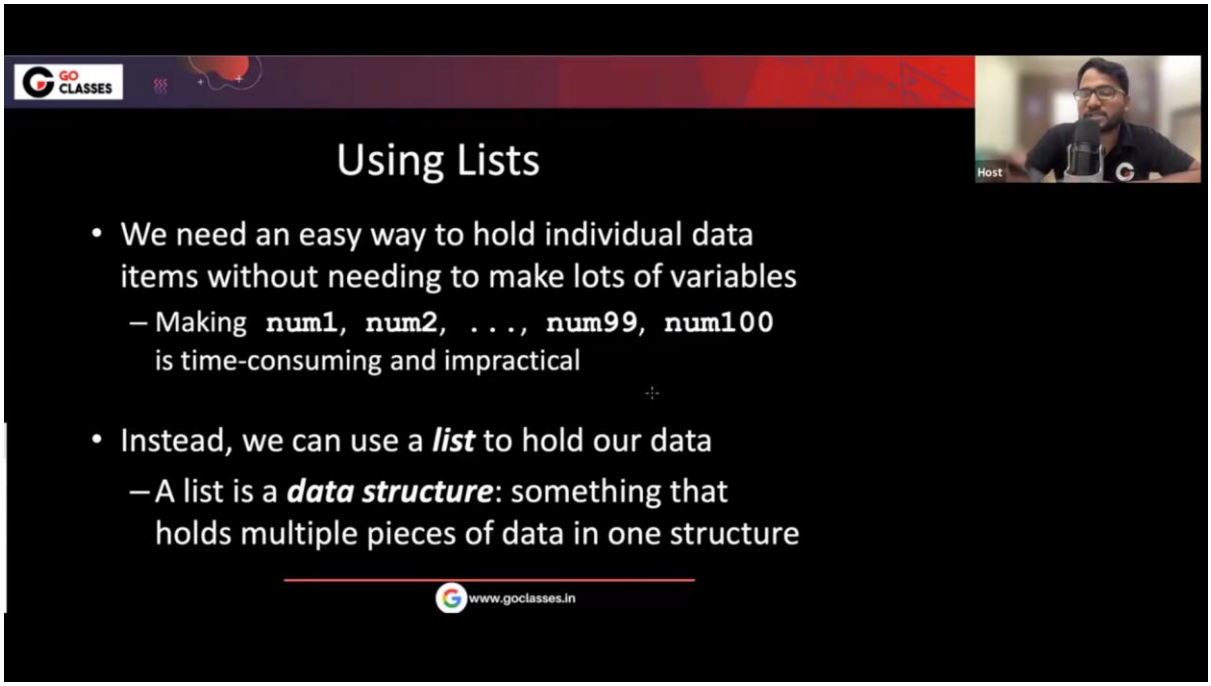
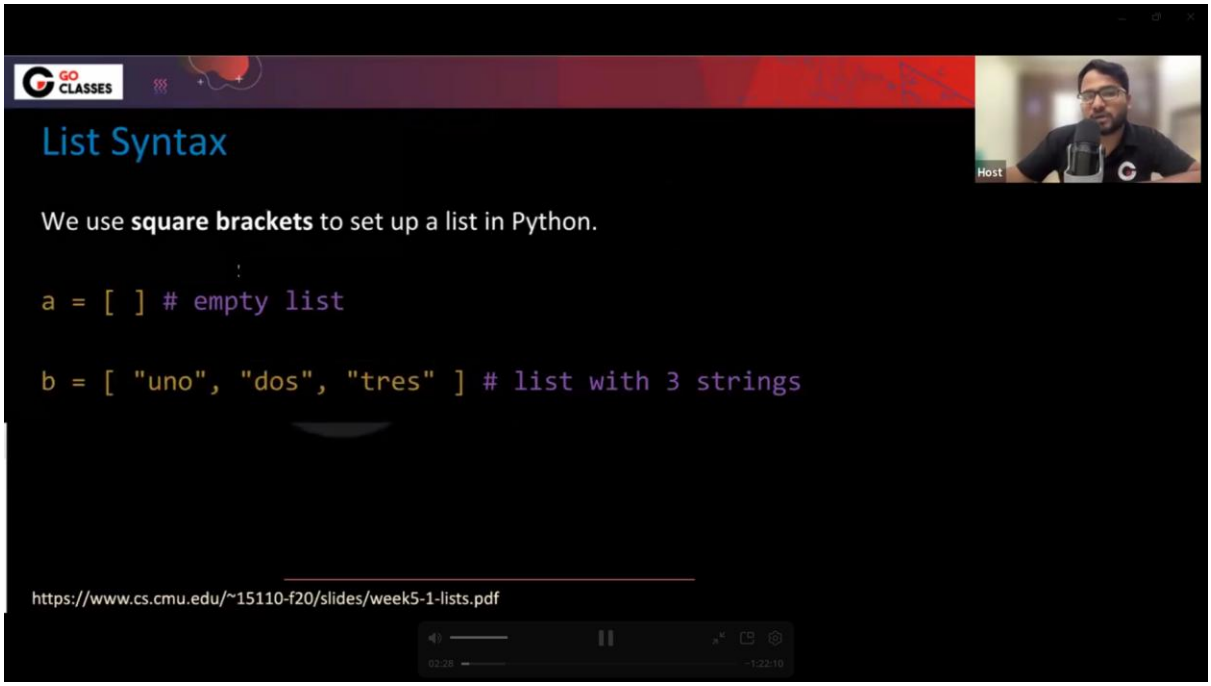
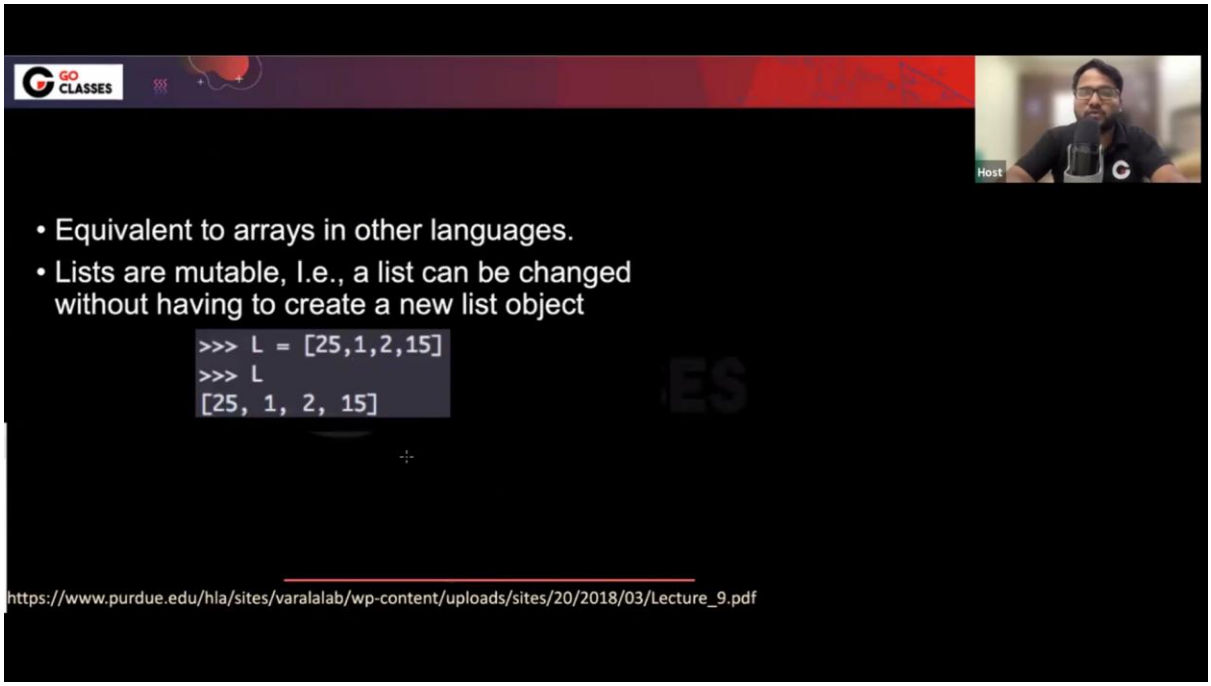
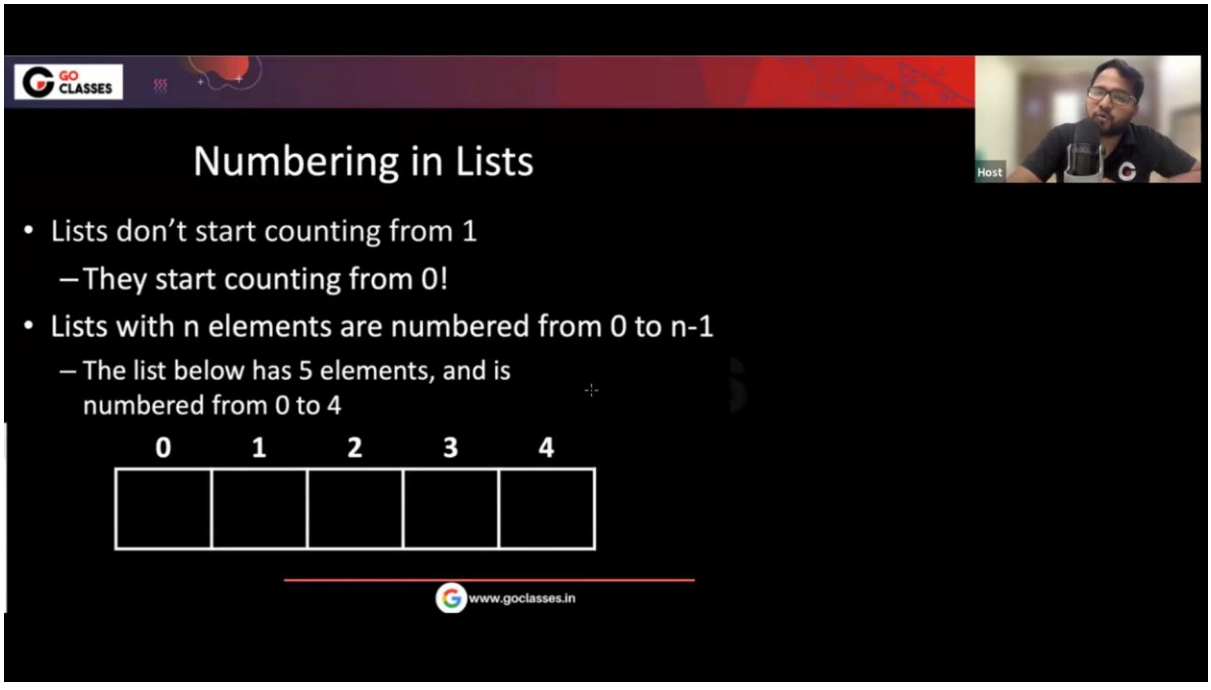
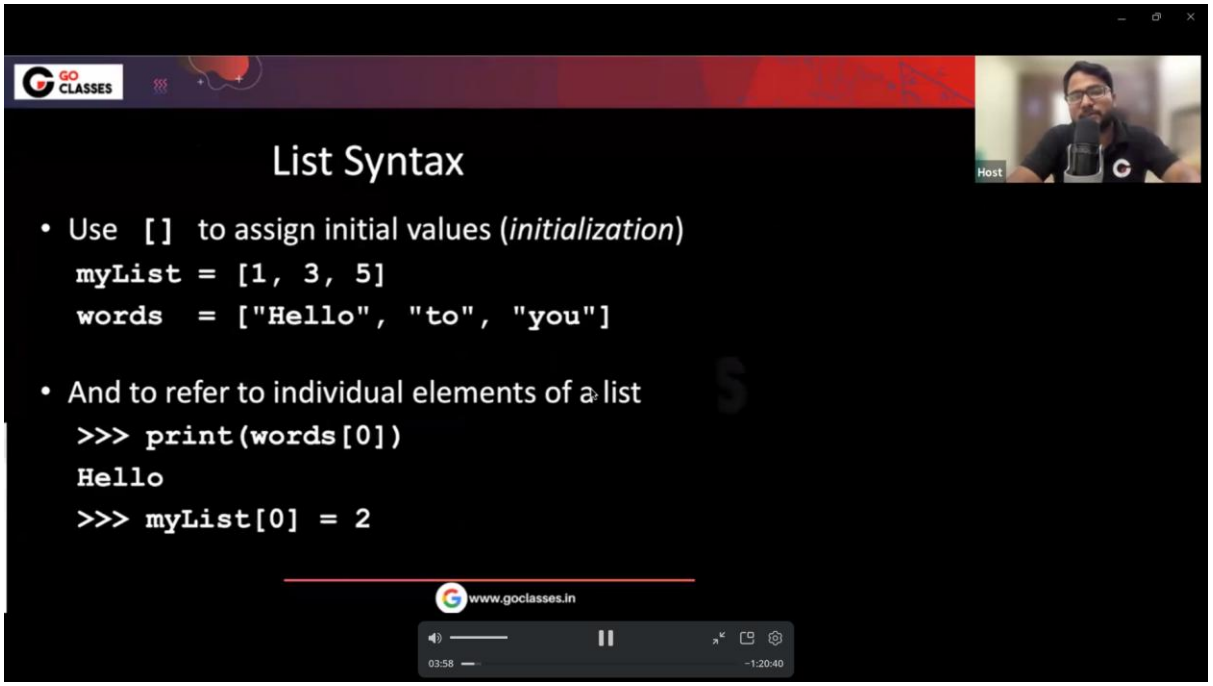
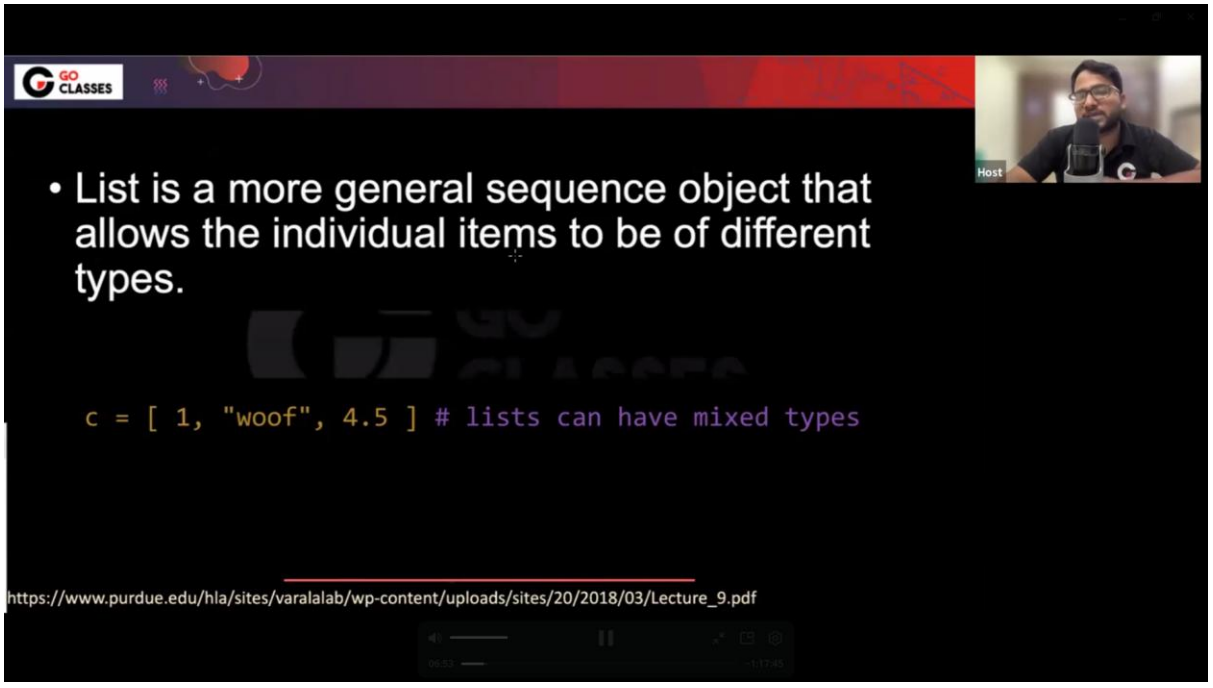

