter as in the zeroth letter of first string which is a, which is matching to the zeroth letter of second string. Same, the second letter which is b, next letter c, next letter d, and then next letter e. All these letters are matching. In first case after e there is one letter which is f, but on the right hand side of that letter than operator there is no letter to compare after e because of that computer end up comparing f with nothing and due to that the output is false because f cannot be less than nothing. Next string concept is string indexing. But in this lecture we will see different kind of string indexing called as negative indexing. Let us declare one string. s is equal to Python, print s of 0, s of 1, s of 2 and so on and the output is t, y, t, h, o, n as expected. This is something we have already done. What if we do something different this time? Instead of giving these indexes 0, 1, 2 and so on can we give negative numbers? For example, minus 1, minus 2, minus 3, minus 4, minus 5, and minus 6, what do you think, what will be the output of this particular Python program? Can we even access minus 1 letter in this particular string? We already know p is zeroth letter, y is first, t is second and so on, n is the fifth letter. Is it even possible to access negative index for a string? Let us execute and check. Yes, it is possible. If you see we are not getting any error. In fact, we are getting the exact letters from the string but this time they are in the reverse order, minus 1 that is showing n, for minus 2 it is showing o, for minus 3 it is h, for minus 4 it is t, and so on, for minus 6 it is showing p, which is the first letter of the string. Which means whenever we do minus 1, it access the last letter of the string, minus 2 is second last letter, minus 3 is third letter from the end, minus 4 is fourth letter from the end and so on. This particular indexing is called as negative indexing. Let us see the next

concept related to string. First let us declare one string variable and let us have some very big value again this variable, some random character. Let us try to print s of let us say 100, we do not know whether we have 100 letters in the string or not. The string looks big enough but may not be big enough to have 100 characters. Let us execute and see what happens? It shows an error which says String index out of range , which means we are trying to access the 100th character in the particular string but as per the error it says index out of range, which means the 100 character does not even exist in the given stream, due to which computer cannot print it. Now, the question is if we have such a big string, maybe even bigger like this, still the output is index out of range. In case of such a big string, how to figure out exactly how many number of characters are there in that string so that we will not end up using some index value which is out of range. For that there is one command called as len, which means length, print of length of s. Let us execute and the output is 36. This number 36 represents the total number of characters in this particular string. Now as we know there are 36 characters, which means now we should be able to execute s of 36. What you think will this code execute or still we will get error? Let us see, still we are getting the same error, string index out of range, but we already calculated that the length is 36, still it is giving error. Can you tell me why this particular thing is happening? Because if you remember in Python the string indexing starts from 0, due to which whenever we say there are 36 letters that means the indexes are from 0 to 35, let us make it 35, and see whether we are getting the last character of the string or not? Yes, indeed we are getting the letter f, which is the last letter in that particular string which is the 35th character, which indicates that we can use the index number which is 1 less than the total length of that particular string. In this case the length of string is 36, therefore the highest possible index is 35. Thank you for watching this lecture. Happy learning!

FAQs

I hope you enjoyed coding with us as and always coding involves practice as we have been stressing, programming involves you to code very frequently as much as possible and not just code whatever, we tried coding the terminal, you probably were parallely coding but it is also important that you code a little more than what we have tried coding this week, that is the only way in which you will get comfortable with programming. So, here we are, we will address a few frequently asked questions rather concerns that a student who is trying to program or learn programming for the first time faces, although python is very-very-very programmer friendly, I did not mean to exaggerate it, indeed is the most programmer friendly language that I am aware of from my little experience of programming from the past 15 - 20 years. I find Python really-really easy to understand, I wish python was there, was the first programming language that I learned, unfortunately it was not. So, here are some questions which I will go one by one. Feel free to ask your questions on the forum in case you have more questions than what we have addressed. I cannot remember the syntax, I find it difficult to remember several keywords that Python requires, this is the question. So, to address this, yes, I am with you people. Of course, it is going to be difficult, in fact, Python is a lot more than what one can understand, maybe even in a lifetime, I mean, so is any

programming language, but then try to see the analogy with any other language, look at your own mother tongue. Do you know all the words in your mother tongue or you do you think your vocabulary is full in the sense that there is no word left out? Obviously not, but then you are aware of most of the words, which is enough for you to converse in your day-to-day life. Similarly, as with Python it is enough if you learn a few things, I mean, which will help you open up the world of possibilities eventually. But then as you are learning these few things you will stumble upon the exact syntax, I mean, you will, you would not know how to write a piece of code for instance we discussed strings, there were a few functions that we used, there were a few keywords that we used to sort of manipulate strings. It was not very easy for us to remember them. I mean, but then with time you will start remembering them and you always have internet to look up, you can even have a small cheat code, I mean, the cheat book, cheat sheet that you can keep so that you can write down all the syntax that is frequently used and then look them up as often as it is required, but then in an actual you need not worry, this is a very-very frequently occurring problem, everybody complains that it is too much to remember, but with time you will start remembering it. I get errors when I am programming and I find them very irritating – So, everybody does get errors, even people who are very proficient with programming tend to miss out a few things when they are coding. Syntax errors are very common as you saw, in a simple print statement if you miss out on closing the bracket it throws up an error. This is very common, you should get used to it, nobody is a perfect programmer and you will indeed make errors here and there. And Python, in fact, helps you by telling you what exactly is the error there, not always though, but most of the times you will get to know where you are making an error and you can rectify it immediately. So, to answer your question this is completely normal, you should get used to it. A programmer's life involves fixing the errors more than writing the code, so this is how a programmer's life is and you are entering into the world of programming, so you should get used to not getting upset with errors. Why are we learning Python why not any other language xyz? That is a good question. Of course, we are free to teach or learn any language, but then the reason as we stated. The reason why we took Python is that Python is very easy to learn, easy on the mind for a person who is trying to program for the first time, very powerful, much sought after, the entire world today is talking in the language of Python and most of the jobs are open for python programmers in my humble observation. And most importantly I would say if I were to restart, I mean if I were to start my student hood and assuming I did not know programming, I would definitely start with Python, for a simple reason that most of the open source softwares today, assuming you want to do something creative and you need to look up if someone has done that already, mostly that program will be in Python, it will not be in any other language. This is my observation. So, which means that if you learn Python, you have millions and millions of open source projects available for you, millions and millions of library functions readily available for you to pick and then simply use. That is probably the reason why we thought python would be a good programming language to start. But then one is free to learn any other programming language, but the point is once you learn one programming language, trying to learn a second programming language is not very difficult, it is just like learning a second language, first language as an infant, you and me probably

