

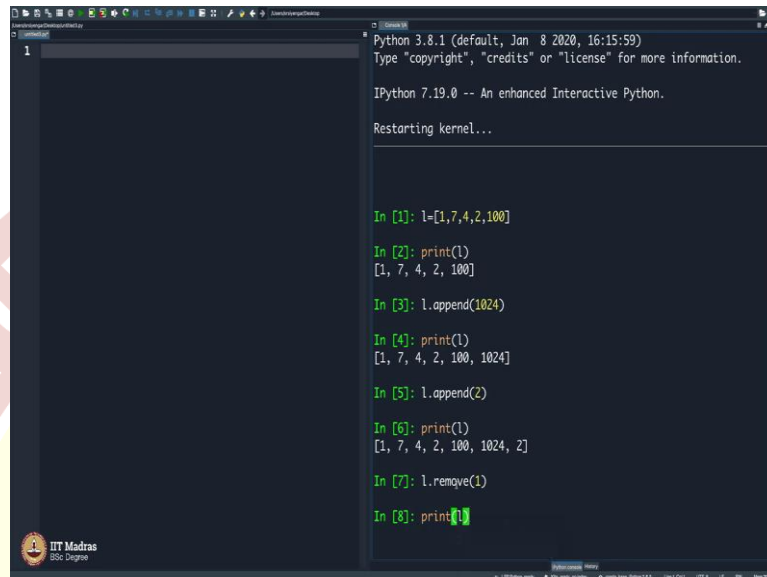


# IIT Madras

ONLINE DEGREE

**Programming in Python**  
**Professor Sudarshan Iyengar**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Ropar**  
**Warmup with Lists**

(Refer Slide Time: 0:16)



The screenshot shows a Jupyter Notebook with two main panels. The left panel is a code editor with a single line of code: `l = [1, 7, 4, 2, 100]`. The right panel is the IPython shell, which displays the output of the code execution. It shows the list `[1, 7, 4, 2, 100]` and the result of `l.append(1024)`, which is `[1, 7, 4, 2, 100, 1024]`. The shell also shows the result of `l.append(2)`, which is `[1, 7, 4, 2, 100, 1024, 2]`. The shell is titled 'Python 3.8.1 (default, Jan 8 2020, 16:15:59)' and 'IPython 7.19.0 -- An enhanced Interactive Python.' The bottom of the notebook shows the IIT Madras logo and the text 'IIT Madras BSc Degree'.

```
Python 3.8.1 (default, Jan 8 2020, 16:15:59)
Type "copyright", "credits" or "license" for more information.

IPython 7.19.0 -- An enhanced Interactive Python.
Restarting kernel...

In [1]: l=[1,7,4,2,100]
In [2]: print(l)
[1, 7, 4, 2, 100]
In [3]: l.append(1024)
In [4]: print(l)
[1, 7, 4, 2, 100, 1024]
In [5]: l.append(2)
In [6]: print(l)
[1, 7, 4, 2, 100, 1024, 2]
In [7]: l.remove(1)
In [8]: print(l)
```

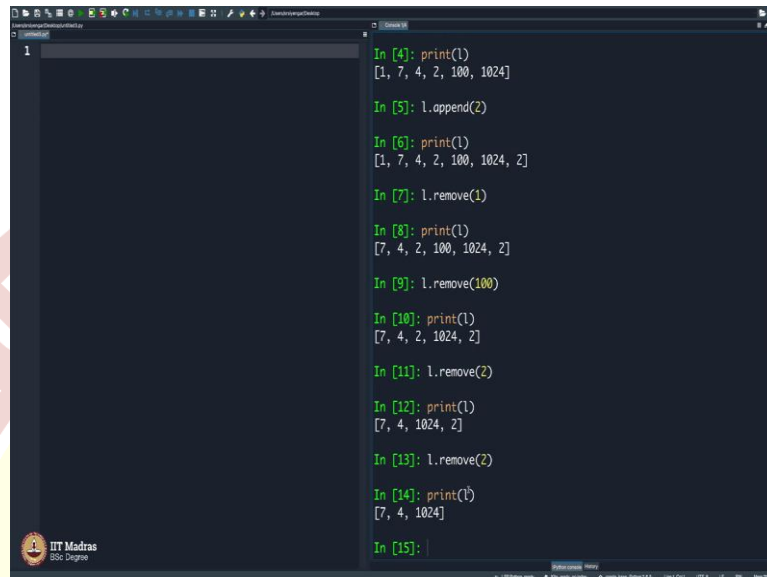
So, if you remember we did talk about lists, `l` is equal to let us say 1 comma 7 comma 4 comma 2 comma 100 and close the big bracket and this becomes a list. When I print this, it shows the values in this list. So, from now onwards, we will be using the IPython shell. This is called the interactive Python shell on the right as you know and as I expect, I believe you are familiar with this interactive Python shell on the right. On the left will be the code that we will be writing.

