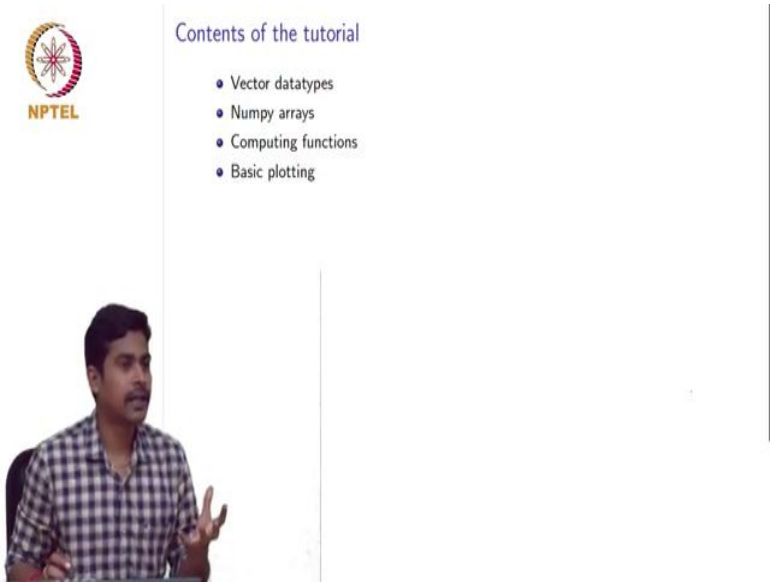
Optical Engineering
Prof. Shanti Bhattacharya
Raghu Dharmavarapu
Department of Electrical Engineering
Indian Institute of Technology, Madras

Lecture – 34
Python – part 2

Good morning welcome to the second lecture of the introduction to Python lecture series. Today we will see the continuation topics in python.

(Refer Slide Time: 00:28)



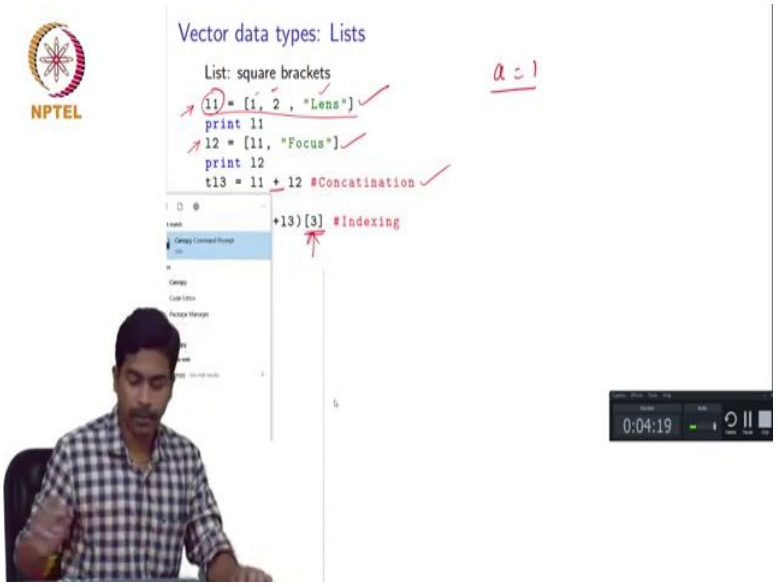
So, in the last tutorial we have seen the basic building blocks for any programming language to write a simple program. And they are the variables of different types of operations on the integers Boolean values and float values. And then we have seen conditional operations like, greater than less than comparison operations. And finally, we have seen iteration operations for doing the same computation over and over again.

So, those are the things that are required to write any simple program. So, now let us go into some more detail or in depth options in python. So, in this lecture we are going to cover these topics. So, that is a vector data type. So, these are similar to arrays in C programming and

after that I will introduce a numpy module. So, which is basically an array, but it is very powerful.

So, you can compute you can manipulate these arrays much faster compared to a normal list. And right later on we will see how we will use these numpy arrays to compute 1 day and 1D and 2D functions. And finally, I will show you how to plot data or measures in python. So, that is the content for today's lecture.

(Refer Slide Time: 01:46)



So, let us go ahead and do a vector data type. So, in the previous lecture we have seen variables for example, integers floats or Boolean values, but they are single values. So, for example, if we say it equals 1. So, a can store only a value 1. So, if you want to store multiple values for example, multiple integers or multiple floats. So, there is something called list in python. So, in lists we can. So, this is the syntax for a list in python.

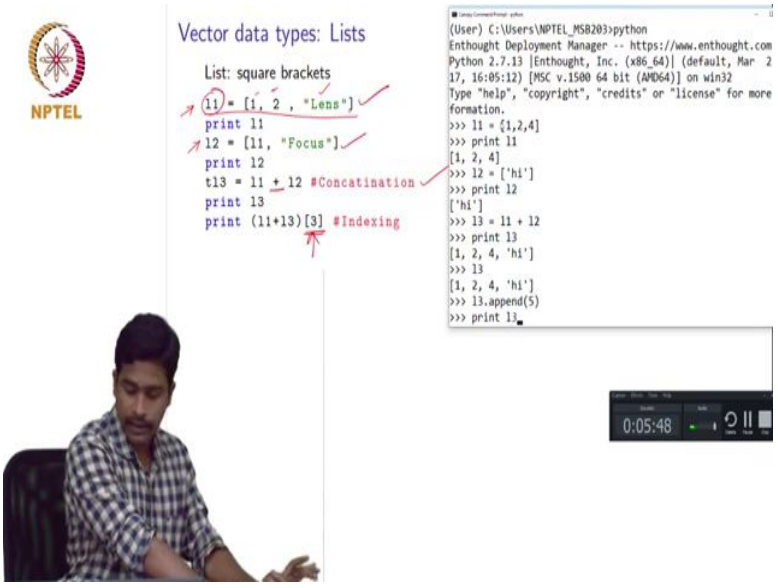
So, where this is the name of the list and the list opens with the square braces and then all the values are separated by commas and finally, we have to close the list using a closed brace. So, this is the syntax; unlike the arrays listed in C or Java programming. So, the python list can store multiple data types if you see so, the first value and second value are integers and the third value is a string value ok.

And if you see the second statement here. So, a list also can store another list ok. So, here we have declared l_1 has a list in the second list l_2 the first element is l_1 itself and the second element is a string ok. And these are the there are some of the commands or some of the operations that we can do on lists. For example, if you have two lists l_1 and l_2 we can use this plus operation to join these two lists. And if you and none of them are just like we have seen in the last lecture.

So, if you want to address a string if you want to find the first element or second element in a string. So, we use this notation. Similarly if you want to find the nth member of a list you have to use index again inside a closed and open square braces. And again, the indexing starts from 0 not from 1 in a similar MATLAB. So, now, let us try these commands in the python shell.

So, in order to open the python shell as I said in the last lecture. So, you have to open this canopy command prompt, which I have already opened for you.

(Refer Slide Time: 04:00)

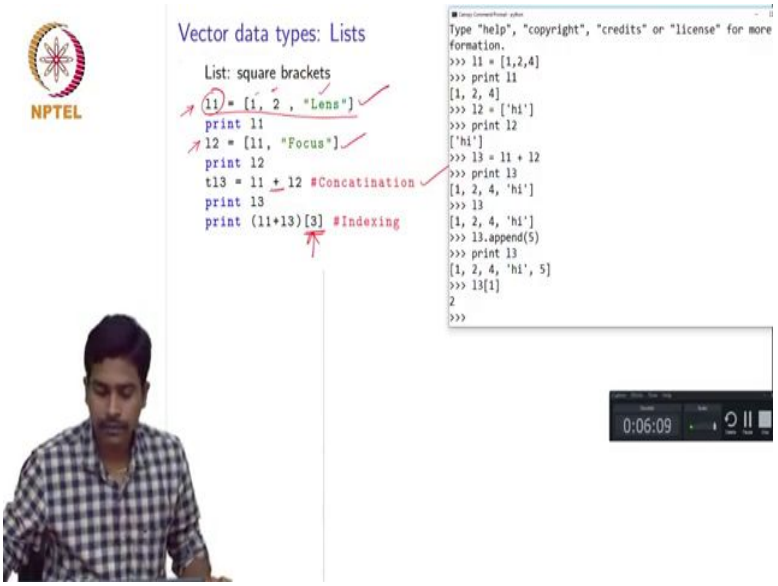


So, after you open this you just have to type python. So, now, it will open the python shell. So, if you do not type python and if you try to run your program it gives error once you type python it goes into the python shell ok.

So, now let us try out the commands that I have shown here; let us say l_1. So, print l_1 give the values l_2 and 4 and let us create another list. So, this list has only strings say and print l_2 and give the value. So, let us create another list l_3, this time it is the concatenation of l_1 and l_2. So, now, it has the values of both l_1 and l_1. And if you want to add just 1 element to the already existing list. So, there is another command called append.

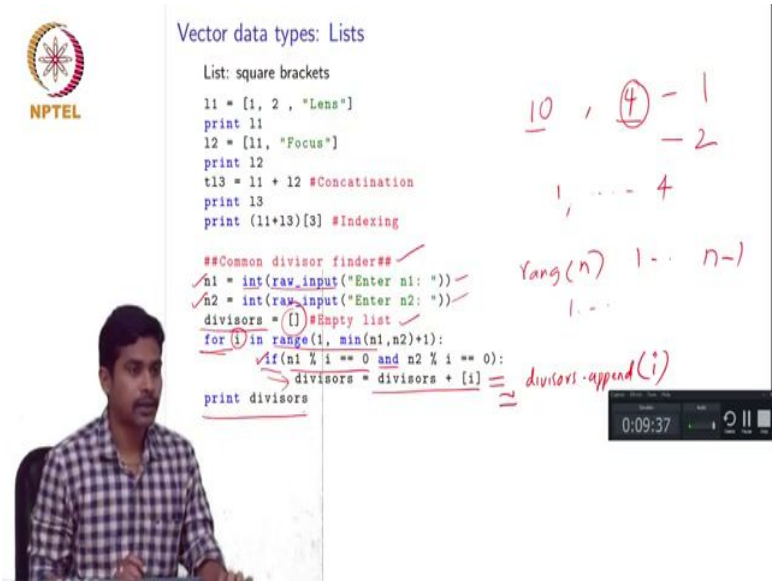
So, now, I will show you how to use an append command. So, let us take the string l_3. So, which has 1 2 4 and this string if you want to add another number. For example, you want to add number 5.0 to this list. So, the command is l_3 dot append and the value that you want to append ok. So, now, this value is already added to the l_3. So, to see it , write the list again.

(Refer Slide Time: 05:24)



So, now if you see here the value 5 is added at the end of the existing list. So, now, let us see how to index a list. So, if you want to print the second element of l_3. So, the second element means indexing is 1 not 2. So, it has given the value 2 as the output. So, these are the different basic operations that we can do using list.

(Refer Slide Time: 05:53)



So, now let us see an example where we have to use a list. So, this program you can open the same program that I have given to you. So, here what we are doing is we are finding the common devices of two numbers ok. So, you all know what is a common divisor. For example, if you have numbers 10 and 4. So, what are the common divisors? So, 1 is a common divisor for any 2 numbers and the second 1 is 2 I think that is all. So, 1 and 2 are the common divisors for the given two integers.

So, now let us try to see what this program here is doing. So, the first n_1 and n_2. So, we are using the same raw input command that we have used in the previous lecture and then we have converted them into integers using this integer command; after these two statements. So, we get the values of 2 numbers from the input. Initially we have created an empty list which means. So, we can create an empty list using this open and closed braces without giving anything inside it.