found it very difficult, it took a long time, but then an immediate second language was not really very difficult. So, once you learn Python for instance, if you do not know C or C plus plus, you learn Python really well, start coding, understand what is logical thinking, how to, what is programming logic and eventually you will see that learning C and C plus plus is only a matter of knowing the equivalent syntax in that language, which you have already learned. So, Python that way gets you started as a beginner. I do not think that it is easier for one to learn a programming language like Pascal or C or C plus plus, Python is relatively easier, that is the reason why we thought one should learn Python and that is the reason why we are teaching you all Python as the first programming language. How do I efficiently learn coding? How do I get better at coding? So, the answer to this is as and always how do I get better with a sport, let us say table tennis or chess or let us say badminton, the answer has always been practice makes one perfect. So, you may want to practice as obvious as it can get. But then while practicing there are few tips that we can give you. Number one is it is important for one to type a piece of code for a particular question repeatedly, that is one skill that I think would come very handy for a beginner wherein just in case you are given a question to let us say sort a few numbers, you will be seeing these programs eventually, although we did not cover it in the first week. You will be seeing sorting of numbers is, something that is taught in the first few days of programming, you will see that sorting numbers, writing a piece of code can actually be daunting for a beginner, but then once you finish writing the code, you should not move to the next program in my humble opinion, you should open the browser, open the compiler and open the editor and then type the code afresh. You should repeatedly do a code for the same question multiple times, I mean, of course, by not looking up the code that you typed already, but assuming that you have forgotten the code you should try redoing it. This is a very nice way in learning how to get good at coding for a beginner, especially the reason is the following. The reason is we address the question of syntax, remembering the syntax, when you are typing a piece of code there are two things that makes your life difficult, one is the syntax, remembering them and second thing is the logic itself. First one you will get used to with time, you will understand what is what, you will understand Python keywords and you will be proficient with programming, but then logic it is very question dependent. You also will learn logic with time, logical thinking also comes with time, once, it is like seasoning your mind, you will do it slowly, but then the first one, the syntax is actually memory based comes with time, but logic takes more time for one to learn. But when you do the same piece of question, same question multiple times, you will, number one, you will not worry about the syntax, number two, you will worry only about the logic. That is the reason why you want to solve the same problem multiple times. So, how do we go about it? Take a question, try writing a piece of code and then open the editor once again and do it once more and do it three or four times and try to take a program that is slightly longish, I mean, you do not want to do a two line code multiple times, it does not make sense. This is a good way in which you can get good at programming with time.

Variables : A Programmer's Perspective

Here is an important tip that I am going to convey through a small program. So let us start with this little story of two brothers, Ram and Lakshman. Let us say Ram s bank balance is one lakh and his bank loan, Ram s bank loan is let us say 5 lakhs. Lakshman, his brother's bank loan is, his brother's bank balance is let us say 20 lakhs and his bank loan is let us say 10 lakhs. Do you observe something here? Is it not indeed confusing to call it A, B, C and D? Is it not important that we make the variables sort of self-explanatory? And that we need not break our head on what exactly did I say just now? Was it bank balance or loan? Was it the brother Ram or the brother Lakshman?

So it is easy if I were to simply name this as Ram bank balance, ram_bank_balance maybe and then here instead of B I can say ram_loan. Instead of C I can say lakshman_bank_balance. This character is called underscore that I am typing it looks like dash but it is a little below dash at the line level and then I will say lakshman_loan. Now with this it is sort of easy for me to calculate the, assuming that these two are brothers staying in the same house and I would say the net income of the house is Ram's bank balance plus Lakshman s bank balance.

And let us say the net liability is what they owe. Ram s loan plus Lakshman s loan, makes sense. Now, the final value let us say whatever that is, is net income minus net liability and I will print the final value hoping that there are no errors right now. Let us see. So the final value is 6 lakhs. So the family has, let us say, so the family has 6 lakhs. Let me restart this kernel then we re-execute it. The so, the family has 6 lakhs. If let us say Ram s loan or let us say even Lakshman s loan was a little more, then it would come 0 here thus making it how much 10 lakh, 1 crore. So the family has minus 84 lakhs loan. So, if it is minus, if it is minus it will be liability, if it is plus it is it is the surplus, it is the extra. So is it not easy on your mind when I say, when I give the variables like this than say A, B, C and D. Here is an important tip. Tip number 1; try to make your variables self-explanatory. Second tip is add comments. How do you add comments? You add comments by putting a hash followed by whatever you type here is to be ignored by the computer. Nothing will happen when you put a hash like this. So still, you get the answer as it is, as you can see. So whatever you type after hash, the computer will ignore it. So what I will do is, then why would, so why are people mad enough to put a hash followed by something if the computer ignores, what is a need for us to put that? The need for us to put something after hash is because sometimes when we declare a variable like this; we would know what we are doing, but after a few days we will forget what exactly was the reason, why we declared this variable.

Especially if the code goes to several 1000s of lines we are bound to forget despite our being careful and giving variables that are real life like still we will forget. So what I will say is the Ram s bank balance note that this is positive. Ram's loan, this is his, this is to be returned by him and hence will count, again the next line if you want to come you should put hash once again will count as negative. And Lakshman s bank balance is self-explanatory. Net income, generally you put the comment below it, you can also consider putting above it not a problem, net income is the total bank balance of the two brothers. Net liability is the total loan of the brothers, of the family brother. The final value is the family's final, let us say final

money could be positive or negative.

So the family has so and so, as you can see, nothing changes. I am executing it to one once again and then you see nothing changes when you include comments. Comment is only for you, for your purpose, just to ensure that if you see it after a few days or weeks or months, or sometimes even years, a software engineer writes a piece of code running into several 1000s of lines and someone else takes a look at it. It must make sense. It will make sense only if you have taken care of the variables made it very sort of human friendly, reader friendly, programmer friendly. Secondly, you have included a lot of comments. So what we will do is from now onwards, we will include a lot of comments in our program.

It is a very, very, very good programming practice, did not mean to exaggerate, it indeed is for one to put a lot of comments. It is a very good programming practice. But unfortunately, many, many people do not do it for some reason. Maybe they are lazy or they do not want to kill their flow of thoughts by including comments, but trust me, maybe you can try writing the code the way I did and then come back and then write comments before you close the compiler ensure that the code that you have written is commented. That is a good programming practice.

