Optical Engineering
Prof. Shanti Bhattacharya
Amogh Manthalkar
Department of Electrical Engineering
Indian Institute of Technology, Madras

Lecture – 35
Introduction to Matlab

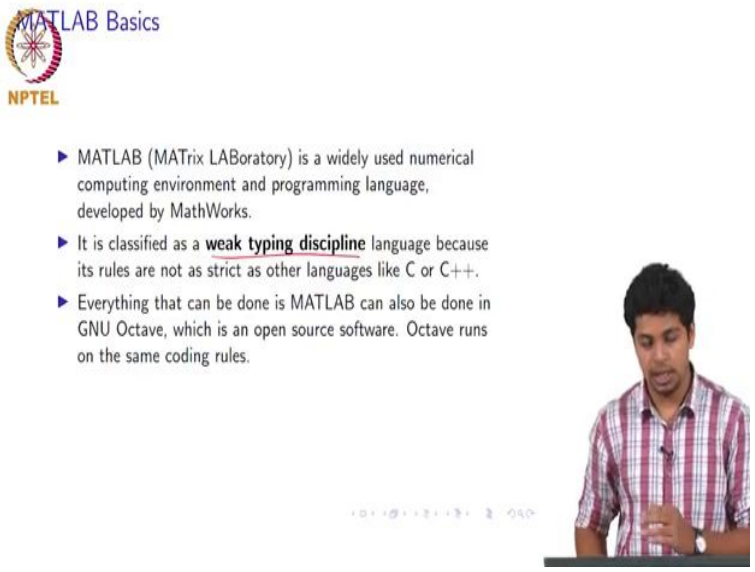
(Refer Slide Time: 00:25)

Hello, I am Amogh and I will be introducing MATLAB programming to you for the course Optical Engineering. This is the brief overview we will go through the basics once, then I will tell you a bit about the programming style in MATLAB; how you are supposed to write codes.

Then we will go through variables; how MATLAB interprets variables, then we will go through some basic operations that will be mostly used in our codes, then we will take a simple example using those basic operations then we will go towards conditional statements and iterative statements, and then we will take a little bit complicated example just to give you a sense of how to use these statements.

And then we will go towards a little bit advanced concepts like arrays and then I will tell you a bit about basic plotting of functions and data let us get into it.

(Refer Slide Time: 01:03)

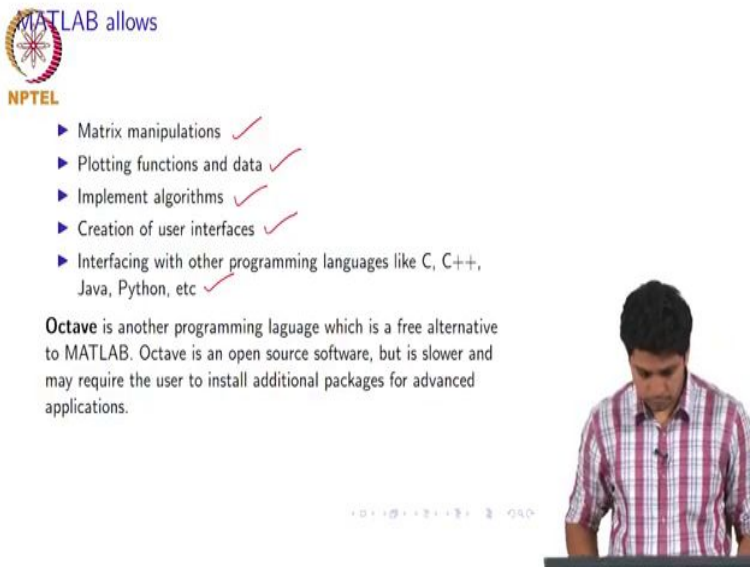


So, MATLAB or MATrix LABoratory is a widely used numerical computing environment and it is also an object oriented programming language; it is developed by MathWorks. And it was started sometime around 1970's and it was primarily used for control engineering, but now it is widely used in many other fields like machine learning and image processing.

It is classified as a weak typing discipline language and what it means is there are not many rules as in some other languages like C or C plus plus. I will tell you what it means later when we discuss an example.

And just like MATLAB there is one more software known as GNU Octave. It is basically the same as MATLAB only it is an open source software MATLAB requires a license and Octave runs using the same coding rules and also it can be run using the same program; you do not have to change any syntaxes. And MATLAB is freely MATLAB has to be licensed, but Octave is freely available on GNU website.

(Refer Slide Time: 02:14)

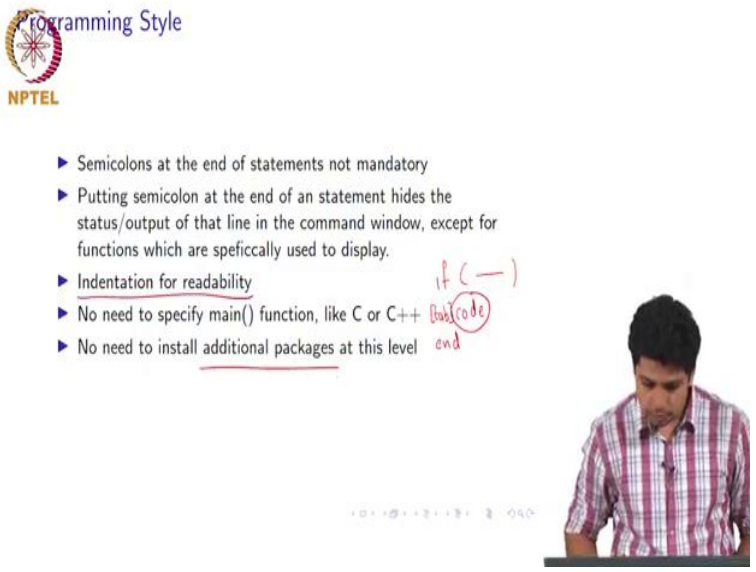


Now, we will just look at what MATLAB is used to do. You can manipulate matrices using MATLAB. What it means is you can create matrixes and then carry out operations on matrices that is the way MATLAB interprets any variables that you give it.

Then you can plot functions and data, you can implement algorithms, you can create user interfaces for interfacing your computer with some devices or use it to interface devices with each other, and you can also interface MATLAB itself with other programming languages like C plus plus and other languages. And you can also use it to interface with other programs themselves like COMSOL multiphysics and some other softwares and you mean you need to do that in order to simplify whatever operations you are doing.

And like I said octave is another language which is freely available it runs on the same code as MATLAB. The only difference being octave is an open source software, so it will not come with all packages pre installed. So, you might have to install some packages additionally, but that depends on the application. So, and at this stage we do not need to worry about installing any additional packages in octave. So, we can just go ahead with it.

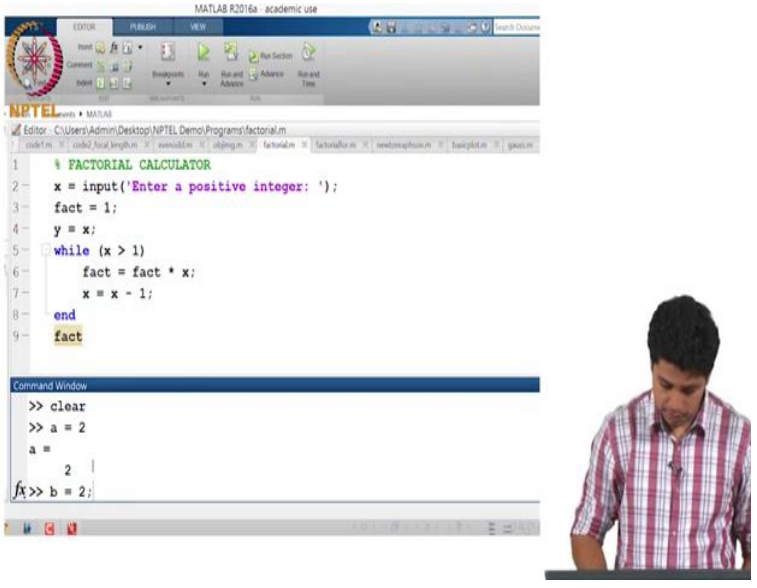
(Refer Slide Time: 03:31)



Now, we will discuss the style of programming in MATLAB. So, as we know in C plus plus or C we have to put semicolons at the end of statements, but in MATLAB it is not mandatory.

Now, what that means, is if you put a semicolon at the end of the statement it just hides it from the command window. Now, I will take this moment to introduce you to how a MATLAB interface actually looks like.

(Refer Slide Time: 03:53)



So, this is the screen of MATLAB and this is the area where you would enter the code it is called the editor, this is the area which is called a command window; here you can enter your code line by line and we will get executed this is the folder where your program will be stored this is workspace.

So, any variables you are using and you are executing the code the variables along with their values and size will be displayed here in this area. Now what I mean when I say that semicolon is not mandatory is; now let us say I use a variable a and I give it a value of 2, and I do not put a semicolon at the end of it.

What happens is in the command window it just sets a is equal to 2 again, but if I put a is equal to 2 and then a semicolon or let us say we will use another variable here b and we put b is equal to 2 and put a semicolon at the end of it then it does store the value of 2 inside b in as we can see in the workspace, but it does not display that is what the difference is.

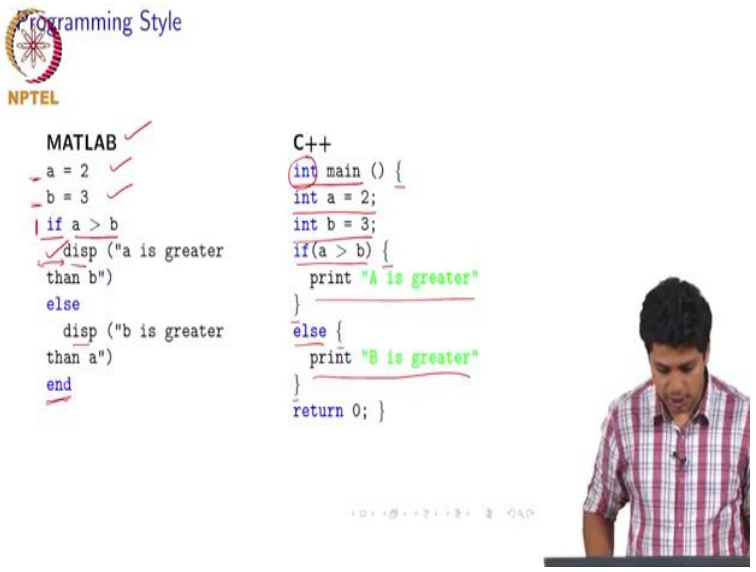
Then we use indentation in MATLAB just to make the program a little more readable. Now what that means is, when we use some conditional or iterative statements; what we do is there might be a situation where you are using code like this.

So, say there is a conditional statement of if and you put some condition here. Now what you want to do in MATLAB is if you want to include some statements within if the condition is satisfied then perform some code. So, what you do is you put a tab here, how that helps is you can now clearly identify what segment of your code is implemented if this condition is satisfied.

And then what you do is you put an end statement here now this indentation is only for convenience unlike python if you do not put indentation it does not make a difference the indentation is only for better readability or better understandability of the code that is what indentation here means. And unlike C or C plus plus you do not need to specify any main function.

So, you can just enter MATLAB and start writing your code in the editor. You do not have to worry about declaring or specifying functions like main and at this stage we will not need any additional packages even for Octave. So, that is a feature that should be kept in mind.

(Refer Slide Time: 06:37)



Now, if you look at the codes here in the MATLAB code this is how I have just put the code here a is equal to 2 assigned, b is equal to 2 b is equal to 3 assigned then there is an if condition here if statement if a is greater than b then it will display that a is greater than b. Now this is the syntax for displaying command disp is the function and you put a bracket and an inverted comma and you put whatever text you want to be displayed in that. Now you can see that there is a space here now this is what I mean by indentation and then there is an else statement then you just display that b is greater than a.

And you end it, but in C C plus plus or you what you would have done is you start the program with int main now what this int mains is your returning value will be an integer, then you put a curly brace.

