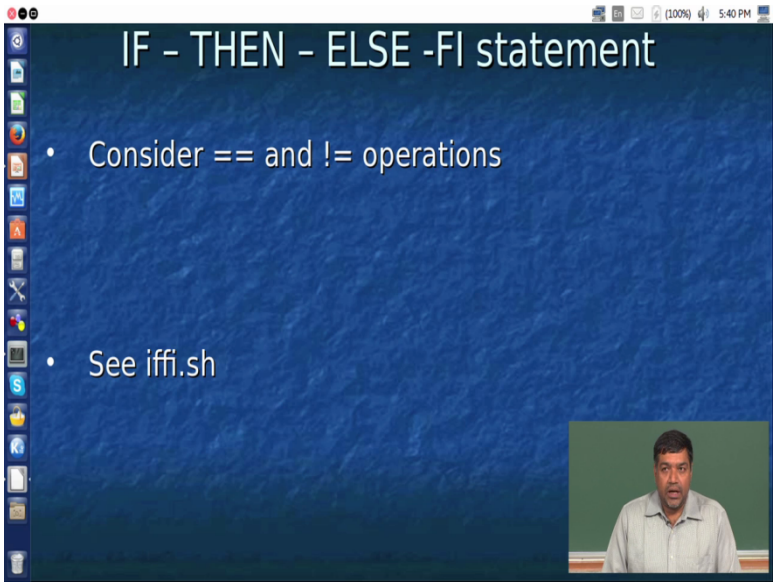
Information Security
Shell Condition and Relation
Sri M J Shankar Raman,
Consultant of Computer Science and Engineering
Indian Institute of Technology Madras
Module 31
Relational Operators

Hi there welcome to this session on relation operators and if statement. We have seen in the last session about arithmetic operations, so before we proceed to relational operations we will take a brief look at the if statement.

(Refer Slide Time: 00:39)

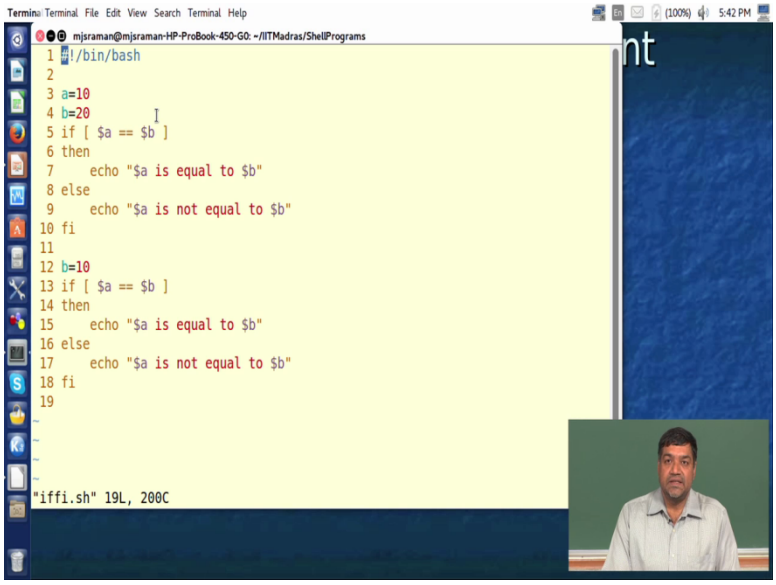


So why do we need if statements, at many points in your Shell scripting when I want to do a bunch of operations I may have how to take a decision so I've to take a decision on whether to go in one path or in another path.

This is a part of structured programming and in this kind of structured programming where structure design I will always come to your note that I've to make a choice between whether I should take a decision based on a certain condition.

So this if then else and FI together helps you move from one path to another Path, usually the conditions or will be either equal to or not equal to so equal to is represented by equal double equal to or not equal to is represented by not equal to let us see this with an example.

(Refer Slide Time: 01:32)



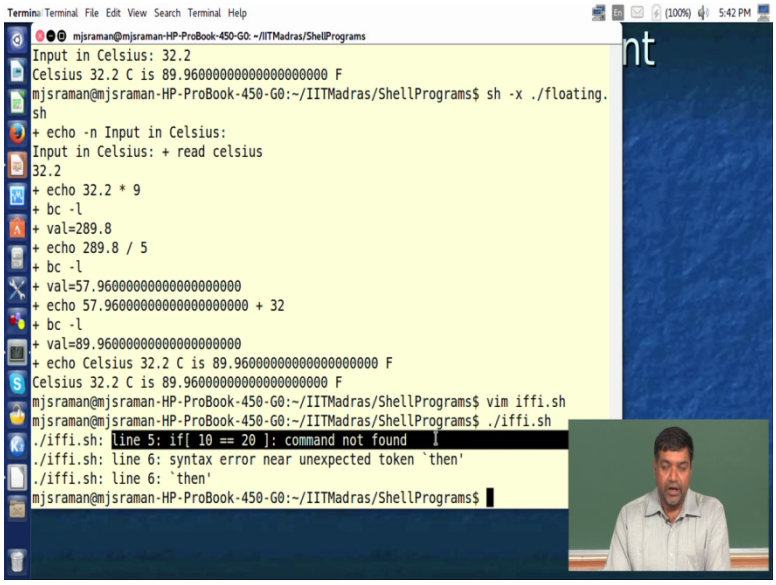
So what we have is a example something like this. So I have two variables a) which are assigned the value of 10 in line 3 of this program and then b) which has a value of 20 that is assigned in line 4 of this program.

So what I want to do right now is I want to see whether the value of a is equal to value of b. Obviously it is not equal and therefore what I do is I write a Syntax like this, I put a if statement, then I leave a space, then I I start a square brace, then leave a space I give the variable name, again leave the space then put the two equal to sign,

Again leave the space, then I put the dollar b the value of b and then I give leave a space and then finally I close this. Now every space that I dictated is important. Even if you miss one value the shell will say it's an error.