Variables Revisited: Dynamic Typing

Let us say, I declare the variable A as 10 and then print its type; if you remember, type prints, what kind of variable A is. It is saying integer. So I need to tell you people that what we are discussing right now is not very important for a beginner. But then for completeness sake, we need to tell you these things. So do not worry much in case you do not understand this really well right now, but eventually you will, I mean as we reach eighth week, ninth week, you may want to go back and then look at what are all we discussed so far and then try to see if things make sense to you or not. So, this is slightly advanced, but do not worry if you do not understand, but make your of attempts to understand what I am saying. So I say A equals 10 and I print the type of A and now I say A equals India and then print the type of a. So what is your guess? Initially, it was int, now it became str. Remember, we were discussing about the jar in your kitchen.

You can use the jar to store rice or maybe even water for that matter, but then we specialize in our, in the containers that we use for storage, we do not use the same container that is used to store rice, we do not use that to store milk. Milk container looks different, is made of a different material and things like that. So similarly, a computer tries to store an integer type using a different kind of memory location and string in a different kind of memory location. Moment you say equals 10, it automatically creates, what does it do? It creates a particular data type. By data type, I mean the memory location where it, where the value of A is stored and the moment you say A equals India, of course, it loses the value of 10 that it was initially assigned to, but then it becomes of the type string.

This is this goes by the name dynamic typing, you all know what is commenting right now. I use comment; the first line will be dedicated to write what program we are writing. We are trying to illustrate what we call the dynamic typing. The word typing here does not mean the typing of the keyboard. The word typing here is the noun form of the word type. Dynamic stands for things can change. In Python, the moment you declare an integer, it does not stay as an integer; you can change A to whatever you want. It provides flexibility for you to use a variable for something as a, as an integer and then make it a string later on. The type of a variable is dynamic; you can change it as per your wish. So let me remove this and then illustrate this with a better example. Let me take a number n equals 10 and then say n equals n by 2. Carefully observe what happens here. I print the type of n and as you would expect, it is int only. Let me clear the screen here. Clear clears the screen. So let me print. It shows you it is of type int. n is of type int is what it is saying.

Now, let me print another print statement here and display the type of n. Now after I divide by 2, let us see what it does. It says int, but the moment you divide, it becomes float. So let me try printing and now and says it is 5.0 now, but then here, if I were to print n it is, it says 10. It does not say 10.0. But the moment you divide by 2 it, the moment division happens, it realizes that oh now the integer might become a non-integer; it can become a rational number or some real number. So, let me be careful. Let me allocate, let me deallocate that integer. Let me try to allocate something that that is easier for me to contain this object; it could be a real number. So I will use float, float stands for real numbers and in the language of computers, int is integers 1, 2, 3, 4, 5; float is 4.3, 7.61 all that comes under float. So let us see what happens? What does Python do even if we divide by 1, even if we divide the moment you use the division operator, Python makes the number, makes the variable float from it being int. So it changes irrespective of you assigning explicitly you did not do n equals a string, you just divided n by 1, still Python was intelligent enough to now change the type to float thinking that oh, this chap is doing some division, he may continue to do more division. Let me try making it float to be on the safer side.

This goes with the name dynamic typing, again as I repeat not very important at this stage. Just know it for completeness sake; we are trying to teach you this. But in case you do not understand you know what to do you go ahead with other weeks, confidently but and then come back and then try to look at these things, this will make more sense once you start coding extensively.

More on Variables, Operators and Expressions

Hello Python students, earlier we have discussed about variables, operators and expressions. We will continue on the same lines and discuss few more concepts related to those same topics. So, the first concept is keywords and naming rules. We have already seen couple of operators like and, or, not, all these terms are considered as keywords. Along with these keywords, there are a few other keywords which we may not have seen yet in Python, but definitely you must be remembering those from computational thinking course like if,

else, for, while and so on. All these terms are referred as keywords as in, these terms or these words are defined as part of the programming language itself, because of that we cannot use these keywords as variable names. For example, you cannot have a variable name and is equal to 5. This will give an error; error says invalid syntax because the term and is predefined, it is a keyword in Python programming language. Hence, it cannot be used as a variable name. Similarly, let us try or, same output. Maybe let us try if, same output; for, same output. So, all these keywords cannot be used as variable names in the Python language. There are another set of rules, which defines what can be used as a variable name. The first rule says, we can use only alphanumeric characters and underscore in variable name. Alphanumeric characters include all alphabets from A to Z in lowercase, as well as uppercase and all numbers from 0 to 9. Python also supports only one special character in the variable name, which is underscore. Capital A can be a variable name. Similarly, small a can be a variable name, a1 is also a valid variable name. As I have mentioned, we can use underscore along with it. So let us try a1_. This one is also a valid variable name, which means we can use all alphabets from A to Z in uppercase, as well as lower case, all numbers from 0 to 9 and underscore in a variable name. In addition to this, there is one more restriction on how a variable name has to be. And the rule says; we must start a variable name with an alphabet or an underscore, but we cannot start it with a number. Which means 1a is not a valid variable name. If we add underscore before it, then it will work as per the rule, which we discussed just now. The last variable naming rule is variable names are case sensitive. For example, we have a variable to store a roll number; roll is equal to 5 where all characters in the variable name roll are in smaller case. In second instance, all characters are in uppercase. In third instance, only the first character is in uppercase, whereas remaining characters are in lowercase. Let us see whether we are getting three different values or it is printing only 15 replacing it with the previous values. As you can see, we are getting 5, 10 and 15 because computer treat all these three variables as unique variables, even though the spelling is same, because variable names in Python are case sensitive, even a small change of uppercase letter or lowercase letter makes that variable a new and separate variable. Next concept is multiple assignment; let us look at this particular code x comma y is equal to 1 comma 2, print x comma y and the output 1 and 2. This type of assignment of a value against a variable is called multiple assignment, where two different values are assigned two different variables in a single line. While using such multiple assignment, we have to be bit careful, because the sequence of variables or the sequence of literals on the right hand side of the equal to sign is very important. In case if we make it 2 comma 1; it will change the values of x and y. Now x is 2 and y became 1. Along with this, there is one more way in which you can assign a value to multiple variables in a single line. x is equal to, y is equal to, z is equal to let us say 10 and print z also. It will print 10, 10 and 10 because the value 10 is assigned to all three variables x, y and z in a single statement. So far, we have only assigned a literal values to a variable but we can do similar with variables as well. For example, x comma y is equal to 1 comma 2, print x comma y, then x comma y is equal to y comma x and then print x comma y, can you guess what will be the output? Let us try it 1, 2 and then 2, 1. Here, this kind of statement simply swaps the values of variables x and y with each other. So far, we have created so many variables, assign values to them, did