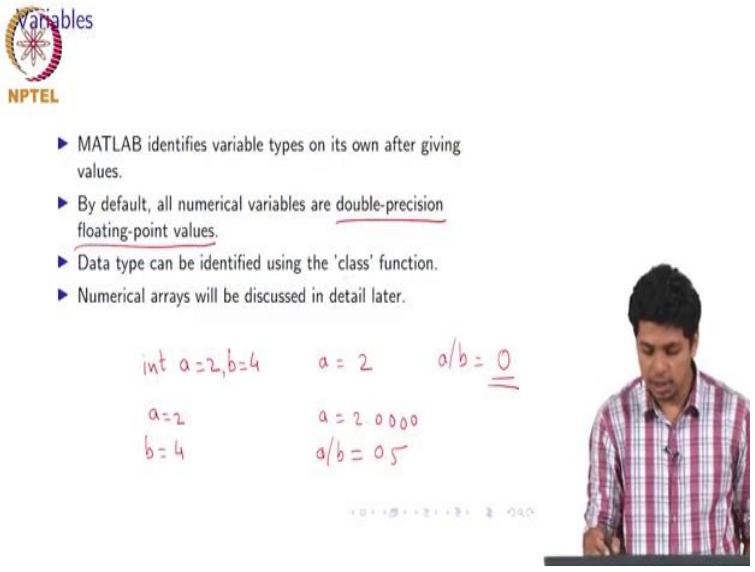
Then int a equal 2, an int b equal to 3 as you can see in the MATLAB code you did not put any variable type. MATLAB identifies the variable type on its own that is a feature of MATLAB. Then you put a condition and then you put again a curly brace you print a statement then you put a curly brace then you put a statement again you put a curly brace open then you print the statement and you put a curly brace.

So, in MATLAB the difference would be you just start an if statement you put your condition whatever statements you want to be executed you put those statements like display. You just

put else you put the second statement whatever you want in the else condition to be displayed and you put an end; this is how conditional statements in MATLAB work or say any statements in MATLAB work.

You do not have to put brackets for these you just have to indicate an end to your if condition that is the difference between C plus plus and MATLAB and as it turns out these differences actually help us a lot, we will discuss how exactly that happens. Now, we will briefly discuss variables.

(Refer Slide Time: 08:53)



Now, as we just saw in that snippet of a code that you do not need to specify the type of variable for MATLAB. It interprets the type of variable on its own, and by default if you enter any number in MATLAB then it is interpreted as a double precision floating point value.

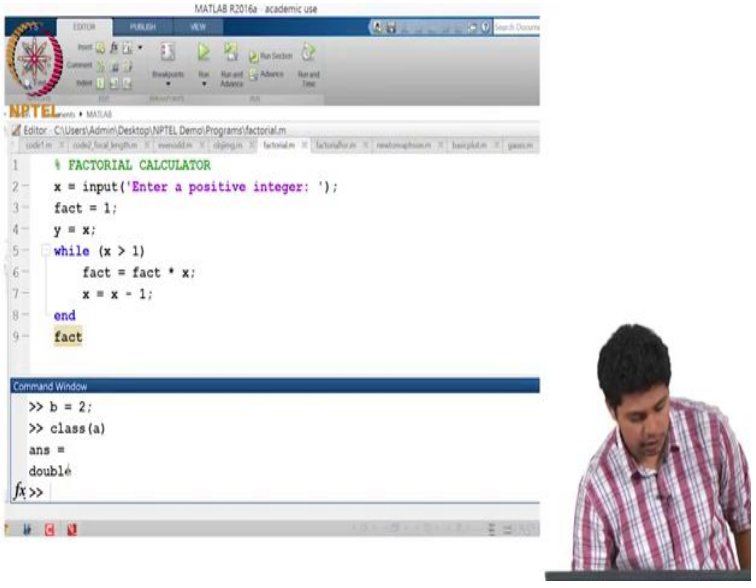
What that means, is if in C you would have said int a equal to 2 then your the a will store a value of 2, but if in MATLAB you say a equal to 2 then by default it will store a equal to 2.000. Now, where this makes a difference is if then you consider another say b equal to 1 in C plus plus, or b equal to 2 in C plus plus and we will just set b equal to 4 here.

So, b will store the value of 4 and then if you perform a division operation say a by b then C plus plus will give you an answer of 0, because both of these are integers and any operation done on 2 integer values should give you an integer output.

But in MATLAB if you just put b = 4 and you perform a division operation it will give you an answer of 0.5 that is the difference. So, by default any numerical value stored is stored as a floating point value and then data type can be identified using the class function. So, suppose you give some value to any variable and then you want to know what type of variable it is.

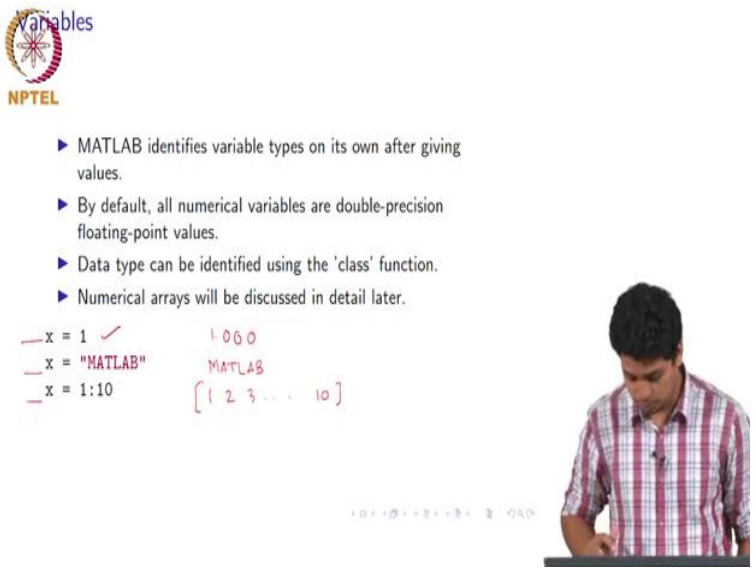
So, you just put class and then the variable name I will show how that happens. So, like we already have a and b values here in the workspace. So, we will just try to identify what type of variable a is.

(Refer Slide Time: 10:57)



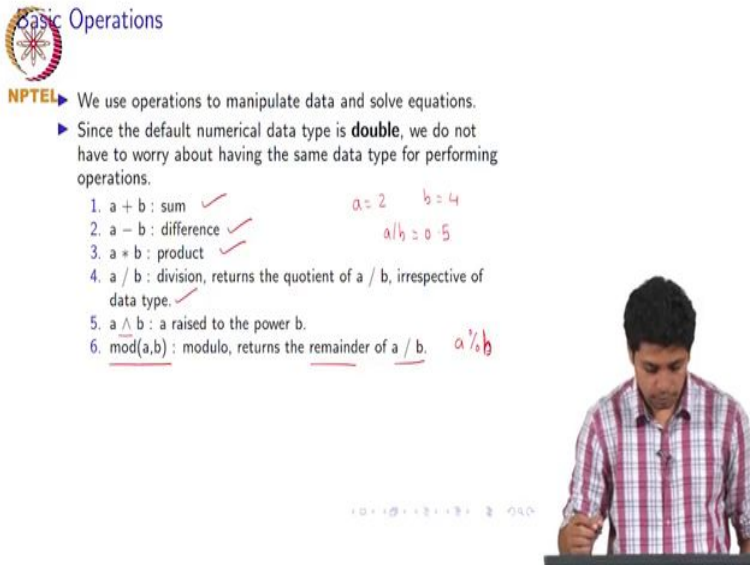
So, what I do is I put a class of a in the command window and I put enter. So, it says double so; that means, double precision floating point a is a double precision floating point value. Then numerical arrays we will discuss a bit in detail, but we will do that later.

(Refer Slide Time: 11:16)



Now, these are some examples that I just discussed with you. So, x equal to 1 it will store it has 1.000. If you wrote; if you wrote x equal to MATLAB it will just store the word MATLAB as it is. And if you say x equal to 1 to 10. So, what it will do is, it will store an array of 1, 2, 3, 4 till 10 by default. So, you do not need to specify any variable type to MATLAB you just need to say x equal to and put any type of value that you want and it MATLAB will interpret on its own.

(Refer Slide Time: 11:52)

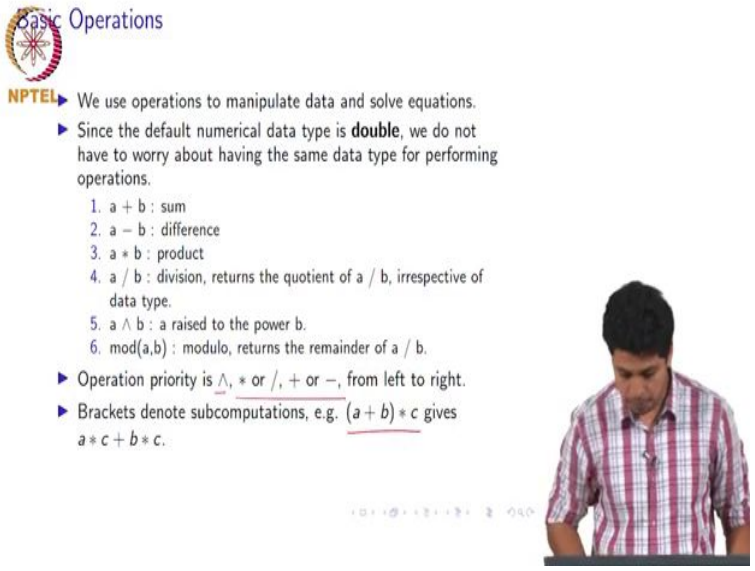


Now, we will discuss a bit about basic operations. Now we use operations to manipulate data and solve equations all the time. So, since the default numerical data type in MATLAB is double.

We do not have to worry about 2 variables having the same data type in terms in order to perform any operations on them. There is an advantage you have to consider for MATLAB. Now these are basic operators that we use plus for sum, minus for difference, asterisk for product, division then actually a by b returns the quotient irrespective of data type.

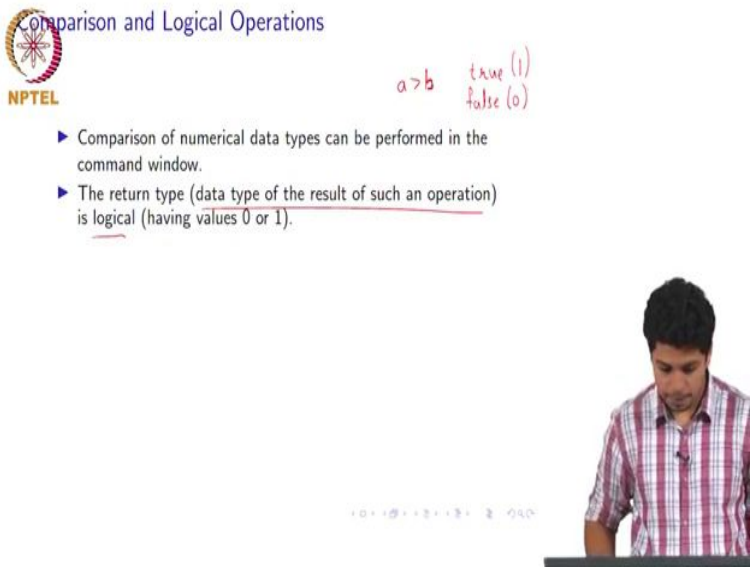
Like we just discussed, even if the value of a is 2, the value of b is 4 a by b will give you 0.5. Then a raised to b this symbol wedge is used for power operations, and then the way we compute modulo operation is we write this command mod a comma b. So, what it returns is the remainder of a/ b. You may know this command in C or C plus plus as a percent b. So, this is equivalent to that.

(Refer Slide Time: 13:03)



And then the operation priority as we all know should be power and then we basically just use the BODMAS rule even the next statement you can see. If you use a bracket then it just follows the BODMAS rule.

