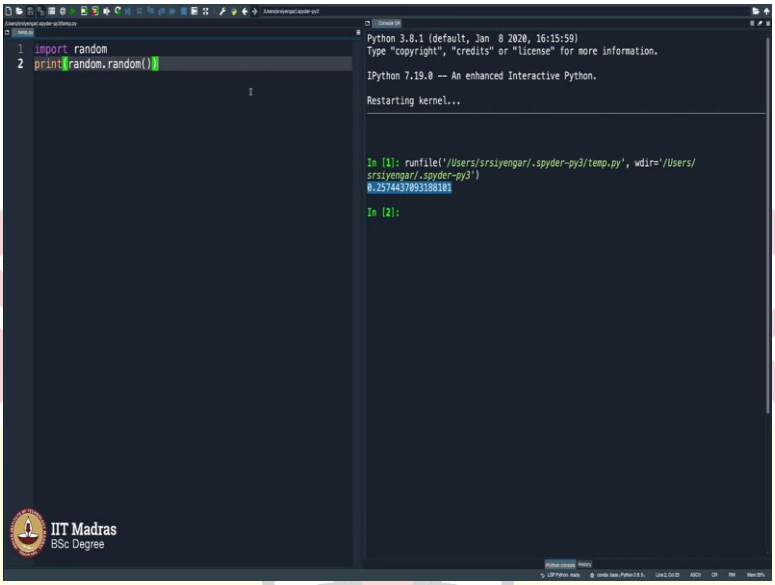
IIT Madras

ONLINE DEGREE

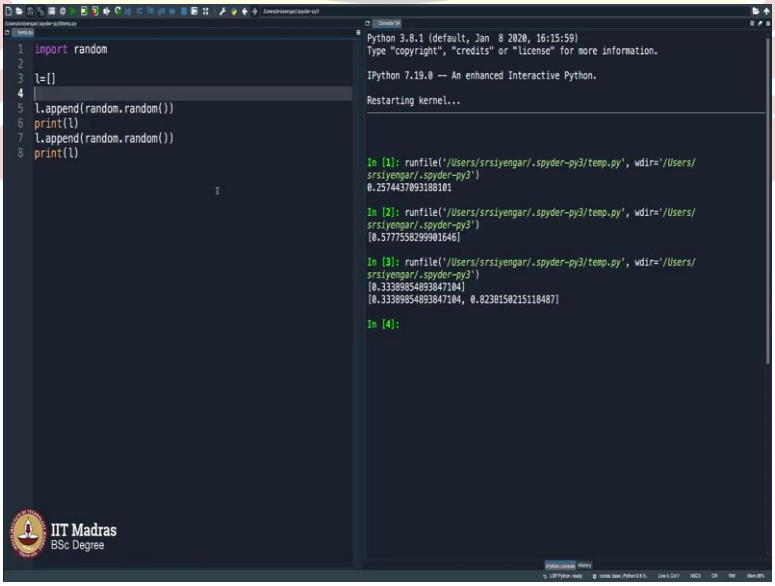
Programming in Python
Professor Sudarshan Iyengar
Department of Computer Science and Engineering
Indian Institute of Technology, Ropar
Birthday Paradox

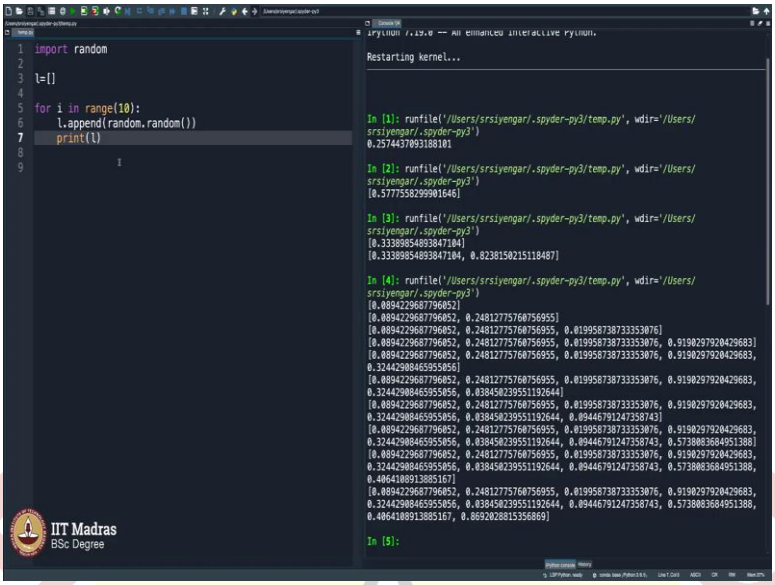
(Refer Slide Time: 0:16)



I am assuming all remember the import random that we taught you a while ago. Remember the import math as well that we taught you. The import random is going to be a very useful tool, I am going to use that now. And then, if I say print random random, you know what that does. Execute this, it shows me a random number between 0 and 1 as you can see in the terminal.

(Refer Slide Time: 0:50)