lot of operations, print the values of those variables and so on. But we never talked about how to remove a variable. As we keep on writing a Python code, we will keep creating lot of variables. So there has to be a mechanism through which we should be able to delete those variables as well and that is our next concept called deleting a variable. For example, x is equal to 10, print x, del, as in delete x, then print x again. Let us execute this code and see whether the variable x is getting deleted or not. Because of first print statement, line number 2, we got the output as 10 and it is giving us error for line number 4, print x. It says name x is not defined, it says not defined, because we have explicitly told computer to delete that particular variable x from the memory location. Hence, at line number 4 does not exist. Next concept is called shorthand operators. Earlier we have seen arithmetic operators. Shorthand operators are kind of arithmetic operators but they have a unique feature. Let us look at this code. We are counting some values. Hence the initial value for variable count is 0. When we count 1 ideally, we write a code something like count is equal to count plus 1. Then when we count second, we repeat the same step and then once again the same and say, finally, we print the variable count, we get the value as 4, which is absolutely correct. But writing this name of the variable count again and again, is bit tedious and unnecessary, that is why Python has a solution to this particular problem and the solution is called shorthand operator, which help us to remove a repetitive use of same variable name in the same statement. We can remove this count from here and instead of writing it like this, we will write it like this, count plus equal to 1, let us replace count plus equal to 1. Let us execute again, still we are getting the same result, which is 4 but now, we have to write this variable only once, instead of writing it twice in the same line. While reading this kind of a statement, we should read it as count is equal to count plus 1, which means at a computer level, the statement in line number 2 and line number 3 are identical, it will give the exact same result, then you must be thinking, why we require shorthand operators? And the answer is shorthand operators are introduced to make our life, as in programmer s life easier because this kind of incrementation or decrementation operations, we have to do so many times in a typical code and the use of shorthand operators to replace the regular increment operation reduces the time. Hence, it helps us code fast. Now, you might be thinking, can we use it only with addition operator or is it possible to use with other arithmetic operators as well? The answer is, yes, it is possible to use such a shorthand operator with all arithmetic operators. Let us try to, let us say count is 10 and this is minus equal to and this is minus let us see what output we get. It is 8, because we decremented the value of variable count twice, first with shorthand operator and second with a regular statement, which is count is equal to count minus 1. Similar thing can be done using multiplication operator, let us execute. It says 40 because initially, the value for variable count was 10. Because of statement in line number 2, the value became 10 into 2 that is 20 and then line number 3, it became 20 into 2 40. Let us try division also, first 10 divided by 2, which is 5 and then 5 divided by 2 which is 2.5. Next concept is a special operator called in operator. It allows us to perform a similar operation, which usually happens with search engines, where we type some keywords and that particular keyword, is checked against all the possible documents and if there is a match, then we get that document in the search results. Similar thing can be done in Python, as well. Let us look at this particular example. First print statement, first string alpha then

the operator in followed by a second string that says variable name can only contain alphanumeric characters and underscores. Similarly, first string is same in second print statement followed by in a variable name must start with a letter or the underscore character. Before explaining this code, let us execute it, then we will see why we are getting a particular output. It says true followed by false. As I explained earlier the in operator checks, if a particular value exists in something else or not, which means a computer is actually checking a string alpha exists inside this particular string or not. If it exists, it returns true that is how this particular in operator works. The result of in operator is always a Boolean value. Now the next concept is related to expressions and it is called as chaining operators. Let us look at a small Python code and see how chaining operators work in Python. x is equal to 5 print 1 is less than x less than 10. Print 10 is less than x less than 20. Print x is less than 10 less than x multiplied by 10 less than 100; print 10 is greater than x less than equal to 9, print 5 is equal equal to x greater than 4. Now, if you observe any of these print statements, you will see, we have used two or more relational operators in a single statement. When we use multiple relational operators in the single statement, then it is called as chaining operators. Let us execute the code and see what kind of output we are getting. First, it says true, because x is 5 and 1 less than 5 less than 10 statement is true, because 5 lies between 1 and 10. Second statement turns out to be false, because 5 less than 20 might be correct, but 10 less than 5 is an incorrect statement. In order to get true as a final output, all possible conditions in that particular statement has to be true. Therefore, the output for second print statement is false. Next print statement, x less than 10 which is 5 less than 10. That is true, less than x multiplied by 10 which is 5 multiplied by 10 which is 50 then less than 100. That is why we are getting true because we are saying 5 less than 10 less than 50 less than 100. Here all relational operators are giving output which is true. Similarly, in the next print statement 10 is greater than 5 and 5 is less than or equal to 9. In the last print statement, where 5 is equal equal to x and that is greater than 4. Hence, once again the output is true. When we use these relational operators in this particular manner, then it is referred as chaining operators. Thank you for watching this lecture. Happy learning!

Escape characters and types of quotes

Hello Python students. In this lecture, we will see two new concepts, namely escape characters and use of quotes supported by the Python language. Let us start with escape characters. For example, I want to print a statement like, It s a beautiful day. As you can see, I have started with a single quote and I have end this particular string with a single quote, which is a valid syntax for a print statement. Let us execute. It says invalid syntax, because over here, It s has this apostrophe and computer does not differentiate between this apostrophe and the single quote, which we usually use to start a string or stop a string, because of that, a computer starts a string with this quote and ends it with this particular quote which is actually the part of that sentence. Now the question is how to solve this problem. One easiest solution is to make it, it is and it will work. But I do not want to do that, I want to keep it like this and this particular problem or this particular error can be fixed