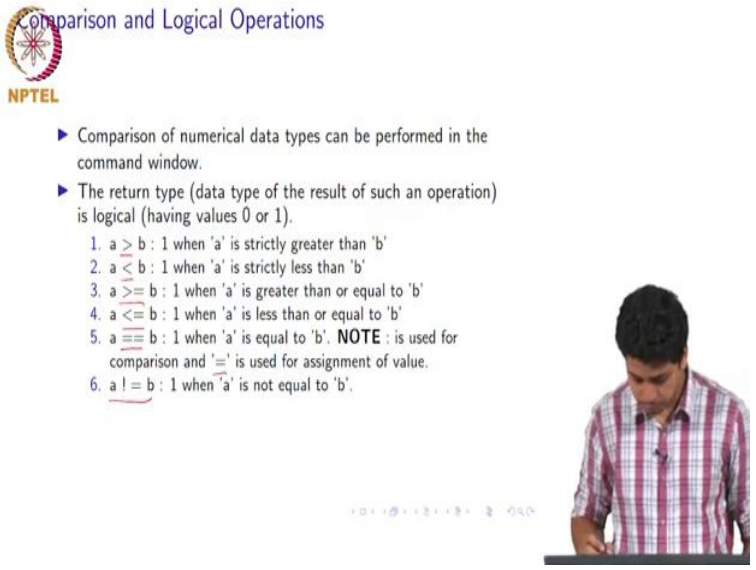
(Refer Slide Time: 13:16)



And now we will discuss a bit about comparison and logical operations. So, we can compare any 2 variables because as we discussed the numerical data type is fixed and it is floating point operations. So, we can perform the comparison operation.

And then the return type that is the data type of the result of such an operation is logical. So, if you compare any 2 numbers then the answer that you are supposed to get is true or false. So, say you have performed an operation greater than b; I mean you are asking if the number is greater than b and if a is indeed greater than b then you get the answer of true and if it is not you get false. So, in MATLAB true has the logical value 1 and false has the logical value of 0.

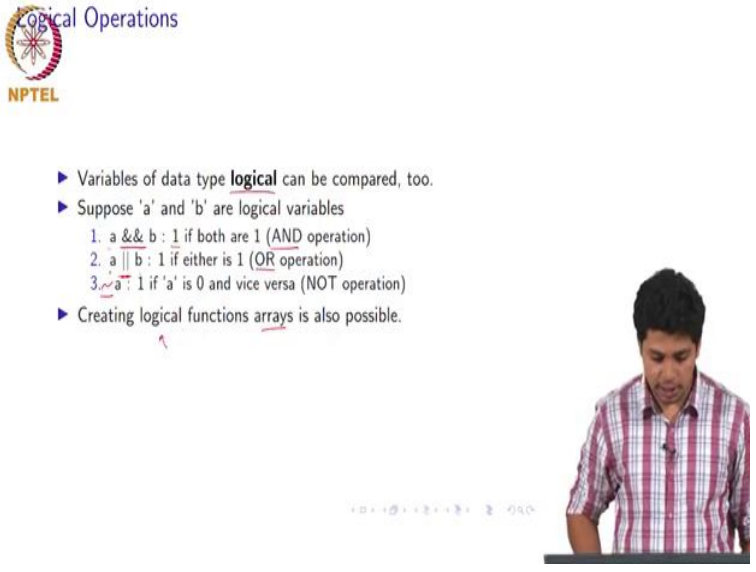
(Refer Slide Time: 14:07)



And these are the different kinds of comparison operations that you can do strictly greater than, strictly less than, greater than equal to, less than equal to then we use an equal to symbol two times to compare if 2 numbers are equal.

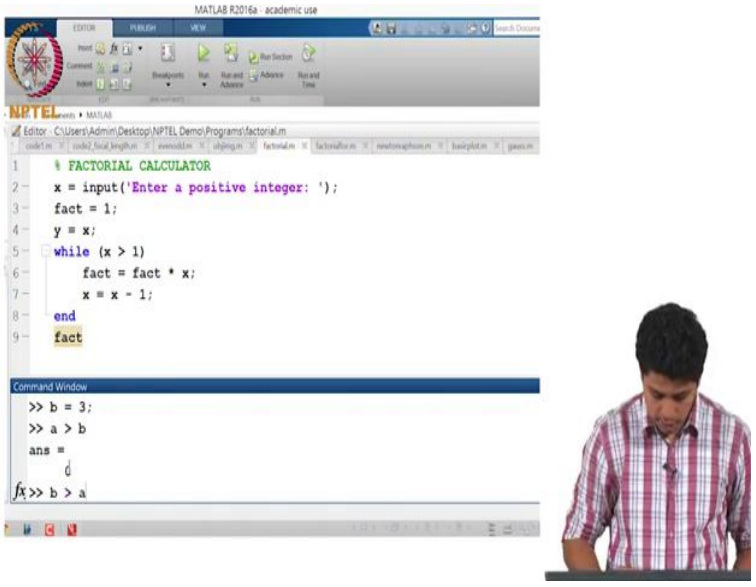
If only 1 equal to symbol is used it means assignment and if 2 are used it means a comparison and then a not equal to b this exclamation mark is supposed to be not a not equal to b.

(Refer Slide Time: 14:37)



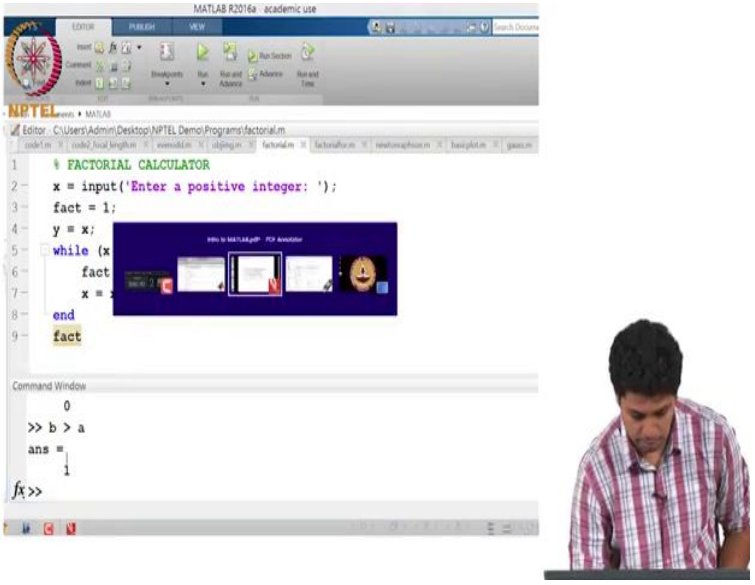
Then we will discuss logical operations, now logical operations can be done on data type logic. Now I will just show you what data type logic means in this code we will just write the code in the command window. Now we know that a is equal to 2 let us give b the value of 3 b is equal to 3 here ok.

(Refer Slide Time: 14:57)



Now, you see the value of 3 stored and it did not get displayed because they put a semicolon in the command window. Now what if I write a statement like this a greater than b now we know that a is not greater than b.

(Refer Slide Time: 15:13)

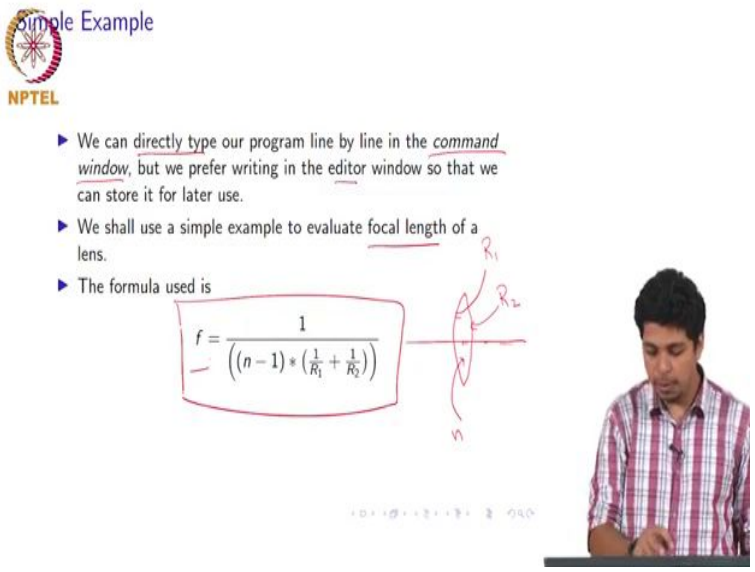


So, it will just give answer as 0 now this is a logical 0 this is what it means, but if I put a statement like this b greater than a; we know that b is greater than a so it will give me an answer of 1; so this is a logical 1, this is what true and false are represented as in MATLAB.

Now on logical type functions we can do operations like this a and and b will be true if both a and b are true. If a or or b means it will be true if either of a and b are true it will only be false when both are false this is the OR operation, the previous one was the AND operation. Then this not a this tilde a symbol is not a; not a is a 1 if a was 0 previously and vice versa.

So, this is basically the tilde is basically the not operation. Then you can also create arrays of logical variables, but we will discuss that later and turns out it is not that necessary at this stage.

(Refer Slide Time: 16:19)

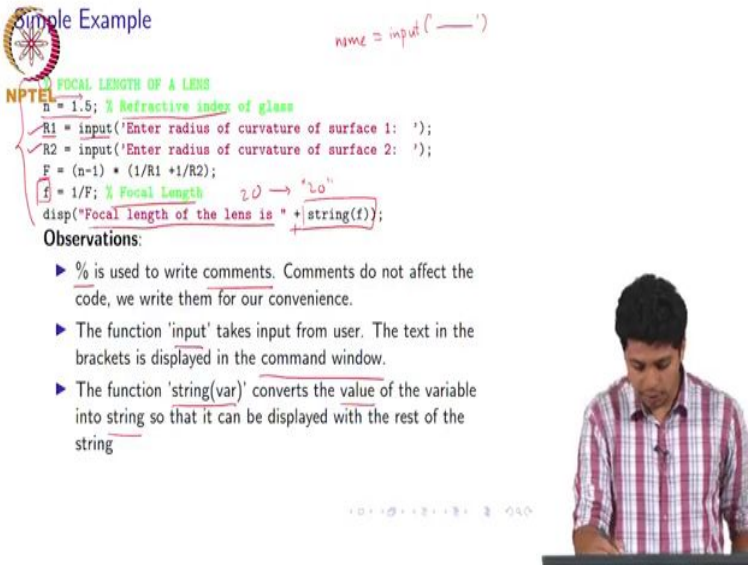


Now, we will just consider a very simple example using all the things that we have learned until now. So, as we know we can directly type our program in the command window, but we prefer writing it in the editor window. So, that we can store it for later use because in the command window we can only execute commands line by line, but if we store it in an editor then we can have a look at the entire code and if there are any errors or we have to make some changes in the code then we can do that that is what the editor is used for.

Then we shall discuss a simple example to calculate the focal length of a lens. Now, the formula that we use is this one; $f = 1/(n-1) * 1/(R_1 + R_2)$. So, if we say we have a lens like this and this is the optical axis then it is a thin lens and we assume that both the surfaces are circular.

So, say if this one has a radius of curvature R_1, and if this surface has a radius of curvature R_2 and the refractive index of the material in the lens is n, then the focal length will be given by this formula. So now what we want to do is we want to implement this formula in MATLAB.

(Refer Slide Time: 17:40)



This is what the code looks like. So, percent is used to comment on the code and anything you write beyond a percentage symbol is not a part of the code, it is only written for our own convenience. So, that we know what section of code we are in we can write any comment there and it will not affect the code. Then we give the value of refractive index 1.5 as you can see I put a comment in there to indicate that it is a value of refractive index of glass.

Then R_1 is equal to input and then I put some text. So, the thing is this function input is used to take an input from the user. What and the syntax that we use is the variable name. Any variable name is equal to input and then in brackets if we want to display any text that we have in the command window. So, that we know what the variable is supposed to be then we put this text in the inverted commas within a bracket that is what this statement is telling us.

Then R_2 is again the input for second radius of curvature then I have used a buffer variable f which calculates (n-1)*($\frac{1}{R_1} + \frac{1}{R_2}$) and then I have used small f as my final answer which will be 1/F, which will give me the focal length and then I have displayed.