Many a times, we may want to execute a few commands by typing them here. So, just now I said `l` equals so and so, it created a list. Then I said `print(l)`, it did print the entire list. I can, if I want to add some value to it, by add I mean, include a value to this, I can say `l.append(1024)` and what will happen, let us see. You get 1024 here. A few more points to note, when I say `l.append` once again 2, it does include 2 twice.

You see, we all have learned in our schools that if you take a set elements are not repeated, but this is not a set, you may want to note this. In a set you put a flower bracket you see in mathematics, you would write a flower bracket in your notebook and you are not supposed to repeat the elements there, that is the convention, it is understood. But here, you can repeat as much as you want.

There is a small thing that I want you all to note which is, there are some inbuilt functions in `list`. Just the way we said `l.append`, we can also say `l.remove` and it is only easy to guess what `l` will be now.

(Refer Slide Time: 2:23)

A screenshot of a Jupyter Notebook interface. The left pane shows a file explorer with a single file named '1'. The right pane shows a series of code cells. The first cell prints the list `l`, which is `[1, 7, 4, 2, 100, 1024]`. The second cell appends the value 2 to the list. The third cell prints the list, now `[1, 7, 4, 2, 100, 1024, 2]`. The fourth cell removes the first occurrence of 1. The fifth cell prints the list, now `[7, 4, 2, 100, 1024, 2]`. The sixth cell removes the value 100. The seventh cell prints the list, now `[7, 4, 2, 1024, 2]`. The eighth cell removes the first occurrence of 2. The ninth cell prints the list, now `[7, 4, 1024, 2]`. The tenth cell removes the second occurrence of 2. The eleventh cell prints the list, now `[7, 4, 1024]`. The twelfth cell prints the list, now `[7, 4, 1024]`. The thirteenth cell prints the list, now `[7, 4, 1024]`. The Jupyter Notebook logo is visible in the bottom left corner of the interface.

```
In [4]: print(l)
[1, 7, 4, 2, 100, 1024]

In [5]: l.append(2)

In [6]: print(l)
[1, 7, 4, 2, 100, 1024, 2]

In [7]: l.remove(1)

In [8]: print(l)
[7, 4, 2, 100, 1024, 2]

In [9]: l.remove(100)

In [10]: print(l)
[7, 4, 2, 1024, 2]

In [11]: l.remove(2)

In [12]: print(l)
[7, 4, 1024, 2]

In [13]: l.remove(2)

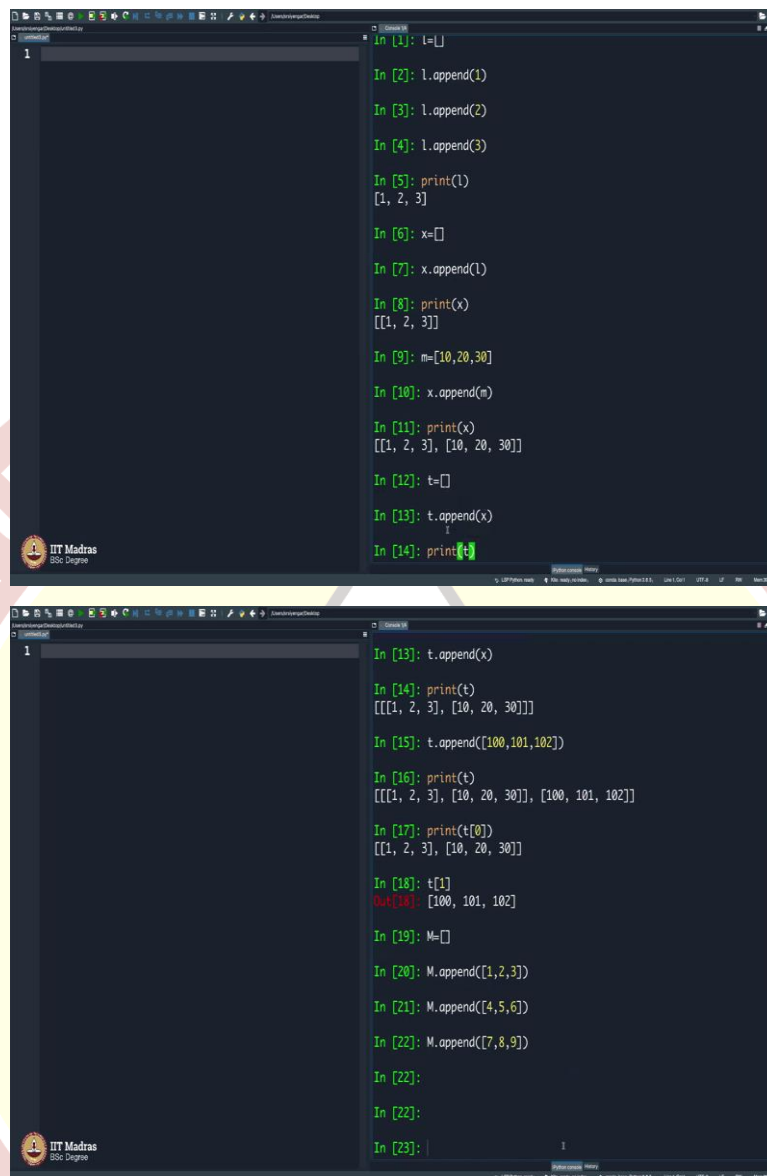
In [14]: print(l)
[7, 4, 1024]

In [15]:
```