using escape character. Escape character is a mechanism which allows us to add those specific characters or those specific symbols in a string in print, which usually we cannot do, as you can see over here, with this single quote. Escape character is written using a backslash followed by the character, which you want to insert in that particular string. In this case, I want to insert a single quote. Hence, I should use backslash before that single quote. As you can see, as soon as I insert a backslash, even the color of that particular statements changes immediately. Now we can observe this backslash followed by a single quote has a different color, because now, these two characters are not considered as separate characters. Computer will translate this particular character as a single entity and it will replace it with a single quote as we want it, just like this. It s beautiful day, it s a beautiful day and this particular character is referred as escape character. Let us try one more example, print. Now, instead of single quotes, I want to use double quotes as we have seen earlier, with respect to print, it is absolutely okay to use double quotes instead of single quotes, we are from IIT Madras, once again computer rates it is starting of the string and this double quote as ending of the string, then it is not able to process this particular term IIT in between followed by starting of a string and ending of a string. Because of this computer is giving us an error, which says invalid syntax, this escape character will rescue us from this particular invalid syntax error, backslash followed by double quote, once again, backslash followed by double quotes. Now we are getting the expected output, we are from in double quotes, IIT Madras. This is related to single quotes or double quotes. What if I want to do something a little bit different than this? For example, print My name is Omkar space, I am from Pune. If you print it, it will print correctly. But what if I want to give additional space between these two sentences? Currently, there is only one space in between these two sentences. But I want to separate these sentences even further. The simplest way, could be to add more spaces, but it is not a good practice. So what to do, usually we use tab to give these extra spaces. But how to use that in the Python language? Let us retain this quote and copy it. Instead of this space if I use something like backslash t, t stands for tab, if you can observe the last two lines of the output here, there was only single space. But now you can see more gap between these two sentences. If we add one more t over here, it will put more gap between these two sentences and now we do not even have to give multiple spaces between two sentences. So this particular escape character is called tab. Now there is a sufficient gap between these sentences. But still, I am not happy with this particular quote. I want to print these sentences on the different lines. A straightforward way is to remove this particular sentence from here and add it as a fourth print statement. But we can avoid that by placing one more escape character called backslash n. n stands for new line. Let us execute this quote. Now you will see the first sentence is getting printed on the third line whereas next sentence is getting printed on the fourth line in the output even though these two sentences are actually written as a part of same print statement. This is happening because of this specific escape character called backslash n, which refers to a new line. Before moving to the next concept let me give you a small exercise, let us copy these two lines, I will change the single quotes to double quotes and remove this escape characters involved. Similarly, I will make these single quotes and remove this particular escape character. Oh, there is a spelling mistake here. That is not important part. The

exercise is, I will not execute these two lines, it is up to you to execute these two lines, see the output and figure out why the specific output is coming like that. This is very important concept of strings. But as it is very simple, I am leaving it up to you guys to study on your own. Next part of this particular lecture is use of quotes supported by the Python language. We have already seen the use of single quote and double quote for strings; x is equal to single quote this is a string, y is equal to double quotes, this is also a string, print x print y. As expected, it will print these two strings. So far it was working. But for example, I want to write a third string which says; first line, second line, third line print z. It says EOL while scanning string literal. EOL stands for end of the line, it says end of the line while scanning string literal. When computer starts scanning this third line; starts with z is equal to single code first line. It is still looking for the closing of this single quote which it is not able to find as part of this third line. That is why we are getting this particular error which means this single quote is useful only when a string is stored in a single line. If we divide the string in multiple lines, then we cannot use single quotes. Let us try double quotes instead of that, still we are getting the same error. Now, the question is how to handle this kind of string? In order to store such a string, which is distributed over multiple lines is stored using three quotes. Let us print and try as you can see, it says first line, second line, third line. Similarly, we can keep adding as many lines as we want, fourth line, fifth line and so on. So, now, we know there are three different types of quotes, which are supported by Python language; single quote, double quote and then triple quotes, which we studied just now, and it is used for storing multiline strings. There is one more advantage is using this triple quotes. Earlier we have studied if we use hash, then computer ignores that particular line from execution, it did not print x and y, it printed only the variable z, but ignored variable x and y which we printed in line number 8 and 9 because the use of this particular character hash. As we know hash is used to comment a particular quote, which means that particular line will be considered as a comment, not a part of Python program. But what if I want to give a comment, which occupies more than one line, I can always say, comment 1, comment 2, commend 3 and so on and it should work. But adding this hash every time is not optimum solution. The similar problem can be solved once again using these triple quotes. Because once again, whenever we use triple quotes, it becomes a comment in Python. So today, we have studied a new type of quotes called triple quotes and they can be used for two different purposes. First; to define a multiline string and second to give a multiline comment. Thank you for watching this lecture. Happy learning!

String Methods

Hello Python students. In this lecture, we will see something called as string methods. We have discussed about strings multiple times in various lectures, we will extend that discussion to string methods. What are methods? Methods are nothing but functions or commands. At this point of time, you do not have to think too much about the word methods. We will come to that in a later stage of this particular course. Right now, you can consider these methods as some kind of commands which we can execute on strings. Let us start with first set of string methods. Lower, it converts the every character in the input

string into a lowercase character. Second, uppercase, this one converts every single character in the string into an uppercase character. Next, capitalize; it converts only the first character of the string to a capital letter. Except that first letter, all remaining characters gets converted to lowercase characters. Title, it converts the first character of every single word in that particular string to an uppercase character, except that first letter of the word, all remaining letters are converted to lowercase characters. The last swapcase, as the name suggests, it simply swaps the characters from uppercase to lowercase and from lowercase to uppercase. In the last two columns you can see a string example is given and all these string methods are executed using that specific example and its equivalent output is displayed in the last column. Let us move to next set of string methods. Next set of string methods are islower, isupper and iftitle. These three methods are very similar to what we saw in previous slide. With lower, upper and title methods, we were actually converting the input string to its equivalent form. But in case of these three methods, we are not actually converting any string we are simply checking whether the input string is in a lowercase form or not, whether all the characters in the input string are uppercase letters or not. And the last method we will check whether the given input string follows all the rules of title or not. Once again, some sample codes along with their respective outputs are given. As you can notice, all the outputs are in a Boolean form as expected. Next set of string methods are isdigit, isalpha, and isalnum. As the name suggests, isdigit checks whether all the characters in the string are digits or not. If that is the case, it will return true or else it will return false. isalpha checks, whether all the characters in the string are alphabets or not. Accordingly, it will return true or false. isalnum is a combination of alphabets and numbers as in the digits. Therefore, this particular method will return true if all the characters in the string are alphanumeric characters, which means any combination of alphabets and numbers. On top of that, whenever we talk about alphabets, we consider alphabets in lowercase as well as uppercase. Therefore, if you observe the last line in this particular table, you will observe that the output is false because the input string has some special characters like at the rate, star and hash, due to which the output we are getting is false because these three characters does not come under the set, which we call as alphanumeric characters. Next set of string methods are strip, lstrip and rstrip. As the description says, strip method returns a trimmed version of the string. As per the given example, it removes all those hyphens or those dash symbols, which are there on the leftmost side of the string as well as the rightmost side of the string. This particular method helps us to remove those extra spaces or extra characters, which appears on either side of the actual string. The same strip function can be split into two different variations first is lstring, which stands for left strip, it removes a specific character from the left side of the string and the second one is rstrip stands for right strip, it removes that specific character from the right of this string. Now, I leave it up to you to find out what happens if that specific character appears somewhere in between the original string whether it will get removed or not. Next set of string methods are startswith and endswith. As the method name such as it returns true if the string starts with the specified value or it returns true if the string ends with the specific value in case of endswith. The value for variable x is Python with capital P and if you observe the output of starts with capital P, and then starts with small p, then you will observe that this particular method is