So, now, in this formula there are in this entire program there are a few observations that we should make like I said percentage is used to write comments then this function called input takes in the input from the user and the text in the brackets within these inverted commas is

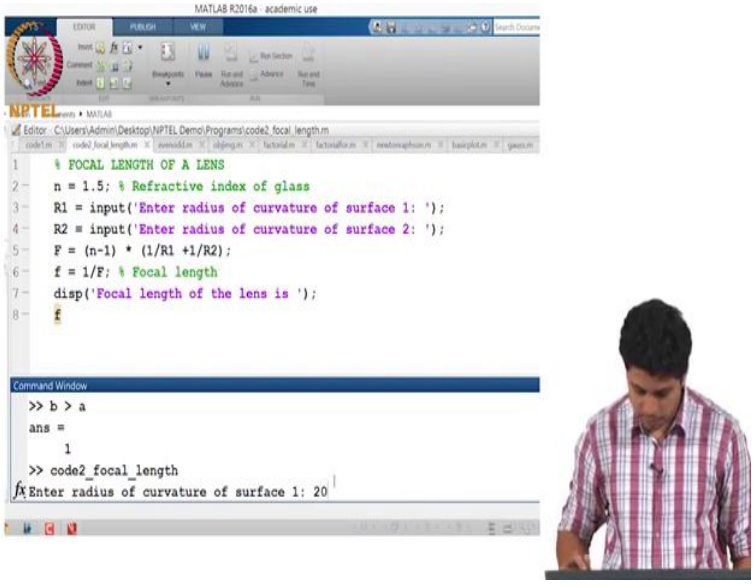
just displayed in the command window; you can enter whatever text you want and it will not affect any further code.

Then this function string and in brackets the name of the variable. Now, this is a special function it converts the variable type I have written the value here, but it converts the variable type into string. So, that it can be displayed along with the rest of the string that is present in display.

So, how that works is say the value of focal length that we get is 20, and what we are displaying is in the inverted commas we have put this focal length of the lenses it will be displayed as it is. And we put a plus here. So, that whatever is there the output of string can be just appended to this one and appending means just adding it to this one.

So, string f will just convert this 2 this 20 in into 20. So, this whole statement will be displayed like the focal length of the lens is 20 that is how it works. It will work in the latest version of MATLAB and octave, but since the version that I have on this computer is an older version. So, the command of string will not work.

(Refer Slide Time: 20:39)

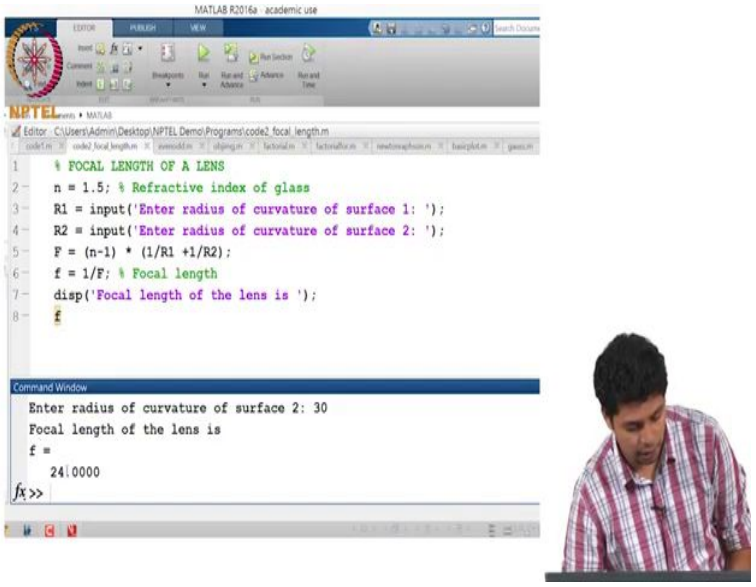


So, I have just entered the code here for your sake. So, this n equal to 1.5, R1 is input of this thing, R2 is again input and then I have used F like this and then small f will be our focal length and I have put this statement here display focal length of the lens is and then I have

just entered the name of the variable. Now if you just enter the name of the variable and do not put a semicolon it will be just displayed in the command window.

Now, we will just execute this code now I have written this code and I just press this run here and in the command window immediately it will ask me to enter the radius of curvature now this text is just here. So, we can display whatever text we want. So, we will just enter say 20.

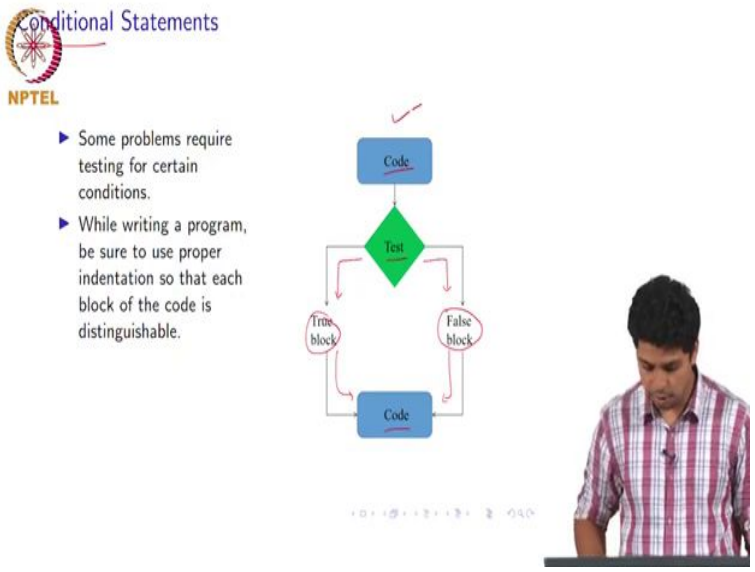
(Refer Slide Time: 21:18)



And when you press an enter it will ask me the radius of curvature of surface 2 which is the immediate next command. So, let us just put 30 here then as you can see it performed operations on the 5 th and 6 th sentence or command and then 7 th command was to display this thing.

So, the focal length of the lens is as it said and then there is nothing no text here. It just instead displays the value of f in this way that is how it is supposed to work.

(Refer Slide Time: 21:51)

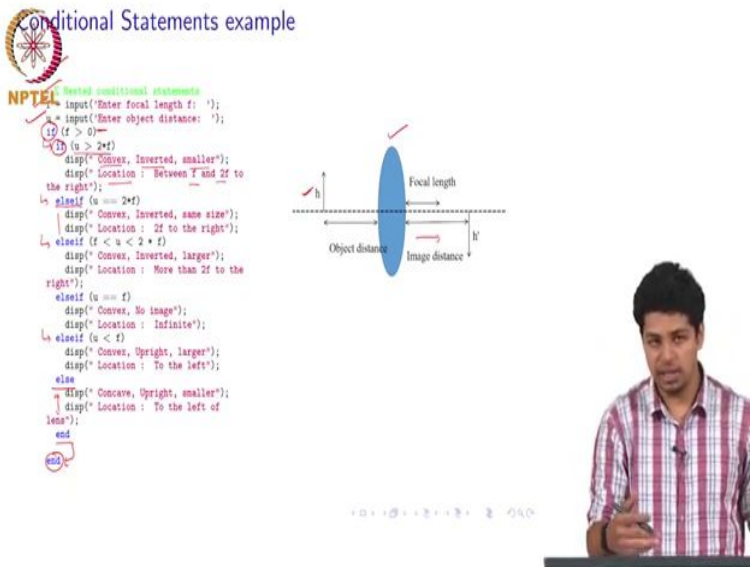


Now we have discussed a simple example, let us just go on to a few complicated or say advanced commands instead of complicated. Conditional statements will be used very often in advanced programming. So, you might require to test our code or test inputs whether or not it satisfies some particular conditions in order to run some statements. So, this is the flowchart of what the conditional statements in MATLAB look like.

So, what we do is there is a piece of code which is executed then there is a test this is this block is where a condition will be tested you can specify the condition. If the condition is satisfied then a set of code is executed which is called a true block we just call it the true block.

If the test is false or it is not satisfied then it will execute a second block of statements which we call the false block. And both true and false block after the execution will again return back to whatever code we want, and we use proper indentation like I mentioned previously. So, that each block of the code within the conditional statements is distinguishable from the rest of the code. So, that is why indentation is important here.

(Refer Slide Time: 23:04)



I'll just give you a brief example of how it is done. So, we all know from school physics that if we have a lens on and we have placed an object somewhere in front of the lens. Then we know depending on the distance of the object from a thin lens where the image will be formed. So, this entire code is basically just testing for different conditions and giving the proper output.

So, a nested conditional statement means when we put a conditional statement within a conditional statement. So, we asked for f that is the focal length input from the user then we asked the object distance then what we do is if the focal length is positive.

Then we know that the lens is a convex lens and if the lens is convex, then all of these then enter this if loop and all of these conditions will be tested. So, now, again we put another if and we asked the for the first condition if u is greater than 2 f then we know that there is a certain place where the image will be formed.

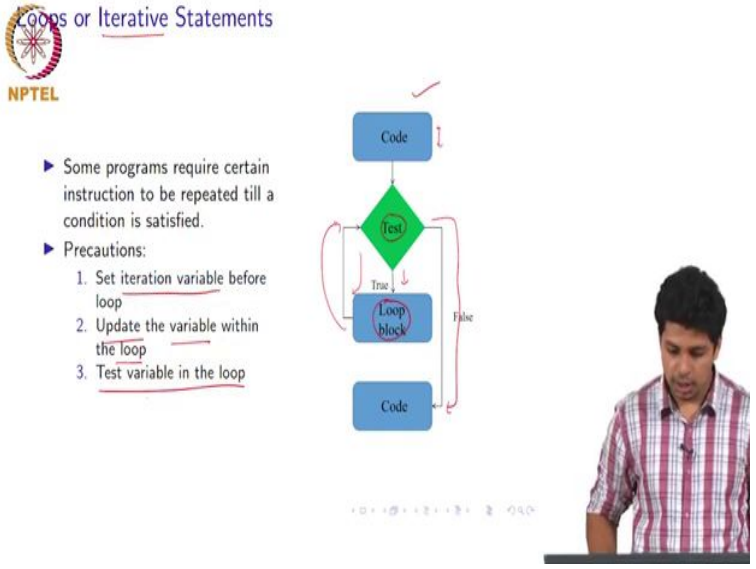
So, if u is greater than 2 f the lens type is convex, the image will be inverted and smaller and the location of the image will be between f and 2f to the right side on this side of the lens. Now, likewise you can test for some other conditions what if u is not greater than 2f we put another command elseif and we put a second condition.

Now, this is exactly like the see command that we use except the syntax is like this we write elseif connected and we put the second condition and then we execute statements for the second condition if even if that condition is not satisfied we can put another elseif put a check for the next condition.

We keep checking conditions until we run out of them and if a certain number of conditions are tested, but our objectives are variable is not satisfying either of them then we can just put an else statement and write whatever statements we want to execute if no condition is satisfied.

Then after this we put an end statement that will end this inner if then it will just jump out and we also end this first if this is just an example to show you how conditional statements in MATLAB work. So, you can write any code like this.

(Refer Slide Time: 25:28)



Now, we will discuss loops or iterative statements here. Now certain programs might require you to repeat a certain number of statements over and over till a certain condition is satisfied. So, for that purpose we use iterative statements. So, we have to take a few precautions if you want to execute statements like this.

So, let us just consider the flowchart shown here there is a block of code we check for a condition, if the condition is true there will be a loop block which will contain statements which are executed if the condition is true.

