

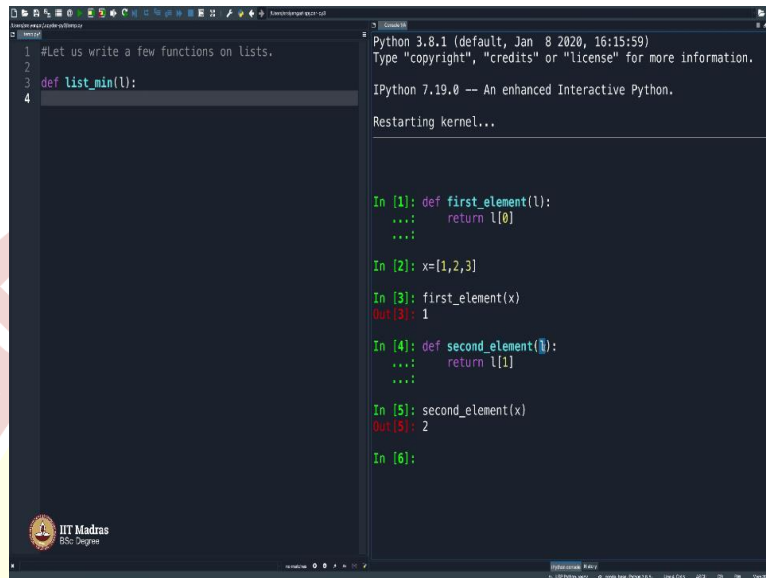


# IIT Madras

## ONLINE DEGREE

**Programming in Python**  
**Professor. Sudarshan Iyengar**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Ropar**  
**More Examples of Functions**

(Refer Slide Time: 00:16)



The screenshot shows a Jupyter Notebook interface with two main panels. The left panel displays a code editor with the following text:

```
1 #Let us write a few functions on lists.  
2  
3 def list_min(l):  
4
```

The right panel shows the IPython console output, which includes the following code and results:

```
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]: def first_element(l):  
...:     return l[0]  
...:  
  
In [2]: x=[1,2,3]  
  
In [3]: first_element(x)  
Out[3]: 1  
  
In [4]: def second_element(l):  
...:     return l[1]  
...:  
  
In [5]: second_element(x)  
Out[5]: 2  
  
In [6]:
```

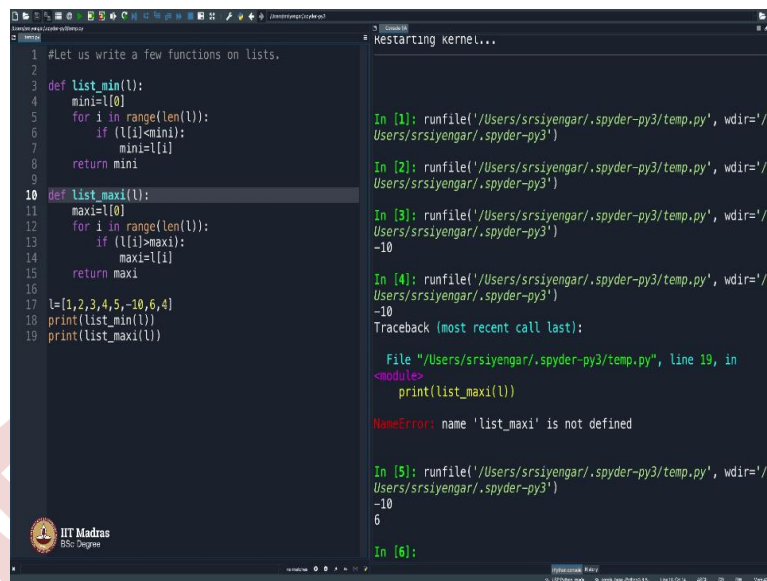
The bottom of the notebook shows the IIT Madras logo and the text "IIT Madras BSc Degree".

So, let us go ahead and write some warm up programs. So, let me do some list manipulations. Let us write a few functions on lists. What are those? Let me first start with finding the minimum element in a list, we have done this program already. The point is to do the same program, again, using a new concept that you have learned in Python.

So, how do we go about it? Let me think, define, let us say the list minimum and then I will input a list here. List can also be input, so I will just show you a small example here, define first element. If I say, define first element in a list L, and then I will simply return L of 0, and then define a list x as 1, 2, 3 and then say first element of x we will get as 1.

If you want to write a function called second element and return L of 1 which is a second element, that also works, as you can see, second element of x is 2. In fact, you can return another list in the function. So, I just want to show you that function can also take list as an input.