`l` will have everything that it had initially with the number 1 removed. As you can see 1 is removed, it starts from 7. So, now, what is your guess? What if I remove let us say 100? It removes 100, there is no questions there. It is very obvious, but what if I say remove 2? The number 2 is repeated twice here. So, possibility one is, it will remove the first occurrence of 2 or maybe the second occurrence of 2 or maybe it will remove both.

So, let us see what it does. Let us print `l` and you see it has removed the first occurrence of 2. Just in case, I typed it once again, it removes the other occurrence of 2. I hope this is clear. This much of list you need to know. Let us get started with some more ideas with lists. Let me clear the shell.

(Refer Slide Time: 3:31)



```
In [1]: l=[]
In [2]: l.append(1)
In [3]: l.append(2)
In [4]: l.append(3)
In [5]: print(l)
[1, 2, 3]
In [6]: x=[]
In [7]: x.append(l)
In [8]: print(x)
[[1, 2, 3]]
In [9]: m=[10,20,30]
In [10]: x.append(m)
In [11]: print(x)
[[1, 2, 3], [10, 20, 30]]
In [12]: t=[]
In [13]: t.append(x)
In [14]: print(t)
[[[[1, 2, 3], [10, 20, 30]]]]
In [15]: t.append([100,101,102])
In [16]: print(t)
[[[[1, 2, 3], [10, 20, 30]], [100, 101, 102]]]
In [17]: print(t[0])
[[1, 2, 3], [10, 20, 30]]
In [18]: t[1]
Out[18]: [100, 101, 102]
In [19]: M=[]
In [20]: M.append([1,2,3])
In [21]: M.append([4,5,6])
In [22]: M.append([7,8,9])
In [22]:
In [22]:
In [23]:
```

What I am now going to tell you is probably one of the most important ideas in Data Sciences. And what is that? That is a matrix and how do you represent matrices? There are actually very sophisticated ways in which you can represent a matrix in Python, in fact there is a library function call the NumPy, numerical Python, we will not get in there. I will now tell you a very naive way of representing a matrix.