And now we are using a for loop to find the different common devices. So, how are we doing it? So, the algorithm is so; obviously, if there are two numbers. So, the common devices can divisor cannot be larger than the smallest of the 2 numbers right. So, in this case for example, if we take these two numbers. So, the common divisor the Mac the largest common divisor is always less than or equal to the smallest number of these 2 ok.

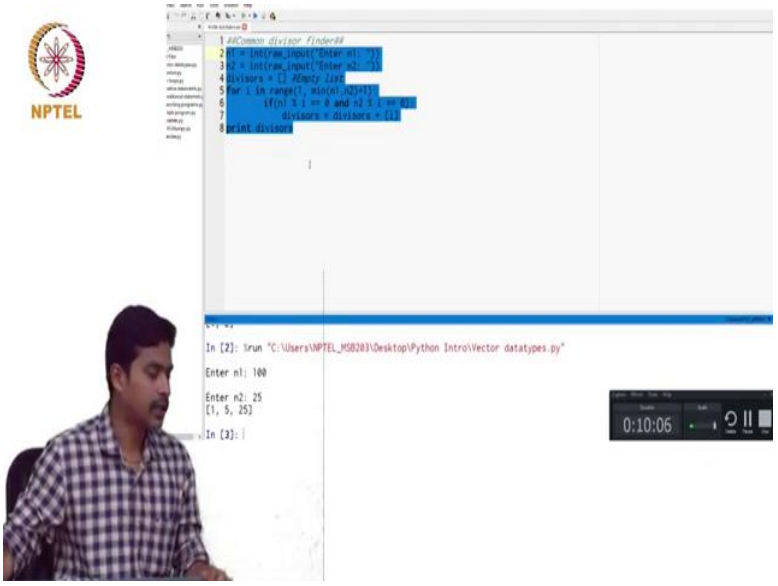
So, what we are doing here is we are taking the numbers from 1 all the way to the smallest number. And we are using an if statement to see if this number is dividing these two numbers or not. So, if the number is dividing the 2 numbers we are adding it to the divisors we are appending that number to the divisors list. If it is not we are going to the next number in the for loop ok.

So, again I have started a for loop and the range of numbers that I am taking here is l_2 minimum of these two numbers +1. Because if you remember in the last lecture we have seen. So, range command will only give the values from 1 to n -1. So, in order to compensate for it. So, we have to get value from 1 to n. We are adding 1 to the smallest number. And in the if statement I am checking if I remember.

So, which is in the value list of values is dividing using percentile operation. I am comparing if it is divided using the percentile operation and again and conditional statement sorry, logical statement to see if the number in the list is dividing both or not. And if it is dividing and I am coming inside the for loop and I am appending. So, in appending as I have said before. So, you can append by using the append statement or you can also create a new list and add it to the existing list ok.

So, this is similar to dot. So, both statements are allowed. So, it gives whichever you want and finally, we are printing the divisors.

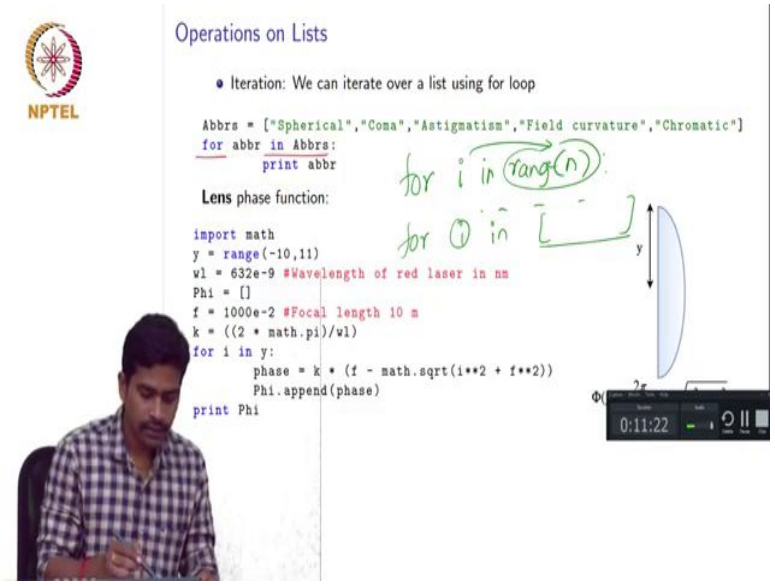
(Refer Slide Time: 09:14)



So, let us go back to the python canopy editor. This is the same program that I have shown you in the slides. So, I am running it. So, let view the 2 numbers that we have seen 10 and 4.

So, now it has printed the values 1 and 2. So, we can try a different set of numbers we can say 100 25. So, 1, 5 and 25 are the common divisors ok. So, now we have seen what are lists and how to access different numbers from a list and how to add numbers to a list. And we have seen a simple example program that uses the lists command.

(Refer Slide Time: 09:57)



So, now, in this slide we will see different operations that we can do on lists. So, as we have seen a for loop in for loops. So far what we are doing is we are using the following statement following syntax we are saying for i in range of some number. And inside this for loop so, the i value is being assigned the value sequentially we can do the same thing in without giving this range command what we can do is we can write for i or some other variable a in a list we can directly give the name of the list ok.

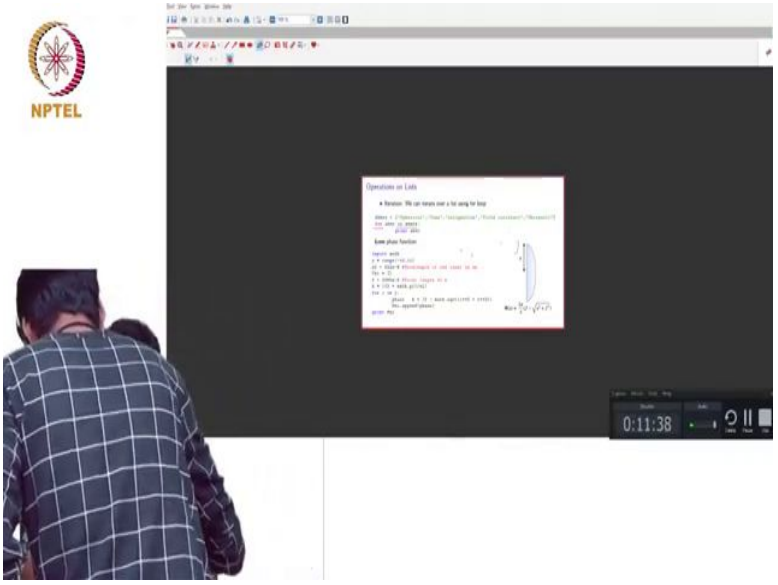
So, to demonstrate that I have, how do I erase this?

Student: (Refer Time: 11:03) in this.

Ok.

Student: No (Refer Time: 11:07) open (Refer Time: 11:09).

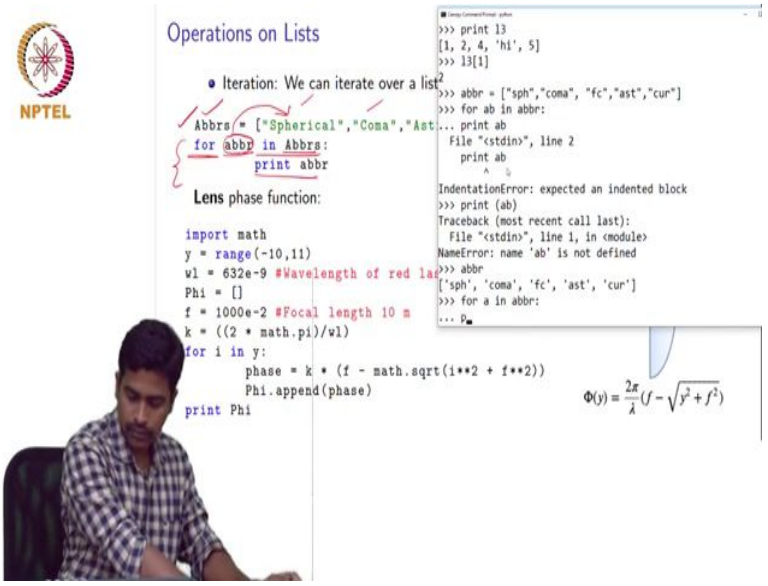
(Refer Slide Time: 11:09)



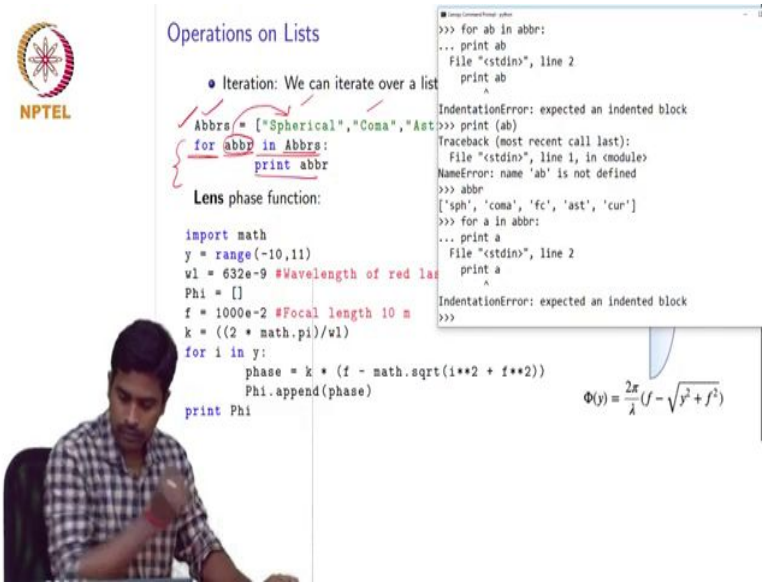
So, to show that. So, this is a simple example. So, Abbrs is a list and the numbers of the lists are different types of aberrations ok. So, you know there are 5 major types of aberrations, which are Spherical, Coma, Astigmatism, Field curvature and Chromatic. So, I am in this simple script. What I am doing is, I am using a for loop to print all these different kinds of aberrations one after another.

So, Abbr is a dummy variable that we are using for the iteration and the list here is the list that we have declared in the previous statement ok. So, now, what is happening here is. So, this dummy variable Abbr will be assigned one of the number members of the list on each iteration. And using the print statement we are printing the values of this variable ok. So, we can quickly try it out in the python shell ok.