GO CLASSES
Using Lists
Host
• We need an easy way to hold individual data items without needing to make lots of variables
– Making num1, num2, ..., num99, num100 is time-consuming and impractical
• Instead, we can use a *list* to hold our data
– A list is a ***data structure***: something that holds multiple pieces of data in one structure
www.goclasses.in

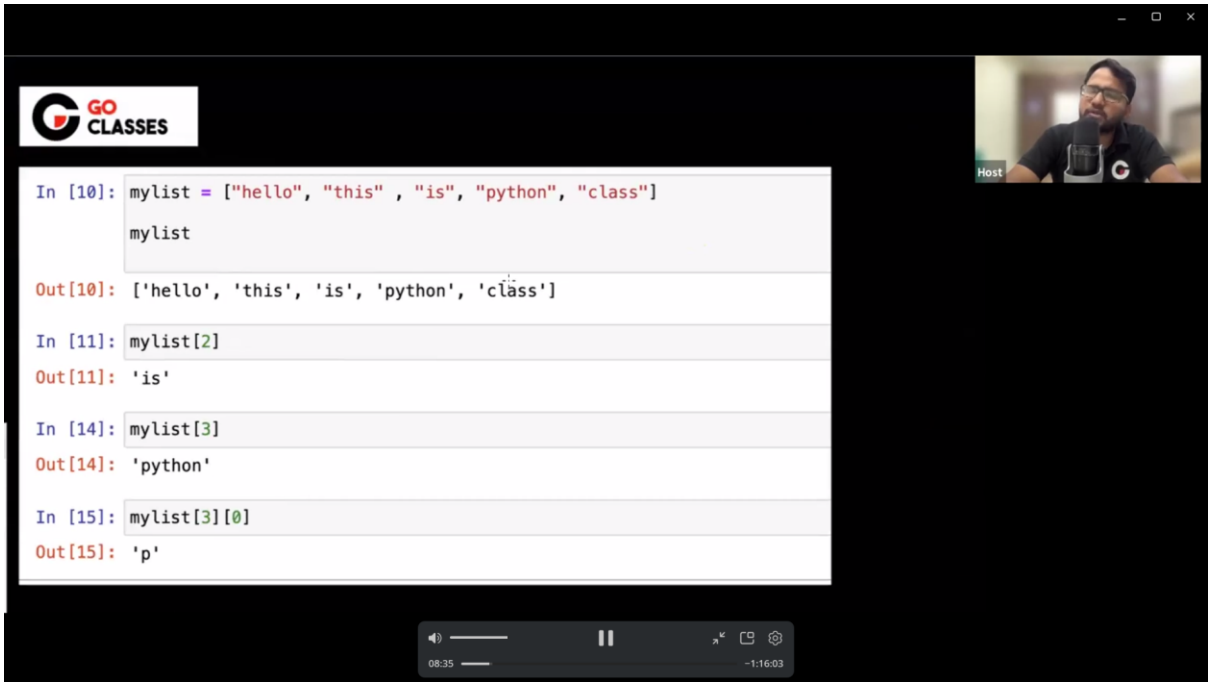

GO CLASSES
List Syntax
Host
We use **square brackets** to set up a list in Python.
a = [ ] # empty list
b = ["uno", "dos", "tres"] # list with 3 strings
https://www.cs.cmu.edu/~15110-f20/slides/week5-1-lists.pdf

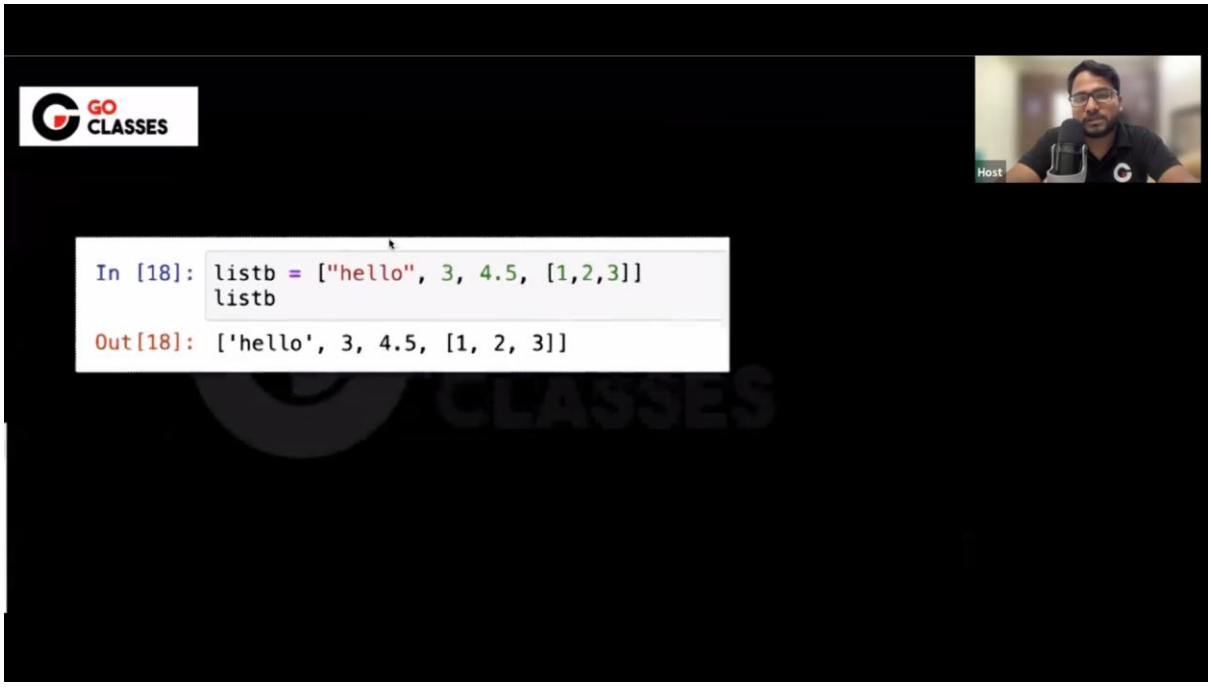

GO CLASSES
Host
• Equivalent to arrays in other languages.
• Lists are mutable, I.e., a list can be changed without having to create a new list object
>>> L = [25,1,2,15]
>>> L
[25, 1, 2, 15]
https://www.purdue.edu/hla/sites/varalalab/wp-content/uploads/sites/20/2018/03/Lecture_9.pdf


GO CLASSES
Host
Numbering in Lists
• Lists don't start counting from 1
– They start counting from 0!
• Lists with n elements are numbered from 0 to n-1
– The list below has 5 elements, and is numbered from 0 to 4
0 1 2 3 4
www.goclasses.in


GO CLASSES
Host
List Syntax
• Use [] to assign initial values (initialization)
myList = [1, 3, 5]
words = ["Hello", "to", "you"]
• And to refer to individual elements of a list
>>> print(words[0])
Hello
>>> myList[0] = 2
www.goclasses.in
03:58
-1:20:40


GO CLASSES
Host
• List is a more general sequence object that allows the individual items to be of different types.
c = [1, "woof", 4.5] # lists can have mixed types
https://www.purdue.edu/hla/sites/varalalab/wp-content/uploads/sites/20/2018/03/Lecture_9.pdf


GO CLASSES
Host
In [10]: mylist = ["hello", "this" , "is", "python", "class"]
mylist
Out[10]: ['hello', 'this', 'is', 'python', 'class']
In [11]: mylist[2]
Out[11]: 'is'
In [14]: mylist[3]
Out[14]: 'python'
In [15]: mylist[3][0]
Out[15]: 'p'
08:35
-1:16:03

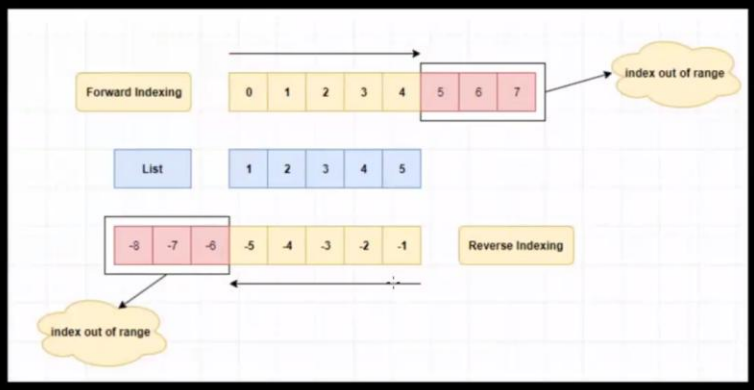

GO CLASSES
Host
In [18]: listb = ["hello", 3, 4.5, [1,2,3]]
listb
Out[18]: ['hello', 3, 4.5, [1, 2, 3]]