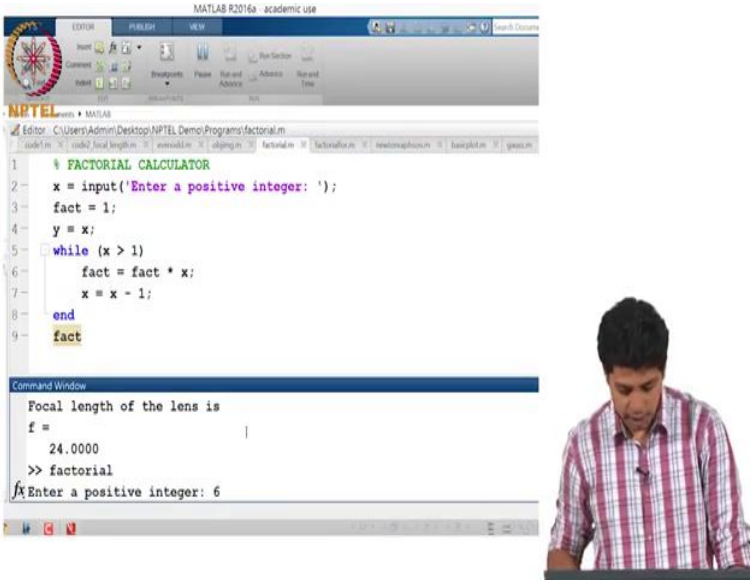
Then after the if block is after the loop is executed it will just go back and check that condition again, it will keep executing the conditions the statements until and unless the condition does not become false. When the condition becomes false it just comes out of the loop and it executes the further code that you have put this is how conditional iterative statements are supposed to work.

Now, the precautions would be we always set an iteration variable before the loop starts. So, there is a variable which will track what happens within the loop then we update the variable within the loop. What happens if we do not update a variable within the loop; is that it will keep executing that loop infinitely because the value of the iteration variable is not changing. So, the condition if it is true for the first time it will never become false, but if it is false for the first time it will just continue, but since we do not want our code to run the same statements infinitely we keep updating and it is generally updated such that the condition eventually becomes false.

So, it just exits the loop and then we test the variable in the loop. So, we test the variable which is being changed or which is being updated. If we check for some other variable there is a decent chance that the other variable might not get updated within the loop and it will again fall into that infinite execution trap.

So, we do not want that to happen. So, we always check a variable that is updated within a loop. I would like to discuss an example here. So, say we want to calculate the factorial of a number.

(Refer Slide Time: 27:47)



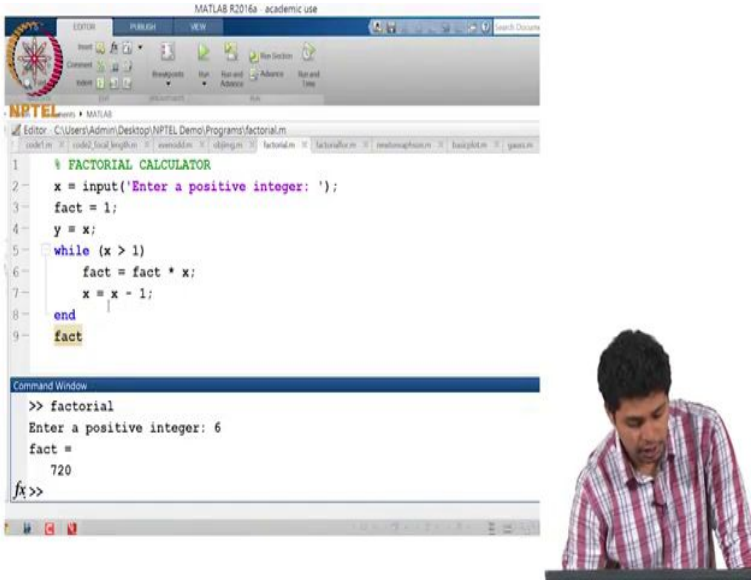
I have written a brief program here. So, if we look at the editor we asked for input x and we asked to put a positive integer here, then I set variable factorial equal to 1 this will be the variable where I will calculate the factorial and then I just use another variable y equal to x. Now, this is just because the value the program that I write will update the value of x and I will not have my variable where I have actually taken the input.

So, say if I want to put a statement here like the factorial of the entered variable is this much then I will not have the value of x that is I have put a value of y here, so that I can retain that value.

Now I write while loop, now this is while is an iterative statement and the way the while loop works is I write while and in brackets I write a condition. So, while x is greater than 1 this set of statements will be executed. So, this data set of statements will be executed till the time value of x is greater than 1 the moment this condition becomes false it will exit the loop and then I just put an end statement so that I know which statements are within this while loop.

And then I just display the variable factorial which I have written as fact. So, let us just execute this program and see how this works. So, it will ask for a positive integer in the command window let us give it a value of 6 and see what the answer is.

(Refer Slide Time: 29:21)

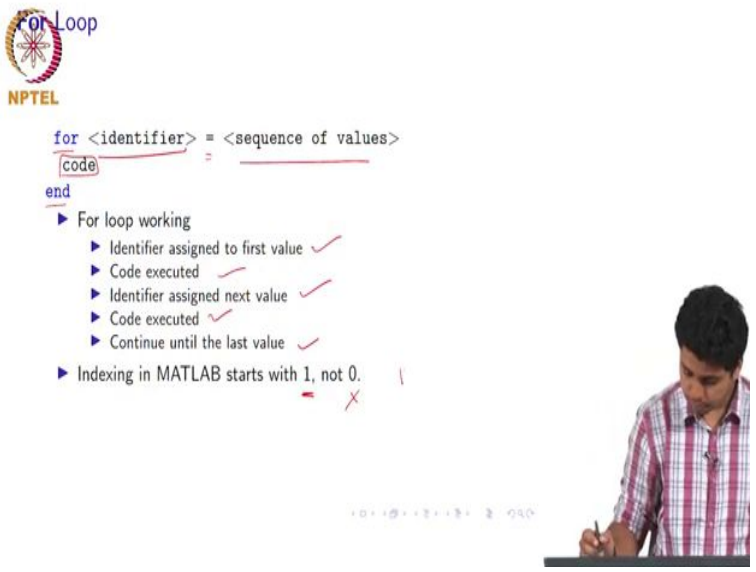


It will give me 720 which is the right answer. So, what happens is when I put 6 here. So, it will check if 6 is greater than 1 it is greater than 1. So, it will just multiply factorial equal to factorial which is 1 into 6.

And then this x or 6 will become 5 because x is equal to x minus 1 and then it will end. So, again goes back to while 5 is still greater than 1. So, what happens is 6 is equal. This factorial will become equal to 6 which was the previous value of factorial into 5 because x is now 5 and this keeps happening till x becomes 0, when x becomes 0 this statement is not run.

So, factorial into 0 does not happen and the moment x becomes 0 it just exists and it executes this statement the 9 statement factorial and the value of factorial as we can see is 720. This is the way a while loop is entered in MATLAB.

(Refer Slide Time: 30:14)

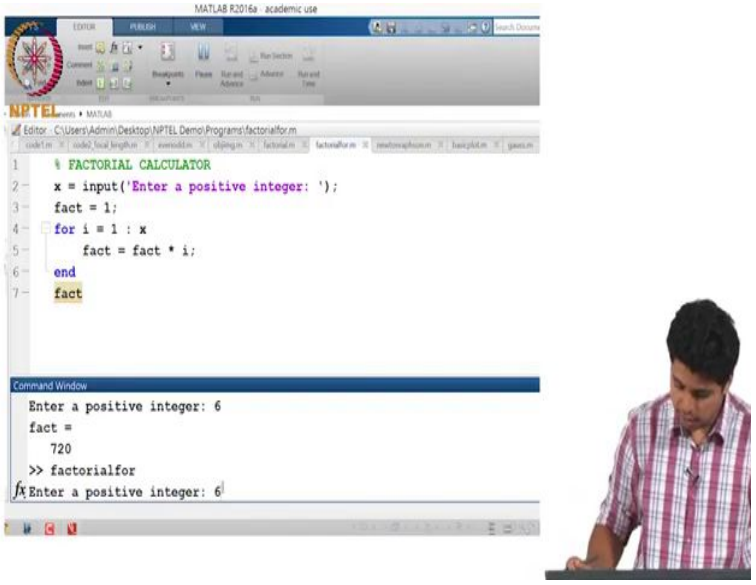


Now, we will discuss whether the loop and way of for loop is executed in MATLAB or it is written in MATLAB if we write for we put the variable identifier or variable here and then we assign it a sequence of values. So, what happens is this variable takes all values within this sequence of values and it executes the code, till each value is set is assigned to the identifier and then it will just exit the for loop.

Now, I have just explained the working in brief here. So, the identifier is assigned to first value first value then the code is executed, then immediately the identifier is assigned the next value then again the code is executed, and this carries on until the last value is assigned and after the last value is assigned the program will exit the for loop. And there is one very key point to remember that indexing in MATLAB starts with 1 not with 0 in C or C plus plus or even python the indexing will start from 0, but in MATLAB the indexing will start from 1.

So, the first value that you put is the index of the first value in an array will be 1 and will not be 0. So, I have the same example in for loops.

(Refer Slide Time: 31:36)



So, we say we want to execute to calculate the factorial of a number, but we want to use for loop instead of a while loop. So, again we ask for input and we store that value in x again create a variable called factorial, we assign it a value initial value 1. Then what we do is we create another variable called i then we say the syntax of the form is as follows for the identifier or the variable which is called i is equal to 1 colon x.

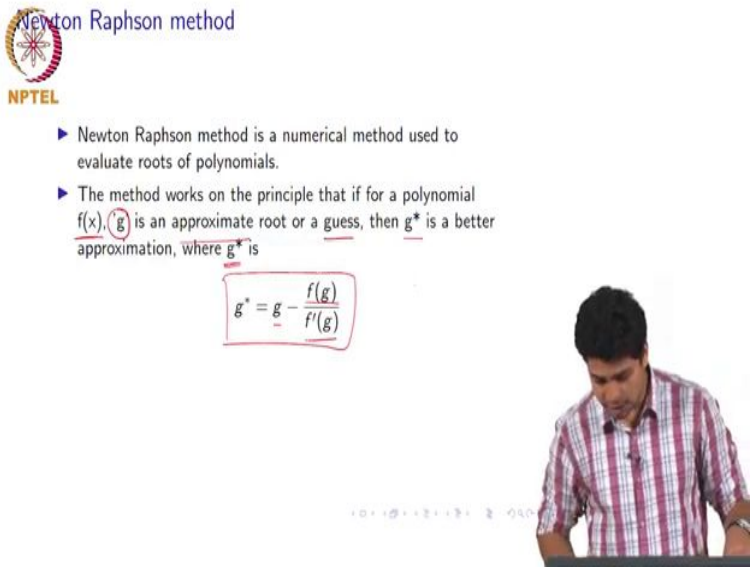
Now, 1 is the first value colon x is the last value. Now typically when this syntax is entered what happens is if x is an integer then 1 is the first value 2 is the second value 3 is the third value and it carries on till x. So, the step size or the difference between 2 consecutive values is by default 1, but we can also specify the step size. We will discuss that later, but for now we are calculating a factorial. So, the step size is 1 so, what happens is in the for loop we say factorial is equal to factorial into i. So, the initial value was 1.

So, 1 factorial is equal to 1 into 1 because the first value of i is also 1 then again it goes back to the for statement and the second value of I is 2 then factorial becomes factorial is equal to 1 which is the previous value factorial and 2 which is the current value of i then factorial becomes 2 then it again goes up i becomes 3. Now then factorial becomes 2 into 3 this is how it keeps executing until the last value of this variable occurs which is x which is what we took the input.

So, it calculates the factorial in this way now we will run this program again and we will see if the syntaxes that I have put are correct. So, we will again put the value 6 here and it is giving me 720.

So, this is just another way of writing iterative programs. So, now, we have a choice between while loop and for loop.