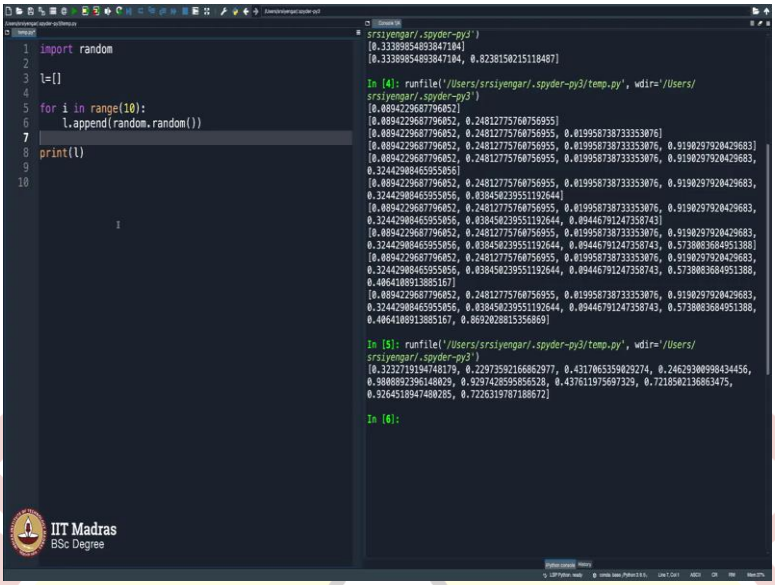


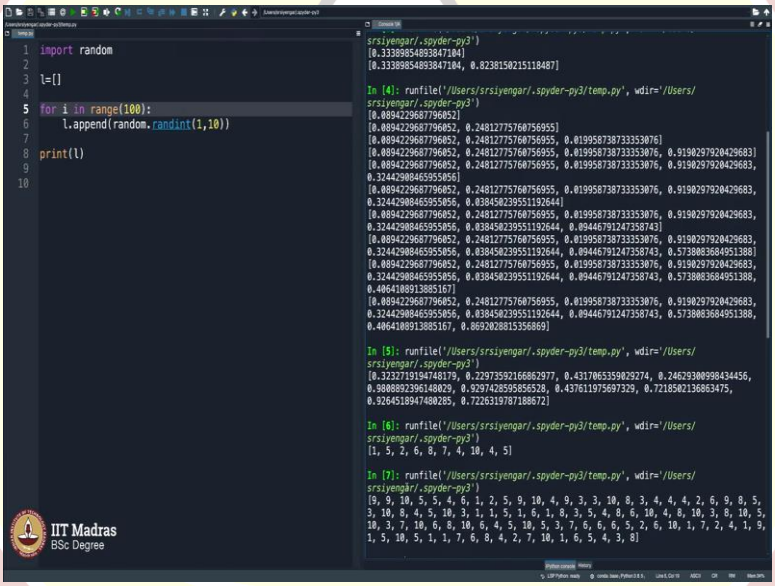


So, what I will now do is instead of printing a random number, I delete this, I will initiate a list l and I will append a random number. And then now if I print l, what will it do? It will create a list and it appends a random number between 0 and 1 and then assuming I keep doing this, then again print l, it will keep, it displays l with one number included and then with the same l I append another random number and here is another random number and that again gets displayed here.

So, spend a minute, pause the video and try to see what am I typing here and what is the output here. So, what I will now do is I will not type this twice, I will type this maybe 10 times. So, how do I do that? For i in range 10, yes perfect. I say l append random random and then print l, this will give me, you see what is happening, I am printing the list every time, so it is displaying every time a random number gets appended.

(Refer Slide Time: 2:16)