GO CLASSES

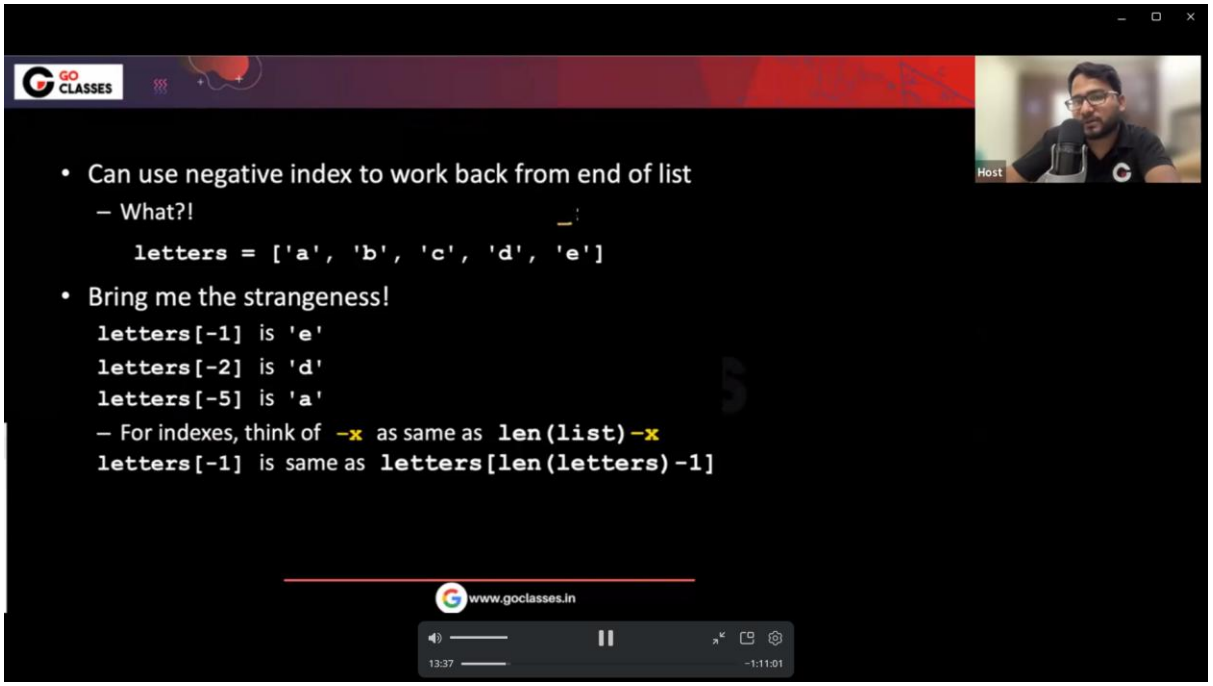
Properties of a List

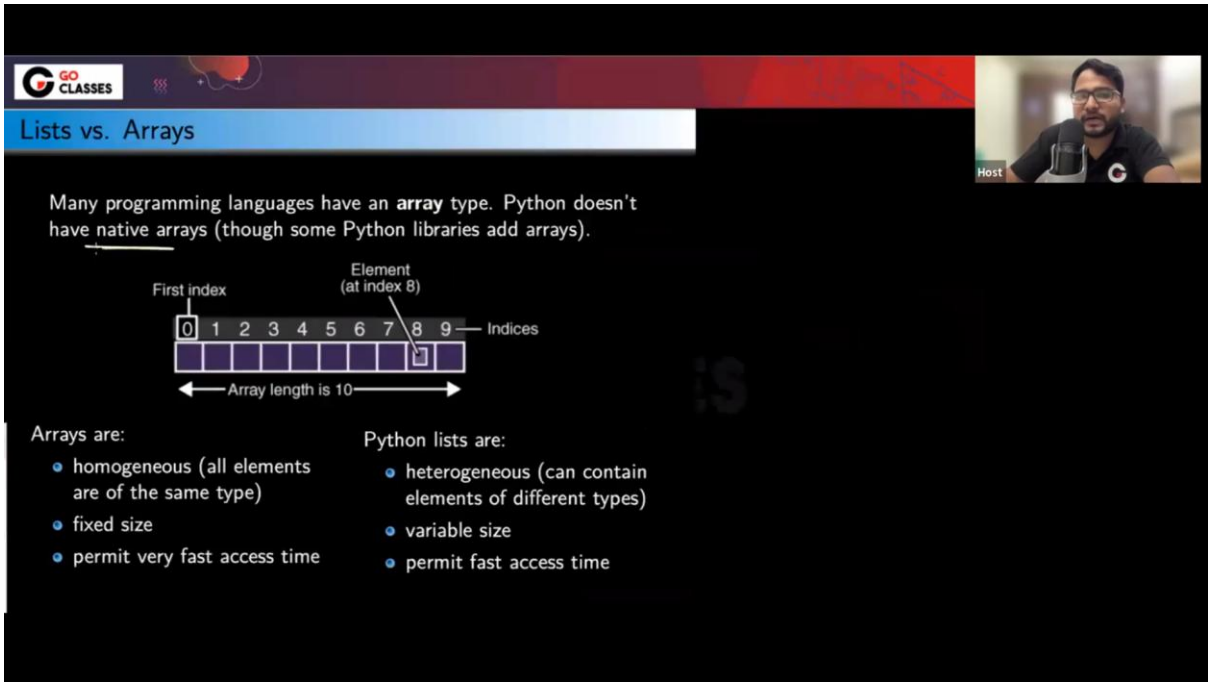
- Heterogeneous (multiple data types!)
- Contiguous (all together in memory)
- Ordered (numbered from 0 to n-1)

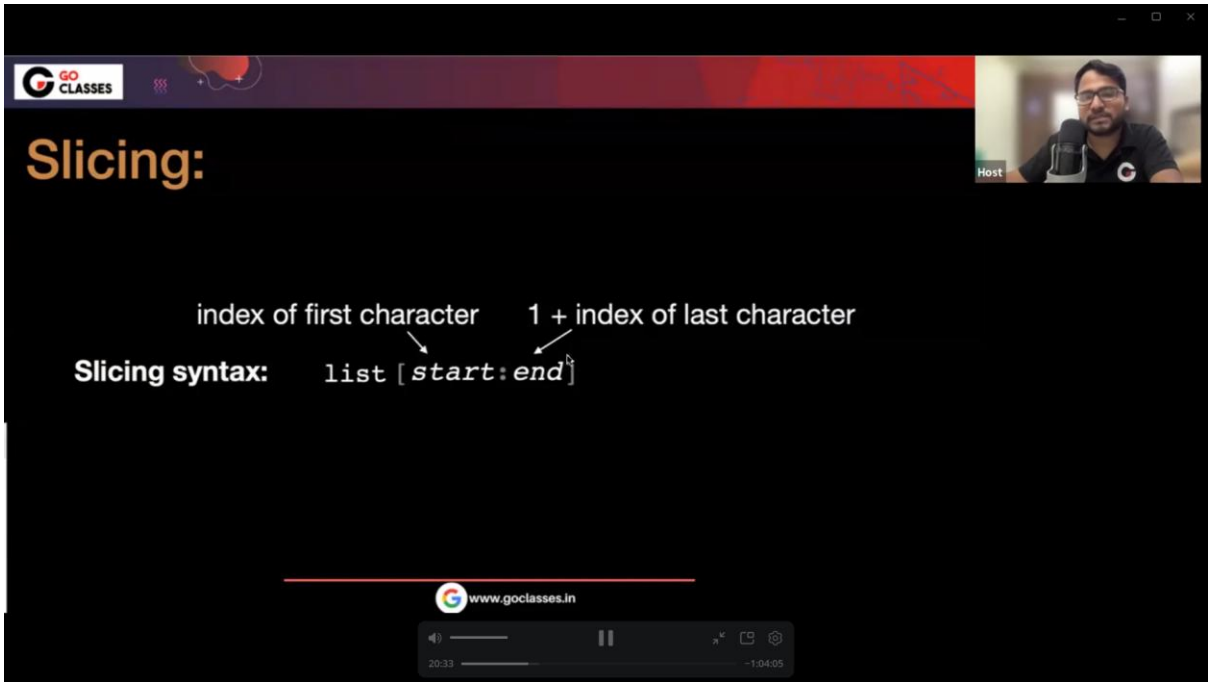
GO CLASSES

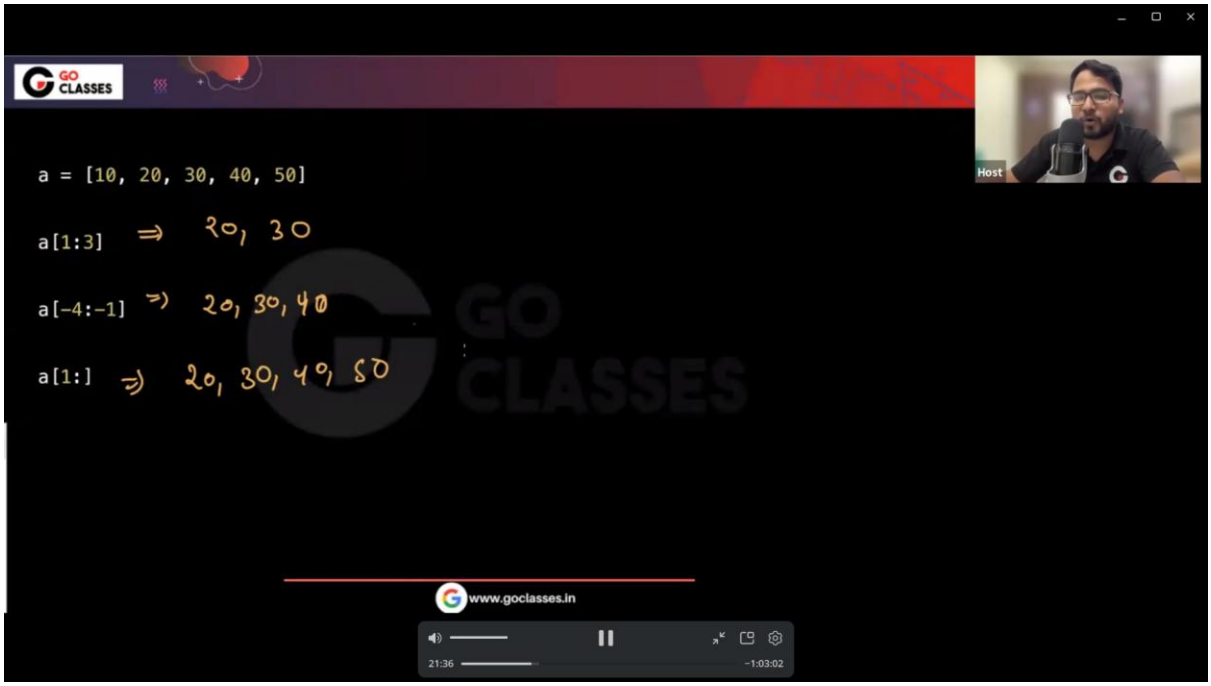


13:15 -1:11:23


GO CLASSES
Host
• Can use negative index to work back from end of list
– What?!
letters = ['a', 'b', 'c', 'd', 'e']
• Bring me the strangeness!
letters[-1] is 'e'
letters[-2] is 'd'
letters[-5] is 'a'
– For indexes, think of -x as same as len(list)-x
letters[-1] is same as letters[len(letters)-1]
www.goclasses.in
13:37
-1:11:01


GO CLASSES
Host
Lists vs. Arrays
Many programming languages have an array type. Python doesn't have native arrays (though some Python libraries add arrays).
First index
Element (at index 8)
0 1 2 3 4 5 6 7 8 9
Indices
Array length is 10
Arrays are:
homogeneous (all elements are of the same type)
fixed size
permit very fast access time
Python lists are:
heterogeneous (can contain elements of different types)
variable size
permit fast access time


GO CLASSES
Slicing:
index of first character
1 + index of last character
Slicing syntax: list[start:end]
Host
www.goclasses.in
20:33
-1:04:05


GO CLASSES
a = [10, 20, 30, 40, 50]
a[1:3] ⇒ 20, 30
a[-4:-1] ⇒ 20, 30, 40
a[1:] ⇒ 20, 30, 40, 50
GO CLASSES
Host
www.goclasses.in
21:36
-1:03:02

Converting Strings Into Lists


```
>>> a = "hello"
>>> list(a)
['h', 'e', 'l', 'l', 'o']
```

- The **list** function can convert a string to a list.
  - Always works.
  - **Very handy** if we want to manipulate a string's contents and create new strings based on them.

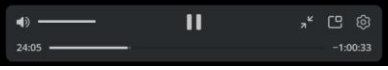
https://crystal.uta.edu/~athitsos/courses/cse1310_summer2013/lectures/7a_strings.pdf