(Refer Slide Time: 33:37)



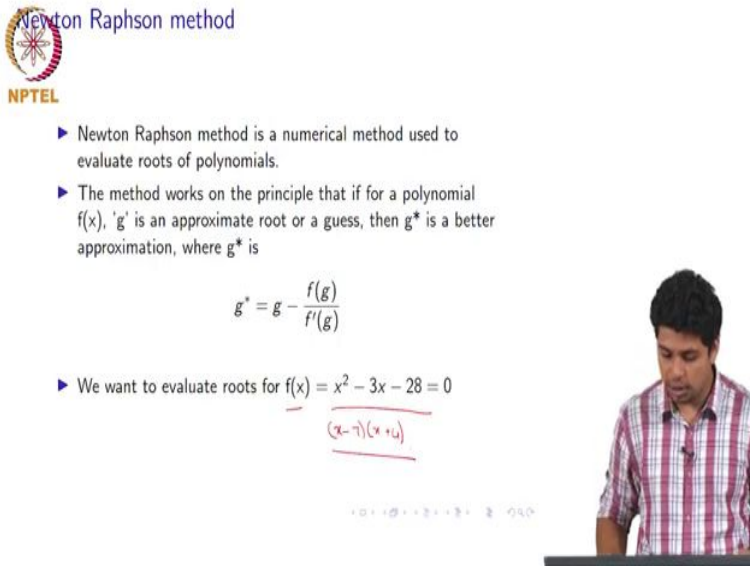
Now, we will use all the concepts that we have discussed until now; and we will apply them for a particular application. The Newton Raphson method is a numerical method used to evaluate roots of polynomials of complicated polynomials of a higher degree. So, this method is used when analytical solutions cannot be found. So, say you have a polynomial with a very high degree and it is not always possible to find an analytical solution.

So, what we do then is we find a numerical method we give different values to the variable in that equation and we see if it goes to 0. So, are as close to 0 as possible. So, the value of x for which the equation is as close to 0 as possible is a solution, and this method is called a numerical method.

Now how the Newton Raphson method works is if for a polynomial f of x g is an approximate root; that means, g is a value of x which we have put as an initial guess or an approximation then the Newton Raphson method says that g star is a better approximation.

That means f of x is closer to 0 for g star then for g where g star is given by this equation. So, g star is equal to g minus f of g which is the function f of x putting g as the value divided by f dash of g where f dash is the first derivative this is what Newton Raphson method says.

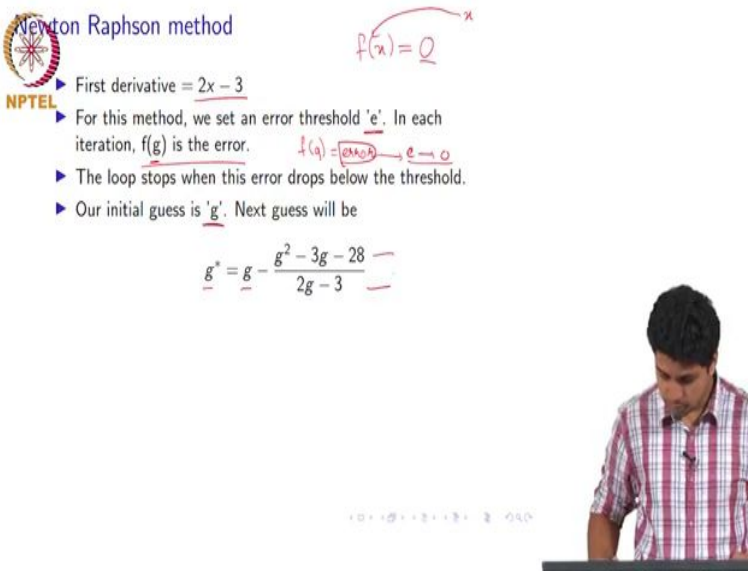
(Refer Slide Time: 35:11)



Now, we will take an example and we will use a simpler example just because we want to understand the Newton Raphson method. So, say we want to evaluate f of $x = \sqrt{x^2 - 3x - 28}$. Now, here just for our own sake we know that we can do this. We know that it is (x- 7)*(x - 4) this is the solution that we already know.

So, we will keep this just for our sake and we will see how Newton Raphson method is evaluating the answers.

(Refer Slide Time: 35:46)



Now, according to Newton Raphson method we want the first one to calculate the first derivative. So, 2 x minus 3 will be the first derivative now in this method we set an error threshold e and in each iteration that we are running because essentially when we are updating our guess what we are doing is we are iterating the entire method and generating a new value of the guess each time.

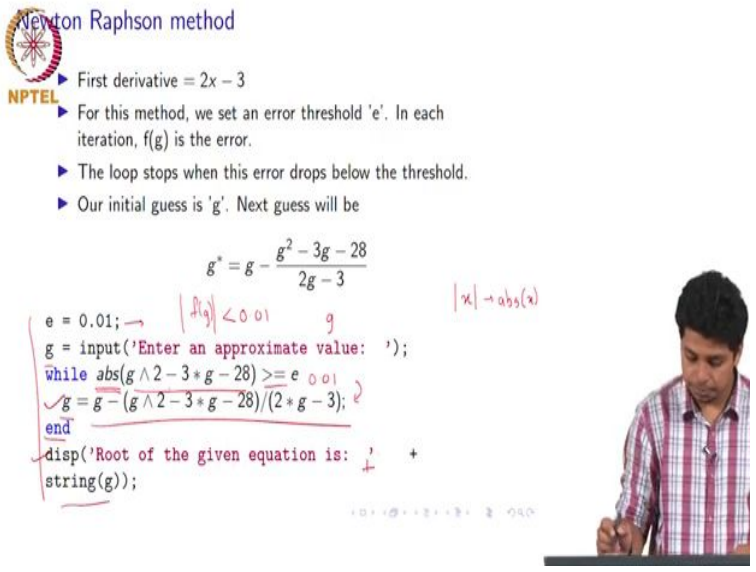
So, for each iteration f of g will be the error because typically the way we find the root of an equation is say we have f of x as a function and we say that x is a given x is the solution for which $f(x) = 0$. So, if I put f of g here and my answer is supposed to be g 0 f of x for a root is supposed to be 0. So, whatever value of f of g I have if it is nonzero then it will be my error and what I want is my error to be as small as possible. So, my f of g is going to be e which is my error and I want e to be as close to 0 as possible.

Now, we set this is the error actually and I set my e as close to 0 as possible and the loop stops when the value of this error becomes less than this e. So, why that is important is because we want to stop the guessing at some point we do not want the root loop to run infinitely.

So, a little bit of error is tolerant or tolerated. So, that is why we want to set a threshold for the error and as soon as the error falls below that threshold we will accept that g as our guess

as to be as close to the solution as we want. So, our initial guess is g the next guess will be $g - f(g)/f'(g)$.

(Refer Slide Time: 37:43)



Now, this is the code that we put for Newton Raphson method in MATLAB. So, say our error threshold is 0.01. So, if the value of $f(g)$ for any particular guess is a false absolute value of f of g false below 0.01 we will accept that particular g as a solution.

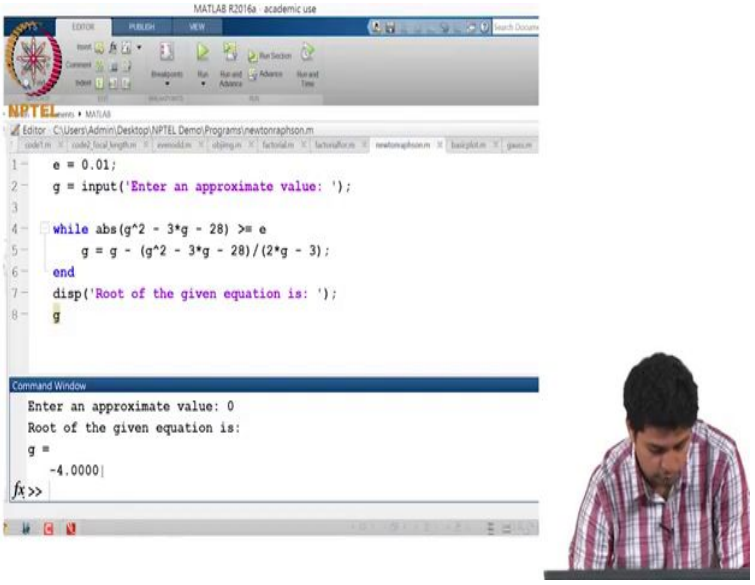
So, this is why we set a threshold and then we put a statement g and we take the input for the first value; for the initial guess. Then I put in a whole statement. So, remember the condition that you put for a while is such that when the condition is false the execution stops. So, we put a condition which is true right.

So, we put the absolute value so the way we enter mod of x is in MATLAB if we could use this function absolute value of x this will give us the mod of x. So, if absolute value of g^2 -3 g - 28 is greater than or equal to 0.01 then these statements will be executed then what we do is we update the value of g and this is the formula that we use is the Newton Raphson formula this is how we execute the Newton Raphson formula and then we end.

So, this statement will be executed as long as this absolute value of the function is greater than e. The moment the value of this absolute value of $g^2 - 3f - 28$ falls below e the execution of the loop will stop and we will accept that g as a solution and then the last

statement is displayed the root of the given equation is string g. So, I mean string g as in the value of g is converted into a string and it is appended to this statement.

(Refer Slide Time: 39:48)



Now, we will run this program and we will see if it will give us the right answer. So, I just enter the same program here except because this version does not support the function string. So, I just put this g comma g variable name here and the rest of the code is actually exactly the same.

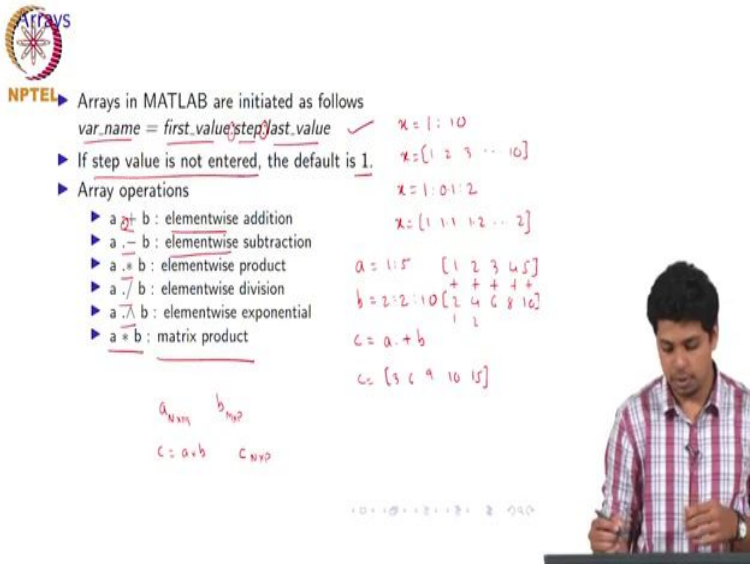
So, we put a while absolute value of g square minus 3 g - 28 is greater than or equal to e and then I have put this updating value of g and then I have ended it. So, we will just run this code and we will see so we will give it an approximate value.

So, say the first approximate value that we want to give is 0 we will see what happens now. So, it will give me the value minus 4 now we know that -4 was a solution. So, a Newton Raphson method works and the reason why we discussed this was we used or implemented this while loop and found the solution of an equation.

So, this is a place where we can make use of iterative statements now this is a very basic example, but you can use this in more complicated examples as well more complicated applications as well. Now just as an exercise, I would like you all to just enter different initial guesses and see if you can arrive at the other solution; that means, if we want the other

solution that is plus 7 to be our output. So, just play around with the initial guess and see what happens.

(Refer Slide Time: 41:17)



Now, we will discuss arrays in MATLAB. So, MATLAB arrays are initiated or declared in this way. So, we put the variable name we put the first value you put a colon here after the first value then you put a step then another colon and then you put the last value. If the step value is not entered the default value is 1. So, if I enter a statement like this x = 1 to 10. So, x will store values like this 1, 2, 3 till 10, but if I happen to enter a statement like this x = 1 colon 0.1 colon 2; then what it will do is x will store values like this 1, 1.1, 1.2 till 2 this is how arrays work.