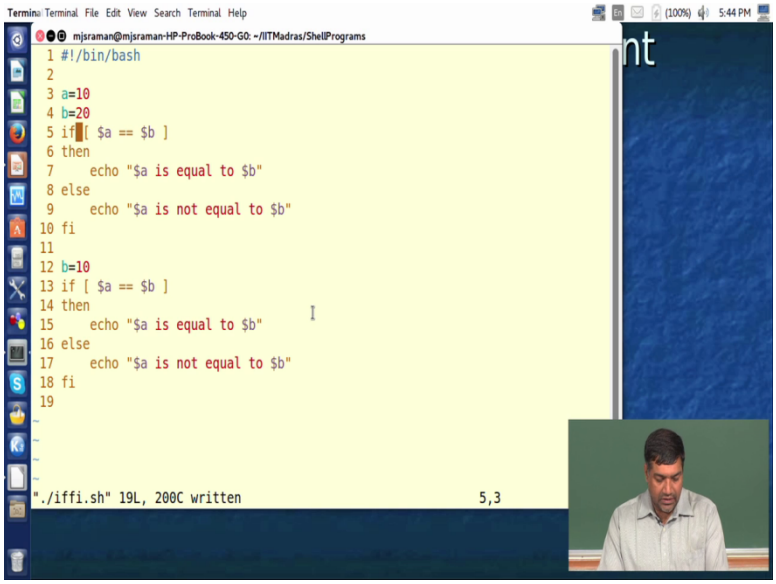
So let us try to see whether the shell shows an error so let us assume that I do not put it like this now if you look at this since I am using a good editor it tells you immediately that there is an error. Anyway let's not bother about it.

(Refer Slide Time: 02:53)



Let's try to run this program. So it immediately says in line number 5 there is an error command not found so you've to be extremely careful when you type this kind of programs in shell.

(Refer Slide Time: 03:08)



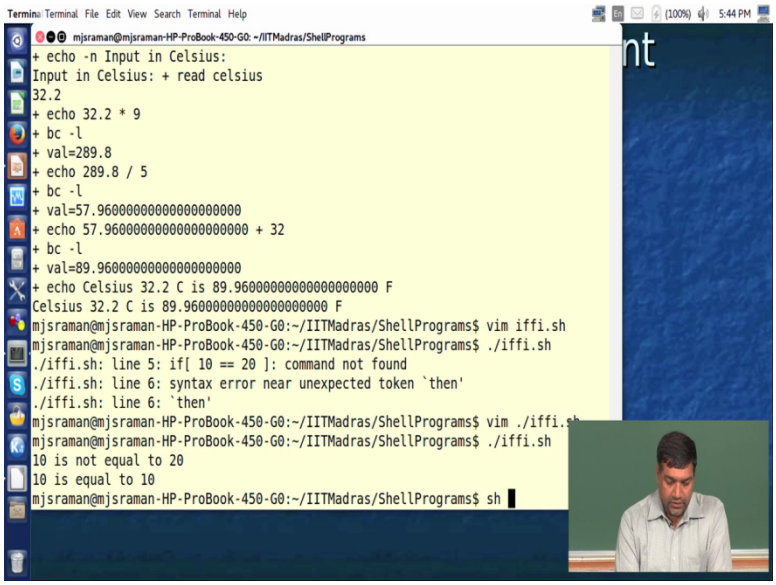
So let's return back. So in this case what I do is I compare these two values, now the if statement works in such a way that if I if these result of this turns out to be true ok, then the statements following then upto else is executed, see here I've written only one statement but you can have many statements written here.

So in this case for example if I give a value of 10 for a and 10 for b then this statement this becomes true this condition becomes true and once this condition becomes true then this part of the shell script will be executed. If suppose the condition becomes false as in this case for example here I have a 10 and 10 is definitely not equal to 20 then the else part of the code is executed.

Remember that when the then part of the code is executed finally the code skips the else part and then comes to the statement after this (04:14feed) so this is a marker ok this whole thing forms a line or a sentence but let's not get into those details so what happens is that so if you take this program sh the first statement that you get printed is this that 10 is not equal to 20 now what we do is now we try to make b also equal to 10.

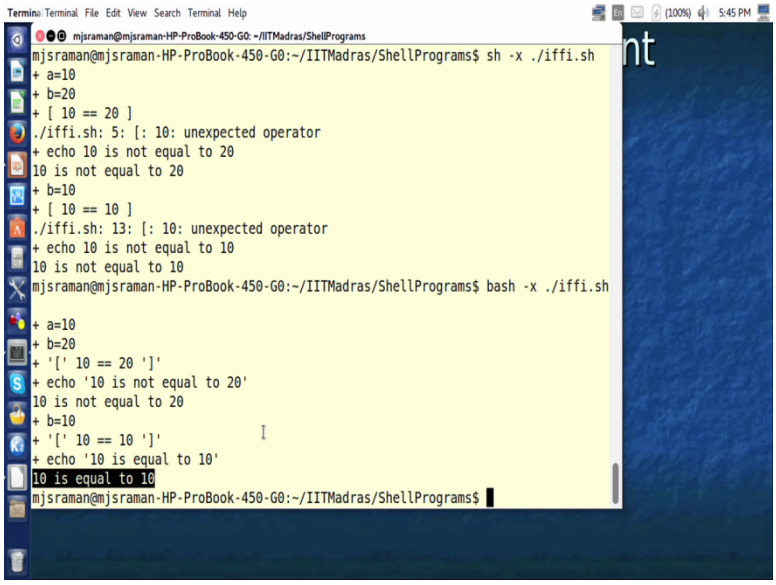
So in this case and then repeat the same program and as discussed before this will print a is equal to b so let us see whether whatever we had seen is getting printed here.

(Refer Slide Time: 04:54)



So what we do is I run this program so it definitely Prints 10 is not equal to 20 and 10 is equal to 10.