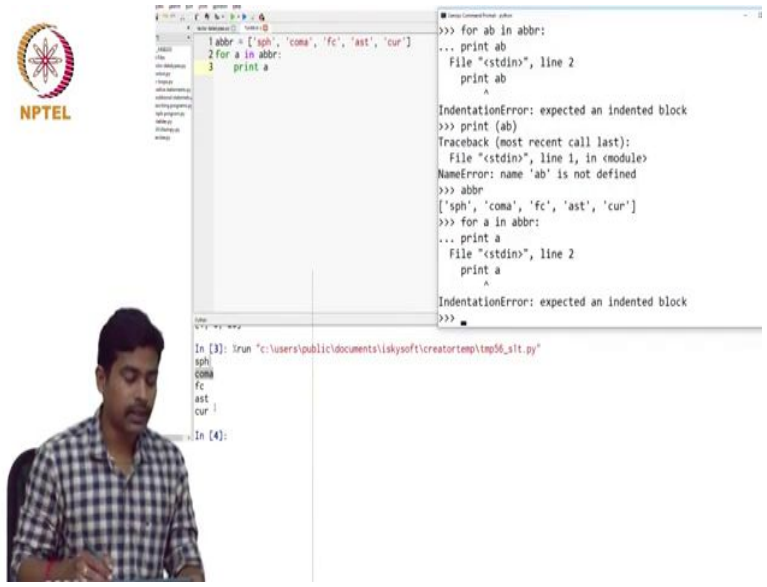
(Refer Slide Time: 12:37)



(Refer Slide Time: 13:37)



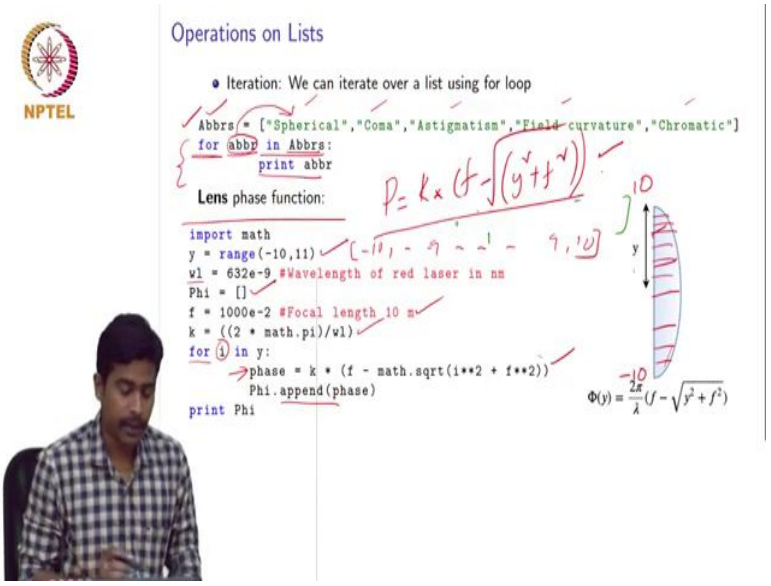
(Refer Slide Time: 13:55)



.

So, now I have declared this list abbr. So, we have declared a list that has the value that has different types of aberrations we used for a loop and a is the dummy variable here. And on each of the iteration step a is assigned one of these strings and we are printing the strings in the output command.

(Refer Slide Time: 14:39)



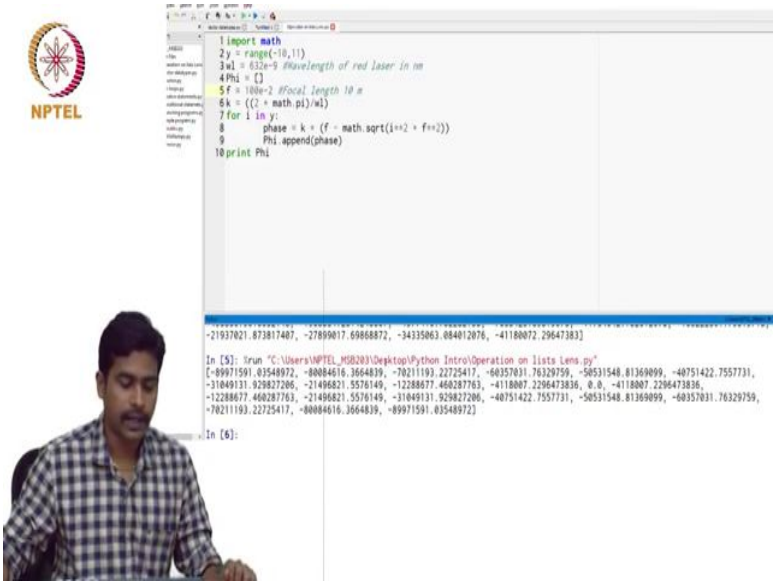
So, now let us see an example function. So, where we want to calculate the phase function of a lens. So, the phase function of a lens is given by this equation. So, P is given by k into. So, this is the phase function of a lens.

So, as we. So, this is the lens and you can see the thickness of the lens in the y direction is varying with this profile of the lens ok. So, which means the thickness of the lens is a function of its position along the y axis. So, now, let us create a sample of this lens into 10 equal points and that is done using this strain statement.

Let us say this is -10 and this is + 10 and we have around 20 points along the y axis and let us say the wavelength is 632 nanometers. And we have initialized an empty list, which is the phase which is going to hold the values of phi along these y axis and let us say the focal length is 10 meters and k we know $2\phi/\lambda$.

So, it calculated this using this equation. And in the for loop and i is the dummy variable that is going to take the values in the range minus 10 - 9 and so on. And this is the equation that I have written here to calculate the phase. And after calculating this phase I am adding these value phase values to the total phase list using this append command.

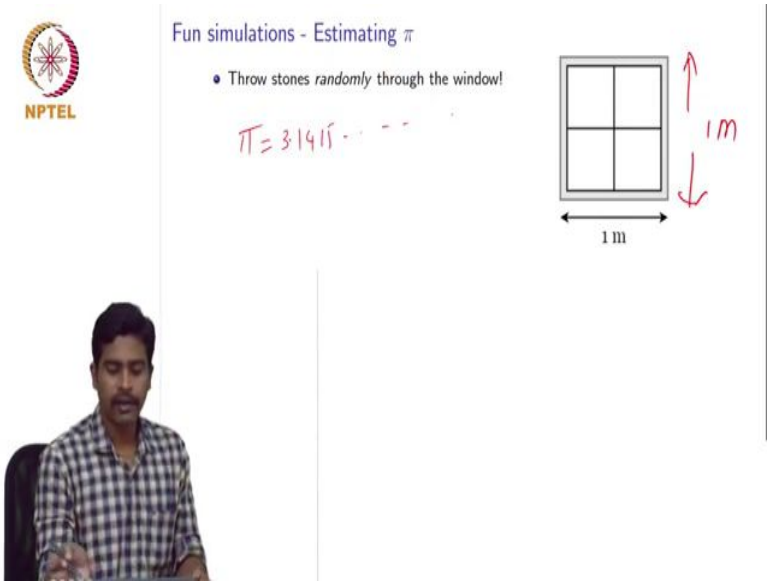
(Refer Slide Time: 16:30)



So, let us see how this works. So, this is the same program. So, let us run this. So, with the given values, these are the phase values. So, that the program has calculated. So, we can try

giving different focal lengths or you can try giving different wavelengths based on the fact that the phase values of lengths are going to change. So, this program I have given so that you will know how to manipulate the list and how to do different operations on lists.

(Refer Slide Time: 17:01)

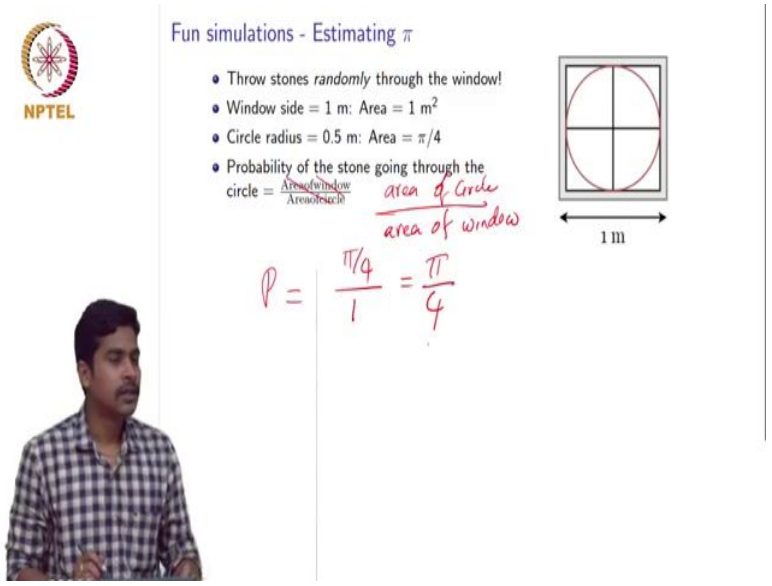


So, now let us move on to some fun program. So, now, you all know different aspects of the python program. So, now, let us see if you can write a simple program. So, if you have computers, I suggest all of you to follow this simple simulation and you can write your own program and let us see what this program is.

So, now, let us assume there is a window in your room and this is the window and the windows length is 1 meter. So, this is 1 meter and this is also 1 meter.

And now you are asked to randomly throw some stones in the window ok. And let us assume all the stones are going through the window not somewhere on the walls or anywhere. So, now, using the simulation how to find the value of π we know π is the ratio of a circle circumference with this with its diameter. And π is also an irrational number and the value of pi is given by 3 4 3.14 15 and a lot of decimal points.

(Refer Slide Time: 18:15)



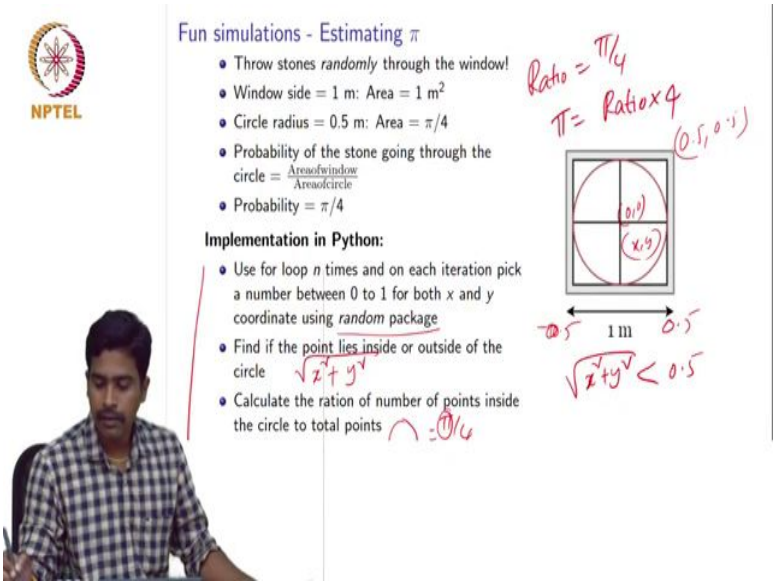
So, now let us see how this simulation is going to give the value of π to us ok. So, just imagine this imaginary circle inside this window and the diameter of this imaginary circle is also 1 meter just like the window ok. Suppose if you are throwing some 100 stones inside that window and if you somehow can figure out how many stones are going through this imaginary circle.

So, if you divide the number of stones that are going inside the circle, with the total number of stones that you have thrown inside the window. So, that is going to roughly estimate going to be the ratio of the areas of the circle and area of the rectangle. So, am I clear?. So, now, let us see as I said the side of the window is 1 meter. So, from that we can calculate the area of the window is 1 meter square. And I said the radius of the circle is 0.5 meters and the diameter is 1 meter.

So, from this we can see the area of the circle; area of the circle is $\pi/4$ but we do not know what pi we want to see and what is π using the simulation. So, suppose if you are throwing a number of stones let us say and if after throwing each of these stones, you are random you are throwing systems randomly. But you are not preferentially throwing ok, this should go inside the circle or this should avoid the circle, but you are randomly just throwing the stone.

So, after throwing each of the stones you are measuring if it is going inside the circle or if it's going inside the rectangle, but not inside the circle. So, if you know this from probability we can say the area of sorry this is wrong. So, the probability of the stone going inside the circle is given by the area of the circle divided by the total area or area of the window ok. So, the probability is given by pi by 4/1 which is $\pi/4$ ok.