case sensitive. As in if we give input which is a specific letter, which matches not just the letter the case as well, then only we get it as true otherwise, the output will be false even though the letter is same, but the case is different. Similarly, endswith here, if the letter n is small, then only we are getting true, if n is an uppercase letter, then we will get false. Next string method is count. It returns the number of times a specified value occurs in a given string, which means it will count how many number of times a specific character or specific value appears in that particular string and the final count value will be given as the output. As you can see, here, the count with character t is 3, but count with character s is 1. We are getting it one because once again, this particular string method is case sensitive, which means as we are looking for lowercase character s, it will count it will count only all those occurrences of lowercase character s. It will ignore that S which is in uppercase form. Moving to next string method, which is index. It searches the string for a specified value and returns the position where it was found. The computer will start reading the string from left to right and whenever it gets that specific value first that particular index will be returned as output. As you can see, t appears three times in that particular string. But the first occurrence of letter t from left to right is at index 2, because we all know string index starts from 0. Similarly, index S is 20. Once again, this particular index method is case sensitive; it is displaying the index of the last s in the particular string. And the last string method is replace, it returns a string where a specified value is replaced with another specified value. If you can see the example, we are performing an operation which says x is equal to x dot replace capital S comma small s. It works exactly like a Find and Replace feature, which most of the common text editors support. Computer will start scanning the string; it will look for all the occurrences of letter S in uppercase for all such occurrences will be replaced by a small s. Similar thing we are doing with capital M as well in the next line. At the end, when we print the particular string x, we are getting the output where the capital S is converted to small s and capital M is converted to small m by this particular replace method. You might have noticed, this is the first time we are conducting a lecture where actually we are not executing any piece of Python code. Instead, we are simply going through slides, we are conducting this lecture in this manner because string methods is an exercise for all the Python students. I expect you to execute all these string methods on your own with different types of input and observe how all these string methods behave with respect to the given input. So, I am leaving it up to you guys to execute all these string methods on your own with different inputs. Thank you for watching this lecture. Happy learning!

An Interesting Cipher: More on Strings

We allow them to play around with strings and we are going to learn something nice. So let us start with our alphabet set and we call them alpha and alpha will be all the alphabets; abcdefghijklmnopqrstuvwxy and z. So after this when I say print alpha of 10, what will it do? Let us start with 0 maybe, it will print the first letter in the alphabet set here, alpha of 0 will be the first letter here; alpha of let us say i and I will say i equals 10 it will display the 11th alphabet k. 10 is, if 0 is a, 1 is b, 10 will be your k. Clear. So far so good. Now just in case I go on like this and I display the 10th alphabet rather ith alphabet and then display the alphabet

after i, which is i plus 1, as simple as that. And then I say alpha i plus 2, then print alpha i plus 3, so what does this do? Let us see, if I start with i equals 0, it gives me a b c d, it is very obvious what this is doing, I do not have to explain it to you. By looking at the code you can tell me what exactly is happening.

I declare the variable alpha which is a string into z, all letters in alphabetical order in lower case as you can see all 26 letters are here abcdefghijklmnopqrstuvwxy and z. And I say i equals to 0 and I display alpha of i, alpha of i plus 1, alpha of i plus 2, alpha of i plus 3. If I were to do i equal 10, it will start from k as you can see and then go to l m n.

If I go to 20, it starts from u v w and x. The best part now is if I say 25, it will start with z of course, alpha of 25 is z, well, you do not guessed it right, alpha of 26 will throw an error. Precisely what we see here; string index out of range. Your alpha is only 26 letters long, which means you can go up to alpha of 0 to alpha of 25, you cannot go beyond that, while i equals to 25, i plus 25 becomes this becomes 26, alpha of 26 becomes outside the range, precisely what my, what python is telling me. String index is out of range, well no problem, there is a way to sort this out. So what do we do? Observe. Let me delete this for the time being. And then say alpha of 20, I will print this, just this nothing else, just this. It shows (()) (3:47). Let me clear the screen. It simply shows the 21st alphabet letter which is u, as you can see it is u here. But then, if I say i equals 30 and then if I just print, let me just comment this, if I just print i modulo 26, this is modulo operator; it gives you the remainder when you divide the number by 26. So 30 by 26, the remainder happens to be 4. Observe, if i equals 26, i mod 26 will be 0. If i equals 27, i mod 27 will be 1 as it is displayed.

Now you know where I am getting, what I will now do is I will print alpha of i modulo 20. And then when I say i equals 25, it displays the 25th letter, I am sorry it should be 26 here, i mod 26, the 25th i equals 25, means the last alphabet z here but the moment I say i equals 26, it goes to 0, because 26 modulo 26 is 0 and alpha of 0 will be a. Correct?

Let us see what happens when I say i equals 27, 27 modulo 26 is 1, so alpha of 1 should be b, so on and so forth. So do you see what is happening? If I were to type alpha of 24, what is that 25th alphabet, which is y, correct. Then I say alpha of 25, good way to do this is call this 24 and keep calling this i, I mean they are all one and the same, I am trying to do it in a way, that it is easy on the mind. It is good to use variables as much as possible. So alpha of i, alpha of 24, alpha of 25, alpha of 26. These should be y z and a, correct? Think about it. Yeah I get y z and there is an error. Why the error? Because I did not do the modulo 26. When I do modulo 26, it will not go beyond 25. If it is beyond 25, it becomes 0, 26 is 0, 27 is 1, 28 is 2 and so on, i modulo 26. I hope this is clear to you all. Pretty simple, I am in fact going very slowly assuming that for some people modulo operator maybe new. So y, z and a, if I well to make i equals 25, you see 25 mod 26 is 25, it is z alpha of 25 and a, b. You see it is getting rotated, it goes from abcdefghijklmnopqrstuvwxyz abcdefgh, after z it again starts with a. So this has a very interesting application which I will tell you in a minute now. Yeah, so far so good.