So, as you can see I created a list l and I appended 1 to it, I appended 2 to it, I appended 3 to it. So, sometimes I may not want to type the entire command, a simple up arrow will repeat the previous command. Another up arrow will repeat the previous to previous command and so on. So now, when I print l, it shows me 1 2 and 3 exactly what I have appended here as

you can see. So, as I have been telling you people, a computer is indeed a very dumb machine, it cannot do anything that you do not tell it to do. Fine.

So now, what is your guess? What if I create another array x and then I append l to x? You see that strange. Let me try doing that. Well, it accepted, there was no error. So, what is it even mean, I am appending, I am creating an array x and I am appending another array l to x. How does it look like? Let me print x and see it. You see it is an array and it contains another array.

Let me create just for example, let us say m equals 10, 20, 30 and let me take x just the way I appended l into x, let me append m into x. Now when I say print x, it prints 1, 2, 3, 10, 20, 30. So, it is a list within a list. That is a very popular idiom in English called wheels within wheels. So, it is a idea within an idea, you would have heard this. So, a problem within a problem. So, you see, this is a list within a list.

So, as you all know there are many living sort of, many microorganisms inside our body, so a living organism within a living organism, something like that. So, anyway you will get used to it. In fact, there is no limit here, you can have a list of a list of a list of a list of a list, we can go up to infinity like that. So, but just to complicate to make your life tough and well, why should we not teach you complicated stuff? Why should it always be easy?

Let me complicate it by typing another list let us say t I will call it a list and I will append x to it and I will print t. You see what is happening? Now, here is a list containing a list containing two lists. And I will append another thing to t which is 100, 101, 102, another list. If I print t, something else is coming, a list first you see this is a list 10, 20, 30, you probably are wondering why is he wasting time, so this is important. Do not worry, with time you will realise what I am doing. In fact, it will come naturally to you, the idea of list of lists.

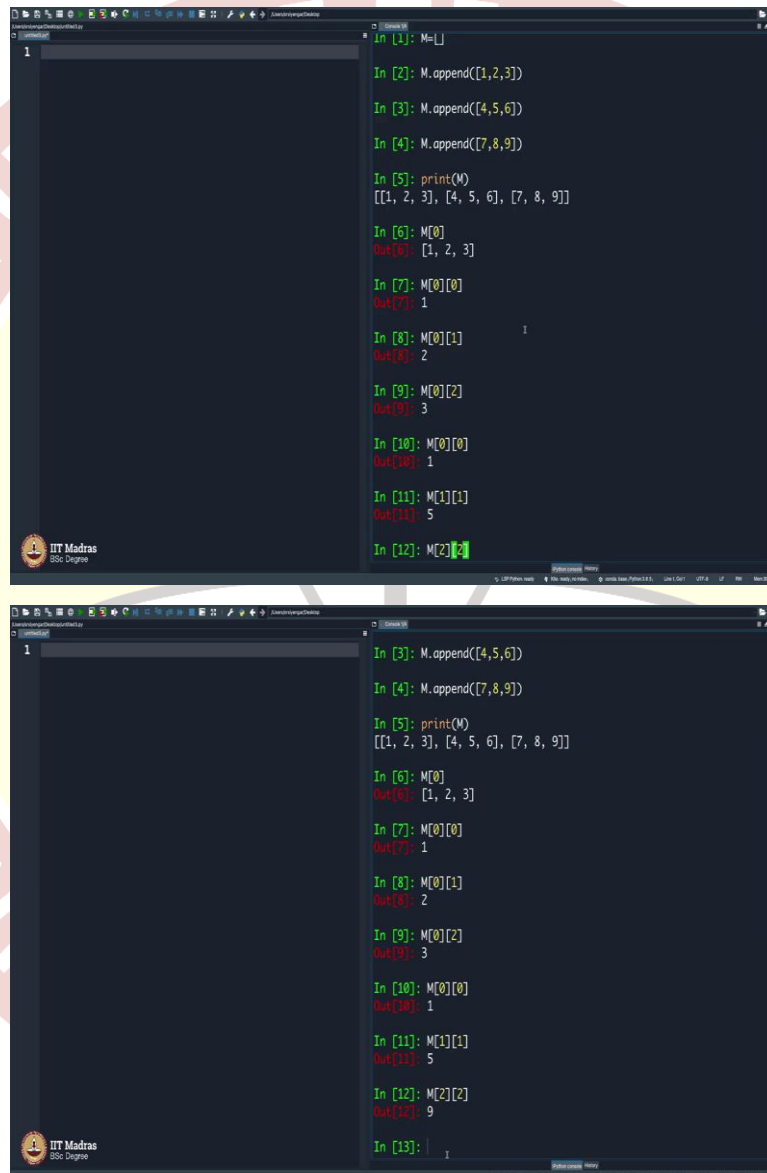
So, and then, this is another list 1, 2, 3; another list 10, 20, 30 and these two things put together is another list as you can see. This much, these two things put together is this list, let me start carefully, then say this list. This is an element of the bigger list and the bigger list has the second element as this. So, it will be clear when I say t of 0, what do you expect t of 0 to be?