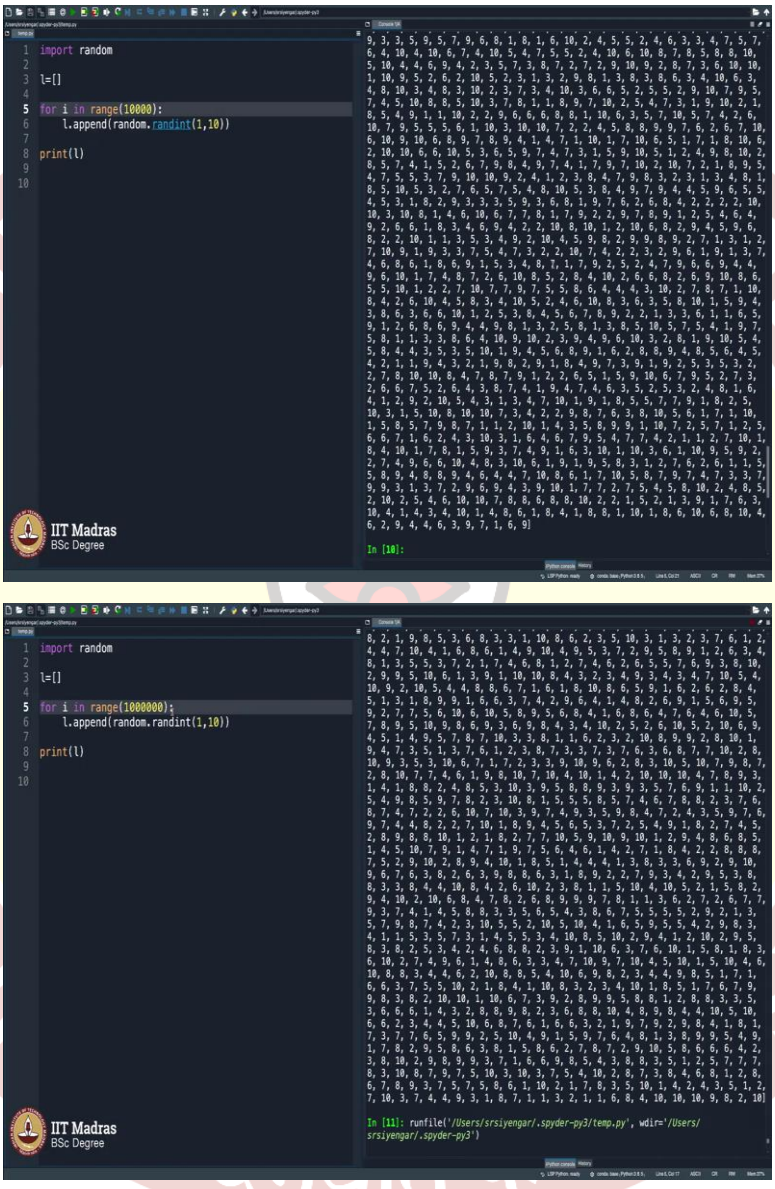


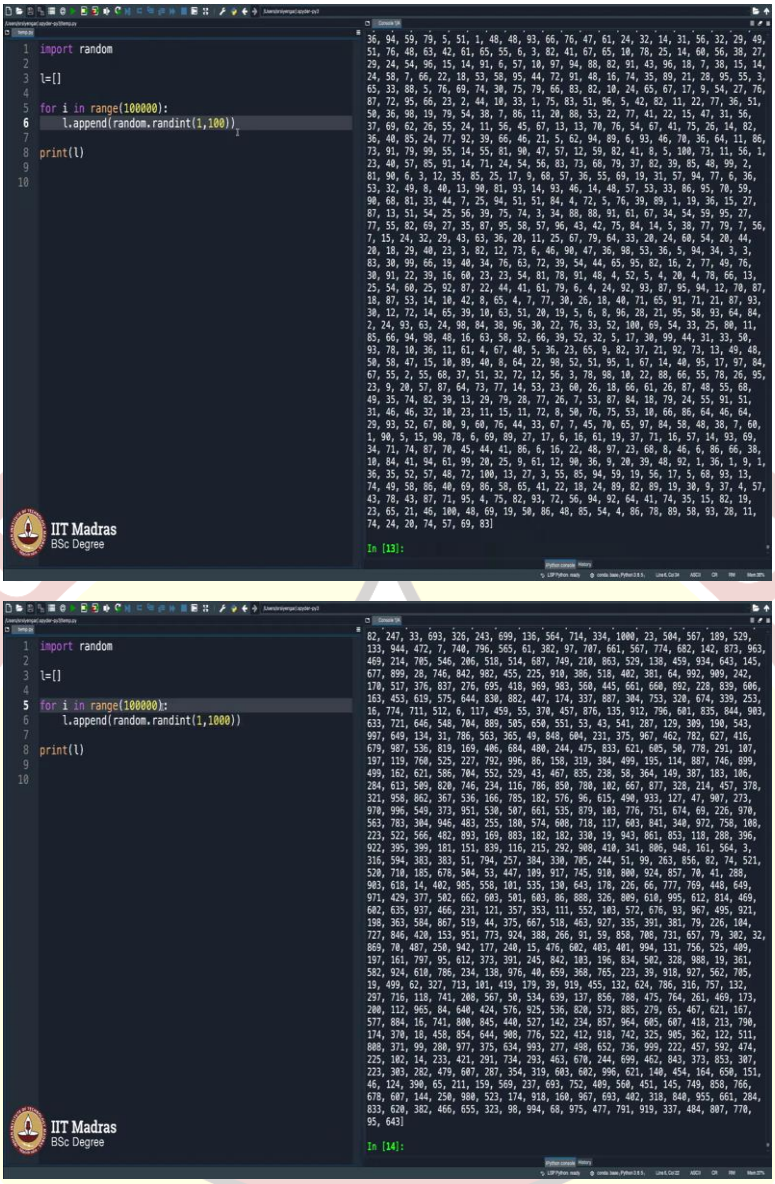
So, if I remove the tab, it means this comes outside the for loop and as you would expect, what should be the output here, let us execute it. It will show the list only once because here it is displaying it only once, you have removed that from the for loop. To avoid confusion maybe it will also help if I put another enter here; empty line here. So, I do have these 10 random numbers here.

As you know we also have what is called the random randint and I can say 1 comma 10, let us see what this does. It simply appends a random number between 1 to 10 here into this list. So, the fun part about computers is that you can simply increase these numbers and boom, it becomes 10 became 100. I am appending 100 numbers here.

So, please note, it started off with 1, but it starts with 9 here that is because it is a random number that you are appending and you executing the program afresh, that is why it does not append to this list, but starts a new list every time we execute this program. So, what I will do is I will keep this in the middle of the screen so that you can see it.

(Refer Slide Time: 3:36)