Now what I will do is I will take my name, you can take your name. So I will say s equals sudarshan, what I will expect to see as an output is I want to print tvebstibo, you know what this is? This is simply, let me comment this; I expect to output tvebstibo, what is that? This is exactly one letter shifting of this string; s u d a r s h a n, if you shift that by one letter, you will get this and whatever you want to type here, India, Chennai, your name, your friend s name and any English word, it should shift it by one letter.

How do I do that? It is very simple. Let us try it out, as you know I am going to use this string function called index which we have discussed recently. So I believe you know of it, if not please go ahead and revise your previous videos and understand what index does. It is basically self-explanatory, you need not go through it, but in case you are in doubt, you can go through it. So let us start, I would say t will be my new string where I will be creating one letter shift of this string that I have here. How do I go about it?

I first say, I will check s of 0, what is it and what is its index in alpha. So s of 0, let us see what this does, patiently you must observe s of 0 is s, the first letter s. And what is its index, what is s s index in this alpha, I mean which alphabet is s? If a is 0, s is what? Let us check, 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18; this is 18. Let us check that. Print alpha of index of s of 0 has to be 18. Yes. It says 18 here as you can see. Good.

So I am just clearing the screen. So fine alpha of index of s of 0 is 18 and what I want to do now is I do not want s of 0, I want 1 shift of it. But then when I shift by 1, I explained a problem that you will encounter, I am just trying to close the bracket that I have opened; I hope I am right, may be one bracket needs to be closed. So now this will print the index of s and plus 1 to it. And what is the alphabet, I basically want the alphabet. So for that I may have to say this is a number, so s of so much and I want to print this, maybe I have used one extra parenthesis, good, more the parentheses, less the confusion. So this should print; index is out of range, what is happening?

So there is a small mistake in the programming, I should not display s, I should say alpha. Why? That is because I take the first letter of s, very simple, just look at this, convert this to plain English. I look at the first letter of s, look at its index in the alphabet. What is its number? Whatever is s of 0, which is s, what is its number in the alphabet set? I add 1 to it and then display that, by display that I mean what is the, which alphabet is that when you add 1, when you shift by 1 unit, what alphabet do you get?

Whatever it is, you will see that here. So it is t, I executed this, it is t. If instead of sudarshan I want to say, let us say India, i becomes j here. That is the first letter and what I want to do is I want to, I do not want to print this, I want to append this to t. So what I will do is I will say t equals t plus, remember, plus of a string appends that to it. If we say print t, it will show me j. Correct? So far so good.

So next let us say I will try to take the next letter which is s of 2, so for that a good idea would be to say i equal 0 here and then make this my i. And then now do this and append the next letter in the string s. Add 1 to it and then take modulo 26 and then append that to t. Things are getting a little complicated, so let me try to explain this slowly to you, the string t will be equal to the string t append that with the next letter of the i plus 1th letter which is n. So in fact before going any further, it is probably a wise idea to print and see what is t. So what do you expect t to be, i becomes j and gets appended to t, n should become o and should get appended to t. Let us see if that is happening. Perfect. i is becoming j and n is becoming o, so far so good.

Now let us see what is next that needs to be done, I will copy paste this to make it easy. So I say t is equals to the existing t, concatenate that with the, take the index of s of i plus 2 it should be the next letter. And shifted by 1, take modulo 26 because in case it goes beyond z, then it should rotate as I explained. And then print it; j o e, as expected, d becomes e.

Now let me do it for i, I will go slowly once again, I am sorry I am doing repetitive again in the interest of people who are seeing this for the first time. Please note that programming can get complicated but if you make it easy on your mind by going slowly, staying patient, using a pen and paper and trying to write a diagram of what is happening, things will be very easy for you.

So t will be whatever t was, concatenate with the alphabet which is the shifting of s of the 0 1 2 3, the 4th letter. Then close the bracket, close the bracket, and then add 1 to it, because it is shifting and do modulo 26 to it just so that you do not mess up when it is the last letter. So in d indi becomes so much, very good and this is not looking neat, let me remove the spaces here, so it will look uniform, perfect. So as you can see, I get j o e j b, India becomes j o e j b, perfect. So as you can observe, I go up to the last letter here, if it were to be one letter more, let us say I take Chennai instead of India which is 1 2 3 4 5 6 7, seven letters. And I have to type this seven letters, otherwise what will happen, let me just execute this and see what happens. Chennai is getting shifted here d i f o o. Let me clear this screen and then execute it. Chennai is becoming, c is becoming d, execute this, c is becoming d; h is becoming i; e is becoming f; n is becoming o; n is again becoming o; a i (()) (16:10) to do. So let me copy paste it and then do it two more times, let us say i plus 5, i plus 6, the entire Chennai right now is shifted by one letter.

Now what if you want to shift to b by two letters. Then instead of 1, I just need to put 2. Let me make that a variable here. Now you see an application of variables, I say k equals 1 and make this 1 k, k, k. Why would I do that? You will see that in a minute, I only explained that if I want this k to be different, I do not want to shift once, I want to shift twice, let me execute it. Perfect, it is the same answer. But if I want to shift it twice, then I may want to put k equals 2, when you put k equals 2, look Chennai is getting shifted twice k equals 2 and c is becoming d, e; h is becoming i, j; and e becomes g; n becomes an o, p; n becomes again an o, p; a becomes c and i becomes k. Very simple. If you want to shift even more, let us say by 20

units, whatever that is. So c becomes w; h becomes b because it goes beyond z, as you can it is getting circled and it is coming back. Very good. So as you can see, points to note here is that point number 1, programming can get complex, you are seeing that it is getting complex here, you may have to think through it. In fact, as I was compiling this, I ended up getting confused a little, but then when I wrote it down, it was pretty clear to me.

It is just a matter of practice, you will get used to it, also please note that errors are very common in programming. As in when you type, it is never flawless, as in when you type, you will face errors and you must fix them on the fly. So it was complex here, so point to notice, things can get complex. Point number 2, the moment you type whatever you type here, you must type it so many number of times here. Point number 3 is you can keep this k as a variable instead of putting it 1 1 1 1 here in k or 2 2 2 2 instead you can simply put k equals whatever number you want and execute it, you will get a shift. This is another quick, importance of why use variables. Perfect. Try coding this, you probably of wondering is there any way we can avoid writing so many lines here, if it is a very long word like abracadabra, it is a very long word or let us say we have IIT, indianinstituteoftechnology, if you want to convert this, this will only convert up to this point, it will not go beyond that. So, it is already throwing error here saying that the substring is not found. That is because we have you know capital I here, see such small mistakes one has to be very careful about. So capital I here created a problem and now it is going good.