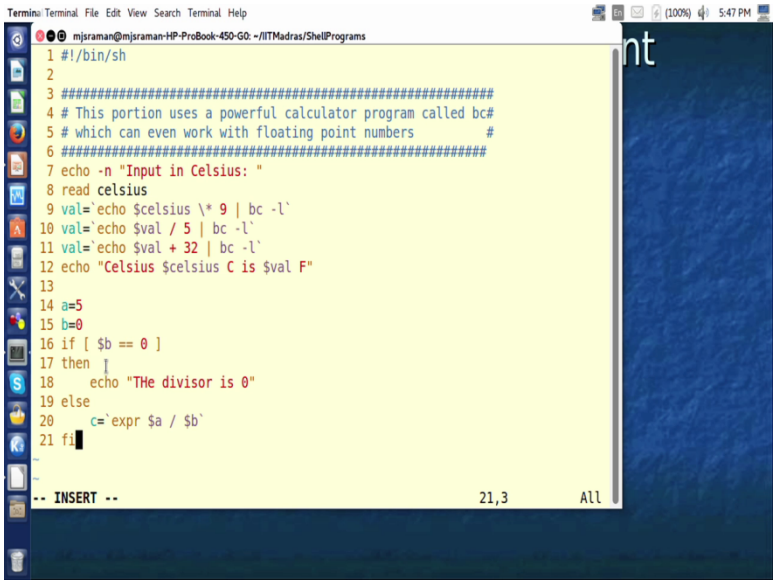
(Refer Slide Time: 05:03)



And we can also see this with the debug mode and in the debug mode you can see It is in the debug mode ok so in the debug mode if you see first is assigned the value of 10 to a it assigns the value of 10 20 to b then it compares a with b and it finds out that the result is not equal I mean this 10 is not equal to 20.

Therefore the else part is executed and it prints 10 is not equal to 20 now assigned a B to be value of 10 now if you look at this is 10 is equal to 10 therefore we print 10 is equal to 10. So this the Syntax for if statement I am just introducing this if statement now to tell you the number of possibilities using this what all are the things we can do.

(Refer Slide Time: 05:58)

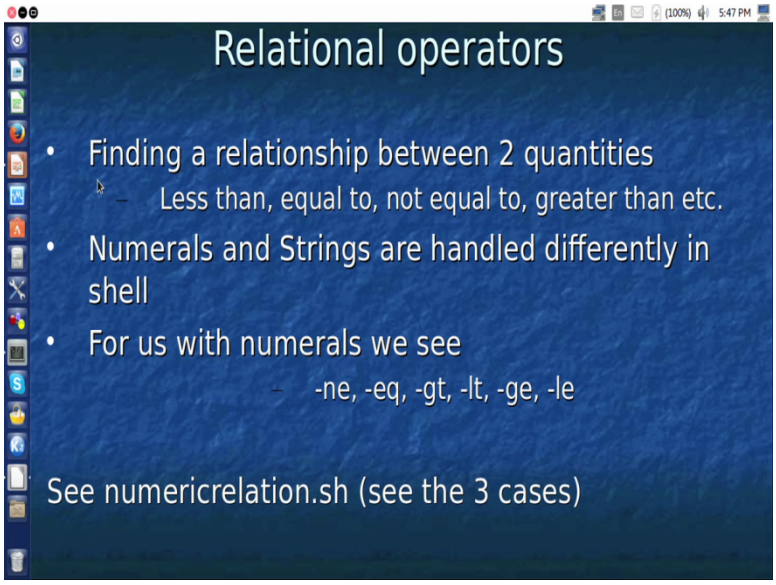


For example when in the previous code let us say we have a we have a let's say I just typed this so let's say I have a variable like this a is equal to 5 and then I want to say C is equal to say expression of dollar a divided by let's say dollar b ok

And let's assume that dollar b so b is given a value of zero now in this case the arithmetic operation itself will give if I divide a by b since b is zero it will give an error ok which you don't want to happen I mean sorry here so if a by b if b is zero dollar b is zero then it will give an error.

So what you could do is you could actually check by putting a condition like this if dollar b equal equal to say something like 0 then I can say then I can say print something like this echo the divisor is zero else we can go ahead and divide so such logic can be implemented in the code so that is the reason we saw why we have we have to use the if statements.

(Refer Slide Time: 07:38)

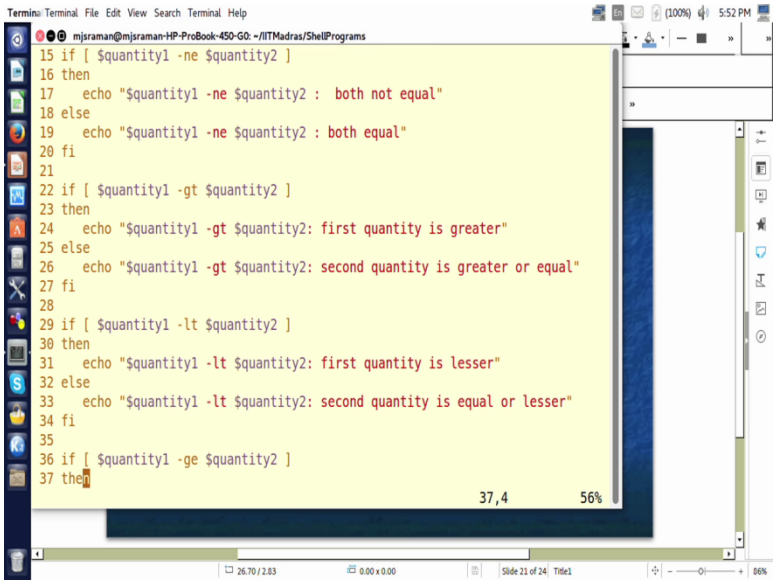


Now let us come to the relational operators ok a relational operator gives relationship between two quantities. What we can do is we can mix and match the relation operators along with the if statement so that is one of the reasons we introduce the if statements before so when we have two quantities then mathematically you can either have a I say two numeric quantities.

Then they can either be less than one can be less than the other one can be equal to the other one cannot be equal to the other one can be greater than the other etc now in shell the numerals and strings are handle in the same fashion ok therefore what we will be seeing in shell is for numerals we will be using these symbols minus me minus equal to minus gt minus lt minus g minus l which represents not equal to equal to greater than or greater than less than greater than or equal to and less than or equal to.