(Refer Slide Time: 01:50)



```
1 #Let us write a few functions on lists.
2
3 def list_min(l):
4     mini=l[0]
5     for i in range(len(l)):
6         if l[i]<mini:
7             mini=l[i]
8     return mini
9
10 def list_maxi(l):
11     maxi=l[0]
12     for i in range(len(l)):
13         if l[i]>maxi:
14             maxi=l[i]
15     return maxi
16
17 l=[1,2,3,4,5,-10,6,4]
18 print(list_min(l))
19 print(list_maxi(l))
```

Restarting kernel...

In [1]: runfile('/Users/srsiyengar/.spyder-py3/temp.py', wdir='/Users/srsiyengar/.spyder-py3')

In [2]: runfile('/Users/srsiyengar/.spyder-py3/temp.py', wdir='/Users/srsiyengar/.spyder-py3')

In [3]: runfile('/Users/srsiyengar/.spyder-py3/temp.py', wdir='/Users/srsiyengar/.spyder-py3')

-10

In [4]: runfile('/Users/srsiyengar/.spyder-py3/temp.py', wdir='/Users/srsiyengar/.spyder-py3')

-10

Traceback (most recent call last):

File "/Users/srsiyengar/.spyder-py3/temp.py", line 19, in <module>

print(list\_maxi(l))

NameError: name 'list\_maxi' is not defined

In [5]: runfile('/Users/srsiyengar/.spyder-py3/temp.py', wdir='/Users/srsiyengar/.spyder-py3')

-10

6

In [6]:

Alright. Now, let us get back. Let me find the minimum of the list L. How do we go about it? First, I will say minimum, I will say mini, because min happens to be a built-in function. So, you should not use the, you should not say print equals 0, because print is used to print, the print command in Python. You should not use the built-in command as variables, which we have mentioned already.

But in one of my previous videos, I did use min as a variable. Thankfully, it did not throw any errors. It was probably in third week I do not remember. Anyway, sorry for the digression. Let me get back, and write a piece of code for finding minimum of a list. So, I will say my minimum is L of 0. And then I will go through a for loop for i in range len of l, so what should I do, I am just thinking. Just in case if this minimum is less than is, if my l of i less than mini let us say which means whatever you have found as minimum. If some entry of l is less than that, then you say minimum, is that.

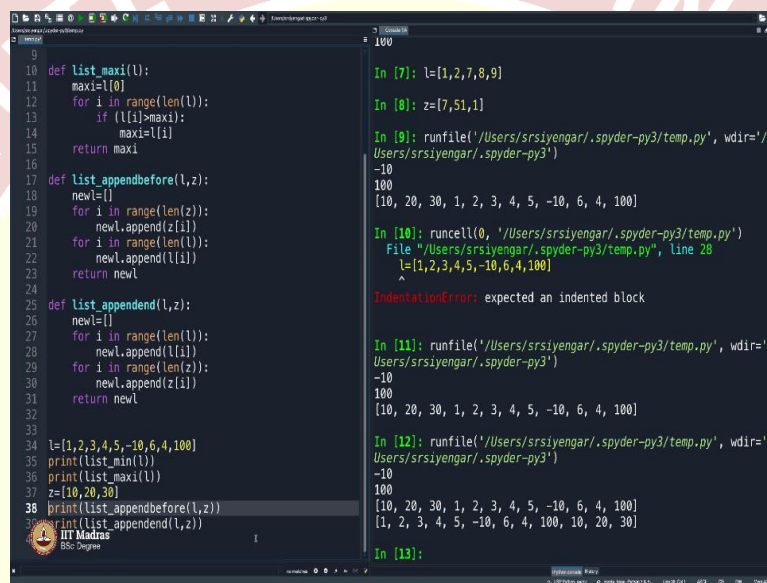
So, pretty simple and then finally you return mini. A quick code. I did not even give it a thought, but let us see whether it works or not. Alright, so what I will do is I will execute it and maybe I should write 1, 2, 3, 4, 5 minus 10, 6, 4, a list. And then I should say print list min of l. Let us see, this should pop out minus 10. Let us see if it does that, perfect. It is giving me minus 10. This function works fine. I have just no verified.

Now similarly, I want to define list max. I am doing it without a lot of energy that is because you guys know what is happening. In fact, this should be boring for you. Maximum again, max is a built-in function so I will say maxi, which is equal to l of 0, I will call the first element as a maximum. And in case I encounter a new element here, which is greater than the

maximum found so far, I will declare that as maximum not explaining much because we have seen this in the previous weeks.

