

Introduction to R Software
Prof. Shalabh
Department of Mathematics and Statistics
Indian Institute of Technology, Kanpur

Lecture – 09
Missing Data and Logical Operators

Welcome to the next lecture on Introduction to R Software. Now, in this lecture we are going to understand that how to deal with the missing values in R and how to handle the logical operators in R software. So, first let us try to understand the missing data.

(Refer Slide Time: 00:38)

Missing data

R represents missing observations through the data value **NA**

We can detect missing values using is.na

```
> x <- NA # assign NA to variable x
> is.na(x) # is it missing?
[1] TRUE
```

R Console

```
> x <- NA
> is.na(x)
[1] TRUE
> |
```

2

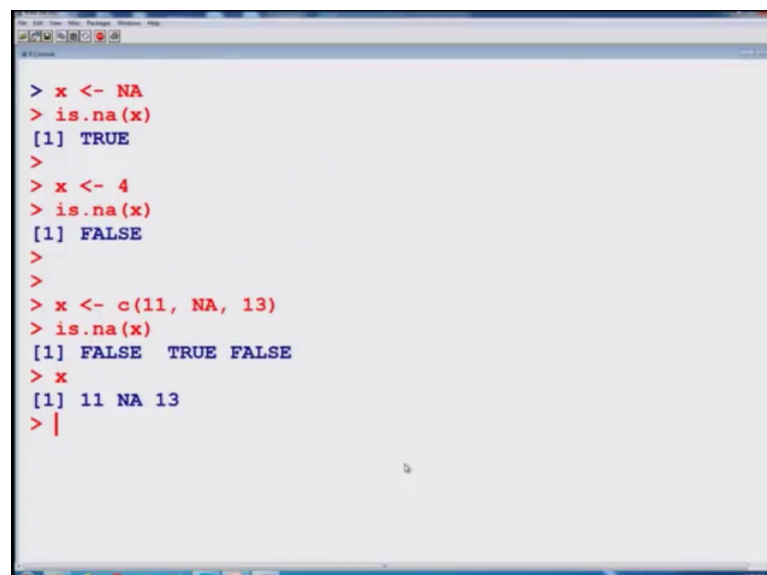
The first question comes what is missing data well it is possible that suppose I have got 5 values and one of the value is missing for a while. So, now, I have 2 options whether I try to delete it and I try to handle 4 data values or second option is that I declare that this value is missing. Then the advantage will be that at least anybody who is handling with that data set will come to know that this particular value is not available. So, now, whenever we are trying to do calculation this is a better option to report the truth if data is not available that should be reported as not available and then the question comes how we can do the mathematical operations over such a condition. So, now, when we come to the R, in R the missing data is represented through the 2 letters NA, NA means not available.

Now, the second question comes suppose you are dealing with a very huge data set suppose there are 5000 values or 20000 values, you cannot scroll through with all the data sets to know whether the data is available or not. So, first question comes that how to know whether there is any missing data in the entire data set or not for that in R we have a command which is something like `is.na`, is dot not available and then inside the bracket we try to give the name of the data set or name of the variable in which the data set is contained right.

So, just for the sake of illustration we try to take one variable say here `x` and we intentionally assign it a value `NA`, `NA` is a special value that is given only to denote that the value is missing and the value is not available. Now I try to operate this command `is.na` and inside the bracket I will write the variable name `x` this means I want to know is there any value which is missing in `x` and the answer comes here `true`; that means, this is true that there is missing value in `x` and that we can verify here right.

So, before we go further let us try to do it over the R console and try to see do we get this thing or not. So, I try to copy this data set over here and I am trying to check whether `is.na(x)` answer comes out to be true.

(Refer Slide Time: 03:57)



```
> x <- NA
> is.na(x)
[1] TRUE
>
> x <- 4
> is.na(x)
[1] FALSE
>
>
> x <- c(11, NA, 13)
> is.na(x)
[1] FALSE TRUE FALSE
> x
[1] 11 NA 13
> |
```

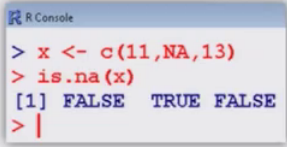
And suppose I give here the value here `x` say here 4 and then I try to check is there any missing value in `x` answer comes out to be false that means, there is no missing value.