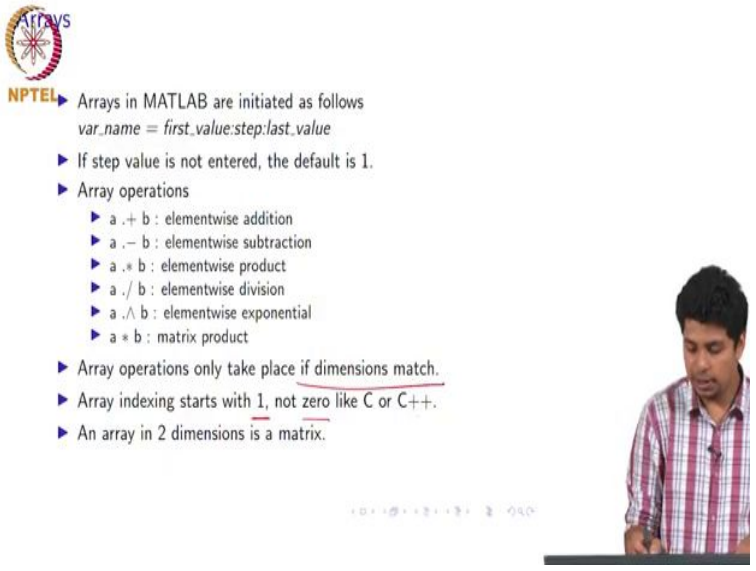
Then you can also perform operations on arrays just like we perform them on individual variables: a. + b. - . multiplied by dot divided by and dot rise to. Now, what this dot means is element wise and say suppose I have a which is storing say 5 values from 1 to 5 and I have b which is storing some values from 5 values again which will say these are like 2 to 10 or 2 and this will not store 5 values we will just put step size as 2 and this as 10 last value as 10.

So, what dot plus will do is this say if you put a variable C= to a. + b what that will do is it will add these elements of a are 1, 2, 3, 4, 5 and b are 2, 4, 6, 8 and 10. So, what dot plus will do is it will just add individual elements of a and b according to the index.

So, the first index values will be added like this, the second index will be added and so on and then finally, c will store these values 3, 6, 9, 10 and 15. So, that is what element wise means. And if we just put a into b it will evaluate the matrix product now we all know how matrix products are evaluated.

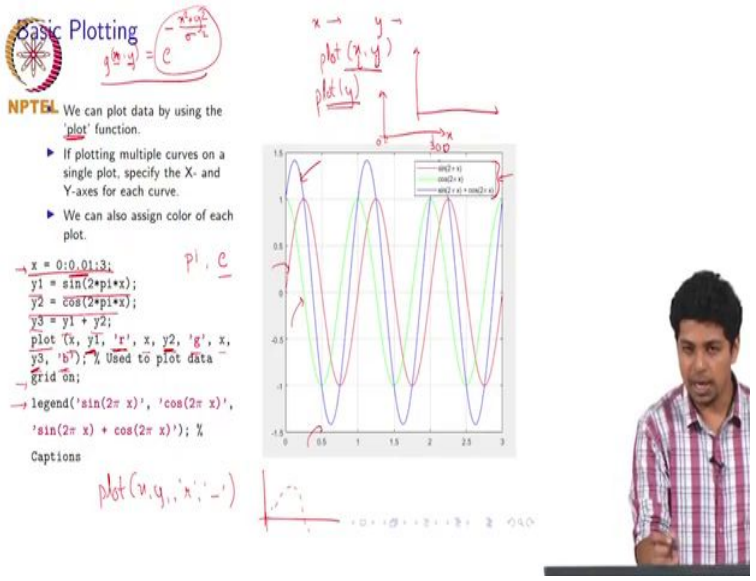
So, if you have a matrix of dimensions n by m and you have another matrix of division of these dimensions mxp and if we put c=a*b then we know that c has dimensions nx p. That is what this matrix product will mean. So, if you want to do element wise operations be sure to remember that you put a dot in front of whatever operation you want to do.

(Refer Slide Time: 44:20)



Now, array operations will take place only if the dimensions of both variables which are storing arrays will match. If this does not happen then it will MATLAB will give you an error and again I would like to remind you that the indexing starts with 1 not 0 like C or C ++ ok. Then if you have a 2 dimensional array then it is basically a matrix.

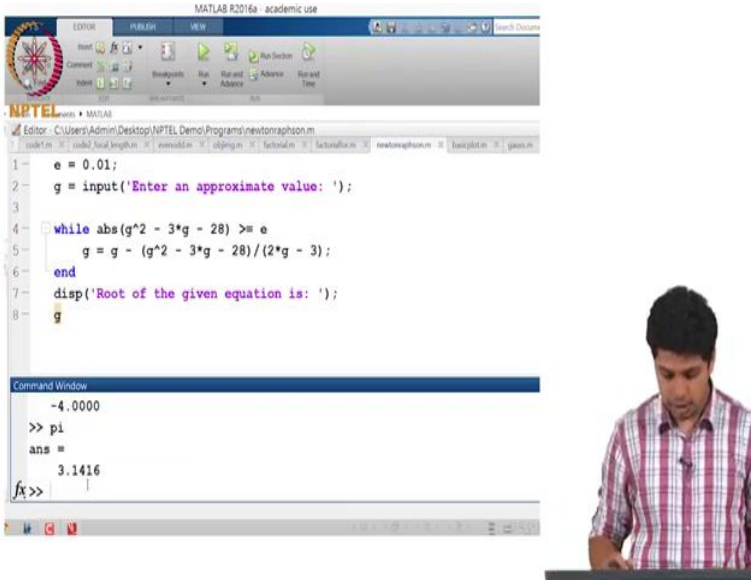
(Refer Slide Time: 44:44)



Now, we will briefly discuss plotting different functions in MATLAB. So, we can just use this command plot in order to plot different functions. So, we will use arrays here. So, in the code that I showed here x is storing values between 0 and 3 with a step size of 0.01.

Now why I have put y 1 is sin of 2 into pi into x. Now I should probably tell you this right now, that MATLAB has I mean inbuilt stored the values of π and e. So, I will just explain to you on the MATLAB command window.

(Refer Slide Time: 45:33)



So, if you just in the command window enter pi then it will give you the answer and that is I mean it already has the value of pi stored inside it we can also likewise check for the value of e as well.

But, since I have already used it in this code for some operation it will give me that value, but if it does not have any value stored in the workplace workspace then even its value will be displayed in MATLAB. So, for this example y 1 will store sin of 2π x, y 2 will store cos of 2 π x, and y 3 will be storing a sum of both of those variables.

Now, the way we plot these waveforms in MATLAB is we use the command plot then as you see here I have put x, comma y 1 and then I have put in these inverted commas r, then I have again put x comma y 2 then in inverted commas I put g then x comma y 3 and inverted commas b.

So, what that basically is saying is you always should enter your variables in an ordered pair. So, say you have x values as your input and you have y values as output. So, what you should do is whenever you use the plot command and we should always specify an ordered pair like this.

So, what MATLAB will do by default is it will consider if you put an ordered pair the first variable to be on the x axis in the display and the second variable it will consider on the y axis if you only put plot y.

So, it will just plot the values of y and it will on the x axis it will just display the number of points or the index of the points. So, if you have a hundred values it will just display 1 to 100, but since here we have like 0 to 3 with the step of 0.1 if you put x y then the first value will be 0 the next value will be 0.201 then the last value eventually will be 3. But if you do not do that if you only put plot y then the first value will be 0, but then the last value will be 300 because it is the 300 and first point and the first point is 0.

So, there are 300 points so the last value will be 300 that is what will happen if you only put plot y. So, as a good practice to code in MATLAB we always put the variables to be plotted as an ordered pair. Now, we will also notice in this code that I have put in these inverted commas I have put certain other letters as well like r, g and b. So, what this means is I can put individual data points and plot them in different colors.

Now, for y, x, y1 I have put the color red, for x, y2. I have put the color green, and x comma y3 I have put the color blue. And as you can see here since the first one is sine wave and it is displayed in the red, the second is a cos wave and it is discussed it is displayed in the green color, and there is sum is displayed in the blue color.

So, you can plot different data using different colors and what you can also do is you can also plot different styles. So, if I put something like this plot x, y 1, in inverted commas r and if I also put this one a dash here or a hyphen here. So, what it will do is it will give me a dashed plot instead of a continuous plot.

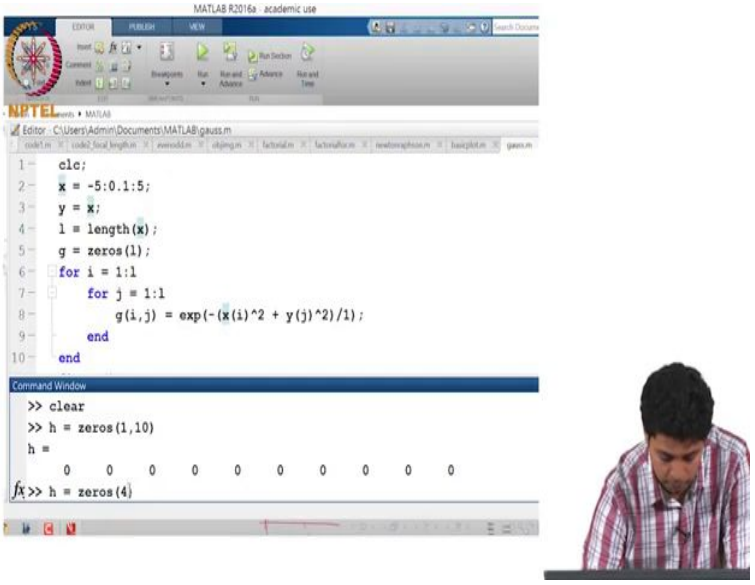
So, this one will give me something like this. So, that is what the difference is you can also put styles of plotting. Now, you just one more thing you can do then the next command is grid on what grid on will do is it will just start this; or it will put this graph line on it that is just for our own convenience. Say, if you want to get a nice guess as to what values are contained inside and so you can put lines like this. So, this is done by this grid on command.

Now the next command is legend now legend is used to put captions. So, because we have 3 data points 3 sets of data points plotted here. So, we can put 3 different legends. So, the first legend as I have put here you can see is in inverted commas I put sin $(2\pi x)$ and I put a comma then I put cos $(2\pi x)$ as the second one and sin $(2\pi x)$ + cos $(2\pi x)$ is the third one and it just gets displayed here this is what the legend does it to just gives captions to your graph.

Now what we will try to do is this is in 1 dimension. So, our input variable was just x it was in 1 dimension. What we can also do is we can input in different or say 2 dimensions also and plot it in the form of an image. So, I have a code for plotting a Gaussian distribution in 2 dimensions. Now, if we have a Gaussian distribution in 2 dimensions the equation will be so, say I am storing it in g of xy. So, the way I will display a Gaussian is or we know as we know the equation of Gaussian is $xe^{-(\frac{x^2+y^2}{\sigma^2})}$

So, this is the equation for a Gaussian now what I want is, I want x a certain range of values for x certain range of values for y and using this I can just generate this one as a 3 dimensional plot. So, 2 dimensions will be x and y and the third dimension will be the value of g which is dependent on x and y. I will show you how to do it. I have a program.

(Refer Slide Time: 51:31)



So, we just use the clc command to clear anything in the command window like I will show you here the clc window clc command is just to clear the command window. So, that we do

not have any thing displayed there to confuse you and we can likewise use a command called clear, what it will do is it will clear this workspace; it will delete all the variables and all their values and you will have a you can assign a new values to the variables without being confused.