(Refer Slide Time: 20:36)



Now, we are doing a random simulation in python. So, here what we are doing is. So, we are picking some random numbers between. So, let us just say this is 0.5 and this is 0.5 and this is 0 , 0 this is 0.5, 0.5 so on ok.

So, now we are picking numbers randomly between 0 - 0.5 and 0.5 for both x and y coordinates. And we are computing to see if this randomly picked number is lying without the circle or not. So, how do we do it? So, if you have a random x comma y number. So, we can calculate the distance of this point from the origin using the square root of $\sqrt{x^2 + y^2}$ and if the distance is less than 0.5. So, we can say the number is lying inside the circle.

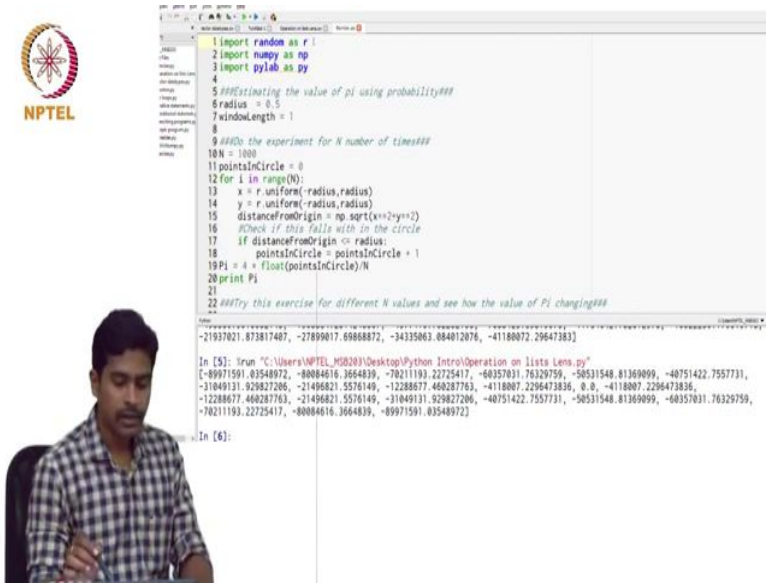
So, this is the idea that we are going to have and write the program and this is the implementation step. So, what we are going to do is we are using a for loop to simulate a number of trials or a number of stones throwing into the window and on each of these iterations. So, we pick a number between 0.5 to 0.5 or 0 to 1 for both x coordinate and y

coordinate. So, after you use the random package in python ok, I will show you what a random package is.

So, after picking these two numbers, we have to calculate the distance of this point from the origin ok. So, using this formula square root of $\sqrt{x^2 + y^2}$; based on the distance we have to find if the value is lying inside the circle or away from the circle. So, if the value is lying inside the circle. So, we have to increment some dummy variable that we create at the starting of the program. So, after finally, as you run through the n number of iterations you just have to take the ratio of the points that lie inside the circle and the total number of points and this ratio is going to be $\pi/4$

So, we do not know what is π, but we do not know what this ratio is going to be at the end of the program. To calculate the pi we just have to multiply the ratio with 4 and it roughly gives the value of π.

(Refer Slide Time: 22:49)



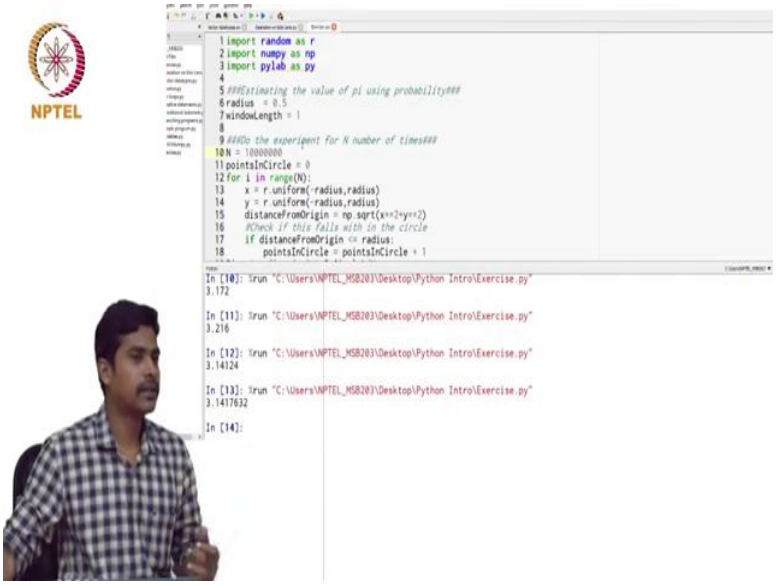
So, I will show you the program I have already written or if you want I will give you 2 minutes you can write it by yourselves otherwise I can just open. So, this is the program and here if you see the first line of the program. So, we are importing a package in python called random and we are giving it some nickname r; because writing random dot random is going to take a lot of effort.

So, from now onwards we refer to the random package as r and we are importing the numpy package, which I have not discussed, but I will show you later and finally, the some other package pylab. So, here I am saying the radius of the circle is 0.5 and the side of the window is 1 and I am taking some 1000 number of random trials ok. So, again I am using a for loop and on each of these trials I am taking two numbers. So, x and y from within -0.5 to + 0.5 again y coordinate is also lying within minus 0.5 to 0.5.

So, I am calculating the distance using this equation that I have shown before. So, I am using this if statement to see if this point is lying inside the circle or not. So, that I am doing the distance that I have calculated in this step if it is less than the radius, I am incrementing this variable which I have declared here point inside a circle initially it is 0. So, whenever a point that is randomly picked lies inside the circle I am incrementing this variable using this command and finally, as I have said in the.

So, the ratio that we get from the program holes to $\pi/4$ and the value of π is given by a ratio multiplied by 4. So, I am using the same equation here. So, where π is given by so, point inside the circle by the total number of points and finally, I am printing the value of π So, let us run this program and see what would be the estimate for π. So, with maybe let us just start with 100 trials I am running the program. So, with 100 trials the program is estimating the π value as 3.08 ok.

(Refer Slide Time: 25:15)

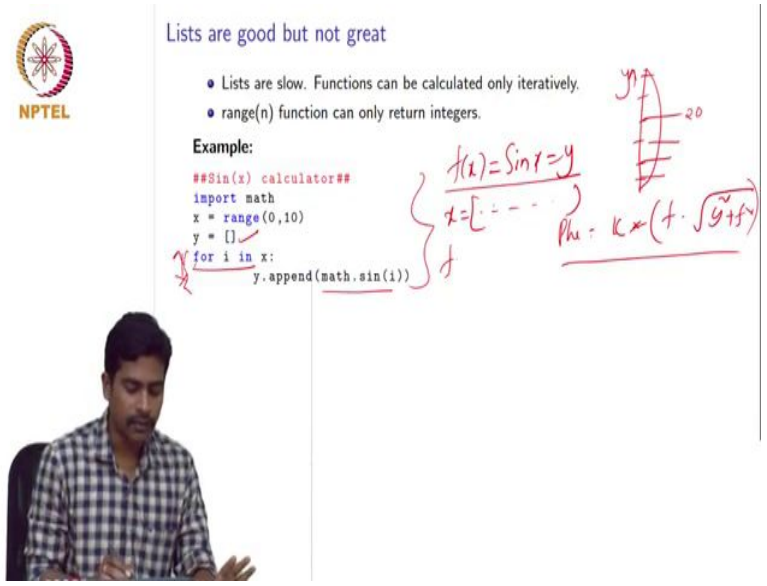


And if you run the program again. So, it's going to give some other value because we have use the random package and each time we are getting different different set of values we are not going to get the same number of values that we got in the first run ok. So, let us run it again. So, it's giving 3.36 again 3.48 and so on. Let us try to know the pi value is 3.14 15, but the values that are given are close to the pi value, but it is not very accurate.

So, in order to make it more accurate. So, we can increase the number of random trails and let us increase it to 1000 and see. Now it has come down to 3.17 which is very very close to the actual pi. So, let us run it again. So, it's giving 3.26 maybe we increase the number of trails to a large number like 1 million. So, it's getting very close to the actual pi value right 3.14. So, if I still increase the value it's going to take a little time; because it has to iteratively calculate this let us wait.

Let that run. So, after doing this iteration it has set 3.14 17 is the value for pi. So, if you keep increasing the iterations, it's going to converge to the actual value of pi ok. So, I hope you found this little exercise interesting.

(Refer Slide Time: 26:52)



So, next now let us go into something very useful in python and very powerful. So far we have seen lists in python, basically a list is the group of different types of integers of loads or strings. These are good, but they are not very great. If you are doing this you know sequential kinds of operations.

If you are I will show you what I mean by sequential operations in this slide. So, as I have shown in the previous exercise we have calculated the phase function of a lens using the list right. So, we had a lens and we and this is the y axis and we have sampled this y axis around 20 points. And then we use the equation $\pi = kf - \sqrt{y^2 + f^2}$ got its equation to calculate the phase at each of these points.

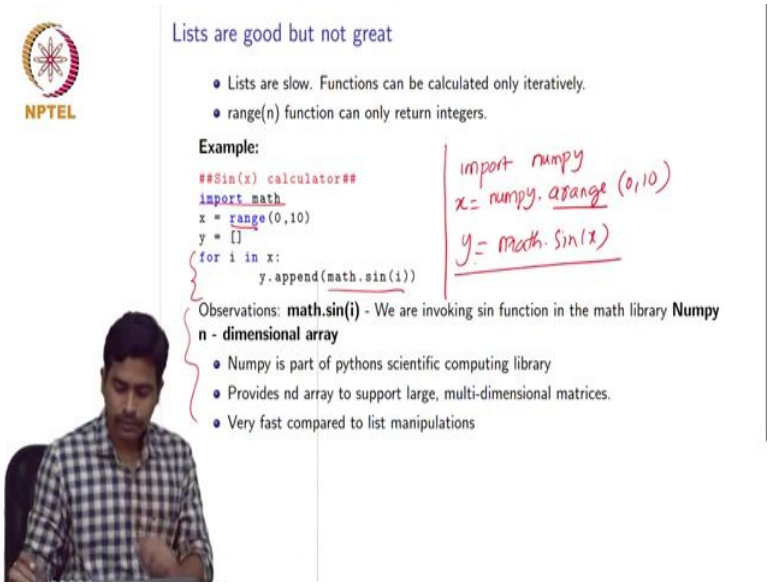
So, we did this using a for loop. So, this is and this is reasonable when you have some very small number of y values. So, like 100 or 1000 points in your list, but as the number of values or number of integers in your list keeps getting more and more like 1000, 10,000, 1 million.

So, this for loop operation is not very useful and it gets very slow. So, which we have seen in the previous example. So, where we have tried to calculate the pi value using 10,000; we have seen the program has slowed down quite a lot. It took some 10 seconds or so, to calculate the value.

So, in order to calculate this equation at 1 go, there is something called a NumPy in python. So, I will just show you as a brief example here. So, in the previous case what we are doing is. So, if you want to calculate a for example, a function. So, the function is $f(x) = sin(x)$ or this is y. So, in order to calculate this what we are doing is, we are taking x values from so on. So, 0 to 10 and we are using a loop here and on each of these for loops we are calculating the value sin x and we are appending this 2 y ok.