So when you are doing numeric comparisons you are supposed to use these symbols this is slightly odd ok because in general we will be using the keyboard values of less than greater than and all that so for those people who had done other programming languages this will looks slightly odd but that's the way shell is.

(Refer Slide Time: 09:06)



So let us try to identify how to do this? So we will look at numeric relation dot sh. So this is a slightly complicated program where we try to explain all the the quantity so let us start from line number 3 so we are comparing our idea is to compare two quantities

So we are taking up quantity one which is a value of 1000 and quantity two which is value of 100 what we will do is when you are demonstrating program we will actually change the values from 1000 to 100 or 100 and 1000 then we will interchange the values and see how the program works.

So let us go through the program line by line and this is one of the methods by which you also debug your program see the the writing shell script is excellent but one of the things that you should do while writing a shell script is to do a line by line activity ok this is a technical term for this known as the code walk through ok what I do is I actually behave like a computer and then see whether my code works as desired.

So let us do this let's start with line number 3 so the quantity 1 is assigned the value of 1000 and quantity 2 is assigned the value of 100 so let's remember this so the next statement actually echo saying that we are comparing quantity 1 with quantity 2 now so third statement line number 8 says that if quantity 1 and if you'll look at this this minus e eq which says that equal to.

So this tries to check whether quantity 1 is equal to quantity 2 ok and if you will look at this look at the space here after the if now this is very important I am insisting again and again

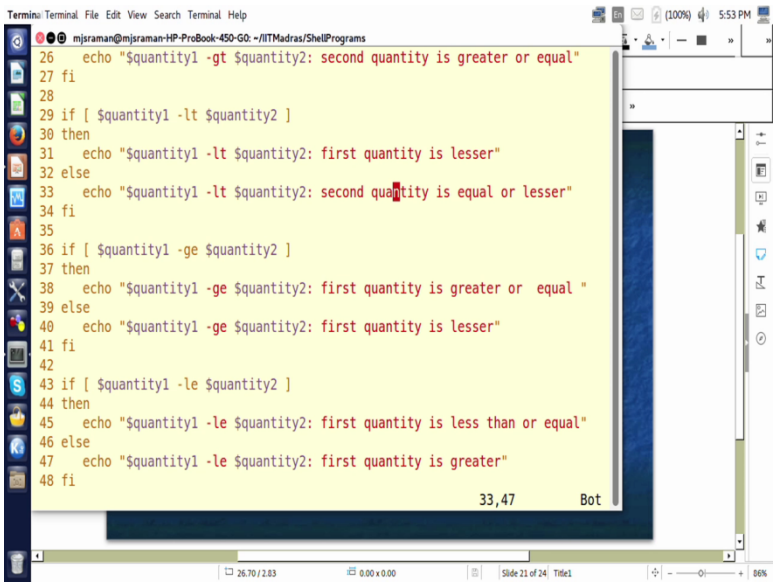
because people tell to forgets this and then it gives lot of errors and if you look at this if quantity 1 is equal to quantity 2 then says both are equal if not they say both are not equal.

So in our case if quantity 1 is 1000 and quantity 2 is 100 then they are not equal therefore we believe that this else part will be printed so let's remember this because when we run this program we should see whether whatever we had gone through as code walk through and whatever is printed program both are the same.

Now coming to line number 15 we are using the minus one that is not equal to quantity so if you look at this then it says quantity 1 is not equal to quantity 2 because 1000 is not equal to 100 then this line will be printed that is both not equal will be printed and let's move on so if you look at this then we will come to the line number line number 22 here these tries to find out whether quantity 1 is greater that quantity 2.

And obviously we had given quantity 1 the value of 1000 and quantity 2 the value of 100 therefore it will print the first quantity is greater moving on to line 29 we are trying to see whether the first quantity is less than the second quantity the first quantity is 1000 the second quantity is 100 and obviously 1000 is not less than 100 therefore what will it print, it will print make a guess so since 1000 is not less than 100 it will print second quantity is equal or lesser is it true let's see let's execute the program and see please note it down.

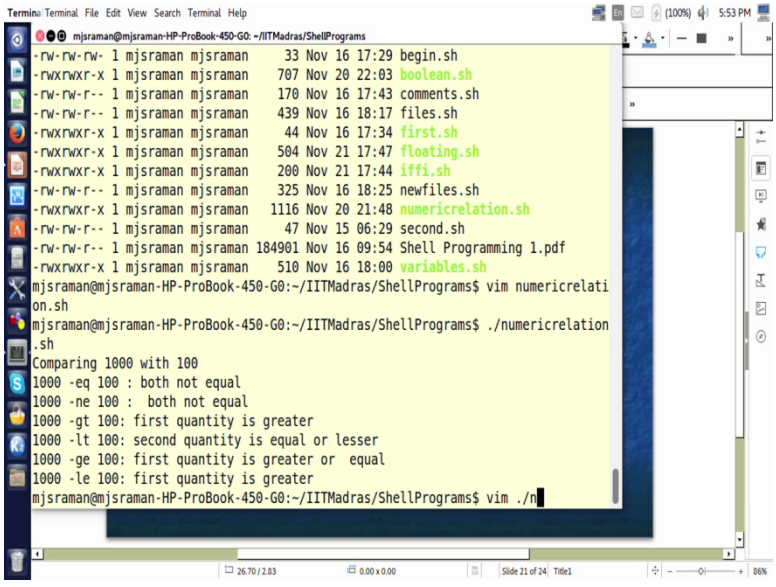
(Refer Slide Time: 12:55)



Now coming to line number 36 we are trying to check whether quantity 1 is greater than or equal to quantity 2. Similarly we are trying to check whether 1000 is greater than equal to hundred which is true therefore it will print first quantity is greater than or equal.