If I were to print t, in fact you can simply set t of 0 also, it will print, but I will not do that. So, print t of 0 will give you the first element of the list t. t of 1 will give me the second element of the list t. So, let us now create a matrix 1, 2, 3, 4, 5, 6, 7, 8, 9. So, how do I do that? A

matrix m, let me say capital M is equal to so much and I will append first let us say 1 comma 2 comma 3 to it, you may not be able to see this.

So, you have M append 1, 2, 3 and then I say M append 4, 5, 6 and then I will say M append 7, 8 and 9, M append 7, 8, 9. So, I think it is clear. So, let me press many return key so that it comes to the centre and you can see it. M append 1, 2, 3; M append 4, 5, 6; M append 7, 8, 9.

(Refer Slide Time: 9:12)



```
In [1]: M=[]  
  
In [2]: M.append([1,2,3])  
  
In [3]: M.append([4,5,6])  
  
In [4]: M.append([7,8,9])  
  
In [5]: print(M)  
[[1, 2, 3], [4, 5, 6], [7, 8, 9]]  
  
In [6]: M[0]  
Out[6]: [1, 2, 3]  
  
In [7]: M[0][0]  
Out[7]: 1  
  
In [8]: M[0][1]  
Out[8]: 2  
  
In [9]: M[0][2]  
Out[9]: 3  
  
In [10]: M[0][0]  
Out[10]: 1  
  
In [11]: M[1][1]  
Out[11]: 5  
  
In [12]: M[2][2]  
Out[12]: 9  
  
In [13]:
```

I will frequently keep clearing the kernels so that it refreshes, in fact, I can also the clear screen, but I prefer doing it like this. So, I would have lost all the content that I had typed, so I will type it once again. M is a list, I will append 1, 2, 3 to it. I will again append 4, 5, 6 to it, it will be nice if we can type these commands alongside with me. 7, 8 and 9 and then I will

print M, capital M and I get 1, 2, 3, 4, 5, 6, 7, 8, 9 and this is how I am going to treat a matrix. By treating a matrix I mean, this is how I am going to store a matrix using lists.

So, you know M of 0 will be so much and M of 0 is a list containing 1, 2, 3 and in that list I can see the first element which is 1. In that list I can say the second element which is M of 0 comma 1, you see in matrices we would write a subscript ij, aij, a11, a12, a13, etcetera. So, similarly here, we say M of 0 2 is equal to so much so on and so forth. So, what did we learned just now?

So, just to end this, I will also show you what is in the diagonal, what will be in the diagonal, M of 0 0 will be 1, so please note you can say print or simply write the variable, it will still display. What is M of 1 1? This is called the diagonal elements in the matrix. 1 and 5 M of 2 2 is 9. I am not going in detail here because I assume you have undergone the computational thinking course and you are used to these one dimensional and two dimensional thinking.

So, we are, we now, while a list is one dimensional, what I am doing is, by making a list within a list, I am creating what is called two dimensional data. So, the word two dimensional data individually makes sense, but put together what does it mean? You will get to know of it in some time. So, that is about lists, you understand something about list right now. The best way to understand list is to go ahead and write a piece of code about it.

From now onwards, we will be discussing almost in every simple code, we will be using list. At least, I have used it in the code that I have done in the recent past. So, I believe even you will be using it. So, at least, in this week's discussion, we will be using good amount of lists, so it is good to get used to what are lists. So, go ahead, open your terminal, start typing your own programs, play around with lists.