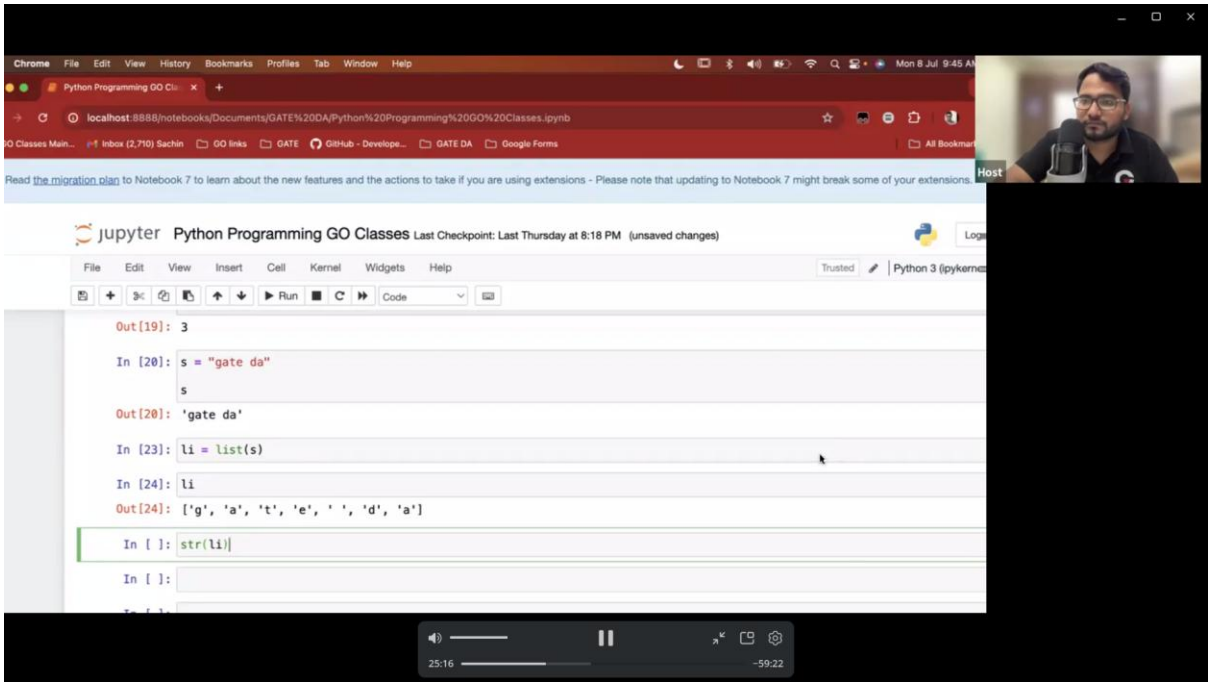
STRINGS to LISTS

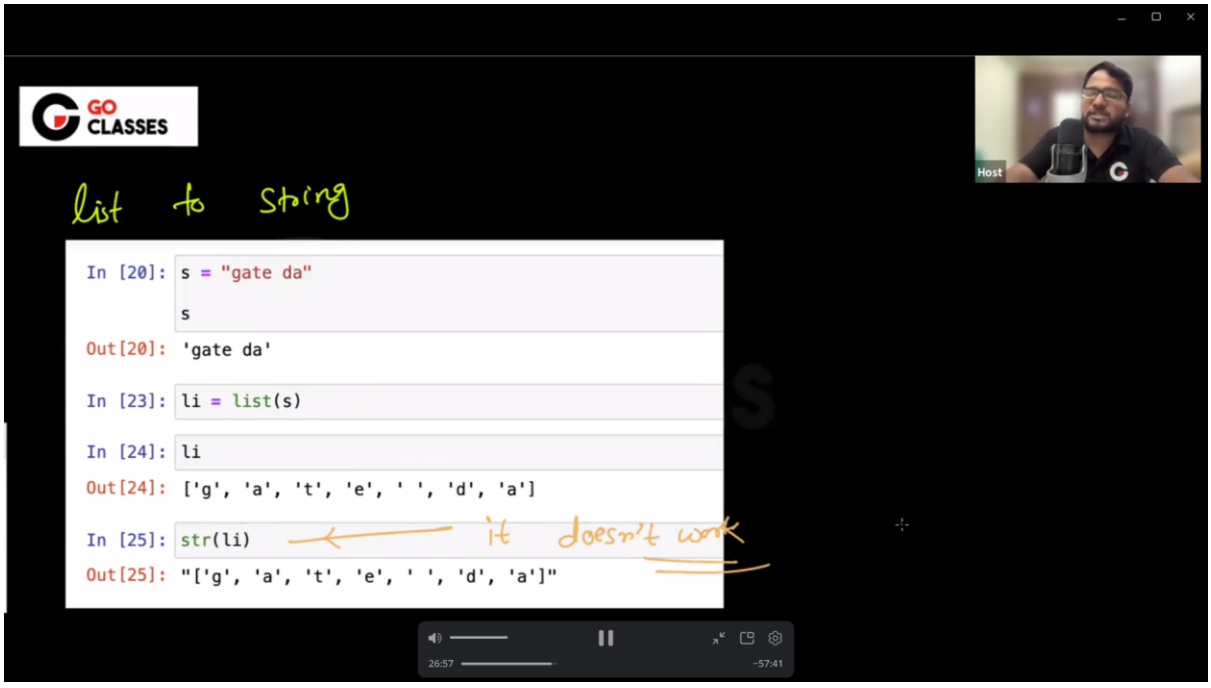
- Convert **string to list** with `list(s)`
  - Every character from `s` is an element in a list

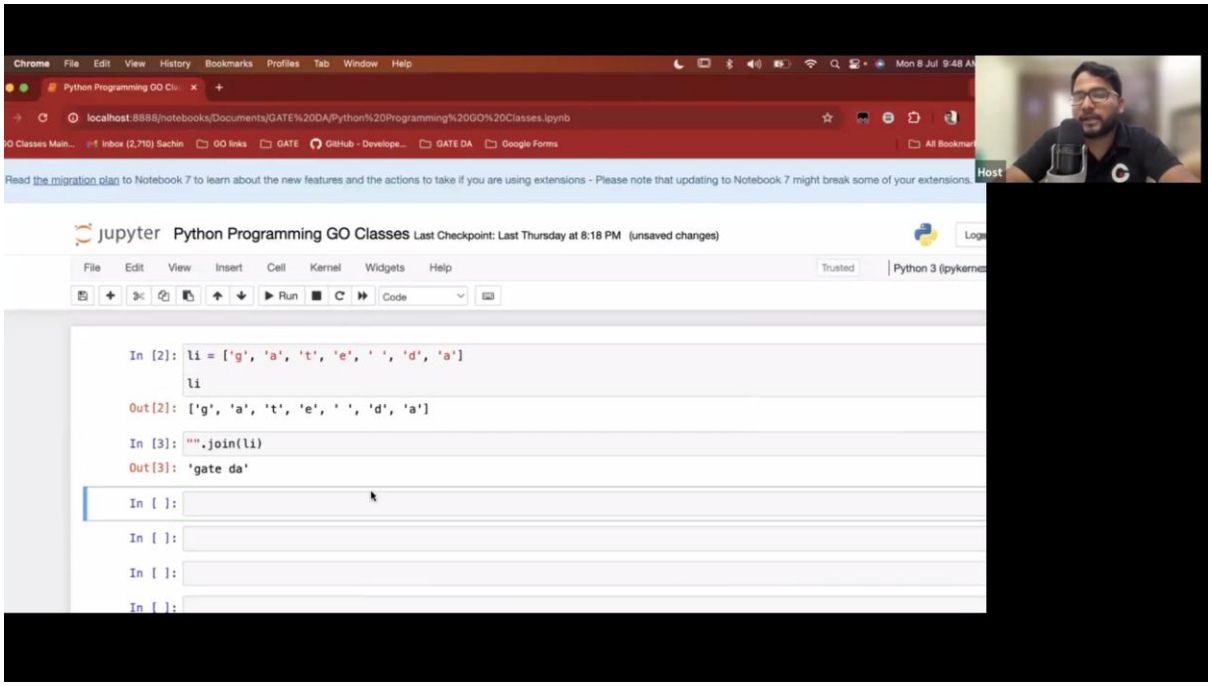
```
In [1]: word = "GATE DA"
        list(word)
Out[1]: ['G', 'A', 'T', 'E', ' ', 'D', 'A']
```

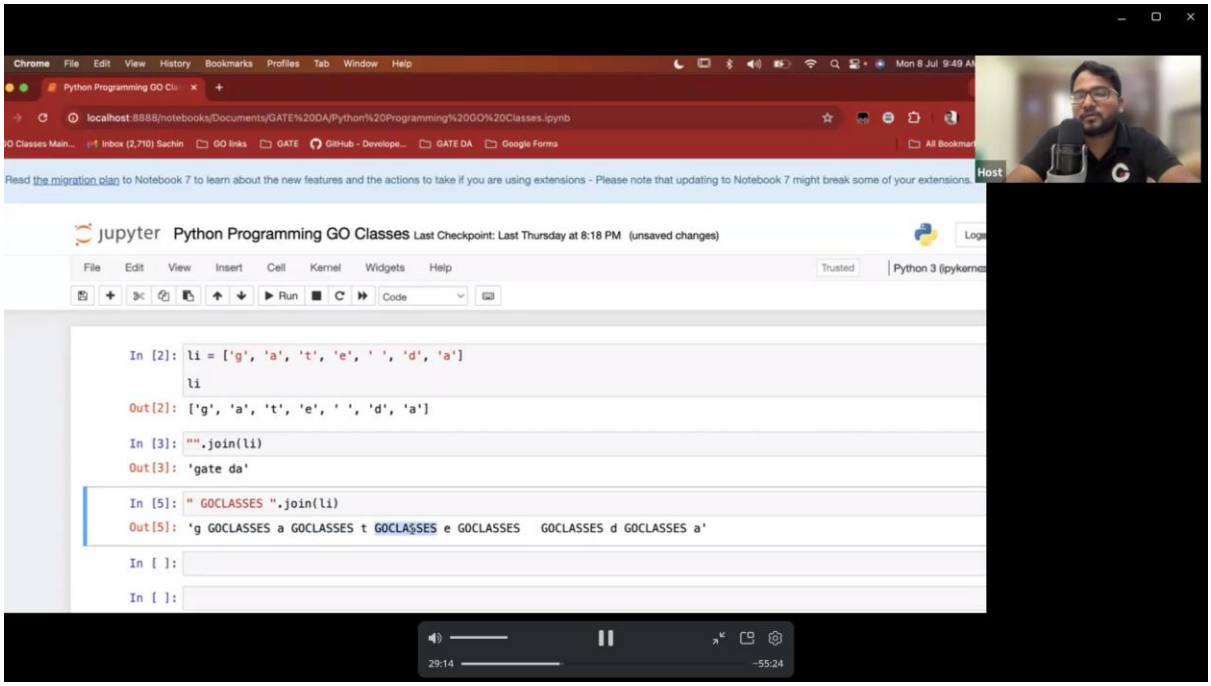
https://ocw.mit.edu/courses/6-100l-introduction-to-cs-and-programming-using-python-fall-2022/mit6_100l_f22_lec10.pdf

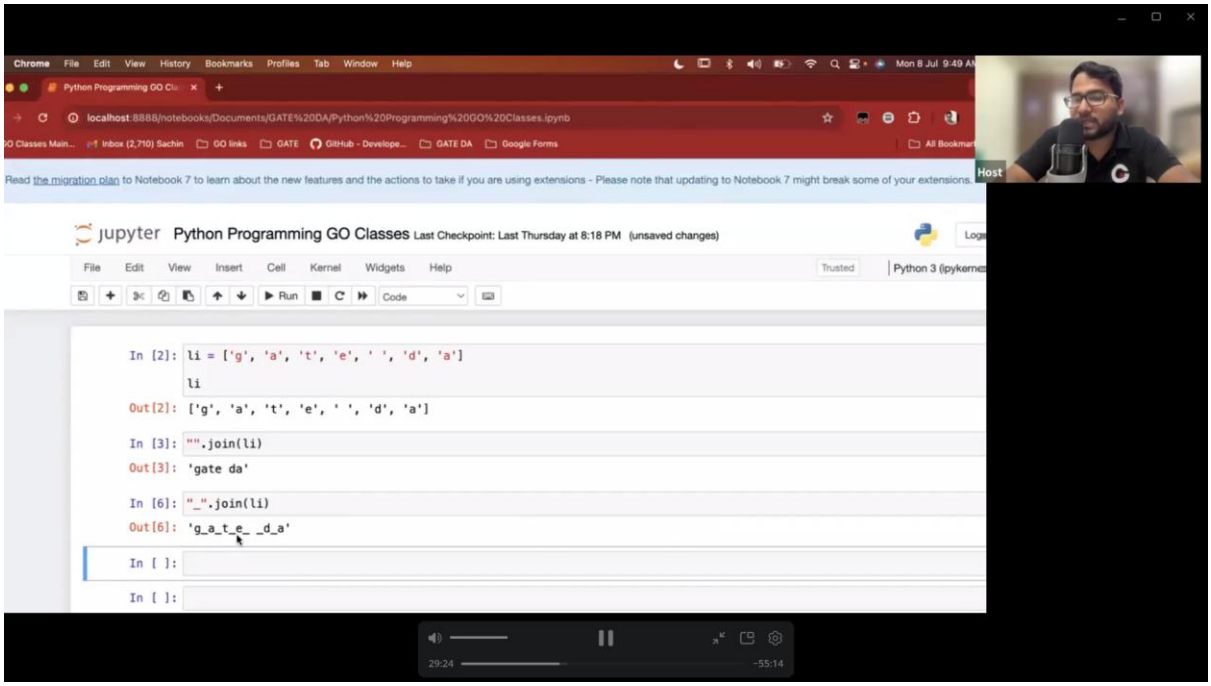


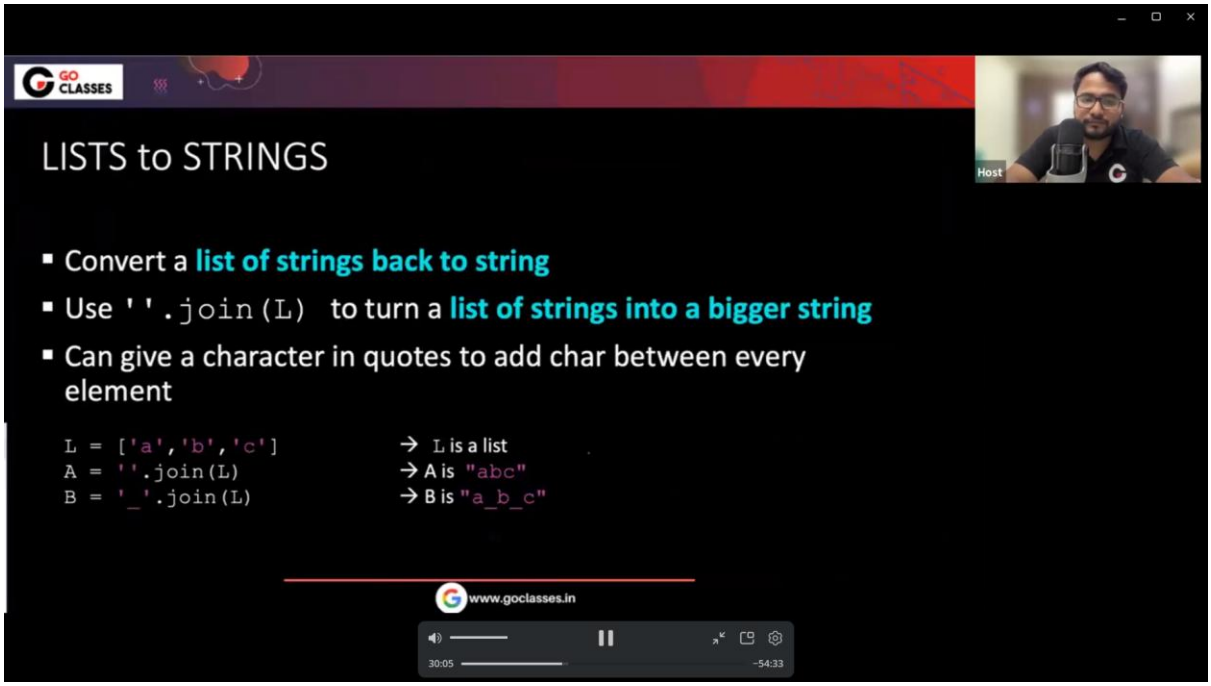

Chrome File Edit View History Bookmarks Profiles Tab Window Help
Mon 8 Jul 9:45 AM
localhost:8888/notebooks/Documents/GATE%20DA/Python%20Programming%20GO%20Classes.ipynb
Read the migration plan to Notebook 7 to learn about the new features and the actions to take if you are using extensions - Please note that updating to Notebook 7 might break some of your extensions.
jupyter Python Programming GO Classes Last Checkpoint: Last Thursday at 8:18 PM (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
Out[19]: 3
In [20]: s = "gate da"
s
Out[20]: 'gate da'
In [23]: li = list(s)
In [24]: li
Out[24]: ['g', 'a', 't', 'e', ' ', 'd', 'a']
In [ ]: str(li)
In [ ]:
Host
25:16
-59:22