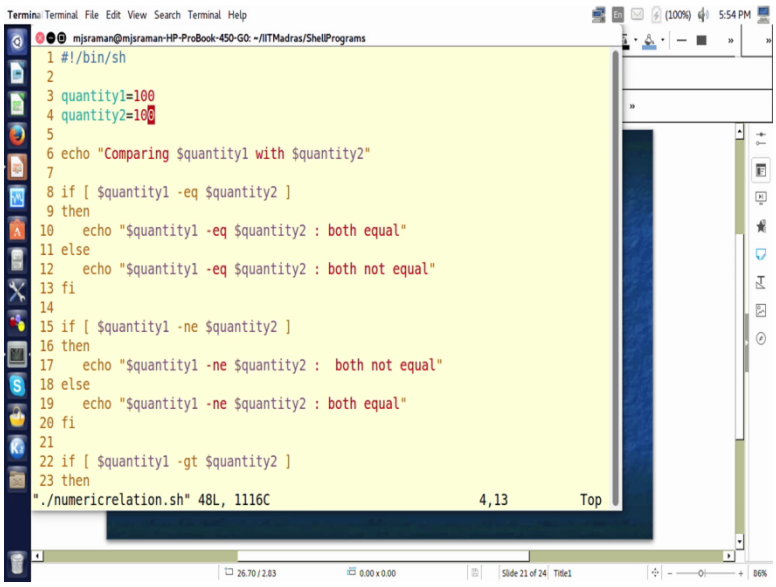
And finally we are trying to see whether first quantity is less than or equal to quantity 2 we obviously know that the first quantity is greater than equal to therefore it will print first quantity is greater. So let us see except for this line we are pretty clear about rest of the lines hopefully let's run this program and then see.

(Refer Slide Time: 13:30)



So we will run this program called a numeric relation and it says both are not both are not equal I think we've got this correct we got first quantity is greater correct ok and second quantity this is I think we got it correct too and I think first we also got this correct so we got everything correct here ok? So this is one way to identify whether your program is working right or not of course this is only one of the test case so what we should do here is?

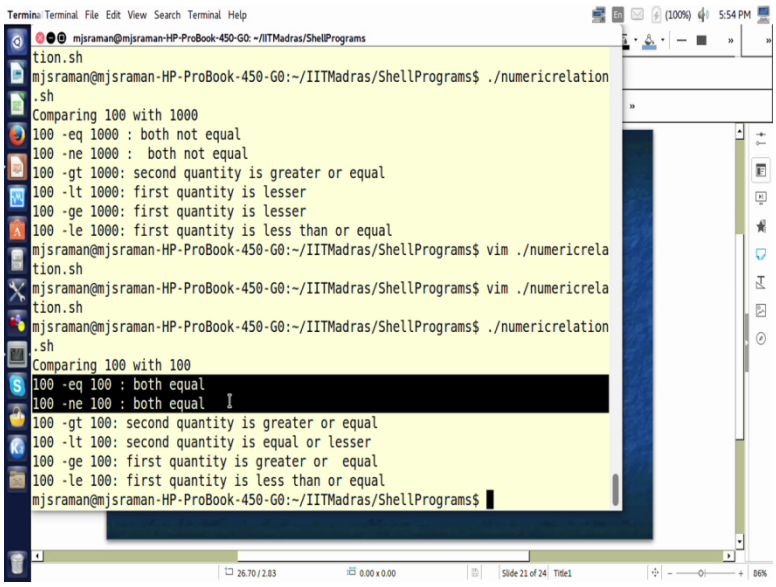
(Refer Slide Time: 14:10)



we should also look at the next test case which essentially uses interchanges you you put 100 here and then put 1000 here and then run this program so you should exactly print the opposite or what it printed last time hopefully it works so obviously so both are not equal both are not equal so it say second quantity is greater.

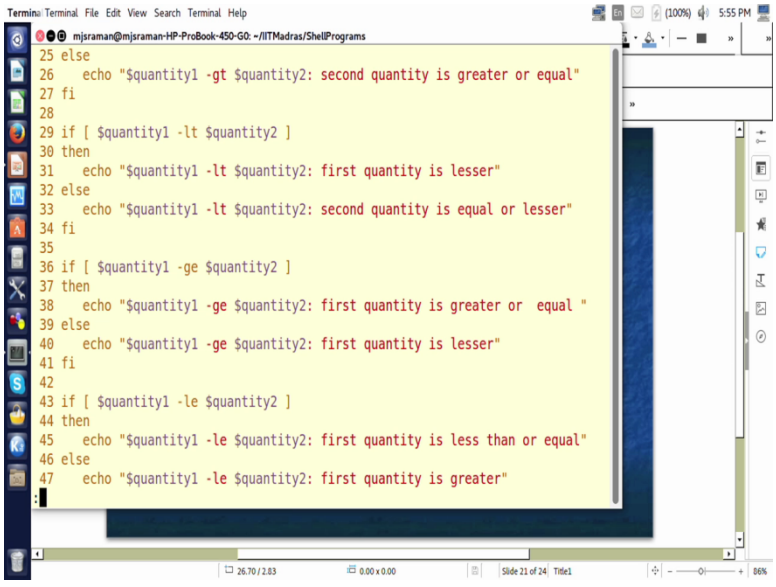
So it say first quantity is lesser first quantity is lesser and so on and finally what we can do is we can print both the values to be the same so I take 100 100 now you'll see something so what we can do is we can run this.

(Refer Slide Time: 14:49)



So this is something so here it says both are equal ok so mathematic so probably I mean why so does it look like a 1 because 100 not equal to 100 ok if it is true then the answer will be different since it is false ok 100 is equal to 100 we are printing both are equal so this also prints correctly ok and then this way we can use the numeric operators now so let's take a look at this program first.