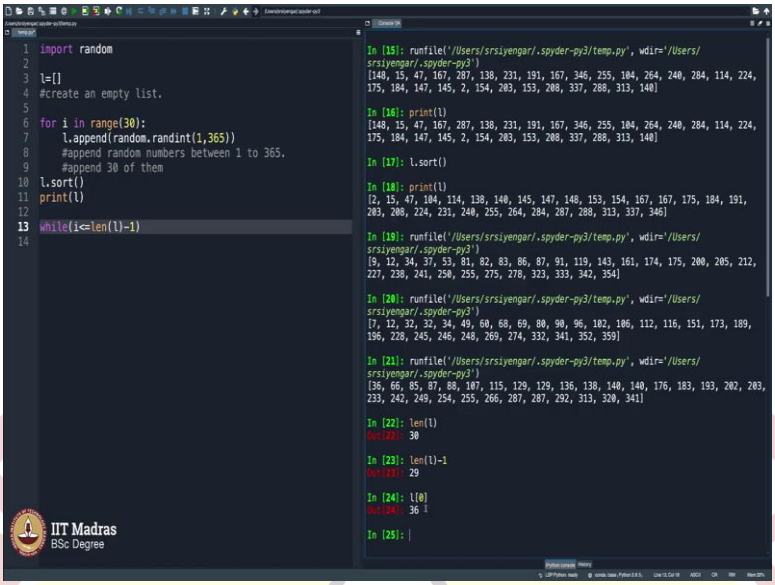


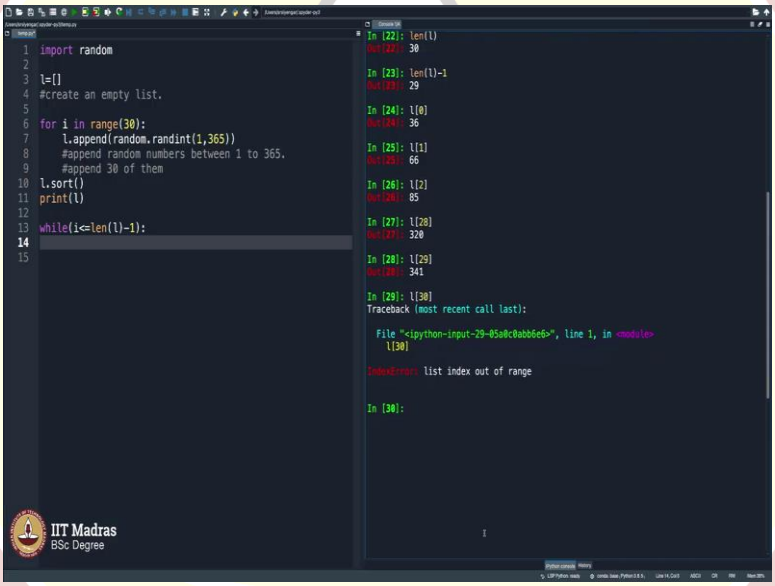


Better even clear the screen. As you can see, now instead of 100 if I do let us say 10000, it will indeed show 10000 entries. Boom, look at that it was really really fast. You can add many numbers and you can observe that it is a lot faster than what you would expect it to be. So, I am now appending 1 million numbers, now I takes time but it does get back as you can see; so many numbers and you can let us say let me just reduce this by, I mean I will keep it 100000 and then which is like 1 lakh and then include random numbers from 1 to 100 or may 1 to 1000.

I am executing, you see numbers from 1 to 1000 randomly put here. Why would I do something like this? So, we are heading towards a topic that you have studied in computational thinking; the birthday paradox. Before that, we would like to see how we can model it using Python. So, what exactly happened in birthday paradox?

(Refer Slide Time: 4:52)





Let me try changing these values and you will probably quickly say Eureka! You found that out in the sense that you understand what I am trying to say. What is this? Let me clear the screen this side. What is this? I am saying create a list l, so as I have been telling, it is important to type comments; create an empty list or rather initialise an empty list. What I do is append random numbers between 1 to 365, how many should you append? Append 30 of them.

Let us see this is what is happening, execute it and you see this is preciously what is happening, you have 30 random numbers between 1 to 365. As you would have guessed, 1 to 365 stands for day in a particular year. So, if I am born on January 1st, the number is 1. If I am born on December 31st, the number is 365. So, when you pick 30 people and ask them for

their birthday, their ddmm day and month, not the yy if you remember, you do not look at the year of birth in the birthday paradox.

If you are wondering what we are talking, you may have to revise your computational thinking course this part where the instructor discusses the birthday paradox. In fact, it is a very interesting discussion as I have watched and there is a surprise factor there that even if you take a very few number of people, you will get two people with the same birthday. So, I have taken 30 people here and I am going to execute it, I have got these numbers, so you are going to tell me if there is any repetition here.

Right, I see, I do not see a repetition or maybe I am bad at seeing if there is a repetition or not. There is a nice way to do this. This was your l, so if you are wondering what I am doing, whatever you have written here, those variables can be called here too. l had these numbers, I have included l, there is something called sort function built within the list. So, when you say l dot sort, it simply sorts it.

Again print l, now it is easy to see if there is repetition or not. Let me check, let me check, 154, 167, there is a repetition and what does it say? It gives a proof that birthday paradox indeed seems to be true. Why? As you can see, when you pick 30 random numbers from 1 to 365, the stat amounts to, this translates to you going on the road and stopping some 30 people and asking them for their birthday. Of course, you should not do that. Nobody will tell you their birthday, especially on the road.

But, if you randomly pick some 30 people, let us say from your friends or in a classroom and ask them for their birthday, you will observe that two people will have the same birthday. So, you must be wondering, is this always true? Not really. This may not really be true. I will show you an example, just a minute.

So, let me say, let me put s, l sort here itself so that it displays. Now a fresh bunch of 30 people, I see is there a repetition, 86, 87, 91, 119, 143, there is no repetition here. Surprisingly there is no repetition here which means this is not always true if you take 30 people; here and there it may fail. I am repeating it once again. 32, 32; bingo! There is a repetition here. I will repeat once again and let us see there is no repetition, 129, 129 is repetition here.

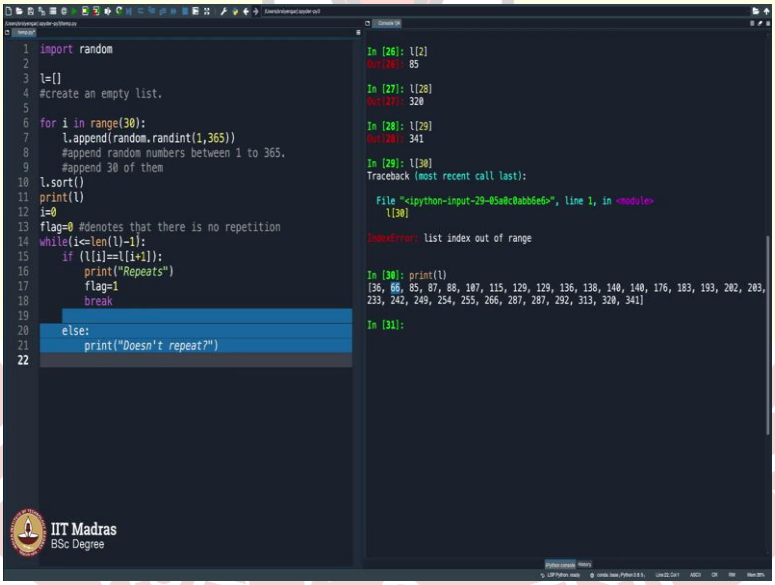
Only here there was no repetition. So, or did I skip seeing that, it is very difficult, so 143, etcetera, etcetera, there are no repetitions here. So, why should we do this manually? Why