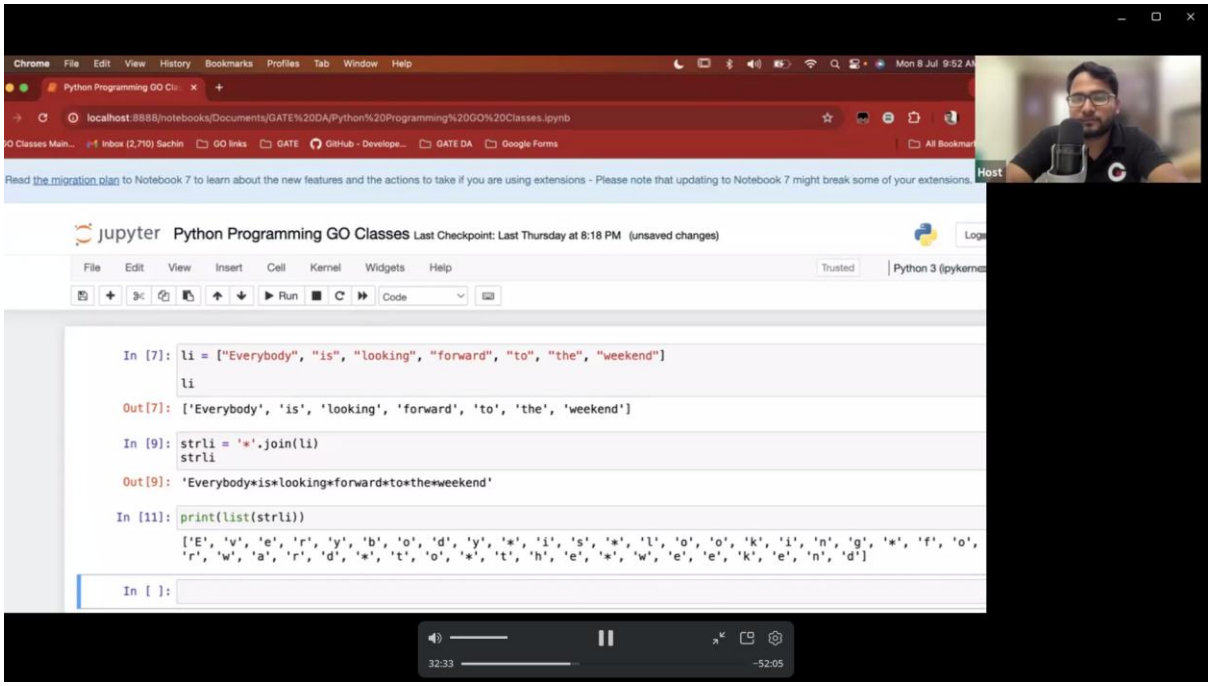

GO CLASSES
list to string
In [20]: s = "gate da"
s
Out[20]: 'gate da'
In [23]: li = list(s)
In [24]: li
Out[24]: ['g', 'a', 't', 'e', ' ', 'd', 'a']
In [25]: str(li)
it doesn't work
Out[25]: "['g', 'a', 't', 'e', ' ', 'd', 'a']"
Host
26:57
-57:41

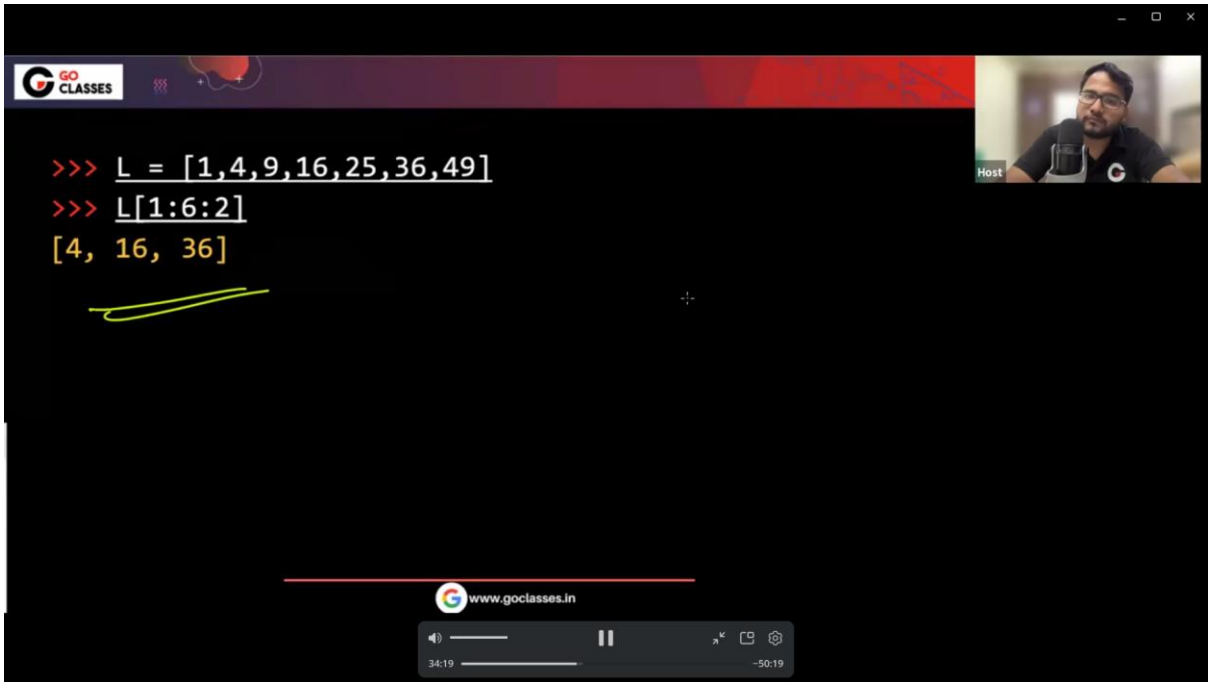

Chrome File Edit View History Bookmarks Profiles Tab Window Help
Mon 8 Jul 9:48 AM
Python Programming GO Cla
localhost:8888/notebooks/Documents/GATE%20DA/Python%20Programming%20GO%20Classes.ipynb
Inbox (2,710) Sachin
GO links
GATE
GitHub - Develope...
GATE DA
Google Forms
Read the migration plan to Notebook 7 to learn about the new features and the actions to take if you are using extensions - Please note that updating to Notebook 7 might break some of your extensions.
jupyter Python Programming GO Classes Last Checkpoint: Last Thursday at 8:18 PM (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
In [2]: li = ['g', 'a', 't', 'e', ' ', 'd', 'a']
li
Out[2]: ['g', 'a', 't', 'e', ' ', 'd', 'a']
In [3]: "".join(li)
Out[3]: 'gate da'
In [ ]:
In [ ]:
In [ ]:
In [ ]:
Host

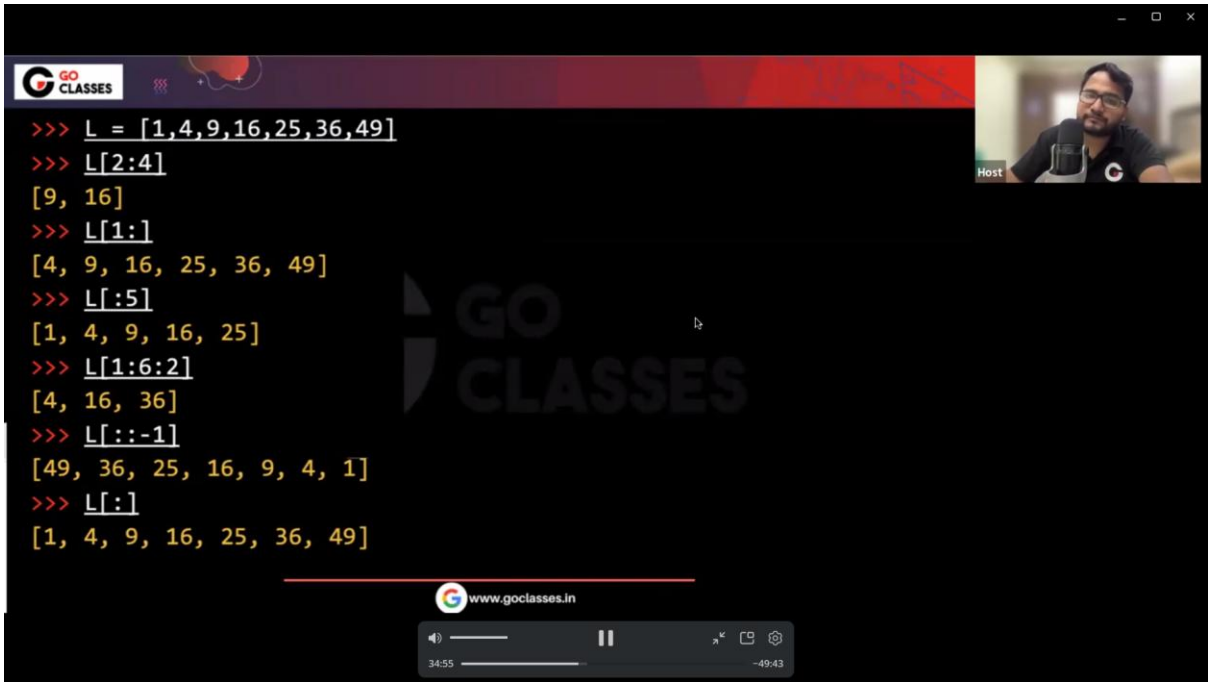

Chrome File Edit View History Bookmarks Profiles Tab Window Help
Mon 8 Jul 9:49 AM
Python Programming GO Cla
localhost:8888/notebooks/Documents/GATE%20DA/Python%20Programming%20GO%20Classes.ipynb
Inbox (2,710) Sachin
GO links
GATE
GitHub - Develope...
GATE DA
Google Forms
Read the migration plan to Notebook 7 to learn about the new features and the actions to take if you are using extensions - Please note that updating to Notebook 7 might break some of your extensions.
jupyter Python Programming GO Classes Last Checkpoint: Last Thursday at 8:18 PM (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
In [2]: li = ['g', 'a', 't', 'e', ' ', 'd', 'a']
li
Out[2]: ['g', 'a', 't', 'e', ' ', 'd', 'a']
In [3]: "".join(li)
Out[3]: 'gate da'
In [5]: " GOCLASSES ".join(li)
Out[5]: 'g GOCLASSES a GOCLASSES t GOCLASSES e GOCLASSES GOCLASSES d GOCLASSES a'
In [ ]:
In [ ]:
29:14
-55:24
Host


Chrome File Edit View History Bookmarks Profiles Tab Window Help
Mon 8 Jul 9:49 AM
Python Programming GO Cl
localhost:8888/notebooks/Documents/GATE%20DA/Python%20Programming%20GO%20Classes.ipynb
GO Classes Main... Inbox (2,710) Sachin GO links GATE GitHub - Develope... GATE DA Google Forms
All Bookmarks
Read the migration plan to Notebook 7 to learn about the new features and the actions to take if you are using extensions - Please note that updating to Notebook 7 might break some of your extensions.
jupyter Python Programming GO Classes Last Checkpoint: Last Thursday at 8:18 PM (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
In [2]: li = ['g', 'a', 't', 'e', ' ', 'd', 'a']
li
Out[2]: ['g', 'a', 't', 'e', ' ', 'd', 'a']
In [3]: "".join(li)
Out[3]: 'gate da'
In [6]: "_".join(li)
Out[6]: 'g_a_t_e_ _d_a'
In [ ]:
In [ ]:
Host
29:24
-55:14


GO CLASSES
LISTS to STRINGS
Host
Convert a list of strings back to string
Use ''.join(L) to turn a list of strings into a bigger string
Can give a character in quotes to add char between every element
L = ['a','b','c'] → L is a list
A = ''.join(L) → A is "abc"
B = '_'.join(L) → B is "a_b_c"
www.goclasses.in
30:05
-54:33


Chrome File Edit View History Bookmarks Profiles Tab Window Help
Python Programming GO Cl
localhost:8888/notebooks/Documents/GATE%20DA/Python%20Programming%20GO%20Classes.ipynb
Inbox (2,710) Sachin
GO links
GATE
GitHub - Develope...
GATE DA
Google Forms
Read the migration plan to Notebook 7 to learn about the new features and the actions to take if you are using extensions - Please note that updating to Notebook 7 might break some of your extensions.
jupyter Python Programming GO Classes Last Checkpoint: Last Thursday at 8:18 PM (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
In [7]: li = ["Everybody", "is", "looking", "forward", "to", "the", "weekend"]
li
Out[7]: ['Everybody', 'is', 'looking', 'forward', 'to', 'the', 'weekend']
In [9]: strli = '*'.join(li)
strli
Out[9]: 'Everybody*is*looking*forward*to*the*weekend'
In [11]: print(list(strli))
['E', 'v', 'e', 'r', 'y', 'b', 'o', 'd', 'y', '*', 'i', 's', '*', 'l', 'o', 'o', 'k', 'i', 'n', 'g', '*', 'f', 'o', 'r', 'w', 'a', 'r', 'd', '*', 't', 'o', '*', 't', 'h', 'e', '*', 'w', 'e', 'e', 'k', 'e', 'n', 'd']
In [ ]:
Host
32:33
-52:05


GO CLASSES
>>> L = [1,4,9,16,25,36,49]
>>> L[1:6:2]
[4, 16, 36]
Host
www.goclasses.in
34:19
-50:19