Clear screen and let me show it to you what is this is doing is simply taking the first 1 2 3 4 5 6 7, seven letters of this and then shifting it by one Indian becomes j o e j b, and you should type this so many times whatever is the number here, I mean how much, I will be give this 1 2 3 4 5 6 7 8 9 10, roughly maybe 25 approximately. So you should type this 25 times, is there any way we can automate this? Yes, you can. We can do this in just a one liner, we are coming there, the next will be an introduction to how to make repetitive things easily some sort of codable, by writing a piece of code. So, what I will then do is instead of going this one by one, one letter at a time, what I will do is, I will tell the computer go through these letters one at a time, until you hit a dead end, there is a way to do that. And we will be discussing that in the next week. But as if now, you be, this is just a warm up exercise for you all to understand that we can meddle around with strings like this. In fact, what we tried doing just now is popularly called the Caesar Cipher.

When k equals 3, which is three shifts. This code let me just write that down in the comment. Just about you can take a look at it; this is popularly called the Caesar Cipher in cryptography. If you want someone to not read your messages, then you can shift it by a particular k and then use that k as a key and tell them that unless you shift back by k units, you will not know what the text is.

That is precisely what we did just now, we just did one way, the other way is to shift it back. You make a minus k here and you will get back the letter. So you can try that out all by yourself and that is it for, I would say in this week this is slightly complicated an example,

but then do not worry, we have to introduce complexity somewhere, we introduced it here. Things will get easier with time and in fact you will be thrilled to see, that this very same code can be done, by the fact end of the third week you will be able to write this code in just a couple of lines.

Conditional Statements

Let us now learn a very powerful possibility in any programming language and that is called the if statement. So it is a very simple statement, but I am not a big fan of explaining the syntax of what is what in a programming language, instead let us see a nice example from which it becomes obvious as to what we are trying to explain. So, here is an example, let us consider the movie Avengers. This is a 13 plus movie, which means only people who are 13 and above can watch it. The rating of the movie it is called PG13, it stands for parental guidance and one should be 13 plus years old. So let us write a small piece of code which checks this. Just to make it fun, I am trying to use this example, although this does not appear like we will be doing in real life. Why would you use a python piece of code to check if someone is 13 plus or not, this is just to add some fun to the example. So, what are we trying to do is, it will be obvious in a while, I will say print please your date of birth. As and always, you take birth year is equal to input, we have discussed this, in fact, you can combine these two things on to a single line of code, but I prefer writing it like this. So it is easy on my mind. You will enter your date of birth and then I will say current year, what is the current year, it is 2021 I am not so sure in which year you are watching my video, but this video is recorded in 2021, so I will put current year as 2021.

Then, in fact there is also ways in which you can actually put the current year as of that day, when the program is executed. But like that is a little off route, so we will not take that route as of now, let us go ahead and say age is equal to the obvious current year minus birth year. And then observe, if age is less than 13, then see as I press enter, the cursor comes a little away from the beginning, you see, that is called a tab character, you leave a tab like that.

So, my editor does it automatically, in fact you should do it manually in case you are not using a nice environment like this, I am using spyder here, which I am typing python code. If you are using a simple editor, you may have to type that tab manually, so details aside, what should we type here, I must say print you are under age, you cannot watch this movie. As I say, the more restrictions we put, more curious the kid becomes to watch the movie, jokes aside. And then I will say else print you are old enough to watch Avengers, enjoy! Perfect.

So, what exactly happens here, I am executing it and it says enter your date of birth assume I am 2010 born, which I am not by the way, I am 2010 born, it says it calculates, it computes that your age is indeed less than 13. Because it is 2010, 2020 minus 2010, age happens to be 11 here and 11 is less than 13 and it comes inside and says, hey you are under age, you cannot watch this movie, which is precisely what it does here. Let me re-execute this and try to see if I type my age greater than 13, will it let me watch the movie. So let me say, execute

this; enter your date of birth, I will say 2000 is my date of birth. That makes me 21 years old, it says you are old enough to watch Avengers, enjoy! So you clearly understand what exactly happened, in fact, there is no much of, there is no details here per se. I clearly say it is very sort of English like a statement. I say if age is less than 13, then do this; else, else means what, if age is not less than 13, then whatever is inside this else you will do this. And then please note, as you come out of this and then say print; have a nice time and execute this, enter your date of birth, I say 1990, it says you are old enough to watch Avengers, enjoy! Have a nice time.

If I again execute this and I say 2012, I become a 9 year old kid, it says you are under age, you cannot watch this movie. If again says, have a nice time. How a nice time is said here too? Which means irrespective of whether this line gets executed or this line gets executed; this part is definitely getting executed. So, if something is true, then do this, else means if that is not true, then do this. In fact, what you can do is you can paste and see, what if you type another line here print wait until you are old enough to watch this movie, you can state this. You are old enough to watch Avengers, enjoy! Do not forget to watch the sequels and prequels. So what will does this do? This will obviously take this to the next line, just so that it looks nice, so this will execute both these statements if the age is less than 13, else it will execute both these statements. That is my guess; let us see what it does. So enter your date of birth, let me reset this. Executing it, so let me reset it once again. I am executing this which is please enter your date of birth, when I say 2015, it says; you are under age, you cannot watch this movie. Wait until you are old enough to watch this movie. I am little crazy about the punctuations, so let me be perfect here. So if I say my date of birth is 1960, it says; you are old enough to watch Avengers, enjoy! Do not forget to watch the sequels and prequels. Have a nice time. Good. So do you see what is happening, this have a nice time is coming in both the places, that is because it is outside the if loop, but then this gets executed only if your age is less than 13, and this gets executed only if your age is not less than 13, which means your age is 13 and above. At least 13 or you are 14 15 16 and more. Anyways this is best not explained in detail, in fact you will get to know more of it as we progress, we will be using as I said if statement is used in almost every piece of code that I have known. 99 percent the code that you write, will somewhere use the if condition.

So with practice you can achieve perfection of how this if statement works. So with this let us end this lecture and let me also try to tell you that you can in fact write if statement within another if statement. All those details will be covered in the forthcoming lectures. Thank you.