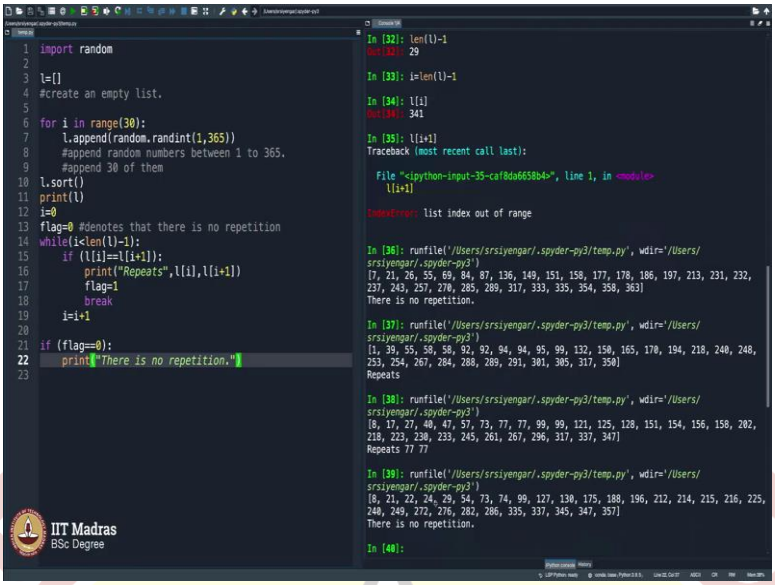
cannot we write a piece of code that checks if there is a repetition or not? How do I go about it? So, it is pretty simple. What I will do is I will say while i is less than or equal to length of l minus 1, what does this stand for? Always length of l will tell you the length.

So, I mean, the how many entries are there in l? I should take minus 1 because I need to go up to the 29th element you see, length of l minus 1 will be 29. Please observe carefully. When I say l of 0, it gives me the first element of l; 36. When I say l of 1, it gives me 66; l of 2 will give me 85 and so on what is l of 28? Can you guess l of 28 is what? 313 or 320? Think about it, it is 320.

So, what is l of 29? The last number 341. If this one is zeroth element, this should be twenty ninth element because they sums up, they totals up to 30. So, what is l of 30? Boom, it will punch on your face saying if is an error. So, I will go up to len of l minus 1. So, what is the syntax? I say colon here, then go here, I do not like this error messages, so I will scroll up a little.

(Refer Slide Time: 10:58)





So, I say while i is less than or equal to len of l minus 1, what you do is just in case if l of i is equal to l of i plus 1, then I say print ‘there is a repetition’ ‘repeats’, else ‘does it repeat?’. It matters for me to make some errors in my code, then you will realize what goes wrong here. So, I should not be doing this you see, you see what will happen. If this happens if l of, it starts with 0 basically i should be equal to 0, i is 0, 0 is less than or equal to len of l minus 1 which is true, it comes inside.

Is l of 0 is the same as l of 1? So, let me go ahead and then see that array here. So, it will just, let me, let us use this as the array, l is this here so let me print l so that it is clear to you people. So, l of 0 is 36 is 36 equal to 66? No, it is not equal. So, it does not get repeated and it says does not repeat. Now that is not true. I want to go until the end and then say it does not repeat. So, I will do a small trick, it may take some time for you to understand what this trick is.

The trick is I will say flag equals 0 which means denotes that there is no repetition. If flag equals 1 it means there is repetition. What does it mean? It will be clear in just a minute. So, whenever there is a repetition, I say flag equals 1 and then once you see the repetition, you may want to come out of the while loop, why should you stay there? Correct? Perfect. And this not required. Why?

Why because translating this to simple English at the end in case you never entre the if loop, it means you have not seen two elements which are equal, two consecutive elements that are equal. Which means, flag never becomes 1 which means your flag is 0. Then I say if flag is equal to 0, then I print doesn’t repeat there is no repetition. Now, there is a small problem here.

i goes up to len of l minus 1, your i should basically be incremented every time i equals i plus 1. Why is that? It is because the while loop ensures that i is less than or equal to len of l minus 1 and you should keep going to the next i, i equals 0, i equals 1, i equals 2; that happens when i equals i plus 1 is stated here. Stare at the code, pause for a minute and stare at the code. It keeps incrementing and the last time i, the moment i becomes equal to len of l of minus 1 which is this case as you can see len of l is 30 equal, len of l minus 1 is 29 and l of len of, let me recall this i.

i equals len of l minus 1 and then I say what is l of i? l of i is 341 in the last step. And then when I say l of i plus 1, it throws an error. Because l of i plus 1 becomes l of 30, l of 30 is not here at all. So, you should not be comparing l of 29 with l of 30 which means you should not go till the last step. You must go to the second last step. So, you should stop when i equals 28 which means you should not go up to the length, you should go up to the length of l minus 2.

This is getting complicated and I am sure you are confused and that is what you mean by programming complexity. When you are thinking, I am thinking aloud and I am coding, I have not typed this code before, I am going very closely just trying to understand what is what, translating my common sense to English and that English to a program.

So, now here is my code, I have a feeling this is perfectly written, this should get executed. Let me see what goes wrong or if it goes right. So, I will execute this. It says there is no repetition. I hope there is no repetition. Let us check. I think there is no repetition. Good. 186, 197, 213, 231, 232, 237, 243, 257, 270, boom there is no repetition. But if I execute the same thing again, it just repeats. Where does it repeat?