(Refer Slide Time: 04:20)

Missing data

Now try a vector to know if any value is missing?

```
> x <- c(11, NA, 13)
> is.na(x)
[1] FALSE TRUE FALSE
```



3

Now let us try to operate this thing over a set of vectors. So, I try to take here 3 values first is 11 second is not available and third value here is 13 and I try to combine them inside the vector x. Now I want to check is there any value missing in this x or are there any values which are missing in x. So, I try to operate this command is dot na and I try to write down here the name here name of the variable x and I get here an outcome like false true false.

What does this mean this false is pertaining to this 11, this true is pertaining to this na and this false is pertaining to this 13. This means when I am asking is there any missing value in x for the first value answer comes out to be false; that means, there is no missing value. For the second value which is actually missing the answer come on to be true; that means, yes the value is missing, and for the third value which is a 13 and that is available that is not missing the answer comes out to be false; that means, the value is not missing. And here is the screenshot of the same thing, but still let us try to do it just to gain some confidence whether it really gives this thing or not.

So, I try to copy this data set over here and I try to say here is na dot x you can see here this comes out to be false to and false.

(Refer Slide Time: 06:11)

Example : How to work with missing data

```
> x <- c(11,NA,13) # vector
> mean(x)          11+NA+13 ??
[1] NA           ↙
                    not available 2
> mean(x, na.rm = TRUE) # NAs can be removed
[1] 12           ↘ remove
                    11+13
                    2 = 12
```

The null object, called **NULL**, is returned by some functions and expressions.

Note that **NA** and **NULL** are not the same.

NA is a placeholder for something that exists but is missing.

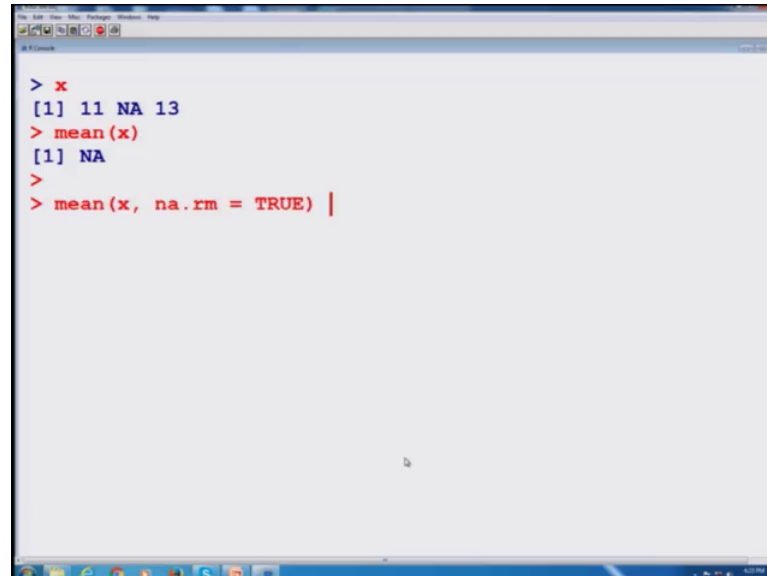
NULL stands for something that never existed at all.

Now next question comes when there is a missing value in the data then how one can do the mathematical calculations. For example, I try to take the same data set 11 NA and 13 which are combined inside the variable x and I simply try to find out the mean of this x. So, you can see here mean of x this is the command to find out the arithmetical mean and you can see here if I try to find out arithmetical mean this is 11 plus NA plus 13 divided by 2, do you think that will I get any value here. NA cannot be added to any value from the mathematical point of view that is only an algebra value you that is not an numerical value. So, the answer comes out to be here NA. Now the question is this what you really wanted? You wanted actually the arithmetical mean of those values which are available in the vector.

So, I have our way out. There is a command that when I try to find out here the mean I should write here mean of the vector in which all the values are contained and then I try to write down here na dot rm, na dot rm means - na means not available, rm means remove and I am saying that na dot rm is equal to true. This means in case if there is any missing value in the data set please remove it and this I am saying that this statement is correct. So, I am requesting my operator to remove the missing data from the vector and find out the arithmetical mean of those values which are available. So, now, it is trying to give me say the mean of values which are available that is 11 and 13 divided by 2 and this comes out to be here 12. Let us try to do it over the screen and try to see whether it is correct or not. So, I try to find out here well I can just copy it here this thing and suppose

this x is already here. So, I can clear the screen. So, for x is here and I try to find out here mean of x. So, you can see here this gives me not available.