In case you are confused, you may want to take a look at those lessons. If  $l$  of  $i$  is greater than maximum, then  $maxi$  equals  $l$  of  $i$ . And then say return  $maxi$ . Let us see if it does what it is supposed to do. So, I will say print list  $maxi$ , of  $l$ . List  $maxi$  is not defined. List  $max$  I say here and then put a  $maxi$  here. So, let me also make it  $maxi$  here. So, yes, it shows the greatest number here is 6.

(Refer Slide Time: 05:10)



```
9
10 def list_maxi(l):
11     maxi=l[0]
12     for i in range(len(l)):
13         if (l[i]>maxi):
14             maxi=l[i]
15     return maxi
16
17 def list_appendbefore(l,z):
18     newl=[]
19     for i in range(len(z)):
20         newl.append(z[i])
21     for i in range(len(l)):
22         newl.append(l[i])
23     return newl
24
25 def list_appendend(l,z):
26     newl=[]
27     for i in range(len(l)):
28         newl.append(l[i])
29     for i in range(len(z)):
30         newl.append(z[i])
31     return newl
32
33 l=[1,2,3,4,5,-10,6,4,100]
34 print(list_min(l))
35 print(list_maxi(l))
36 z=[10,20,30]
37 print(list_appendbefore(l,z))
38 print(list_appendend(l,z))
39
```

```
In [7]: l=[1,2,7,8,9]
In [8]: z=[7,51,1]
In [9]: runfile('/Users/srsiyengar/.spyder-py3/temp.py', wdir='/Users/srsiyengar/.spyder-py3')
-10
100
[10, 20, 30, 1, 2, 3, 4, 5, -10, 6, 4, 100]
In [10]: runcell(0, '/Users/srsiyengar/.spyder-py3/temp.py')
File "/Users/srsiyengar/.spyder-py3/temp.py", line 28
    l=[1,2,3,4,5,-10,6,4,100]
    ^
IndentationError: expected an indented block
In [11]: runfile('/Users/srsiyengar/.spyder-py3/temp.py', wdir='/Users/srsiyengar/.spyder-py3')
-10
100
[10, 20, 30, 1, 2, 3, 4, 5, -10, 6, 4, 100]
In [12]: runfile('/Users/srsiyengar/.spyder-py3/temp.py', wdir='/Users/srsiyengar/.spyder-py3')
-10
100
[10, 20, 30, 1, 2, 3, 4, 5, -10, 6, 4, 100]
[1, 2, 3, 4, 5, -10, 6, 4, 100, 10, 20, 30]
In [13]:
```

In case I were to change this to let us say 100, this must show 100 as the maximum number. Perfect, everything is going good. So, let me write more functions now. What do I write? Let me think. Maybe, I will define list prefix, prefix is a little heavy on the mind. Append before we will say of a list  $l$  and  $z$ .

So, what do I mean by list append before? If you have a list, let us say 1 comma 2 comma 7 comma 8 comma 9, and another list  $z$  equals 7 comma 5, comma 51, comma 1, let us say, I want  $z$  to whatever is the entries here to appear here. Ultimately, the answer should be 7, 51, 1, 1, 2, 7, 8 and 9. This is what I mean by append before, so let us try writing a code for append before.

How do I go about it? It is very easy. Let me call it a new list and in the new list, I will first append the entries of  $z$  and then entries of  $l$ . As simple as that, being a little fast here, that that is how we are going to go in the forthcoming semesters, because I am expecting some maturity from you guys that you have typed and written some code so far.

For or i in range len of z, so what I will do is new list, append z of i. So, this will simply append entries of z into my new l list. And after that, I write another for loop range length of l. And I say new l append. And now I will append entries of l. So, what is happening here, so I am used to rather every programmer is used to simply writing, they are used to simply writing the code, and then trying to check it.

Not worrying whether what they did was right or wrong, because you will get to know here whether it is right or wrong. So, you have l here and I need to put a z here, z equals let us say 10, 20, 30. Perfect. So, I will say print list append before of l comma z, what is this supposed to do, it will this function as you know, will go and see what is in list append before put list the list l list z here, it will go inside and do all these manipulations and then finally, return. See I did not return it.