(Refer Slide Time: 15:34)

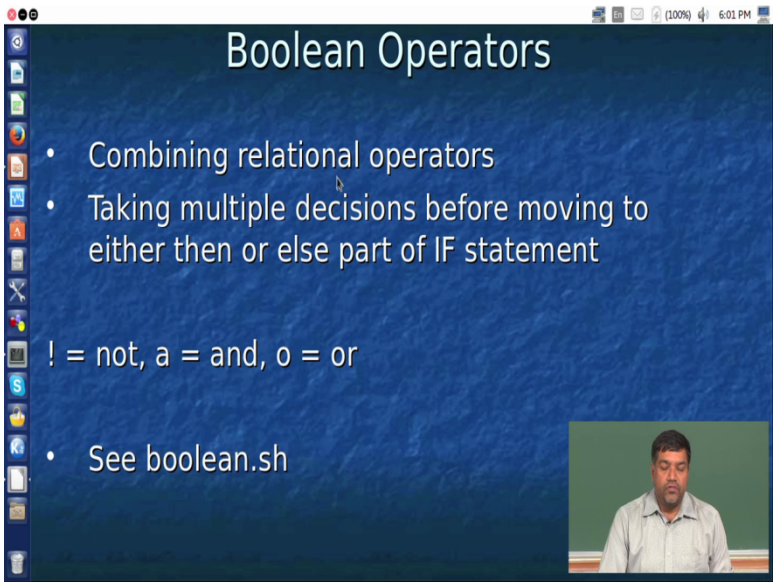


So hope we've clear with this program ok so I would suggest that you type these programs so it is nice to to listen to lectures but scripting only you can do scripting only when you type it in your computer and then check the results.

Now moving on let us so we saw this we also saw the three cases the first case where the first quantity was less than second quantity in the second case the second quantity was less than first quantity and so on and then both quantities being equal.

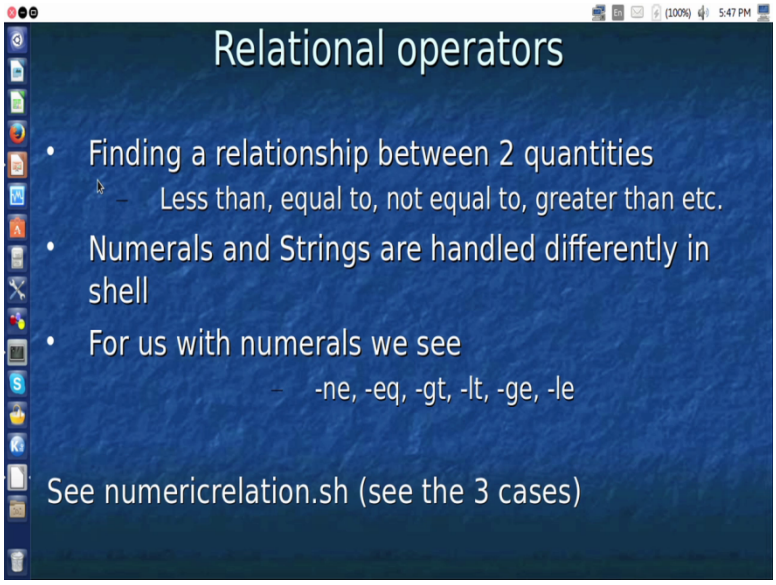
Now you should understand that when you write a shell script you should not only write the code but you should also do a code walk through as well as run some unit testing so what we did was a sample of a unit test to ensure that our shell scripting is correct ok.

(Refer Slide Time: 16:22)



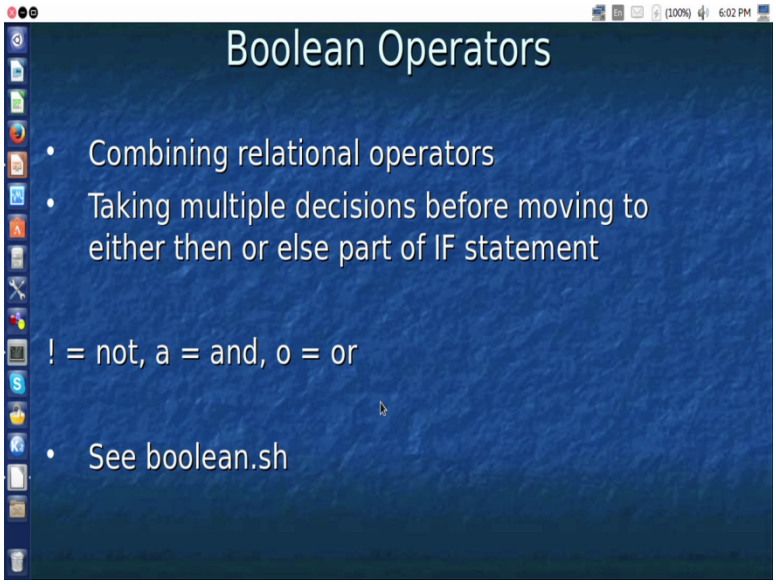
Now let's move on to Boolean operators, so why do we need Boolean operators ok the three Boolean operators that are given are not and and or and these are represented by the exclamation mark as given here the exclamation mark is represents not the a represents and and the o represents r now why do we need these three Boolean operators 1 the first reason is this you can combine the relation operators and take a Boolean decision either true or false.

(Refer Slide Time: 16:55)



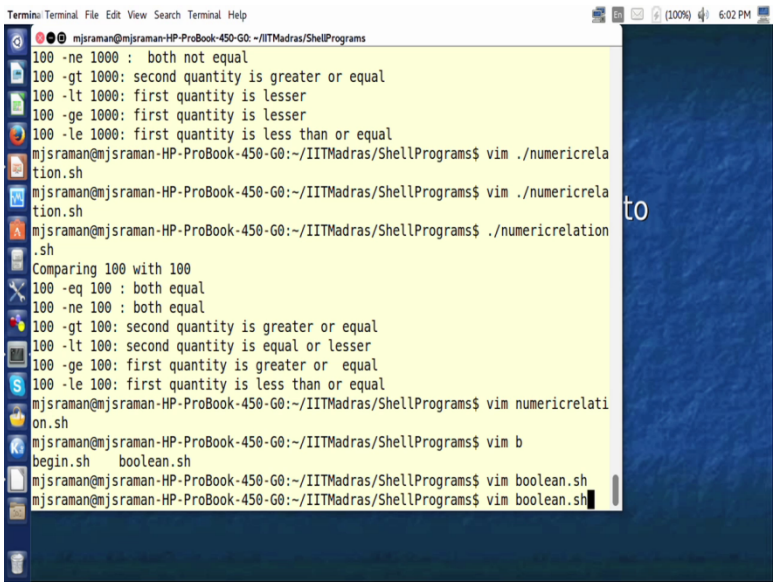
So if you look at what we did with the relation operators we were trying to find out whether one quantity is equal to the other quantity so when we compare two quantities and ask a question whether a quantity is equal to the other quantity the obvious answer could be either yes it is equal or it is not equal so when it is equal we say that with the condition that e equal to condition then we say that this statement becomes true.