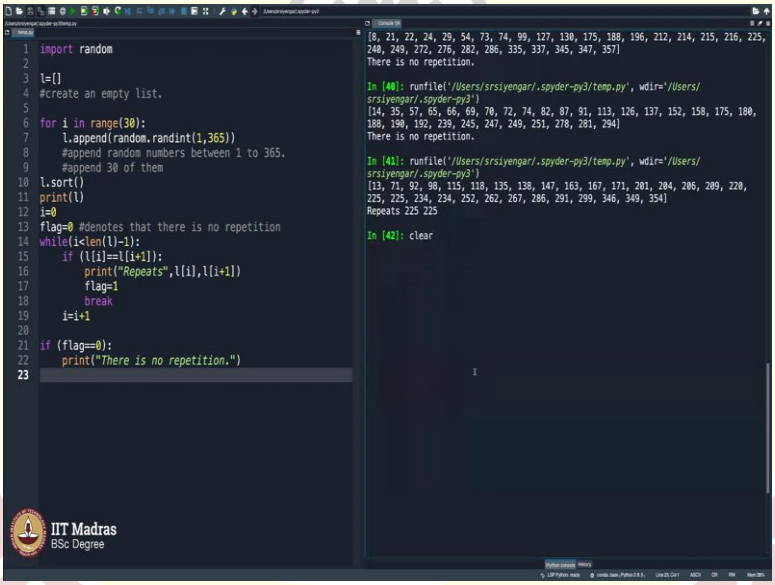
It repeats somewhere here, let us see where is it, 58 there is a repetition. So, it says repeats. So, I can trust my code now. Why should I go and see where it repeats? Why cannot I display where it repeats? When I say print, it repeats. I can also say it repeats in the particular place, let me just write it next to repeats, repeats l of i and l of i plus 1 basically, they both are the same, it should display the numbers twice.

Repeats 77, 77, that is awesome. But then you see 99 and 99 is also repeating, it is not the showing that. That is because I keep saying your computer is dump, it only does what you specify, what you ask it to do. The moment it repeats, you are coming out, you are saying break, stop; I have seen a repetition. When you are checking for 30 people, the moment you see two people with the same birthday somewhere, you stop.

By the way they do not come in this ascending order, we are sorting the list that is why it has been displayed in ascending order. So, you, the moment you see the repetition, you stop. You may not want to do that. Maybe you can modify this code a little, I will leave that as a homework. It is pretty straight forward, will take another 20 minutes of our time, I am not ready to spend that right now, but I want this as an exercise that you can take home and then try to figure out how to do it.

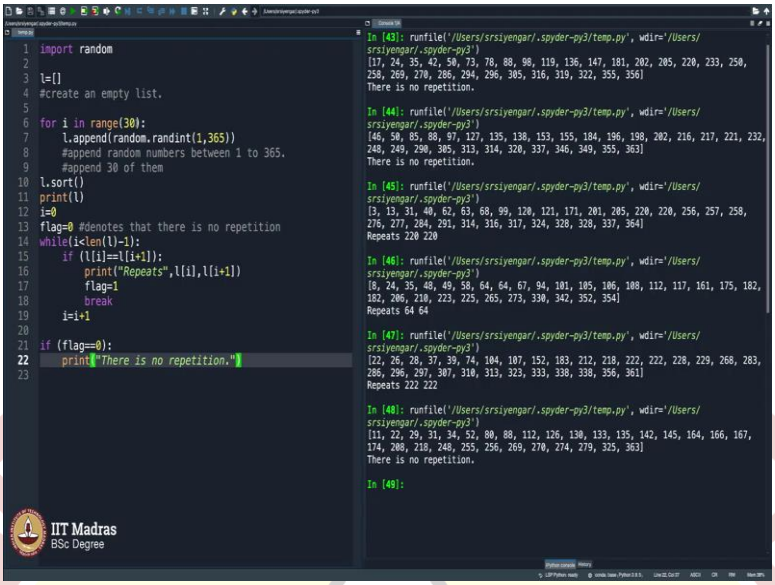
So, what you should do is you should show the repetition and you should also show all the repetitions which means once it repeats, you should not break, you should come once again and then check whether it repeats or not. Sometimes it may repeats thrice or four times, 5 times, you should be able to see that. There is no repetition, repeats, repeats. Looks like every now and then, it does not repeat, there is no repetition once again.

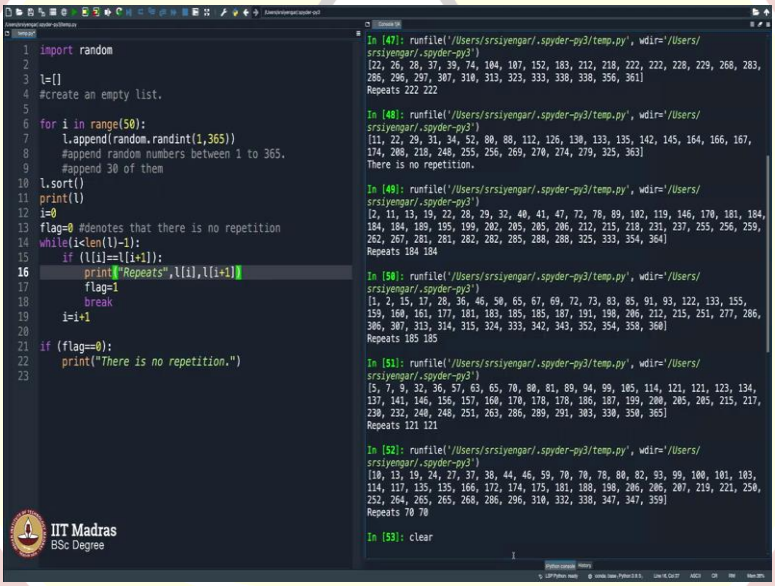
(Refer Slide Time: 18:42)



Let us try executing this again and again, no repetition once again; repeats 225, 225 as you can see. So, let me clear the screen so that it is easier on your eyes.