(Refer Slide Time: 08:47)



```
> x
[1] 11 NA 13
> mean(x)
[1] NA
>
> mean(x, na.rm = TRUE) |
```

But now, if I try to find out here mean where I am trying to say na dot rm is equal to true then I get here 12 so; that means, this mean has been calculated after removing the missing values. Well now here you have to be little bit careful. There are 2 things one is NA and another here is N U L L, NULL when you will be using some more functions sometime you will get the answer as N U L L, NULL this means what – NA and NULL they are not the same, please remember please note that NA and NULL they are not the same. The interpretation of NA is that NA is a placeholder for something that was expected to exist, but somehow it is missing whereas, when we are writing out here null this means that something which never existed that even was not available right from the beginning. So, there is a difference between missingness and not available, the same difference between NA and null also continues further ok.

So, now we come to say this another topic which is about logical operators first question comes what is a logical operator? For example, if I write 8 plus 9 that will give me an answer to 17, that is the mathematical operator, but if I want to simply know whether 8 is greater than 9 or not, then this greater than is an is a logical operator. Suppose I want to know whether the sum of some number is greater than or smaller than or not equal to

certain value or not then you have to keep in mind here that here I am not interested in knowing that what is the value, but I simply want to compare them.

For example, we know that 10 is greater than 2, but 100 is also greater than 2. So, the difference between 10 and 2 is only 8 whereas, the difference between 100 and 2 is 98, but here I am not interested in the magnitude, I just want to know whether it is greater than or not. So, these operators like greater than, less than, greater than equal to, less than equal to and not equal to they are essentially our logical operators. And logical operators are also very very useful in doing any programming for example, I would like to repeat our program until some conditions for example, x greater than 2 wholes. So, these types of thing can be accomplished using the logical operators. So, here we are going to first understand what are the different logical operators available in R and then we will see how to use them.

(Refer Slide Time: 12:28)

Logical Operators and Comparisons

The following table shows the operations and functions for logical comparisons (True or False).

TRUE and FALSE are reserved words denoting logical constants.

Operator	Executions
>	Greater than
>=	Greater than or equal
<	Less than
<=	Less than or equal
==	Exactly equal to
!=	Not equal to
!	Negation (not)

$\geq \Rightarrow >=$

$\leq \Rightarrow <=$

$==$

$!=$

!

5

And before we go further in the logical operators and their comparison we have to keep in mind that there are 2 words one is TRUE and other is FALSE, TRUE and FALSE in capital letters they are reserved. Reserved means they have a particular meaning in R programming. So, please do not use the words true and false to name any variable or say anything they are reserved for R programming that we have to keep in mind.

So, now, in this case you can see here there are several operators first is greater than this has a same symbol what we use in practice. Another symbol is greater than or equal to

now this symbol is written in R is greater than equal to like this similarly smaller than or less than that that is denoted by the classical symbol and less than or equal to this is denoted by say less than equal to like as here. When we want to compare whether the value is exactly equal to something or not then in order to make comparison for exactly equal to we have a symbol here which is containing 2 equality signs like this, and when I am trying to use 1 exclamation sign with equality this is meaning that not equal to and when I am trying to use only the negation; that means, not then I simply use here the sign simple exclamation sign. So, these are various logical operators which are used in R.

(Refer Slide Time: 14:32)

Logical Operators and Comparisons

Operator	Executions
<u>&</u> , <u>&&</u>	and &
<u> </u> , <u> </u>	or

- The shorter form performs element-wise comparisons in almost the same way as arithmetic operators.
- The longer form evaluates left to right examining only the first element of each vector. Evaluation proceeds only until the result is determined.
- The longer form is appropriate for programming control-flow and typically preferred in if clauses (conditional).

6

There are 2 more operators which I have written separately and there is a reason. Suppose I want to make a comparison by joining 2 statements or more than 2 statements. So, the statements can be joined by 2 possibilities one is and another is or for example, I can say where that x and y both holes or other alternative is this x or y any one of them wholes. So, for that I have here 2 types of symbol for and I use the same letter and, but you can see here I have written here 2 types of a statement one is single and here I am trying to write here and 2 times.