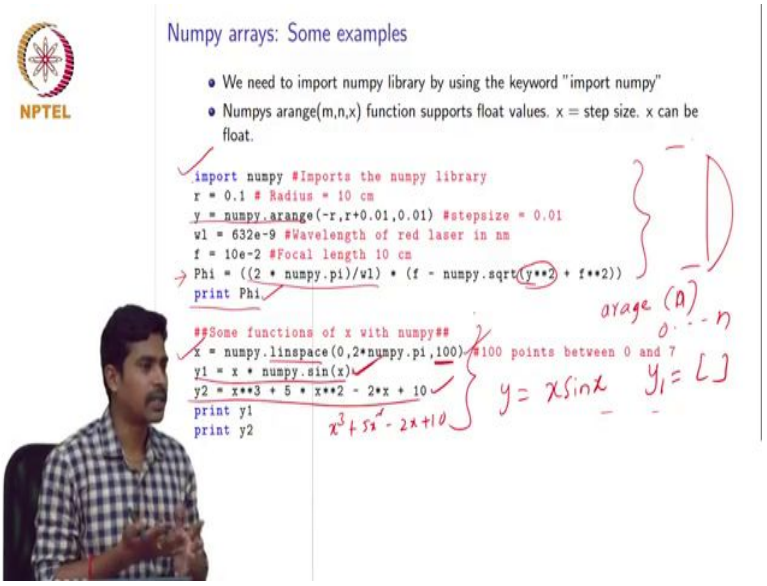
So, this is and this is easy to write, but this takes a lot of time; especially these two lines ok.

(Refer Slide Time: 29:22)



So, these are the observations I want you guys to make at this point. So, here is this command to calculate the sin value, we have to import this math from the library and math dots sin we will calculate the value of sinusoid function for all these i values .

(Refer Slide Time: 29:45)

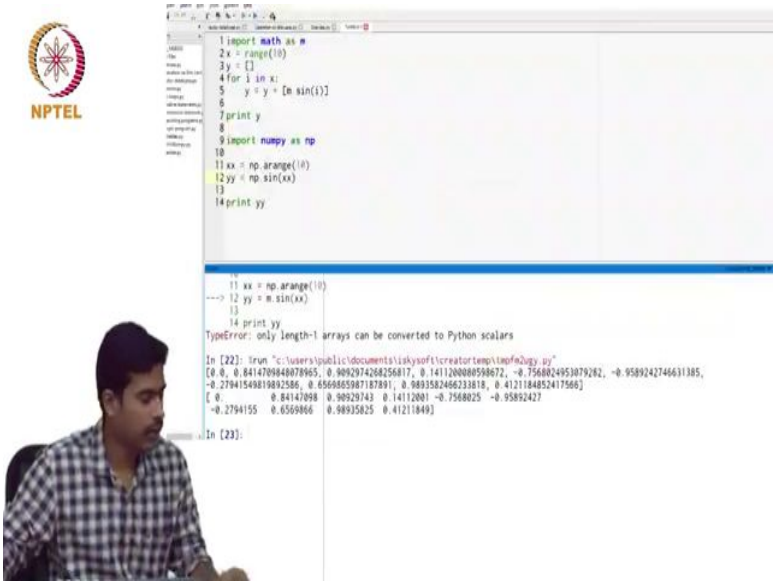


So, now let us see how we do the same thing using a numpy array. So, for this we have to first import the package using this command import numpy. And sorry what I am doing here ok.

So, I will quickly tell you how to do the same thing in numpy. So, the equivalent steps first we have to import the numpy and after that we have to create similarly the range of x values that we want to use to calculate y so, but there is a slight difference. So, earlier we used this range command, which gives a list of values from 0 to 10, but here we are using another command, a range. So, this command will give a numpy array which is different from list. And so, these two steps can be written in just a single step y equals math x that is all, we do not have to write any for loop.

So, the output will be a list of values that are sinusoid of x. So, let us just quickly go into python and write this program ok.

(Refer Slide Time: 31:12)



Student: Sin i.

Sorry.

Student: Sin i.

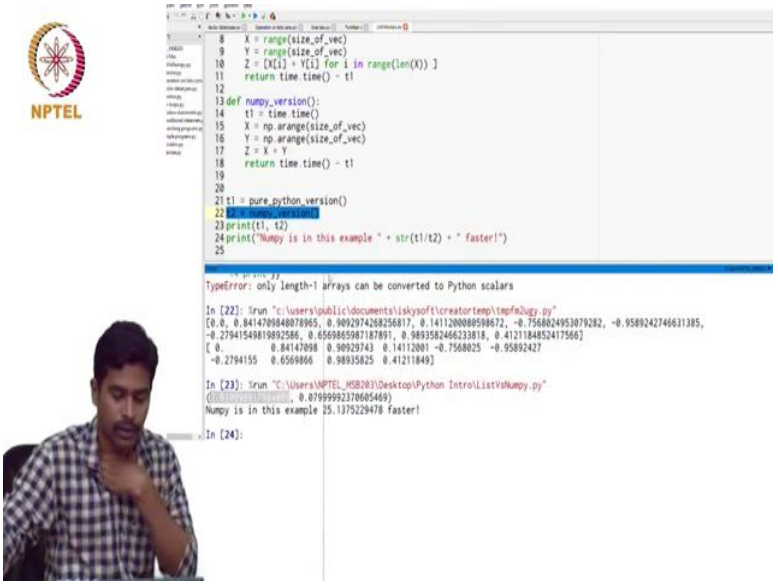
Thanks. So, this is how we normally use for loops to do the same thing as I have said before everything is the same, but we just have to make an array not just. So, for that I have to import numpy. So, these two lines are exactly similar to what we have written here, but here we are making use of the numpy array. And if you run the program if I run ok. Instead of using math dot sin you have to use the sine function in the numpy library otherwise it gives the error that you have seen.

(Refer Slide Time: 34:22)

So, these two are identical and if you are you I will see if I have the program. So, this is a simple program I have written to compare. So, the efficiency and the speed of operations, when you use numpy and normal lists. So, I am taking a large number of iterations. So, 1 million iterations or so, and I am defining a function. I have not explained what the functions are in python.

But for now just imagine it's just a function like any other programming language. So, here I am defining two functions. So, in this function I am just using the for loop to calculate some operation on such a large number of arrays.

(Refer Slide Time: 35:12)



And in this function I am doing the same operation, but I am using a numpy array. So, I am calling these two functions and I am calculating the time that it had taken when you use just for the list operations and if you use the python numpy.

So, I will just run the program and hope it will take less. So, these are the times that I have taken by the for loop the first. So, it took around 2 seconds to calculate using for loops and it just took 0.07 seconds to do the same operations using numpy and in this I am just comparing how fast numpy is? So, for this case its numpy is 25 times faster than for loops. And I can increase this number. I am afraid it may take a lot of time, ok.

Let it run (Refer Time: 36:25) ok, just believe me numpy is very fast because this program is taking a very long time.

So, now we have seen that the numpy array is much faster than the list. So, now, let us see how we do some of the examples that we have seen before using lists. So, in this function. So, what I am doing is, I am calculating the phase function of a lens. So, the only difference here is instead of creating a list using range command, I have used a range command to calculate the steps from like in this lens the values from minus y to y. And instead of writing the for loop here I have just directly written the value of y in this equation.

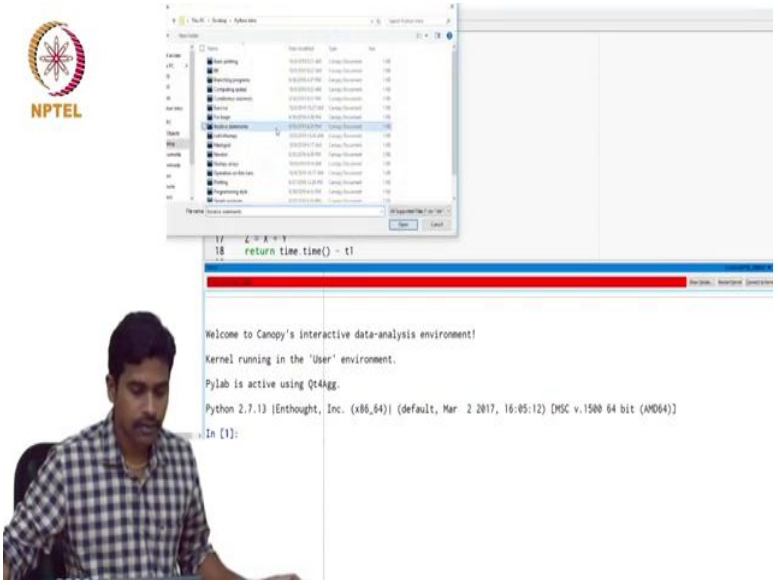
So, this is equivalent to writing an equation type, this is equivalent to just typing an equation you would not have to write for loops ok. And we are printing the value of pi using this command and this is some other example. So, here what I am doing is I am taking x values from 0 to 2π and I am taking 100 samples from 0 to 2π using this line space command.

So, what a range does is. So, a range will give if I say a range of n it gives 0 to n values equally spaced integers, but in line space you can specify how many points how many samples you want in that interval. So, in this case what I am asking the program to do is, give me 100 equally spaced values between 0 and 2π ok. So, the command will give you 100 vial values. So, in this first one what I am calculating is this function. So, y equals x into sin x.

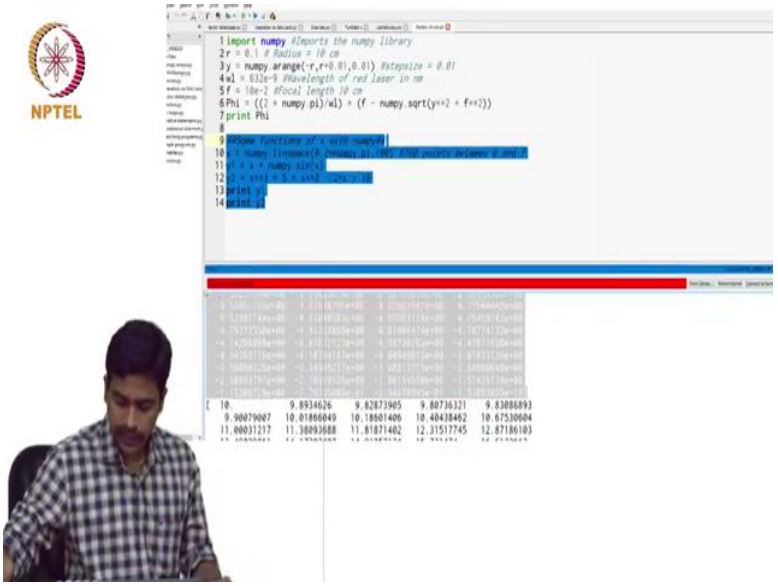
So I just have to write just like any mathematical equation, but it will you know interpret as a list I mean as a numpy array without for loops. But normally it will do the same thing that we have done in numpy. We have to create a list, create a dummy list and write a for loop and on each of the iterations. We have to append the value of the computation to the new array, but in python we can just write this expression and it will calculate the values. And the second expression I am calculating this quadratic cubic equation $x^3 + 5x^2 - 2x + 10$ in just 1 simple statement ok.

I hope you all understood how easy it is to write the equations or expressions in numpy compared to for loop.