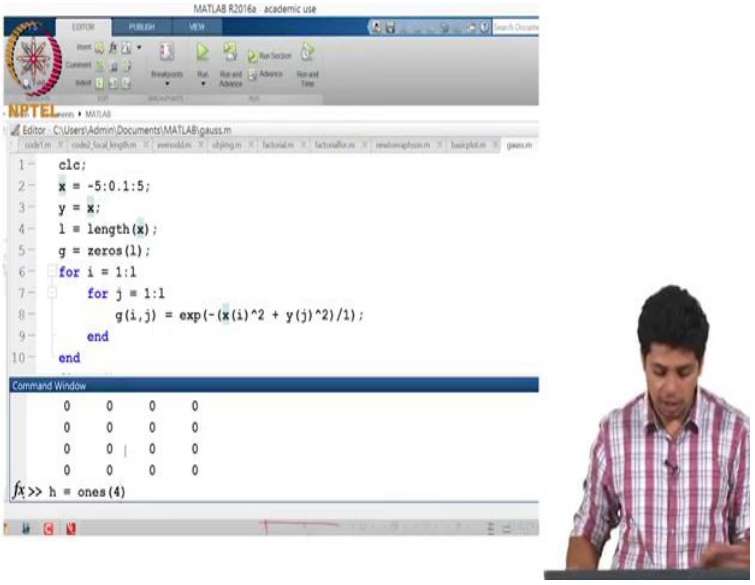
So, in the program x is given the values between minus 5 and + 5 with a step of 0.1 and I have chosen to give by the same values. So, what this will do is it will give me a square space with the site being - 5 to + 5 with a step of 0.1. So, it will remain square space then I have used a variable l and I have assigned it length of x.

Now this command length is used to determine the length of any variable any array that is given to it. So, now we know here from- 5 to 5 if the step is 0.1 then we will have 101 values. So, the l will store 101 then what I have done is I have used a variable g and I have assigned it this command zeros of l.

So, what this command will do is it will create an array of zeros of dimension l, and if I put only l here so it will by default give me a 2 dimensional array. So, what that means is say now we know that the value of l is 101. So, if I put g = 0 101. So, it will give me 101 by 101 matrix and all the values in which it will be 0 that is what it will do. If I want to create just an array of 101 values. So, I have to enter a command like say h is equal to zeros and in brackets I have to put one comma the dimension so 10 it will give me an array with 1 row and 10 columns and all of them will be having the value 0.

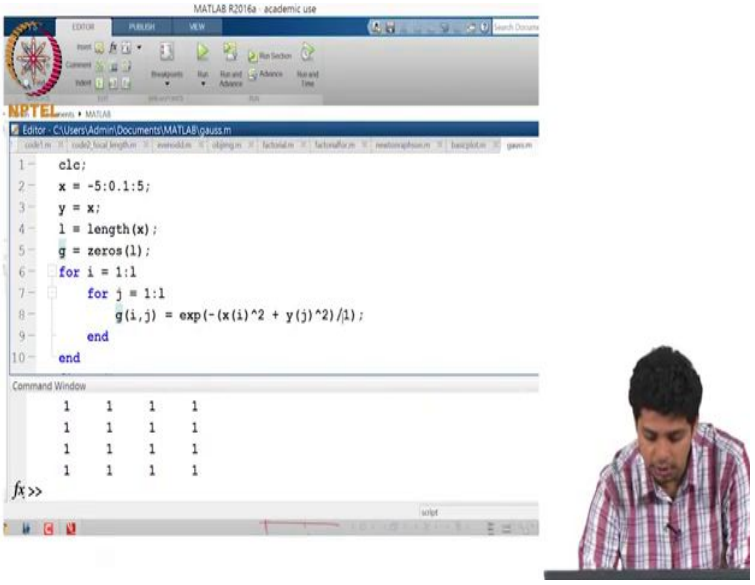
Now, this is what it means, but if I put h is equal to zeros and in brackets if I put say 4. So, what this will do by default is give me a 4 x 4 matrix and all of them will be 0 like this.

(Refer Slide Time: 54:03)



So, this is 4 rows and 4 columns now there is one more command that can be used and it is like this instead of zeros you can put ones and you can specify the dimensions and what it will give me is the matrix with corresponding dimensions and all the values in the matrix will be 1. So, like this.

(Refer Slide Time: 54:24)



So, this comes in handy when you are generating images. So, we declare this g and store 101 by 101 zeros in it. Then we use a for loop in which i a variable goes from 1 to l so it goes

from 1 to 101. Then another for loop j is the variable used in a second for loop and it will go from 1 to 101 again.

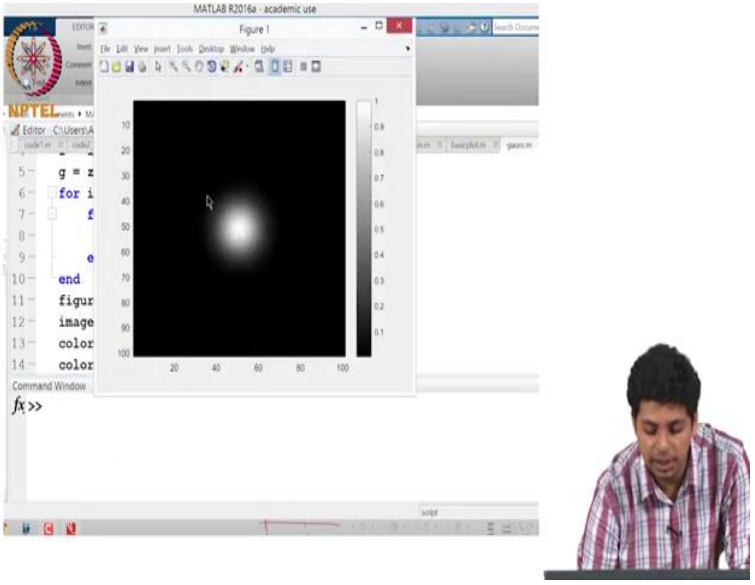
Now what we will use for use this for is g with index I and j so, ith and jth value of g will be given by exp is e to the power ok. So, this is another function that we can use. It is e to the power bracket minus and I have calculated it as x with the value of x square plus jth value of y square and divided by I have just put this 1 as our sigma square. So, here our sigma will be 1, sigma is the standard deviation or here in this case it will be the flatness or sharpness of the curve.

So, this is what I will use the value of as I have just used the value 1. And I end the for loop both for loops end. So, what it will do is it will generate all values of using x and y and it will store these values of the Gaussian terms in this g variable; g matrix. Now I use these 4 last 4 lines to display this image. What I use is the first command is figure it will create an empty figure then I use this image scan command and I put the variable as g.

So, what it will do is it will give me an image of g where different intensities or different colors will represent different values, then the next command will be color map gray. So, what I do is I. So, since different colors are representing different values I want those colors to be represented between black and white.

So, black will be 0 and the whitest part or the one with the maximum value of value within this g will be 1 or whatever value and it will be represented by white. So, 0 will be black and the maximum value will be white and then this color bar is just to show what different colors are used and you will see when I generate this graph.

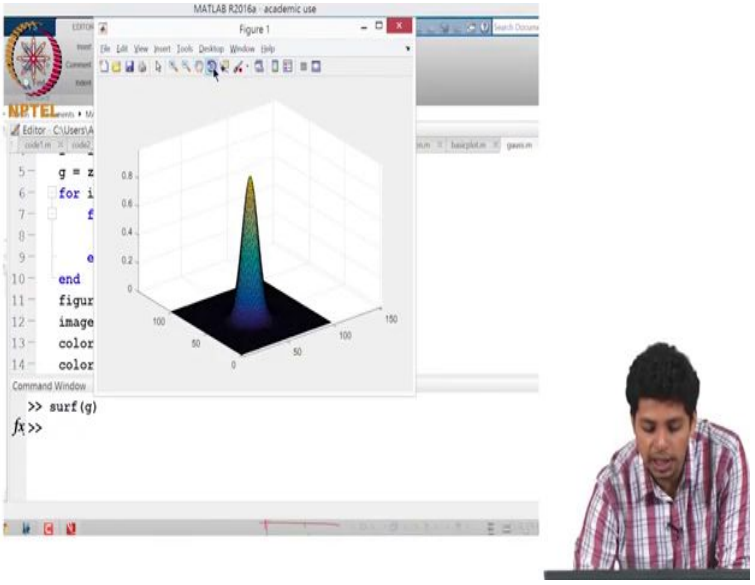
(Refer Slide Time: 56:49)



So, what we will do is we will just run this and this is the output that we get. So, as you can see there are 100 and 100 points by 100 points approximately it there are 101 points as you can see in the workspace, but it this will show an approximate just picture here there actually r 101 by 101 pixels is just displaying 100 and then the darkest part is black which has a value of 0 and brightest part is 1 since we did not give any amplitude value to it.

Now this is how your Gaussian wave will; Gaussian into waveform into dimensions will look like. Now there is one more way in which we can see instead of putting an image scan command here what I can also do is I can use this command surf.

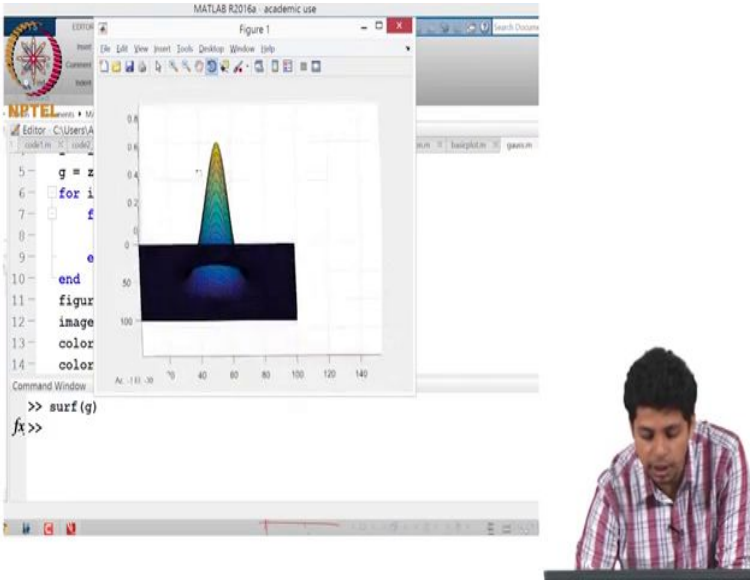
(Refer Slide Time: 57:33)



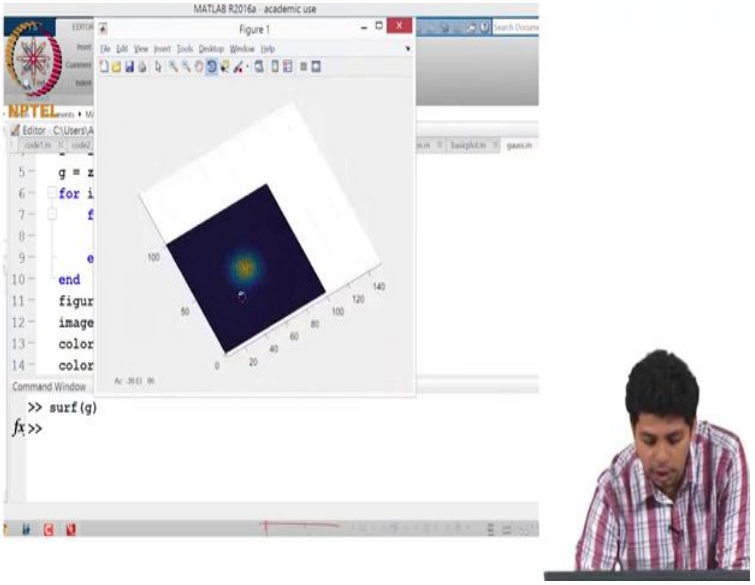
And then I can use this g variable. So, what this will do is this will generate a surface with all the values of g and I can visually visualize it in a different way like this. Now this is a surface plot of g and the maximum value as you can see is at a height and all the other values which will be very close to 0 after a certain distance they are all at a flat at z = 0.

Now we can see that there are different tools in this figure environment that you can use to manipulate the graph in different ways. So, suppose that I am using this hand tool. So, what I can do is I can shift this and if I use this second rotate 3D which looks like a block I can do this.

(Refer Slide Time: 58:24)



(Refer Slide Time: 58:26)



I can rotate it in different ways. I can have a look at it like this. So, if we make it like this it will look flat and the color coding will just give us the values at each point. So, this is how you can also plot different functions and different data points in MATLAB and we will use all of these commands in further lectures or in further experiments using MATLAB and also actually using python.

And further, we will actually use these commands and use this software in order to generate diffractive elements refractive elements in optical engineering and we will see how phase works you can also represent complex numbers in MATLAB so there is also that part to discuss and we will do all of this in the next session.

Thank you.