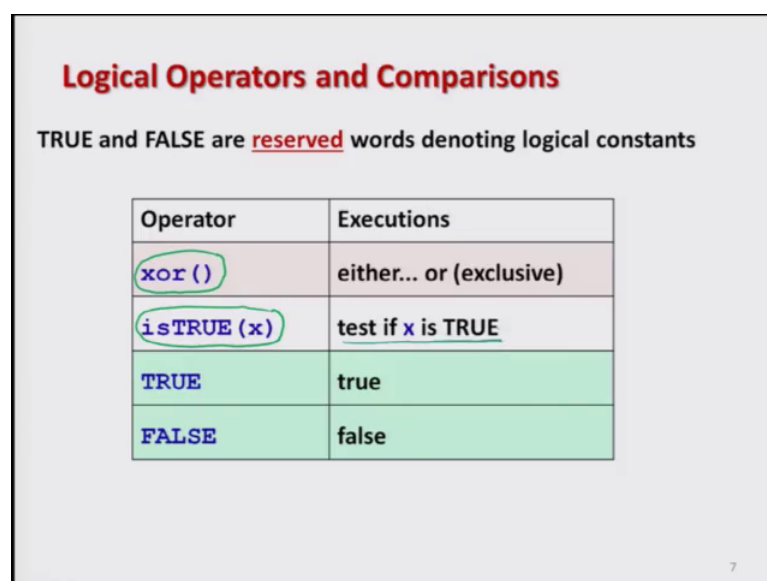
And similarly when I want to use the statement R then for R I am using this vertical bar vertical sign. So, here I am using only one sign and here I am using two signs. Now there is a difference between the use of 1 symbol and 2 symbols the 1 symbol or which I am calling here says the shorter form this works element wise and this makes element wise

comparison in almost the same way just like any arithmetic operator does. Whereas, when I am trying to use the 2 letters like a double and or said double vertical lines then this I am calling as a longer form and this evaluates a vector from left to right. This we have to keep in mind this is very important. This evaluates a vector or examines our vector form from left to right, but it takes care only the first element of each vector this I will try to show you with an example, but here what we have to understand that whenever we are trying to use a vector which has more than 2 values then possibly our possibly from our common sense we think that each of the comparison are made for each and every element in the vector or see between 2 vectors this is not correct.

When I am trying to use the longer form here this is comparing only the first element of the vector from say another vectors first element this is what we have to keep in mind right. Actually this longer form this means that double letters they are more appropriate when we are trying to the program the control flow and typically it is preferred in conditional like as clauses that we will try to see later on when we are trying to do the (Refer Time: 17:31) and other aspect in the programming.

So, now let us try to take a look over some other operators where we are trying to make a comparison of 2 statement or simple comparison with respect to true and false right.

(Refer Slide Time: 17:40)



Logical Operators and Comparisons

TRUE and FALSE are **reserved** words denoting logical constants

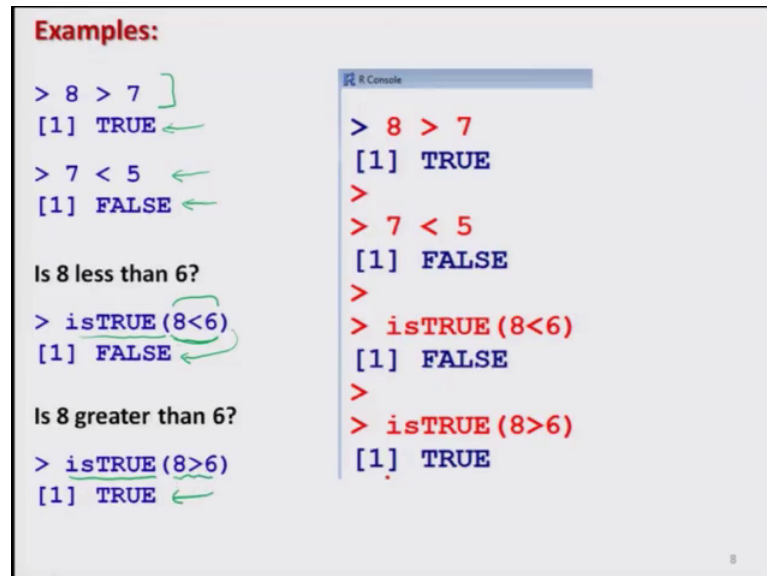
Operator	Executions
<code>xor ()</code>	either... or (exclusive)
<code>isTRUE (x)</code>	test if <code>x</code> is TRUE
<code>TRUE</code>	true
<code>FALSE</code>	false

7

First thing suppose I want to check the truthfulness of more than one statements then this is done by XOR and inside the bracket we write the statement what we want to compare.

Similarly there is another statement is true and inside the bracket say for example, if there is a variable x then it is actually testing if x is 2 or not, that we already have used, but I am just repeating it just for the sake of completeness, and true and false these are the 2 letters which are written in capital they are indicating whether that statement is true or false respectively.