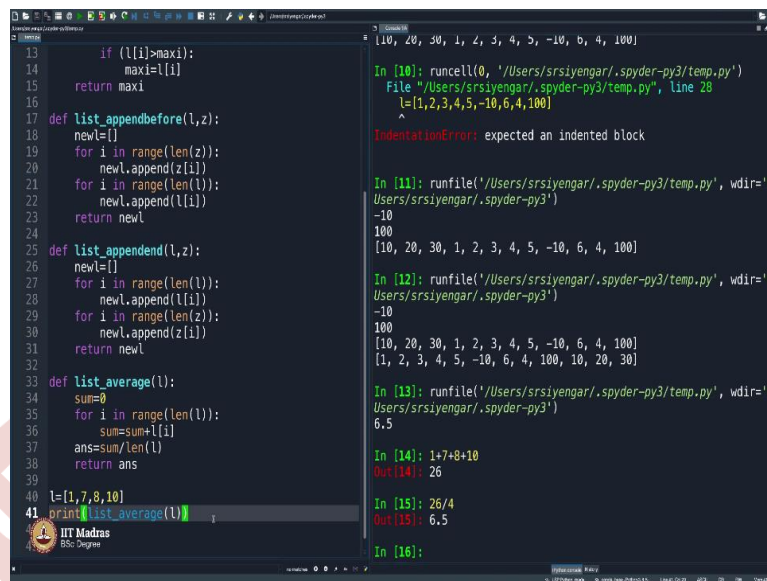
It will return error message if you do not return something it will not do. I mean of course, it will not return an error message but it will not do what it is supposed to do. Return new l and that is it. You see 10, 20, 30 and then 1, 2, 3, 4, 5 minus 10, 6, 4, and 100. Perfect, so we will now go ahead and write another function define list append before we did we will do end now l comma z. The same function.

So, what should what should happen now? What should happen? Do not worry about this error message here, so it probably got executed, as I was typing this code. So, what should I do now? I probably should copy paste this here. I copy paste this here and then whatever goes in the front should instead be appended at the end.

This should work. Let us see if it works or not. So, I am executing it with list appendend l comma z. Print list, appendend l comma z. And there we are. We have 6, 4, 100, and then 10,20, 30. It got appended at the end. So, very good. So, what is it that you observed? You have observed that you can indeed manipulate the lists by using functions.



(Refer Slide Time: 09:40)



```
13     if (l[i]>maxi):
14         maxi=l[i]
15     return maxi
16
17 def list_appendbefore(l,z):
18     newl=[]
19     for i in range(len(z)):
20         newl.append(z[i])
21     for i in range(len(l)):
22         newl.append(l[i])
23     return newl
24
25 def list_appendend(l,z):
26     newl=[]
27     for i in range(len(l)):
28         newl.append(l[i])
29     for i in range(len(z)):
30         newl.append(z[i])
31     return newl
32
33 def list_average(l):
34     sum=0
35     for i in range(len(l)):
36         sum=sum+l[i]
37     ans=sum/len(l)
38     return ans
39
40 l=[1,7,8,10]
41 print(list_average(l))
```

In [10]: `runcell(0, '/Users/srsiyengar/.spyder-py3/temp.py')`  
File `"/Users/srsiyengar/.spyder-py3/temp.py"`, line 28  
`l=[1,2,3,4,5,-10,6,4,100]`  
IndentationError: expected an indented block

In [11]: `runfile('/Users/srsiyengar/.spyder-py3/temp.py', wdir='/Users/srsiyengar/.spyder-py3')`  
-10  
100  
[10, 20, 30, 1, 2, 3, 4, 5, -10, 6, 4, 100]

In [12]: `runfile('/Users/srsiyengar/.spyder-py3/temp.py', wdir='/Users/srsiyengar/.spyder-py3')`  
-10  
100  
[10, 20, 30, 1, 2, 3, 4, 5, -10, 6, 4, 100]  
[1, 2, 3, 4, 5, -10, 6, 4, 100, 10, 20, 30]

In [13]: `runfile('/Users/srsiyengar/.spyder-py3/temp.py', wdir='/Users/srsiyengar/.spyder-py3')`  
6.5

In [14]: `1+7+8+10`  
Out[14]: 26

In [15]: `26/4`  
Out[15]: 6.5

In [16]:

Just a warm up exercise for you all. So, let me just write one more piece of code. I am in the mood for writing more functions. So, what will I do? Let me think. So, let me write a piece of code to find the average of the entries of a list. List average of a list l.

How do I go about it? Let us see. First I must find the sum and then I should divide that by the number of entries in l. Pretty simple, that we bring this to the center of the screen, so that it is clear to you also maybe zoom in a bit. So, what I will do is for i, in range l, len of. Len of l what I will do is sum equals sum plus l of i, this will return the sum total of the entries in the list.