(Refer Slide Time: 39:21)

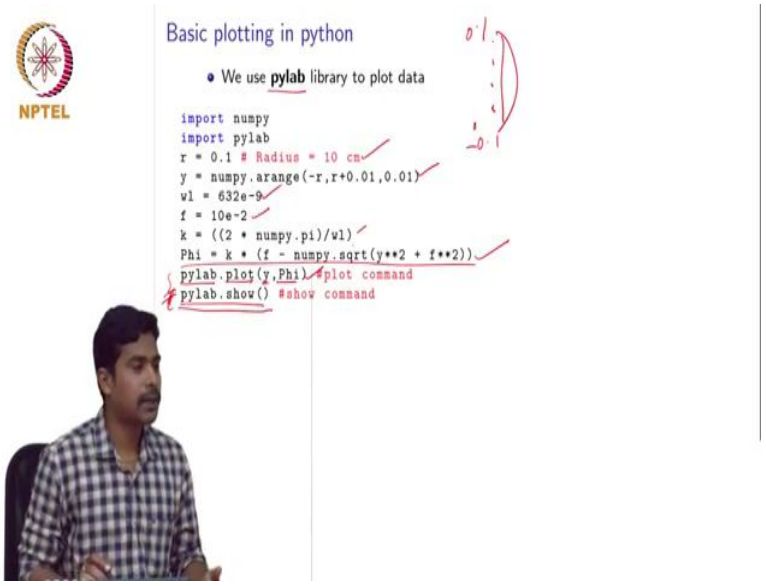


(Refer Slide Time: 39:32)



So, let us run these programs if this is working ok. So, these are the programs that I have shown you before. So, in the first one we are calculating the. So, the radius or the phase values for a lens and these are the values of the phase at different points. In the second function I am calculating different x sin x functions and quadratic expressions. And these are the values printed by the function this is the; this is the y 1 and from here onwards it is the y 2 ok.

(Refer Slide Time: 40:02)

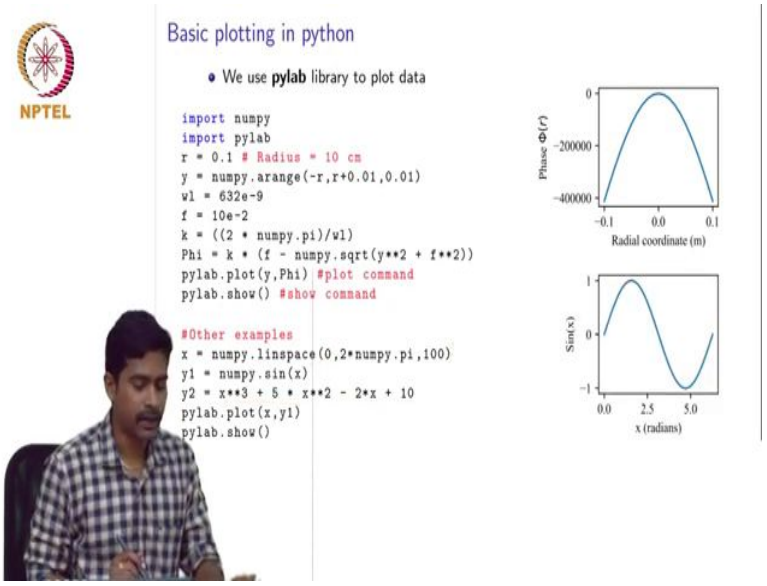


So, now, we have seen how to calculate mathematical expressions using numpy. Now let us see how to plot them in python. It's very easy. So, in order to plot data and python you have to use this library called pylab just like numpy. It's another lab, another library pylab. And so, in this example what I am going to plot is the phase function of a lens. So, that we have seen before. So, the radius of the lens is 10 centimeters and so, this 10 centimeter radius I am sampling at the interval of 0.01 ok.

So, what I am doing here is minus 0.1 to plus 0.1 I am sampling with a spacing of 0.001 ok. And the wavelength is 633 nanometers and the focal length is 1 micrometer r 1 and k; again is the wave vector $\frac{2\pi}{\lambda}$. And using numpy we can write the phase equations in just one expression and these are the two main commands that we use to plot this data ok.

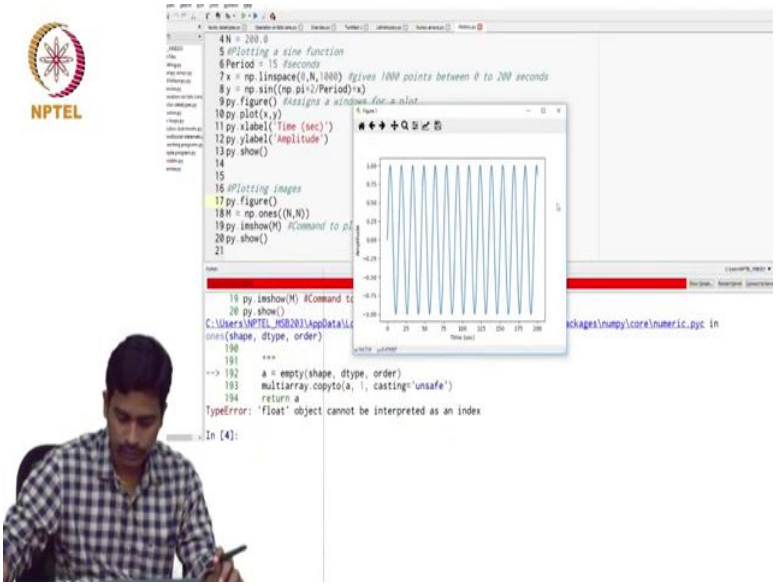
So, pylab is the package that we are using and plot is the command that will plot the data. And for the plot command we have to give both the values of x and y value y coordinate ok. So, here the x coordinate is actually y, because these are the values against which we want to calculate the phase of the lens and y value this is going to be the phase at that particular location. And finally, we have to give this statement a pylab dot dot show to show the plot if you do not write this command. So, it will execute this, but it will not give the plot for you to visualize ok.

(Refer Slide Time: 41:53)



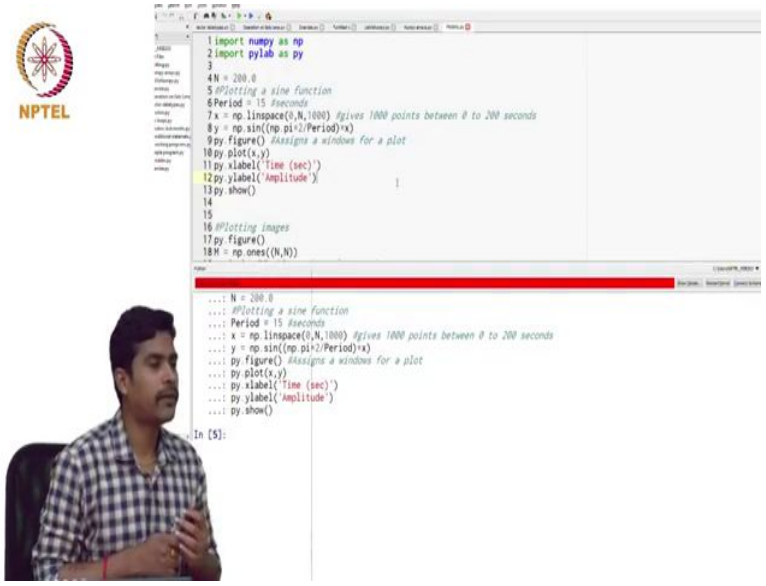
So, you always have to write this statement after you use this camp command this is very simple and. So, I can go back plotting.

(Refer Slide Time: 42:07)



So, this is the program that I wrote to plot the data. So, it opens up. So, sorry.

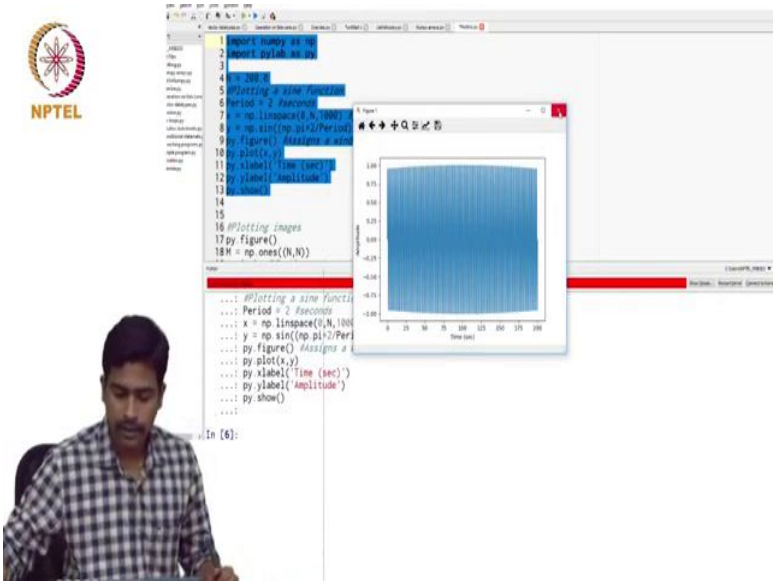
(Refer Slide Time: 42:34)



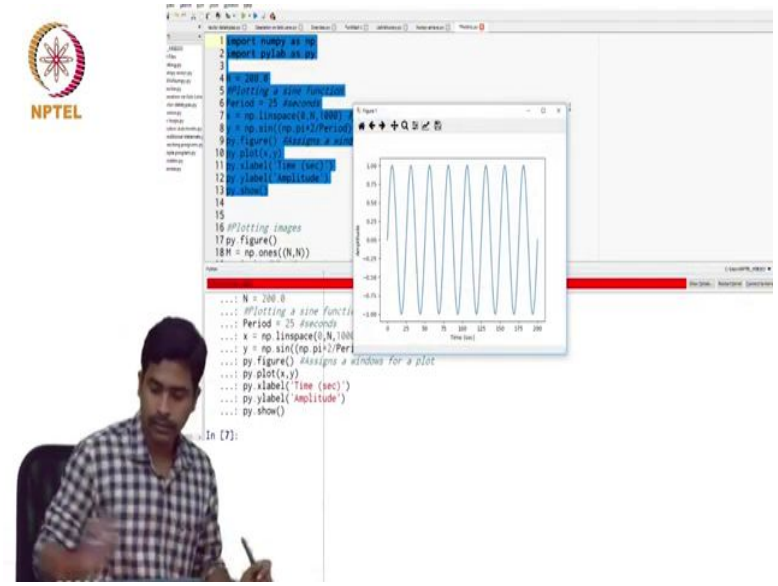
So, in this program I have plotted a sinusoid function. So, in order to plot the sinusoid. So, I took 15 periods to plot and I have taken 1000 points along the x axis and the value of sinusoid is calculated by this function y equals to sin of so and so. And this figure gives a figure canvas to plot and this command, as I have said before, will be used to plot the data. And if you want to give the labels for x and y axis for a for example, what is x axis and what is y axis; you just have to say using this command.

So, in this case it's a I am plotting a sinusoid with respect to time and the y coordinate is the amplitude of the sin. And if I increase the number of periods or decrease the number of periods it will be reflected.