GO CLASSES
>>> L = [1,4,9,16,25,36,49]
>>> L[2:4]
[9, 16]
>>> L[1:]
[4, 9, 16, 25, 36, 49]
>>> L[:5]
[1, 4, 9, 16, 25]
>>> L[1:6:2]
[4, 16, 36]
>>> L[::-1]
[49, 36, 25, 16, 9, 4, 1]
>>> L[:]
[1, 4, 9, 16, 25, 36, 49]
Host
GO CLASSES
www.goclasses.in
34:55
-49:43

Similarly, `step` may be a negative number:

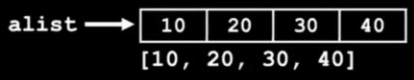
```
a = [1,2,3,4,5,6,7,8,9,10]

a[::-1]    # all items in the array, reversed
a[1::-1]   # the first two items, reversed
a[:-3:-1]  # the last two items, reversed
a[-3::-1]  # everything except the last two items, reversed
```


35:07 -49:31

append() method

- Can add elements to end of list with `.append`

```
alist = [10, 20, 30]
alist.append(40)
```



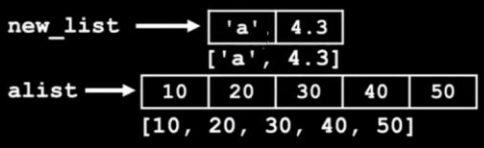
[10, 20, 30, 40]

36:02 -48:36

append() method

- Can add elements to end of list with `.append`

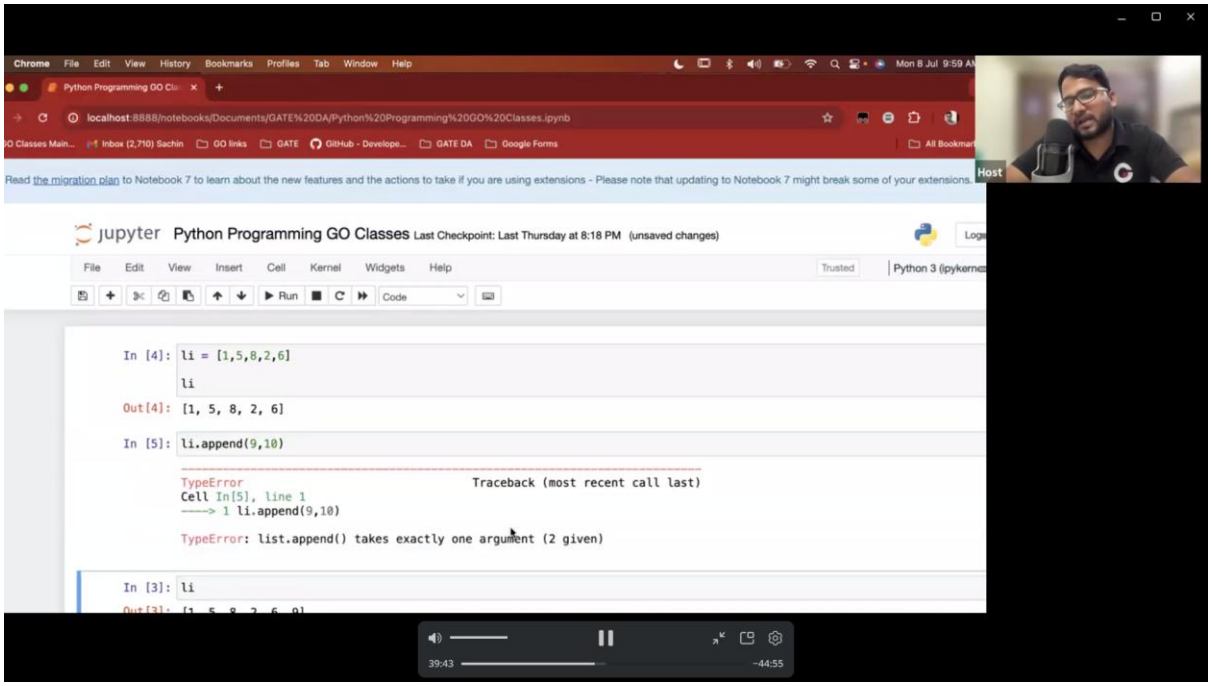
```
alist = [10, 20, 30]
alist.append(40)
alist.append(50)
new_list = []
new_list.append('a')
new_list.append(4.3)
```



extend() method

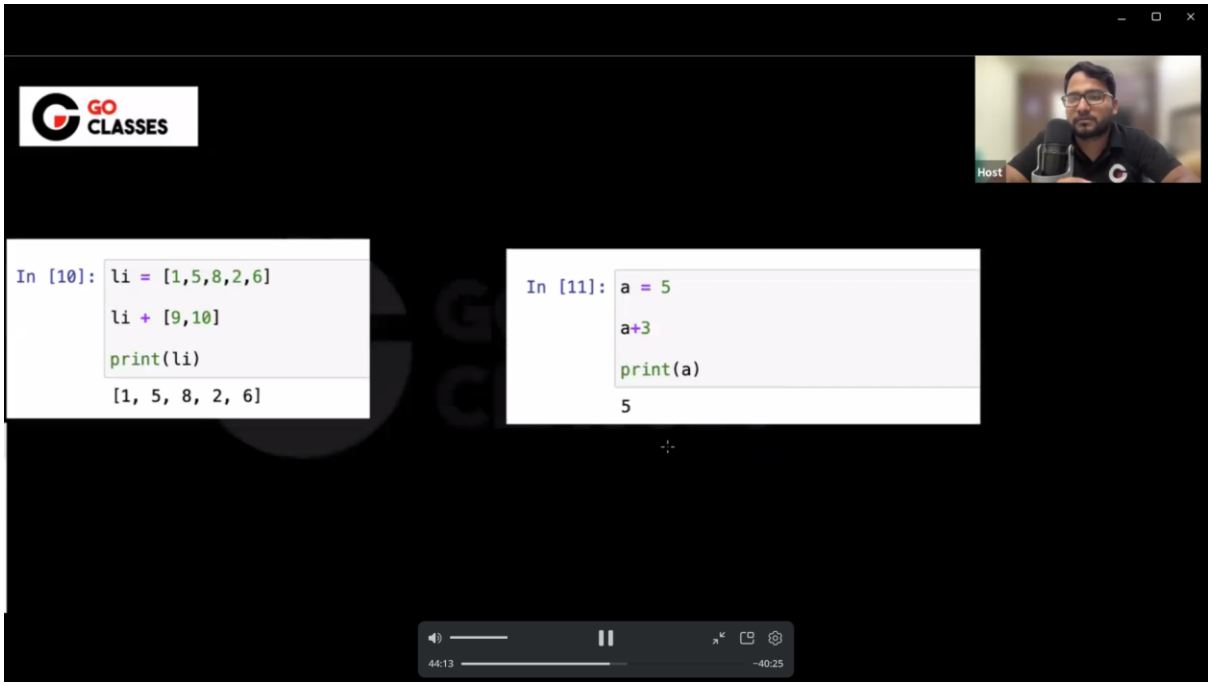
- Function: *list*`.extend(other_list)`
  - Adds all element from other list to list that function is called on

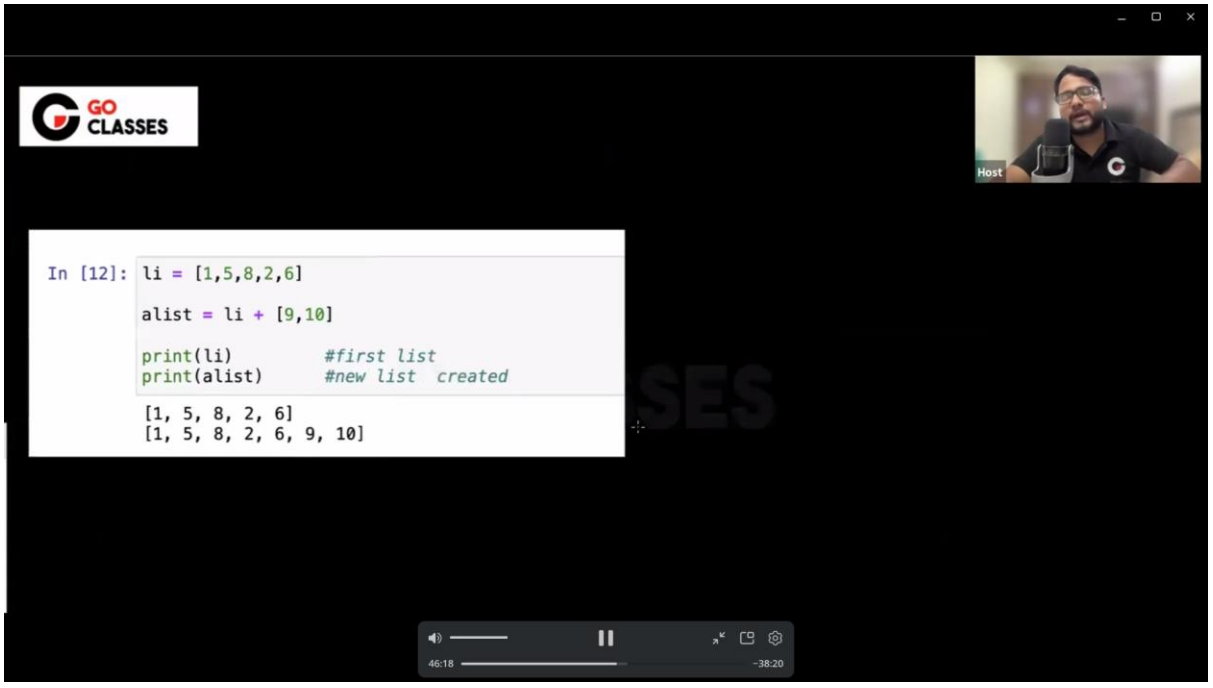
```
>>> list1 = [1, 2, 3]
>>> list2 = [4, 5]
>>> list1.extend(list2)
>>> list1
[1, 2, 3, 4, 5]
```

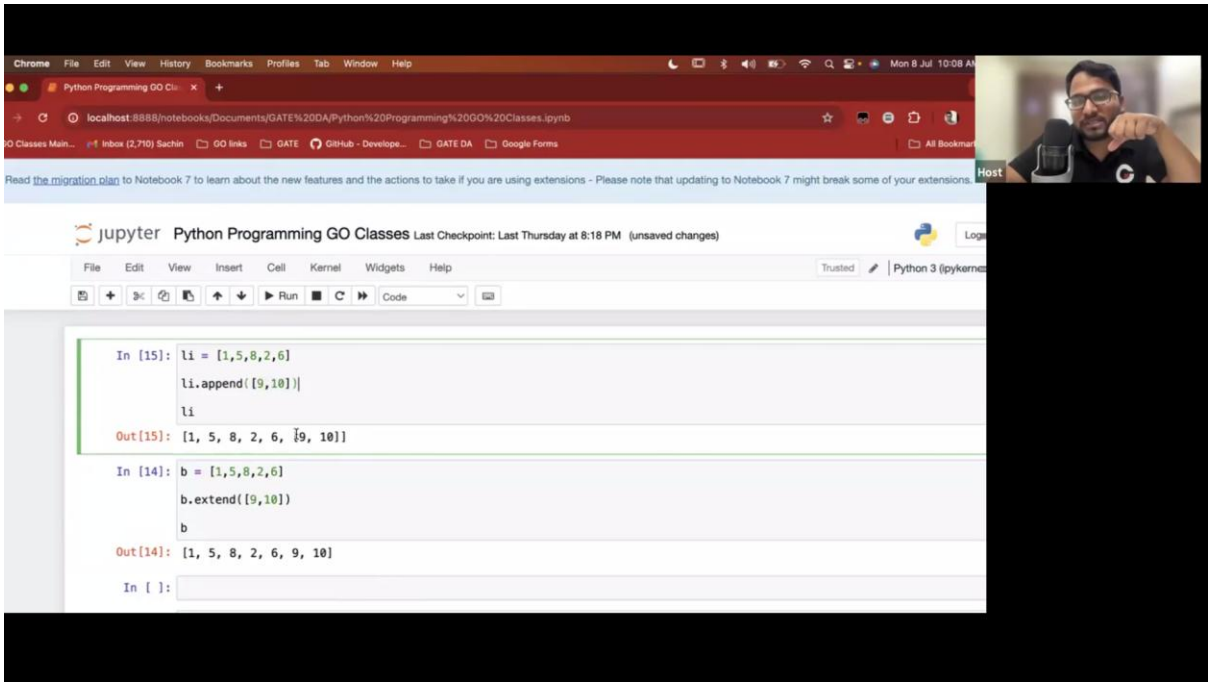


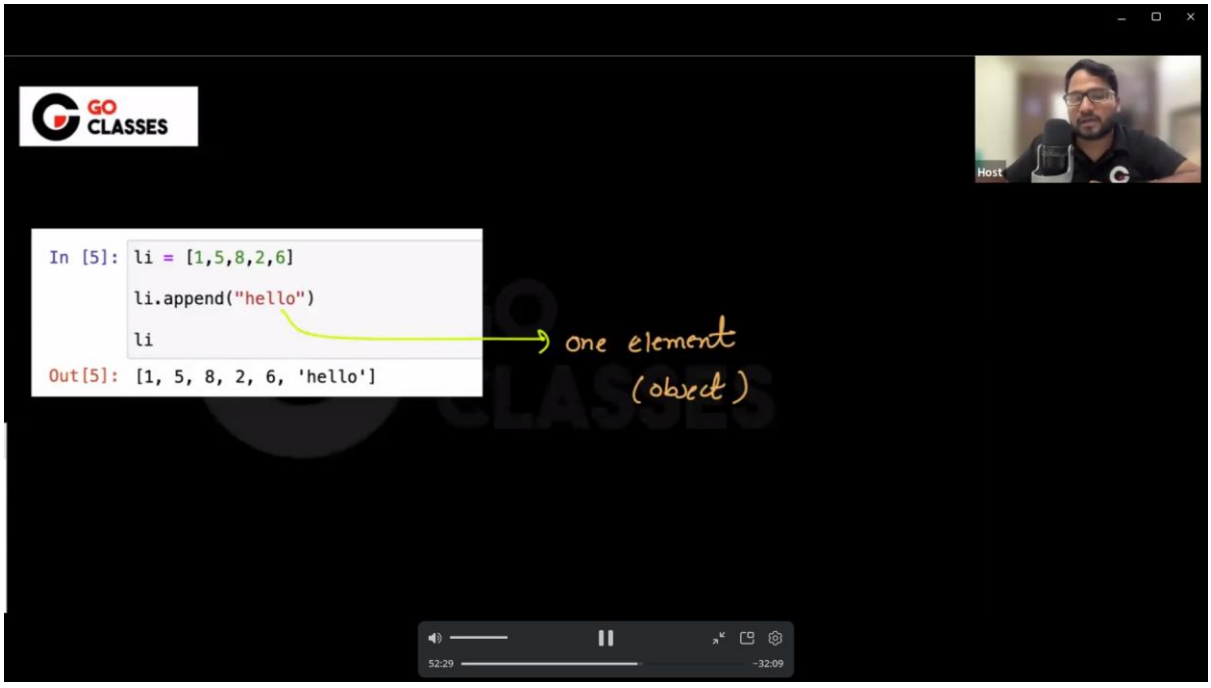
Summary of the difference:

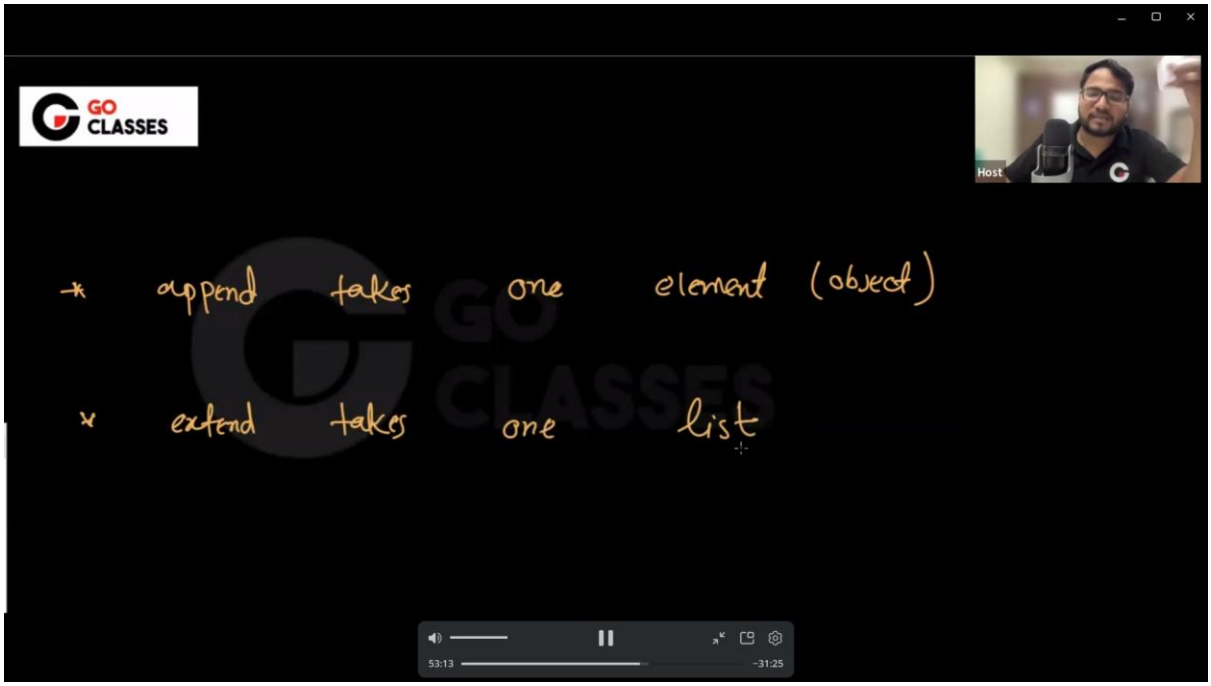
Code	Action	Result
`my_list.extend([4])`	Adds each *element* of the provided list `[4]` to `my_list`.	If `my_list` was `[1, 2, 3]`, it becomes `[1, 2, 3, 4]`.
`my_list.extend(4)`	Attempts to iterate over the integer `4`.	Raises a `TypeError` because an integer is not iterable.
`my_list.append(4)`	Adds the entire integer `4` as a *single element* to `my_list`.	If `my_list` was `[1, 2, 3]`, it becomes `[1, 2, 3, 4]`.

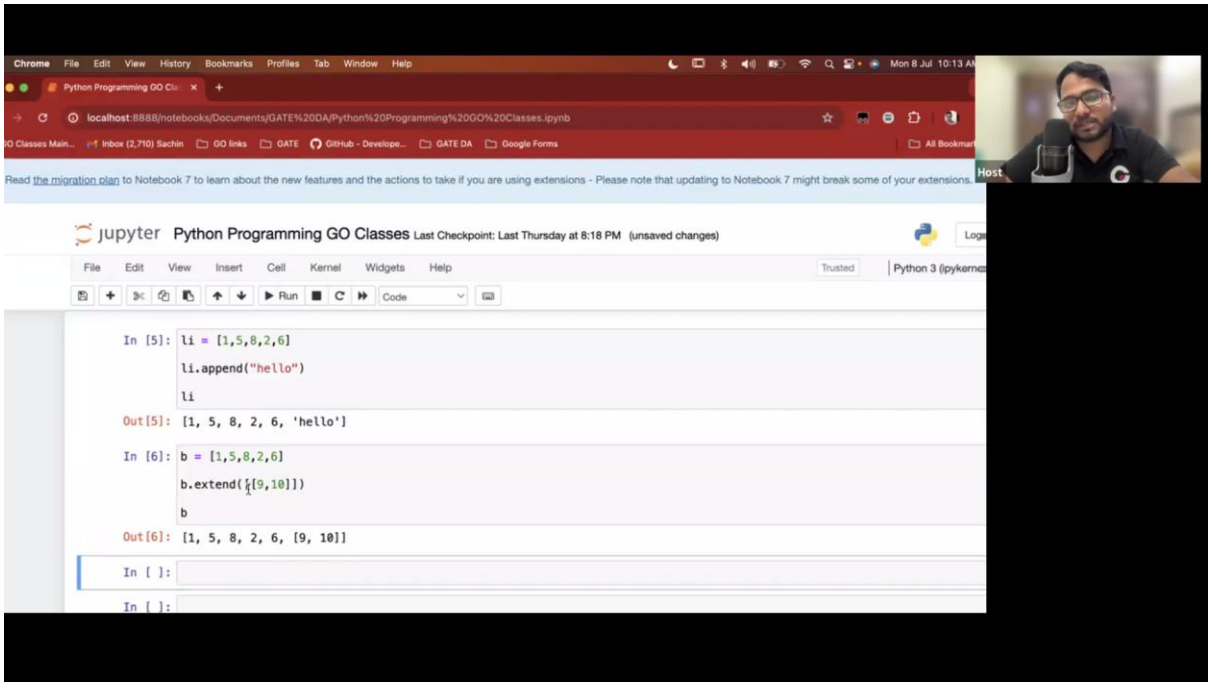

GO CLASSES
Host
In [10]:
li = [1,5,8,2,6]
li + [9,10]
print(li)
[1, 5, 8, 2, 6]
In [11]:
a = 5
a+3
print(a)
5
44:13
-40:25

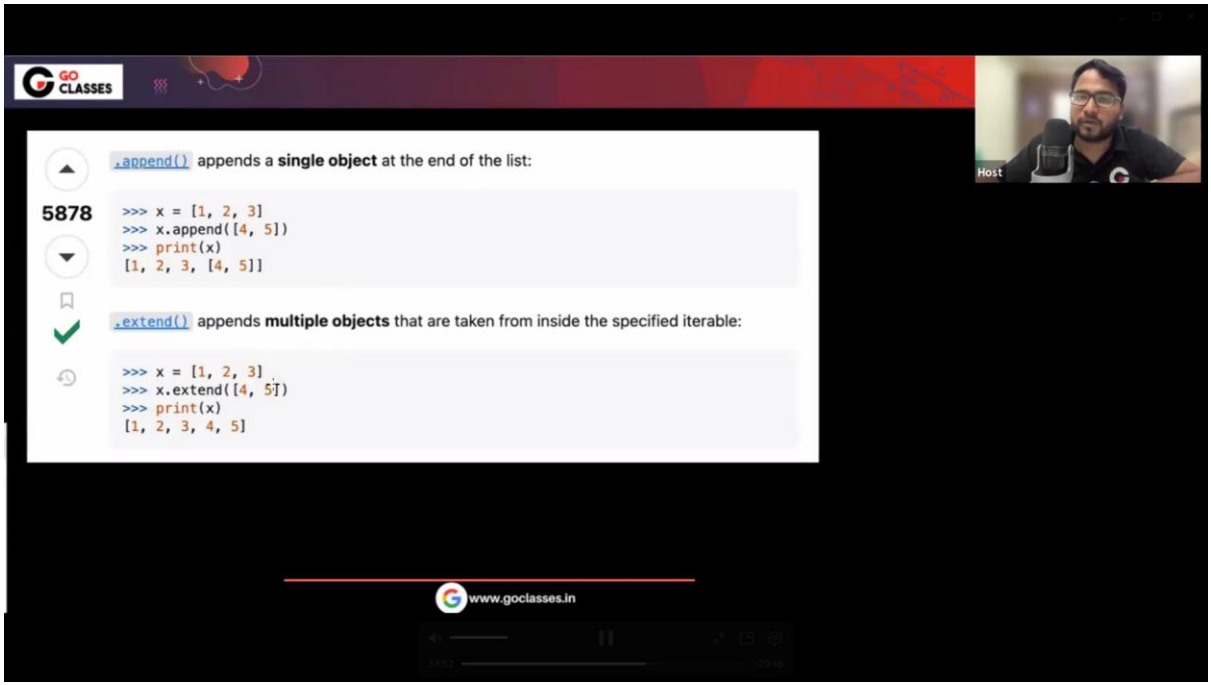

GO CLASSES
Host
In [12]:
li = [1,5,8,2,6]
alist = li + [9,10]
print(li) #first list
print(alist) #new list created
[1, 5, 8, 2, 6]
[1, 5, 8, 2, 6, 9, 10]
46:18
-38:20


Chrome File Edit View History Bookmarks Profiles Tab Window Help
Mon 8 Jul 10:08 AM
localhost:8888/notebooks/Documents/GATE%20DA/Python%20Programming%20GO%20Classes.ipynb
Read the migration plan to Notebook 7 to learn about the new features and the actions to take if you are using extensions - Please note that updating to Notebook 7 might break some of your extensions.
jupyter Python Programming GO Classes Last Checkpoint: Last Thursday at 8:18 PM (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
In [15]: li = [1,5,8,2,6]
li.append([9,10])
li
Out[15]: [1, 5, 8, 2, 6, [9, 10]]
In [14]: b = [1,5,8,2,6]
b.extend([9,10])
b
Out[14]: [1, 5, 8, 2, 6, 9, 10]
In [ ]:
Host


GO CLASSES
Host
In [5]: li = [1,5,8,2,6]
li.append("hello")
li
Out[5]: [1, 5, 8, 2, 6, 'hello']
one element
(object)
52:29
-32:09


GO CLASSES
Host
append takes one element (object)
extend takes one list
53:13
-31:25


Chrome File Edit View History Bookmarks Profiles Tab Window Help
Mon 8 Jul 10:13 AM
localhost:8888/notebooks/Documents/GATE%20DA/Python%20Programming%20GO%20Classes.ipynb
Inbox (2,710) Sachin
GO links
GATE
GitHub - Develope...
GATE DA
Google Forms
Read the migration plan to Notebook 7 to learn about the new features and the actions to take if you are using extensions - Please note that updating to Notebook 7 might break some of your extensions.
jupyter Python Programming GO Classes Last Checkpoint: Last Thursday at 8:18 PM (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
In [5]:
li = [1,5,8,2,6]
li.append("hello")
li
Out[5]: [1, 5, 8, 2, 6, 'hello']
In [6]:
b = [1,5,8,2,6]
b.extend([9,10])
b
Out[6]: [1, 5, 8, 2, 6, [9, 10]]
In [ ]:
In [ ]:
Host

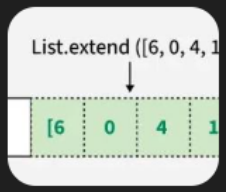

GO CLASSES
Host
5878
.append() appends a single object at the end of the list:
>>> x = [1, 2, 3]
>>> x.append([4, 5])
>>> print(x)
[1, 2, 3, [4, 5]]
.extend() appends multiple objects that are taken from inside the specified iterable:
>>> x = [1, 2, 3]
>>> x.extend([4, 5])
>>> print(x)
[1, 2, 3, 4, 5]
www.goclasses.in

li=[1,2] li.extend([3,4],[5,6]) print(li)

AI Overview

En Listen

The code will raise a `TypeError` because the `extend()` method can only take a single iterable as an argument, not multiple separate iterables like `[3,4]` and `[5,6]`. To achieve the desired output of `[1, 2, 3, 4, 5, 6]`, you should use two separate `extend()` calls or pass a single list containing both sub-lists, like `li.extend([3, 4, 5, 6])`.