(Refer Slide Time: 17:22)



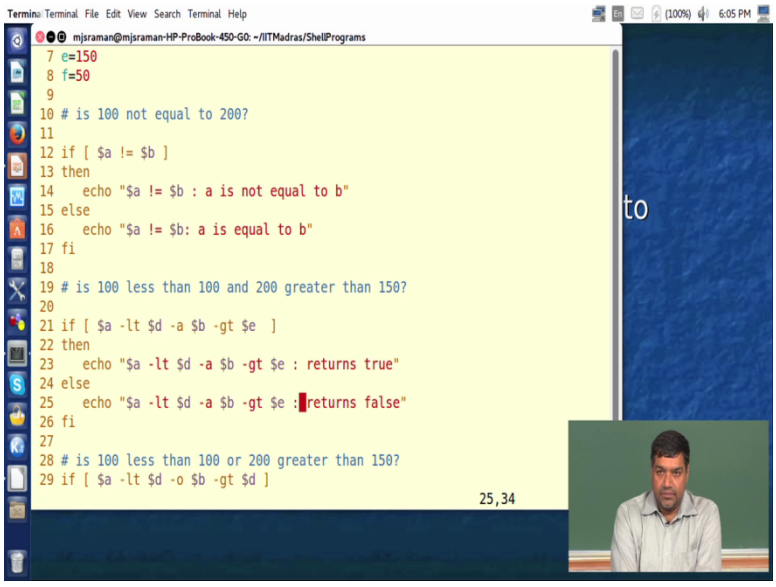
So we can combine such multiple decisions using the relational operator so the idea is so we will see this and we can insert this decision making process into the if statement part we will still see this with an example.

(Refer Slide Time: 17:42)



So what we have is a program called Boolean dot sh, ok?

(Refer Slide Time: 17:47)



And in this case we are trying to have 6 variables a b c d e f ok the variable name should be meaningful but since this is an example we've taken this variables names to be a b c d e f in

your programs we expect meaningful names to be given to these variables so let's start with our program so I assume 8 be value of 100, b to be 200 and c to be 1000.

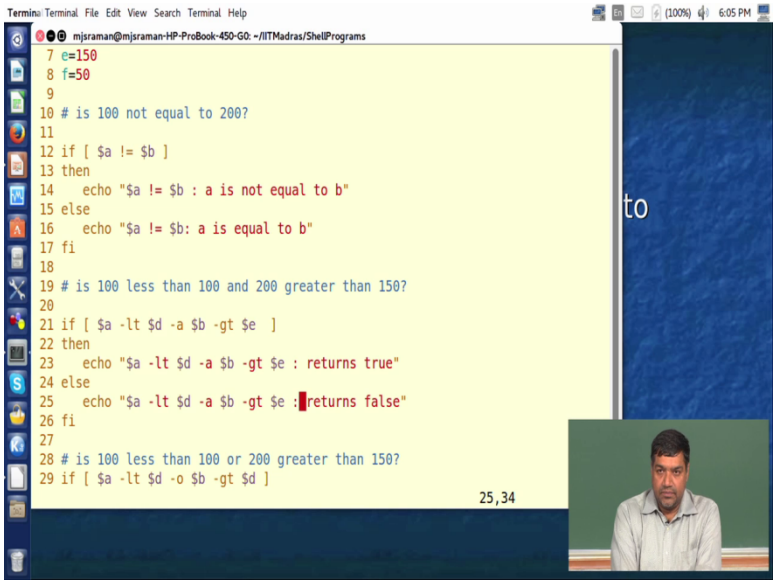
So let us take this statement 12. So we are trying to find out whether a is not equal to b ok? That is 100 is not equal to 200 yes 100 is not equal to 200 therefore this needs to true and if it is true the condition will evaluation of the condition becomes true then the then part is executed. Therefore we will get this printed.

Now the most interesting line now here is line number 21. So here you look at this we are joining 2 relational operators together using the Boolean operator. So if you look at this quantity dollar a is less than d ok?

This is the first condition the relational operation now dollar b is greater than dollar e this is the second relational operation and both of this are combined with an and condition. So what happens is that in this case there are four possibilities so a can be this first expression ok a minus lt dollar d can be 2 and this can be 2 these both are 2 then this 2 statement get executed.

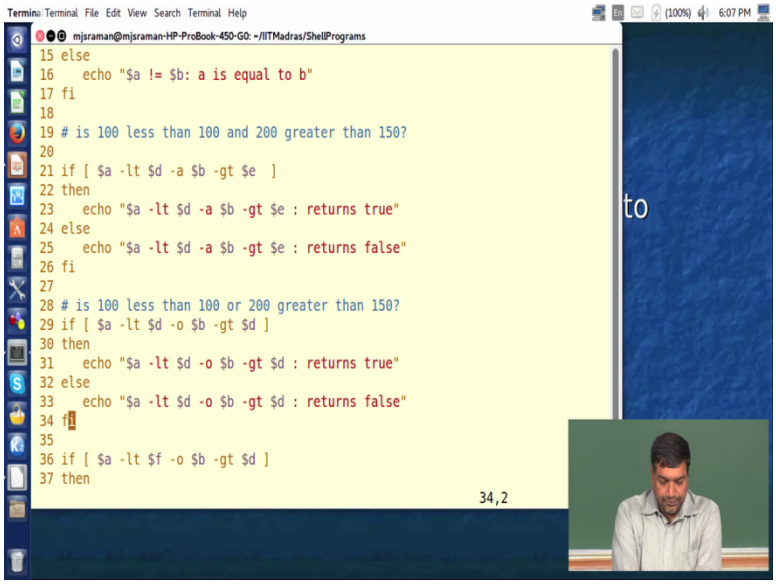
If a less than 3 is less than dollar d is false and b greater than dollar e is false then we will execute this statement which says return false. If the first relation operator evaluates to be true and the second operator evaluates to be false then false will be printed. Similarly if this evaluates to be false and the second relational operator evaluates to be true even then false will be printed. So this exactly follows the Boolean logic of an and gate.