(Refer Slide Time: 18:53)





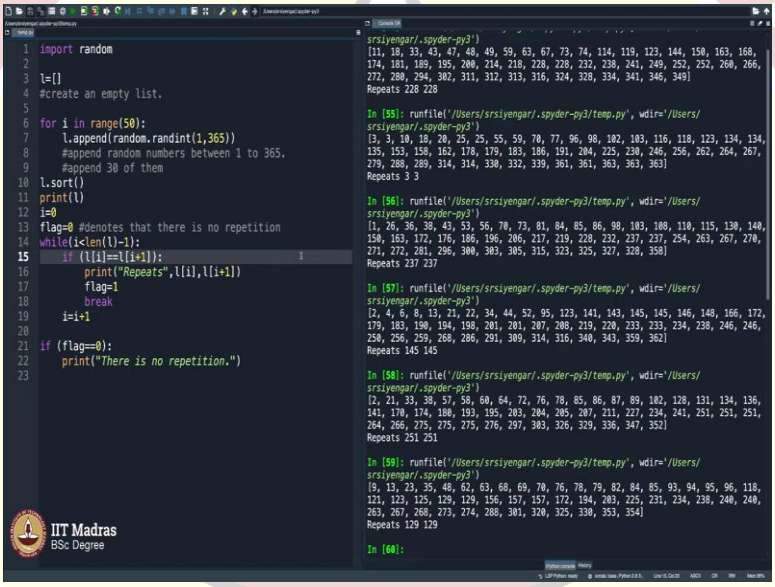
There is no repetition. Once more, no repetition. Repeats 220, 220. Repeats 64, 64. 222, 222. There is no repetition. So, what do we do here? Why is there no repetition occasionally? Not occasionally, almost one in two times when you run this experiment, how many times have we run here on this page? No repetition, no repetition, repeats, repeats, repeats, no repetition. 1, 2, 3 times no repetition. 1, 2, 3 times there is repetition. It means, repetition happens only half the times.

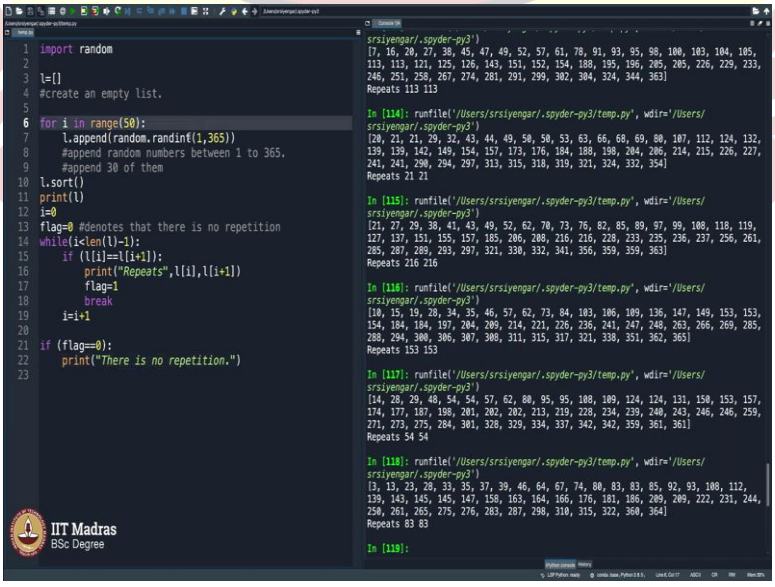
So, shall we go to our computational thinking professors and tell them this repetition is not happening, but you taught us that repetition is very probable, I mean, it, with the high probability it happens, but it is not happening here, it is not happening with a very high probability, it is happening with let us say, probability half, half the times there is repetition.

And your computational thinking professor will say, that is expected. Why is it expected? That is expected because you are taking only 30 people. Try taking 50 people and you will see the magic. I have taken just 50 people, this entire program I just go and simply make this 30 50 that is all. And I run the experiment, I repeat, not the phrase: I run the experiment. I repeat, I run the experiment. You probably are thinking I am mad, repeating the same phrase again and again. There is a reason for it.

I run the experiment. I will tell you in a while why I am saying it again and again. I run it once again, there is a repetition; run it once again, there is a repetition, run it once again, there is a repetition. Let me clear the screen. One minute, clear the screen.

(Refer Slide Time: 20:45)

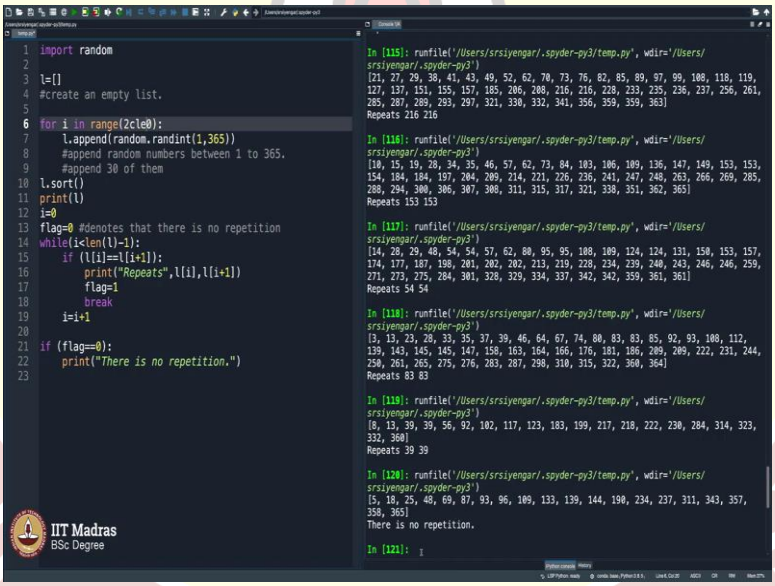




And then repeat the experiment, repeat the experiment twice, thrice, 4 times, 5 times 6 times, wonderful. I repeated it 6 times and at all times, it says repeats, repeats, repeats, repeats, and repeats. We did not encounter a situation where it does not repeat, there is no repetition. Seventh one, it says there is no repetition. But then again repeats, repeats, repeats, repeats, repeats, repeats, repeats, repeats, you see once in a while it says no repetition. I made to encounter no repetition right now.