(Refer Slide Time: 18:39)



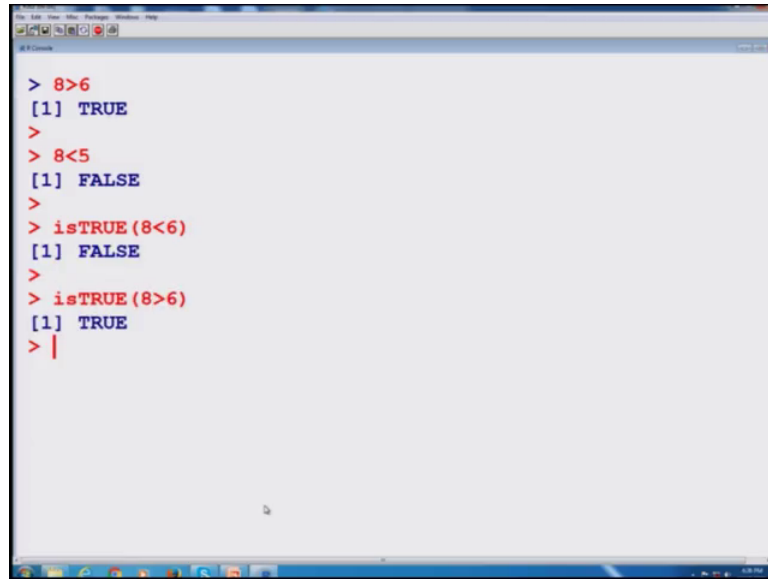
```
Examples:
> 8 > 7
[1] TRUE
> 7 < 5
[1] FALSE
Is 8 less than 6?
> isTRUE(8<6)
[1] FALSE
Is 8 greater than 6?
> isTRUE(8>6)
[1] TRUE

R Console
> 8 > 7
[1] TRUE
>
> 7 < 5
[1] FALSE
>
> isTRUE(8<6)
[1] FALSE
>
> isTRUE(8>6)
[1] TRUE
```

So, now, let us try to take some example and try to understand the application of these operators. So, I am trying to take a very simple example suppose I want to know whether 8 is greater than 7 or not. So, if you try to type here 8 greater than 7 the answer comes out to be true and that is obvious yes 8 is greater than 7. Similarly if I want to check whether 7 it is smaller than 5 the answer is no, so the answer comes out to be here false.

And similarly if I want to see 8 is less than 6 whether this statement is true or false, so I try to write down here is true and inside the bracket I try to write down this statement 8 less than 6 the answer will come out to be here false. And similarly if I want to check whether 8 is greater than 6 or not, so I will try to write down here is true 8 is greater than 6 and then as soon as I enter I get the answer here true yes it is correct. So, let us try to do this example over the R console and try to see how these things can be done right.

(Refer Slide Time: 20:05)



```
> 8>6
[1] TRUE
>
> 8<5
[1] FALSE
>
> isTRUE(8<6)
[1] FALSE
>
> isTRUE(8>6)
[1] TRUE
> |
```

So, now, if I say here 8 greater than 6 answer is true, if I say here 8 is smaller than 5 answer is false and similarly if I want to know whether 8 is smaller than 6 or not because the command is true inside the bracket is less than 6 and so it comes out to be false. Now in case if I want to check here whether 8 is greater than 6 or not answer comes out to be true. So, this is how you can see this logical operator works.

Now, you will be little bit surprising that why I am trying to take such a simple example what is there in this one means anybody can do it without any calculation, but try to consider a situation where you have a complicated expression for which you cannot know just by looking from your eyes whether this statement will give us a value which is greater than 0 or smaller than 0. For example, if I say here $x^2 - x + 2$ whether this expression for x equal to 1 will be greater than 0 or says is smaller than 0 and so on. So, I can do I can check all this statement within a programming language or within a function to just to check whether some condition whole true or not. So, these logical operators are useful over there.

(Refer Slide Time: 21:48)

```
Examples:
> x <- 5
> (x < 10) && (x > 2) # && means AND
[1] TRUE

R Console
> x <- 5
> (x < 10) && (x > 2)
[1] TRUE
```

Now, I am trying to take another example there are 2 statements which I would like to show here. So, here I am trying to take here a value here x less than hyphen 5 that is x equal to 5 and there is a operator which has hash say a double and double and which means here and that we had discussed earlier. So, now, I want to check that when x is equal to here 5 then the statement that x less than 10 and x greater than 2 is this statement true or false.