(Refer Slide Time: 20:12)



Similarly the next statement follows the Boolean logic of the OR gate which tells you that either this relational operation a less than d or b greater than d if any one of them is true or both of them are true then it will return true the statement will return true in any other condition it will return false. Similarly in this case we are again using the OR condition ok? And we can also use the NOT condition as given that we leave it as a exercise so here is an example of we also use in NOT condition if you remember we have also used this during the relational operator numeric comparison case equal to or not equal to.

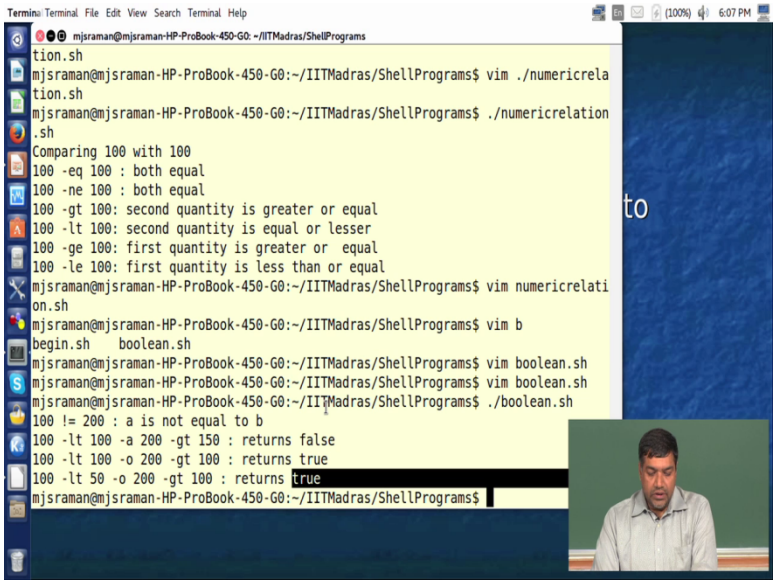
(Refer Slide Time: 20:55)



Now let us try to look at this example ok so is 100 not equal to 200 yes 100 not equal to 200 therefore what will print here if a is not equal to b then if 100 is less than 200 and 200 greater than 150 so 100 is definitely not less than 100 because 100 and 100 are equal and 200 is definitely greater than 100 therefore this is an AND condition so the first condition is false and the second condition is true since we are combining this with an AND the result should be a false.

And the third one is 100 less than 100 or 200 greater than 150, so 100 is not less than 100 , 100 is equal to 100 ok? And 200 is definitely greater than 150 and we are combining with an OR condition therefore it should return a true and then the last statement if a is 100 and f is 50 then the first statement is false and if b is 200 and d is 100 next statement is true therefore it will return true. So let us run this program and see whether we are getting this answer.

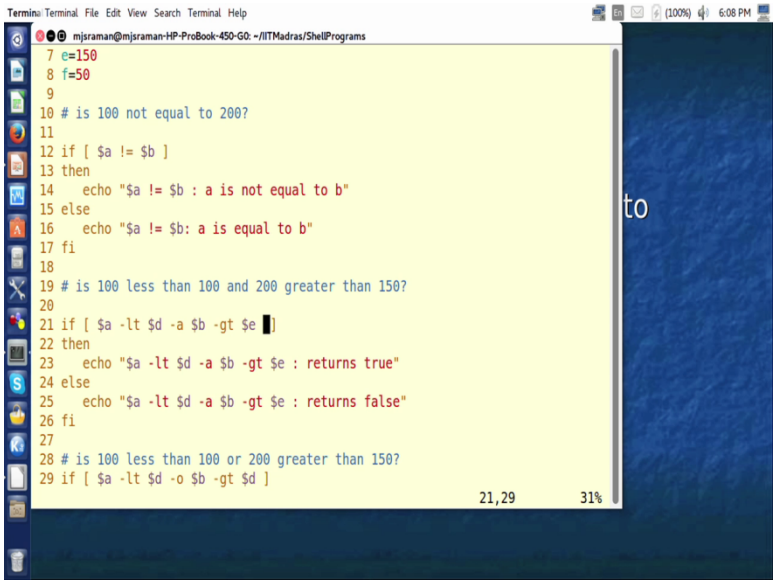
(Refer Slide Time: 22:07)



So if 100 is not equal to 200 so look at this 100 less than 100 is false and therefore the whole statement returns false now 100 less than 100 is false but 200 greater than 100 is true therefore the whole statement returns 2 100 less than 50 is false and 200 greater than 100 is true therefore the statement returns true.

So in this way we can make conditions more and more complex by combining the relational operators and the Boolean operators. One thing that we should remember is ok it is all nice to write such complicated programs but the user should be able to understand the logic that you write therefore the first rule in any programming is that you should avoid very complex conditions, if there are very complex conditions you need to break down into simpler conditions and then do the coding.

(Refer Slide Time: 23:04)



So for example a condition like this could as well be split and this is of course there are only two conditions you can also add OR and AND and so on that would be lot of larger Boolean expressions that you ca evaluate.

But it is always better that you do not have more than two conditions otherwise it becomes very the program becomes very complex for you to read and understand and if there are any problems with coding too it is very difficult to debug sunk kind of problems. Logical errors are always difficult to debug, so please be very careful when you are using these kinds of Boolean operators.

Thank You!