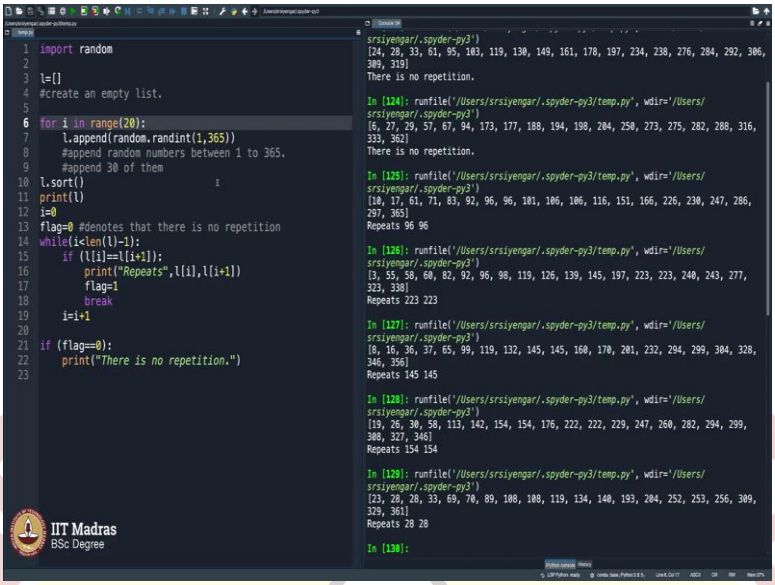
Let me keep going, I have all day repeats, repeats, repeats, repeats, repeats, repeats, repeats, I am getting slightly tired now, repeats, repeats, repeats, repeats. Come on please show us one no repetition so that I can tell them that it is not always green, it can be red too. We indeed saw once but looks like this will go on and on. In fact, you can in fact, put in a loop here and then try to figure out how often does it repeat and how often it does not? So, we had 30 here, we made it 50.

(Refer Slide Time: 22:15)



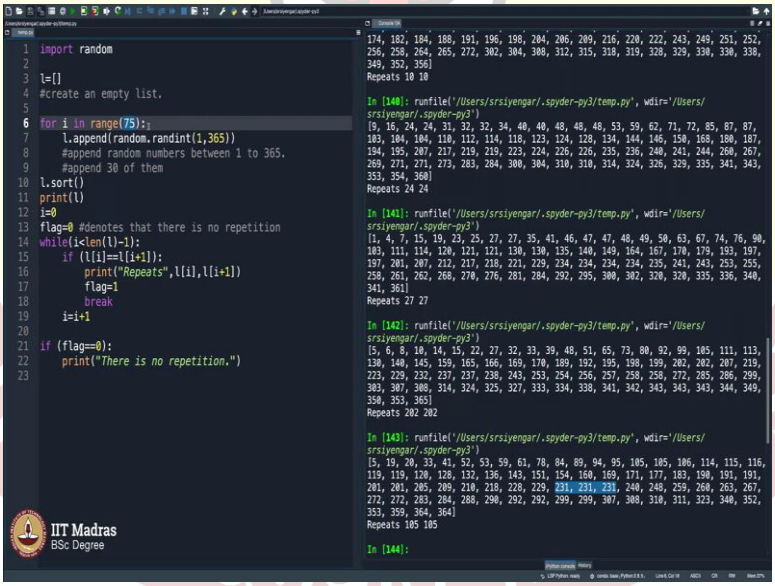
Now I will make it 20, you will see many a times that it will not repeat. You see repeats, no repetition. I will clear the screen.

(Refer Slide Time: 22:28)



No repetition, no repetition, no repetition, no, repeats, repeats, repeats, repeats. What is happening? Repeats, there is no repetition, repeats, there is no repetition, repeats, there is no repetition. So, they, it happens more frequently than for 30. But for 40, 50, it never repeated.

(Refer Slide Time: 23:02)



But if you add, if you some 75 or something you may have to wait for a long time before you will see no repetition. Definitely, there will be repetition. In fact, there will be many such pairs. You will observe that there will be many such pairs as you can see, I am just quickly seeing, there is a 364, 364 here. The repetition 105, 105 is there. It says 105, 105, so 299, 299, I see 231, 231, 231. There is a triplet here, not just a twin.

You see repetitions not just two, but 3, 4 starts off as you increase the number. Let us run the experiment. I said the word experiment, what I mean by that? You need not go to the roads or go to a classroom and then run this experiment. You see, you noted that in the computational thinking course, the professors were using the cards to show you that the repetition happens. Is that really true?

You can come here and run an experiment on your computer, repeat the experiment many times and then say that this is indeed true. When you repeat the experiment, this paradox called the birthday paradox, paradox is something that is tough to believe but is true, indeed is visible here in this experiment where you pick numbers from 1 to 365, uniformly it random a few times, put them into an array, sort it so that it is easy for you to see repetition, you will say that there are repetitions.

I hope you enjoyed the piece of code here, what we did just now is called simulation in the language of science and it is called the scientific simulation. Many many researchers do this and they write research papers based on this. You can in fact simulate how Covid disease spreads just on your Python compiler. You can simulate how the stock market works just on your Python editor.

So, this was a good introduction to what you mean by a simulator, but then what interest us the most right now is confidence to write a piece of code. We just now wrote a nice piece of code which told us indeed birthday paradox is true. So, from now onwards, whenever someone says something, if you think that is numerically possible to simulate, you should quickly switch on your computer, write a piece of code and then analyse the output of the code.