So; obviously, if I try to write down here 5 and here 5 here you can see here 5 is greater than 10 answer is yes 5 is greater than 2 answer is yes so; that means, 5 is less than 10 and 5 is greater than 2 the answer is yes and the answer comes out to be here true. And this is the same screenshot of the same operation which I can do on the R console also right. Now after this another thing I would like to show you here just continuing with the same slide.

(Refer Slide Time: 23:05)

```
Examples:
> x <- 5  x = 5

Is x less than 10 or x is greater than 5 ?
> (x < 10) || (x > 5)  # || means OR
[1] TRUE

Is x greater than 10 or x is greater than 5 ?
> (x > 10) || (x > 5)
[1] FALSE

R GUI (64-bit)
>
> (x < 10) || (x > 5)
[1] TRUE
>
> (x > 10) || (x > 5)
[1] FALSE
>
```

That suppose I try to take here x equal to 5 and I want to check whether x is x less than 10 or x is greater than 5. So, I can write down here is x less than 10 and this symbol here 2 bars they are indicating the statement or x is greater than 5. So, now, if you try to see I have taken here a value here of x as 5. So, 5 is less than 10 answer is yes and 5 is greater than 5 it is not, but it is equal to 5, but here the operator is R. So, if any of this statement or this statement is true the answer will come out to be true and that is what here is the R is telling us that the answer is true. I will try to operate it on the R console to illustrate it, but before that we try to understand it first.

Now suppose there is another question I want to know is x greater than 10 or x is greater than 5. So, I can write down here x greater than 10 and this operator R and x greater than 5, now if you try to see x is taking here value 5 so is 5 greater than 10 answer is no and is 5 greater than 5 then answer is no, so the answer is no or so answer no - no; that means, the this statement is false. So, now, you can see here I have taken here a value here x x was this right.

(Refer Slide Time: 25:00)

```
R Console
> x
[1] 11 NA 13
> rm(x)
> x
Error: object 'x' not found
> |
```

So, now, if you remember in the earlier lectures I had told you that you have to first remove the value here x. So, I am now removing here x and now x is nowhere. So, I try to clear my screen and then I am trying to put here x equal to 5 which is given in this example and I try to take this statement and I copy it here you can see here the answer comes out to be true and similarly if I go for the second say example the answer comes out to be false. So, you can see here I can do this type of logical operations over this R right.

(Refer Slide Time: 25:47)

Examples:

```
> x = 10 ✓
> y = 20 ✓
```

Is x equal to 1 ^{x=1?} and is y equal to 20? ^{y=20?} x=1
x=2

```
> (x == 1) & (y == 20) # == means exactly equal to
[1] FALSE
```

Is x equal to 1 and is y equal to 2?

```
> (x == 1) & (y == 2)
[1] FALSE
```

```
R Console
> (x == 1) & (y == 20)
[1] FALSE
>
> (x == 1) & (y == 2)
[1] FALSE
```

12

Similarly, I try to take here 1 more example. Suppose I try to take here 2 variables now x equal to 10 y equal to 20 and suppose I want to check is x equal to 1 and is y equal to 20 because I want to check. So, I can write this statement as say x equal to 1 now remember one thing when I am writing x equal to 1 this is a mathematical operator and when I am writing x equal to equal to 1 this is a logical operator. So, then I want to write x equal to 1 as a logical operator I have to write x equal to equal to 1 and why equal to equal to 20. So, this I am now trying to check whether this statement is true or not. So, here you can see I have taken the x value here as say 10 and y value here is say 20. So, thus 10 is equal to 1 answer is no this 20 equal to 20 answer is yes, but my joining operator is AND, so this is false and this is true because their combination will be false.

Similarly if I want to check is x equal to 1 and y equal to 2. So, I can write down here x equal to 1 and y equal to 2 using the logical operator and now if you try to see I have taken here the value x to be 10 and y to be 20. So, now, whether what you observe here whether 10 is equal to 1 answer is no this is wrong, whether 20 is equal to 2 again this is wrong. So, this is false again the second statement is false, so false and false is false. And here I am just trying to give you the screenshot so that you can practice yourself also right.

So, we stop here I have started a discussion on logical operator and in the next lecture I will try to continue with some more examples, with some more explanation and in the meantime I will request you try to take some examples from the books, from the assignment and try to practice them, till then good bye.