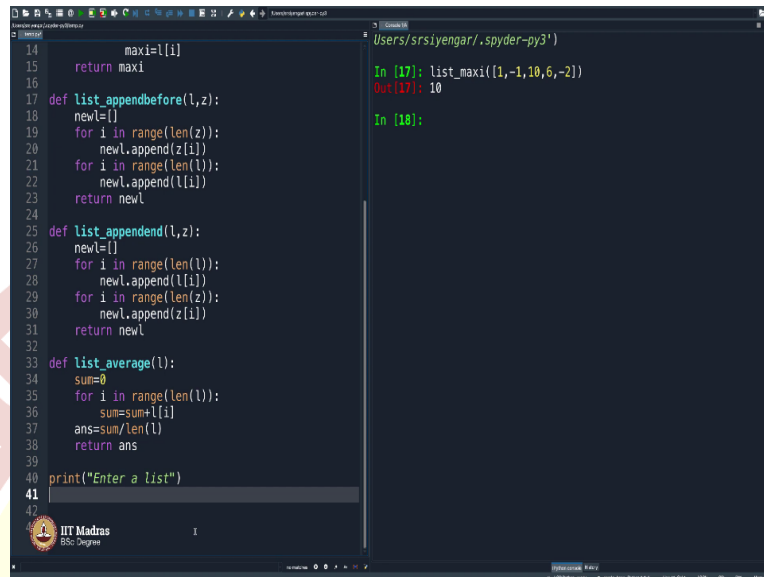
And finally, I should say answer equals sum divided by length of l. This will return my average, is not it? That is how you would compute average and this was taught to you in your school days. I think if I am right, this is the formula for average, then I will say return answer. That is it.

So, what I will now do is, try to print this for a list. So, all these things maybe I can remove it now. Remove it, remove it not required so we have checked these functions, these functions will still be there. If you want, you can use them anytime you want, but I now want to check if I can find the average of the entries of a list. So, let me scroll down and then try to write a list here. My list l is equal to let us say 1 comma 7 comma 8 comma 10.

So, what is the sum of this? 10 plus 8 is 18, plus 7 is 25 plus 1 is 26. 26 by 4 whatever that is. So, print list average of l, now let us see what this comes out to be 6.5. Is that really 1 plus 7

plus 8 plus 10, what is that 26 divided by 4 6.5. Perfect. So, my answer is indeed. So, I have a written list average, you can go ahead and write more functions.

(Refer Slide Time: 12:11)



```
14     maxi=l[i]
15     return maxi
16
17 def list_appendbefore(l,z):
18     newl=[]
19     for i in range(len(z)):
20         newl.append(z[i])
21     for i in range(len(l)):
22         newl.append(l[i])
23     return newl
24
25 def list_appendend(l,z):
26     newl=[]
27     for i in range(len(l)):
28         newl.append(l[i])
29     for i in range(len(z)):
30         newl.append(z[i])
31     return newl
32
33 def list_average(l):
34     sum=0
35     for i in range(len(l)):
36         sum=sum+l[i]
37     ans=sum/len(l)
38     return ans
39
40 print("Enter a list")
41
42
```

Users/srsiyengar/.spyder-py3

```
In [17]: list_maxi([1,-1,10,6,-2])
Out[17]: 10

In [18]:
```

The best part now is, the following. I will remove this. Keep observing, I will execute this. Once I execute this, this entire thing goes and sits in your memory. It sits in your memory. Let me just scroll this down just so that you can see it clearly. It goes and sits in your memory, and you can happily use them here. Use them here, just observe.

So, I would like to use the list maxi function. Simply say list maxi function of a list, whatever you can either type here or declare the minus 1, 10, 6 minus 2, close and close, it will give you 10. You see, this gets loaded to your memory and you have now authored a function. Is not that awesome. So, you can go ahead and write your own functions. And they will work like magic here.

The moment you call the function, you put the name here, it will show you the output. Either here or here itself, you can print them. This is called modular approach to programming where you very patiently, very slowly. You break your problem into smaller pieces and write the pieces as functions and finally stitch them together and you will get a final program.

So, what do I mean by that? By that I mean, print, enter a list, and you can take the list from the user here. And then you can ask him what you need to do. If you need to find the average of the list or find the minimum in the list or maximum in the list based on that you can call the functions and then do it here.

In fact, your life becomes very easy, because you need not write all his code here. I am not writing the code right now, I should go ahead with the next program, but I am just giving you a quick illustration. You can write the entire code here and eliminate the requirement for functions or write down the functions here and call them whenever you want. That will be an easier way to write a piece of code. You will see the power of this in our forthcoming programs now.