(Refer Slide Time: 43:27)

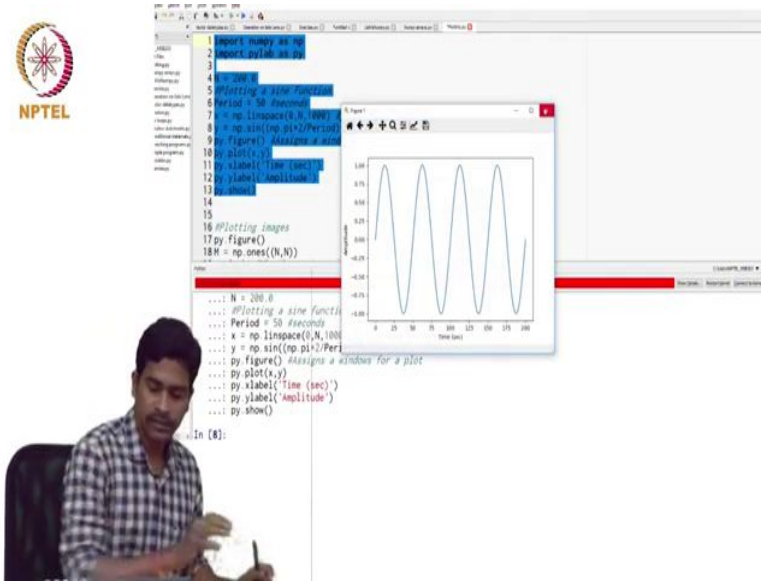


(Refer Slide Time: 43:39)



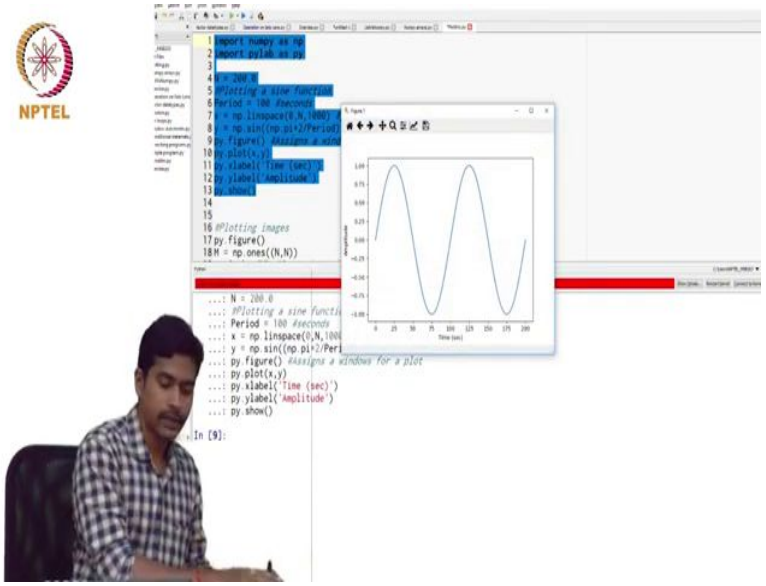
So, this is the time period in seconds ok. So, if I want to increase the time period, which means I will get a lesser number of sinusoids.

(Refer Slide Time: 43:57)



So, I can give 50 as the time period and the total duration that we are plotting the sinusoid is 200 seconds ok.

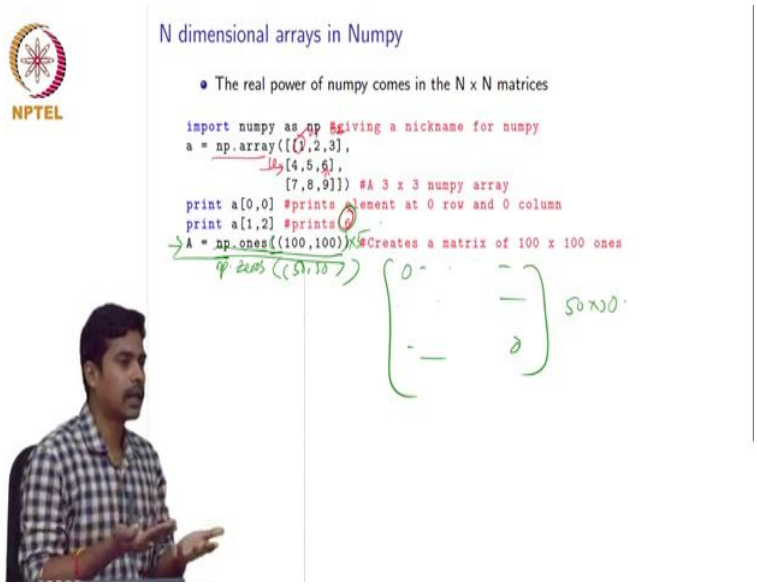
(Refer Slide Time: 44:14)



So, if the total duration is 2 seconds and if the period is 50, we are going to get 4 periods. And if I give 100 as the time period you are just going to see 2 sinusoids just like this ok. So,

this is how you plot data in python. And there are other examples that I have given here. So, here we can also plot $sin(x)$ or this quadratic equation $x^3 + yx^2$ using the same command.

(Refer Slide Time: 44:36)



And this is how these look like. And finally, the most important thing that you guys will be using in your simulations is how to simulate interference and diffraction.

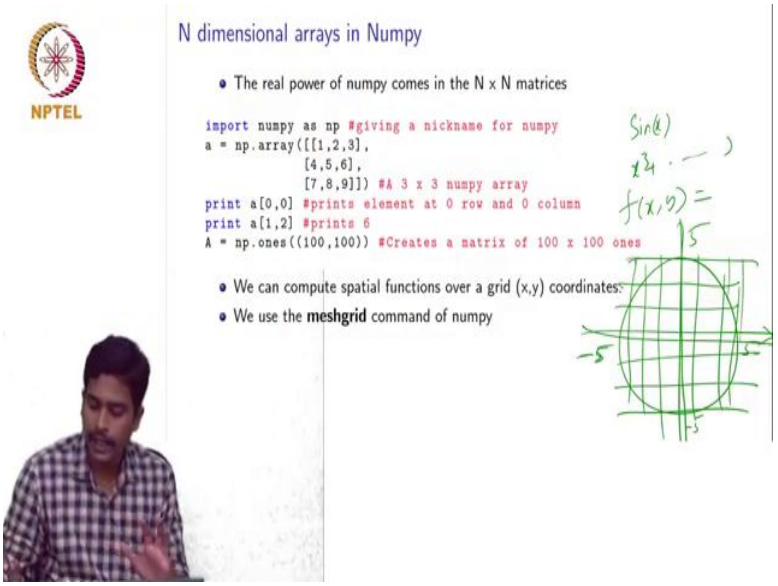
So, for that you need to learn something called a 2 dimensional numpy array. So, far we have seen the arrays to be using 1 dimensional, which means they have they do not have N x N kind of shape they are just 1 x N kind of arrays so, but these are this is the actually the real power of an numpy comes from where you have N x N values and N x N values. So, in python numpy. So, this is how you create an N x N array or here it is 3 x 3 array. So, $N\pi$ dot array and within these curly braces you actually give the array and it will convert it into a numpy array.

So, to print a particular value for example, the first value. So, index 0 , 0 the indexing starts from this point and this is 01, this is 02, this is 10 and so on. So, if you want to print the first value give the A of 00. So, if you give off 1 comma 2 it is going to be first row and second column 012. So, it is going to print the value of 6; it will print the value of 6. And if you want to create an array of 100 by 100 zeros or 100 by 100 ones. So, this is the command used.

So, numpy dot ones of you have to give the shape of the matrix I mean how many 100 or 100 by 100 or 50 x 50 whatever. Similarly there is another command called numpy dot zeros and you have to give the shape; remember there are two brackets here not just 1 50 x 50. So, this will return a matrix of 50 x 50 zeros.

So, these are some handy operations in numpy if you want to calculate a big array of the same values. So, if you want to create a big array of 5's what you can do is you can create an array of 5 also or array of ones now you can just multiply with 5. So, this will convert it into a matrix of all 5's or all 100's whatever you want.

(Refer Slide Time: 46:57)



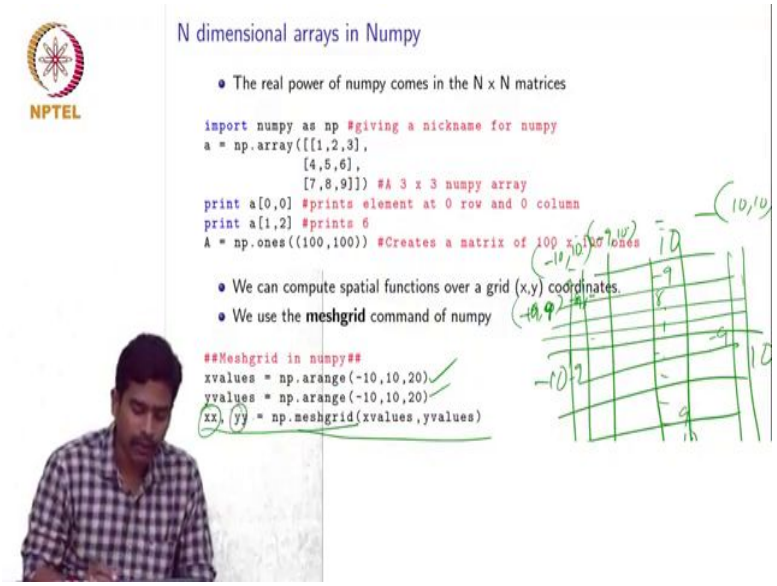
And next important thing is the meshgrid. So, meshgrid is very very useful if you want to calculate functions of 2 variables. So, far in previous example we have seen sin x or x cube some quadratic function, but if you want to calculate a function of two variables for example, if you take a lens. So, lens is a 2 dimensional element it's not just 1 dimensional; if you want the phase at each point of the lens. So, you can create a meshgrid or the coordinate system for a lens and you can just write the lens function. So, let me draw a lens for you and this is the lens and this is the coordinate system.

Let us say if you are a lens is some 10 centimeters in diameter. So, this is going to be minus 5 comma minus 5 this is - 5, y 5 this is the coordinate system of the lens. So, you can sample

this coordinate space into as many numbers of points as you want, you can create like 10 points minus 4 minus this grid or you can create an even finer grid like minus 5 -4.5 - 4 and so on.

So, the more finer you make this matrix. So, the more memory your program is going to consume. So, in this example for example, if you want to calculate a lens function. So, you have to take this coordinate system and the coordinate system ranges from - 5 to 5 in x direction and - 5 to 5 in y direction ok.

(Refer Slide Time: 48:33)

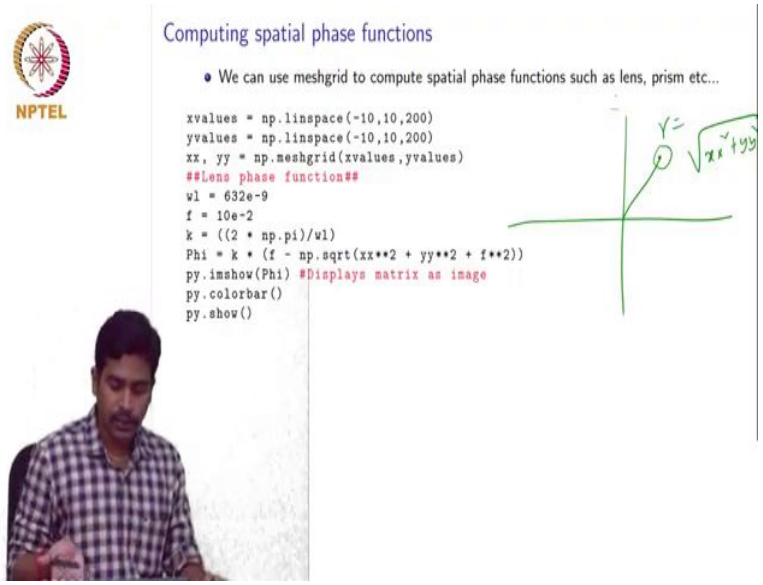


So, now if you want to calculate all the coordinate points pair of coordinates in this meshgrid. So, there is a command sorry this is wrong. So, in this particular example what I am doing is I am sampling the coordinate system like this.

So, minus 10 to 10 in x and - 10 to 10 in y and I am taking like 1 into 1 interval minus 8 and so on similarly 10. So, I have given the range of x values and I have given the range of y values. So, by using this command. So, what it does is take the range of x and range of y values and it will return 2 matrices xx and yy and the xx and yy will hold a 2 dimensional matrix of 20 by 20. So, for example, in this example. So, it is going to be minus 10. + 10. So, minus 9, 10 and so on on all the way till 10 , 10 ok.

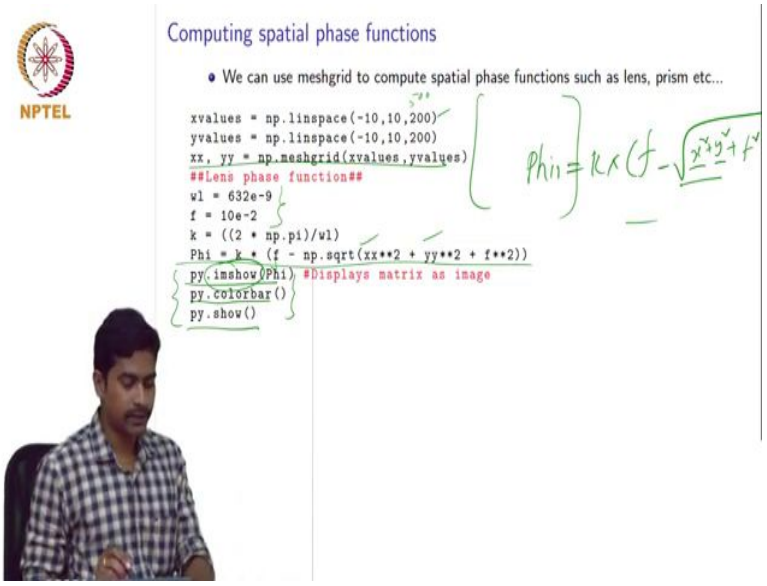
Similarly, this is - 9 ,10 sorry -9 , 9 and - 8 sorry-10 - 9 - 9, 10 and so on. So, this is all the coordinate system is going to look like. So, what this meshgrid command will do is it will take all the x coordinates from these points and then store it in xx and it will take all the y coordinates from these set of points and store it in yy.

(Refer Slide Time: 50:05)



For example, if you want to calculate the lens function, you can just simply write. So, you can convert the xx intro and yy into radial coordinate for example, if there is a point here. So, the radial distance can be given by the square root of x square plus yy square right. And if you say r equals to this. So, it is going to convert all these xx and yy into radial points ok. So, that is what we have done here or if this is too complicated just forget it ok.

(Refer Slide Time: 50:43)



So, for the case of a 2 dimensional lens the equation is the same, but we are also adding the x coordinate is given by $kf - \sqrt{x^2 + y^2 + f^2}$ So, in the earlier case we have seen the phase variation only in 1 direction which is y, but in this example we want to see the phase variation overall in 2D. So, for that we have to add both $\sqrt{x^2 + y^2}$.

So, normally if you are doing the same thing in for loop what you would need to do is you have to write 2 for loops 1 for loop for x coordinate and inside that you have to write another for loop for the y variation. And if you have 2 for loops and each running n times the complexity is going to be n square it's going to take n square number of iterations to calculate. But in numpy you do not have to write any for loop whatever equation you are seeing here you just translate it to normal python statement ok. So, but this xx and yy you have to calculate using the meshgrid you understand.

So, this is how you calculate a 2D lens profile in using mesh meshgrid.

Student: What is the 2 (Refer Time: 51:53).

Sorry.

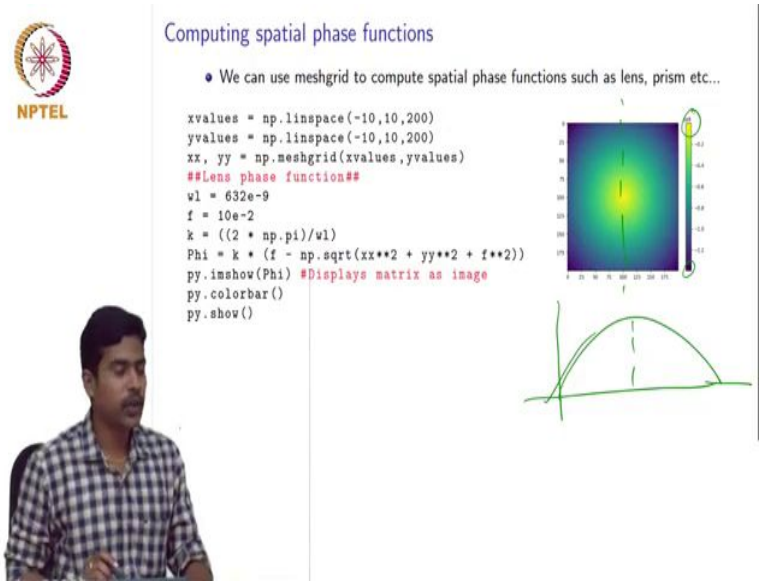
Student: (Refer Time: 51:54) first minus 10 (Refer Time: 51:55) 200.

Here we have taken 200 points between minus 10 comma 10 like minus 10 minus 9.9 and so on. So, we can increase it to 500. So, if we increase it to 500 what is going to happen is you will have 500 by 500 matrix to represent your lens. So, if you want to represent your phase of a lens very accurately, you take more number points like 1000 by 1002. But if you take too many points; obviously, if you want to do some competition like fft or dft. So, it will take a lot of time.

So, there is a trade off between how accurately you want to represent a particular optical element and how much time you want to spend in doing your competition. So, ideally 500 by 500 1000 by 1000 is not a big deal for python. So, it will be very comfortable handling that number of elements, but if you go to 5000 by 5000 or 10000 by 10000. So, it will take a lot of time and depends on your computer's ram and processor and all those things.

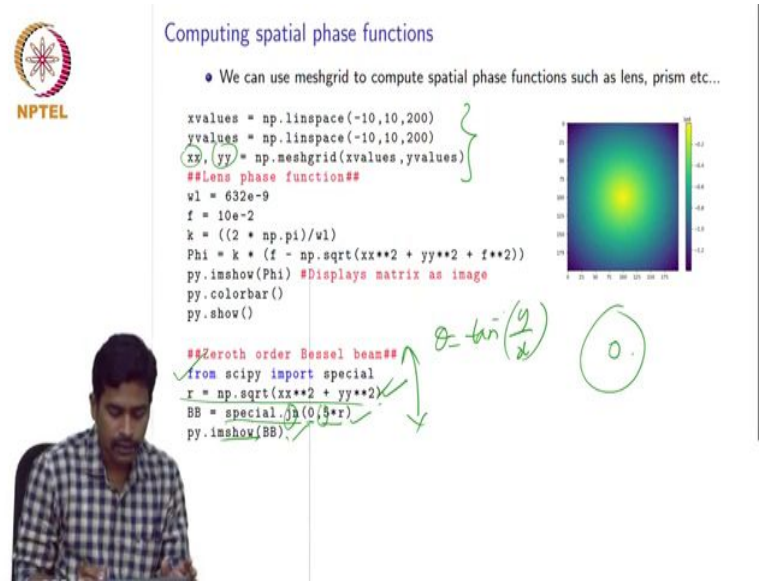
So, now in this example I have calculated the lens function in 2D. So, if you want to represent 2D data you cannot do it using plot. So, you have to represent it and you have to show it as an image right. So, each pixel will have some color and you have a color bar. So, to do this in python we use this command python py dot imshow and for this imshow you have to give the 2D array. So, in this case the phase profile and you can give a color bar to show what color bar is doing. And finally, so, this command you should not forget otherwise it will not display the function.

(Refer Slide Time: 53:31)



So, after I run this. So, I will get something like this. So, this is the phase function of a lens. So, in the earlier case what we have done is, we have calculated the phase only along the y direction and remember and the phase looks like some function like this right, I think it's over and the phase function looks like this. But in this case we are calculating the function in both x and y and this is the color map the phase is maximum at the center from this color and it is reducing in this manner when you go to the extremities, which is given by this color ok.

(Refer Slide Time: 54:07)

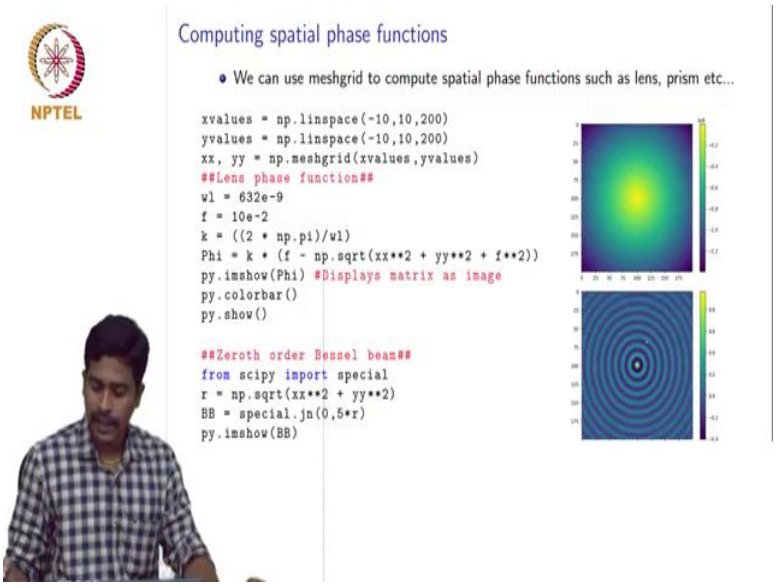


And similarly you can compute different functions. So, to show you I have plotted a Bessel function. So, Bessel beams are very useful which are not covered in this course. If you are interested you can read about them. So, for this I am using another package called scipy that has all scientific functions in it. So, from scipy, I am taking this Bessel function which is given by j n and for the Bessel function not its radial function. So, in order to calculate radial functions I have converted the x and y coordinate into a radial coordinate system or r theta coordinate system.

So, you all know how to convert from Cartesian to r θ r is given by $\sqrt{x^2+y^2}\theta$ is given by $tan^{-1}(y/x)$. So, simply you can get the meshgrid from this equation. And based on these two you can convert it into $r\theta$ just by giving r equals to this and θ equals to tan $tan^{-1}(y/x)$. But in this case I do not need the θ I just need the radial coordinate, which I have obtained using this equation and this is the command to plot the Bessel function.

So, 0th 0 is the 0th order Bessel function you can give 1 or 2 whichever order of the Bessel you want, but in this case I am just plotting 0th order and the radial coordinate into some scaling factor 5 here; you can give order of scaling factor you want. Finally, I am plotting it using the imshow function I have to write this otherwise it will not display.

(Refer Slide Time: 55:42)



So, this is how a Bessel function is going to look like, it has a central maximum and followed by so many concentric rings around the central peak. So, this is all for today's lecture. So, we have seen how to do this sequential operations using normal for loops. And we have seen how to do the same thing using numpy using numpy. It is just like writing mathematical equations in the form of a python statement. And it is we have also seen its very fast using a comparison between lists and numpy arrays.

And finally, we have seen how to plot 1 dimensional data, using plot command. And finally, we have also seen how to plot a meshgrid or a 2 dimensional data using imshow command. So, with this the introduction to python is over. And in the next lecture I will show you how to model interference and diffraction phenomena using whatever we have learned so, in python ok. So, that is it for the lecture.

Thank you.