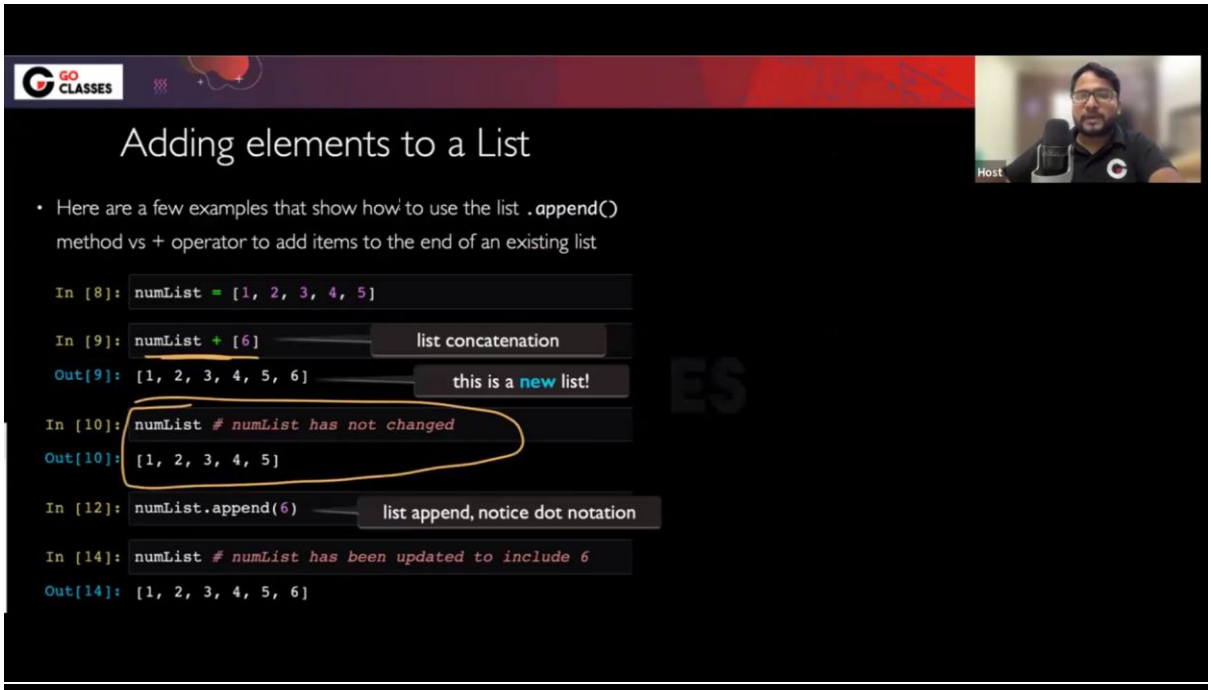
Corrected code

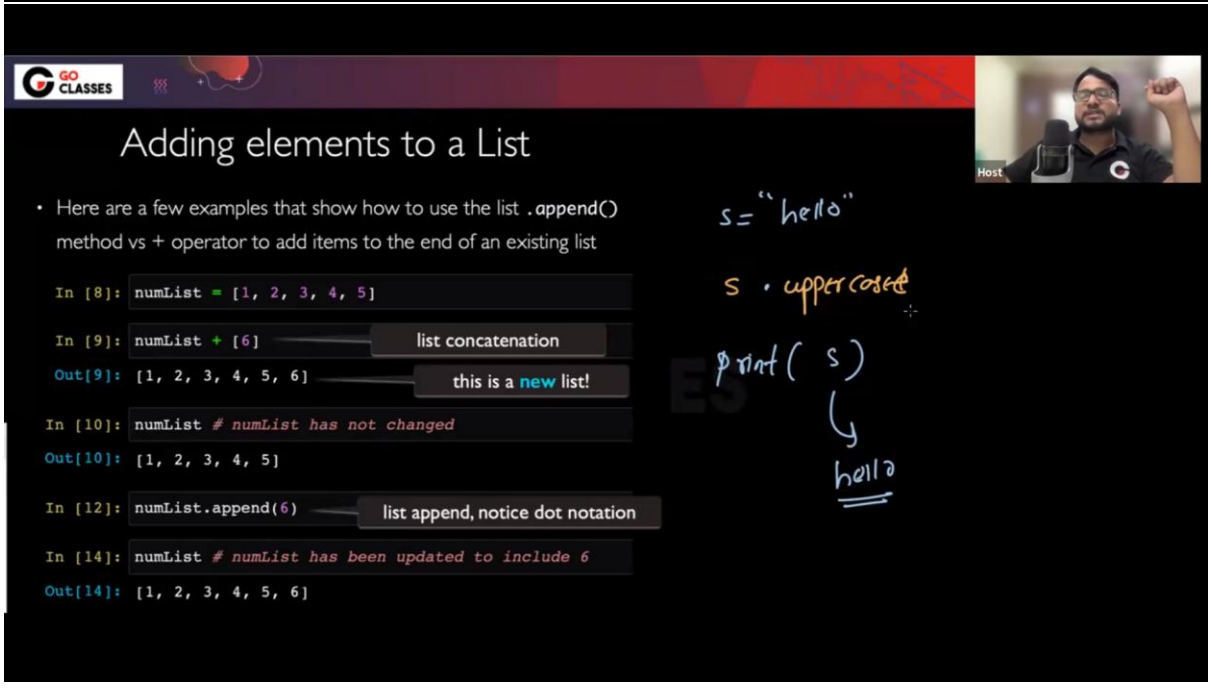
python

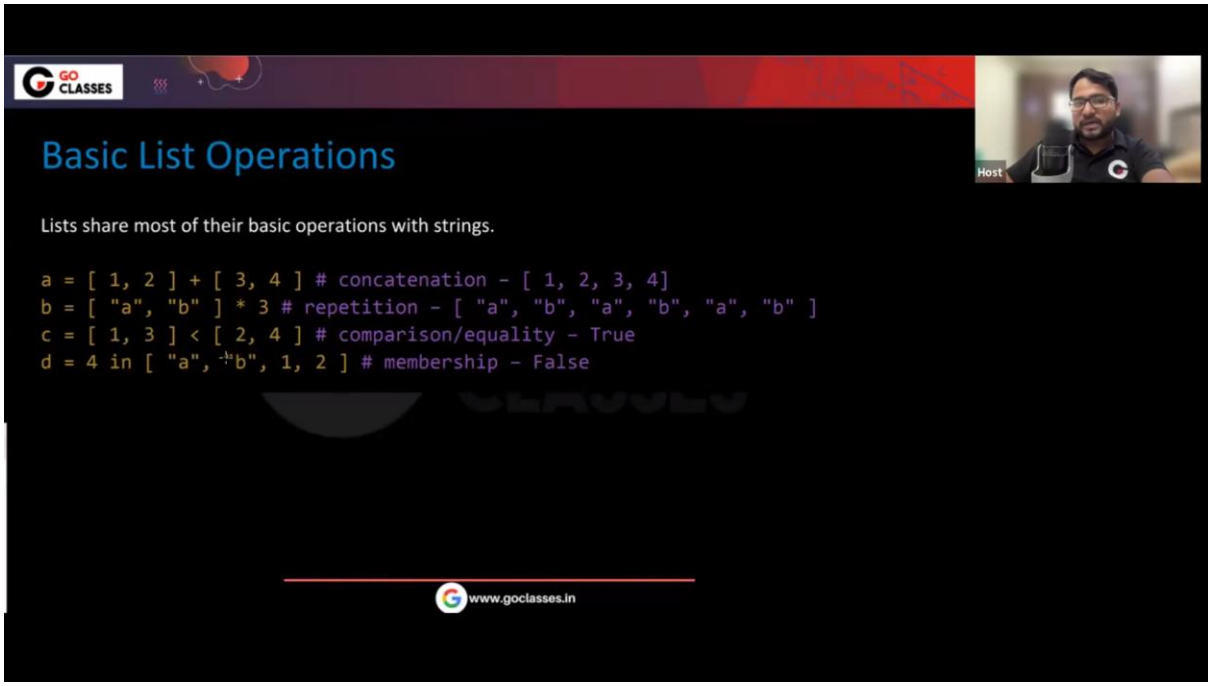
```
li = [1, 2]
li.extend([3, 4])
li.extend([5, 6])
print(li)
```

Output

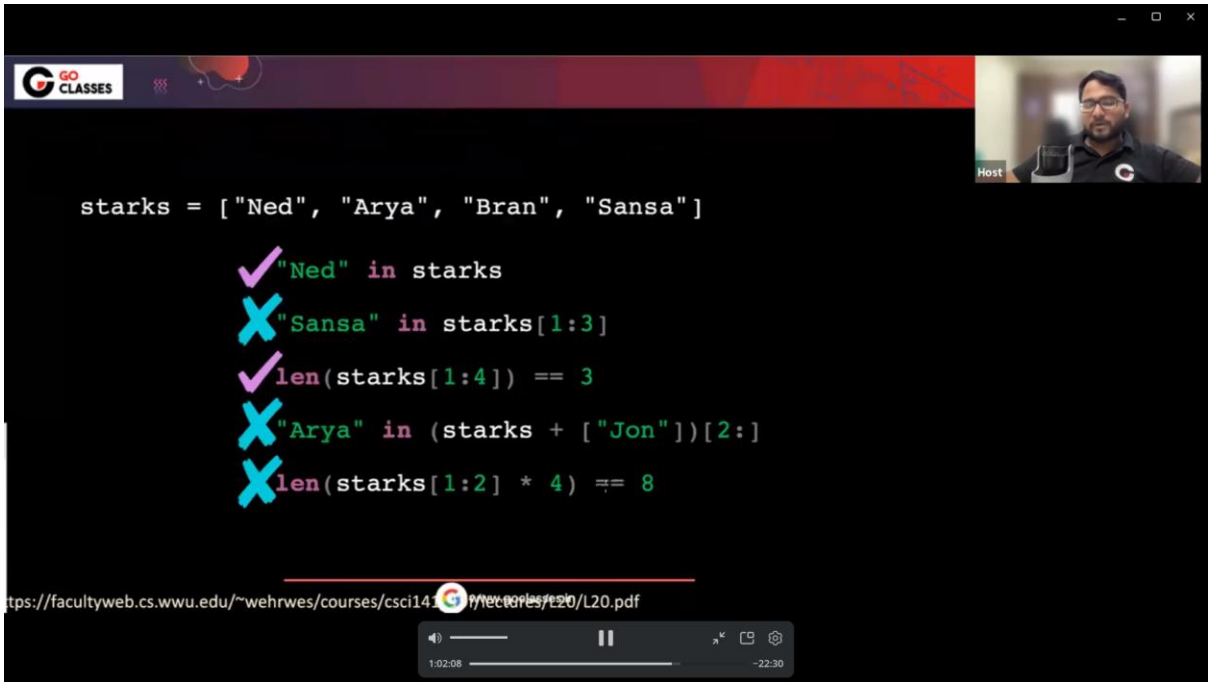
```
[1, 2, 3, 4, 5, 6]
```


GO CLASSES
Adding elements to a List
Host
• Here are a few examples that show how to use the list .append() method vs + operator to add items to the end of an existing list
In [8]: numList = [1, 2, 3, 4, 5]
In [9]: numList + [6]
list concatenation
Out[9]: [1, 2, 3, 4, 5, 6]
this is a new list!
In [10]: numList # numList has not changed
Out[10]: [1, 2, 3, 4, 5]
In [12]: numList.append(6)
list append, notice dot notation
In [14]: numList # numList has been updated to include 6
Out[14]: [1, 2, 3, 4, 5, 6]


GO CLASSES
Adding elements to a List
Host
• Here are a few examples that show how to use the list .append() method vs + operator to add items to the end of an existing list
In [8]: numList = [1, 2, 3, 4, 5]
In [9]: numList + [6]
list concatenation
Out[9]: [1, 2, 3, 4, 5, 6]
this is a new list!
In [10]: numList # numList has not changed
Out[10]: [1, 2, 3, 4, 5]
In [12]: numList.append(6)
list append, notice dot notation
In [14]: numList # numList has been updated to include 6
Out[14]: [1, 2, 3, 4, 5, 6]
s = "hello"
s . uppercase()
print (s)
hello


GO CLASSES
Basic List Operations
Host
Lists share most of their basic operations with strings.
a = [1, 2] + [3, 4] # concatenation - [1, 2, 3, 4]
b = ["a", "b"] * 3 # repetition - ["a", "b", "a", "b", "a", "b"]
c = [1, 3] < [2, 4] # comparison/equality - True
d = 4 in ["a", "b", 1, 2] # membership - False
www.goclasses.in


GO CLASSES
Host
Out[7]: [1, 5, 8, 2, 6, 7, [9, 10]]
In [8]: [1,3] < [2,4]
Out[8]: True
In [9]: [1,3] < [2,2]
Out[9]: True
In [10]: [1,2,3,4] < [1,2,5,0]
Out[10]: True
59:52
-24:46



Yes, `len(st[1:4]) == 3` evaluates to `True` [1].

Here is a breakdown of why:

1. `st` is the list `['ab', 'bcc', 'dddd', 'ghj']`.
2. The slice `st[1:4]` takes elements starting from index 1 (inclusive) up to, but not including, index 4.
3. The elements selected are:
    1. Index 1: `'bcc'`
    2. Index 2: `'dddd'`
    3. Index 3: `'ghj'`
4. The resulting sublist is `['bcc', 'dddd', 'ghj']`.
5. The length of this sublist is 3.

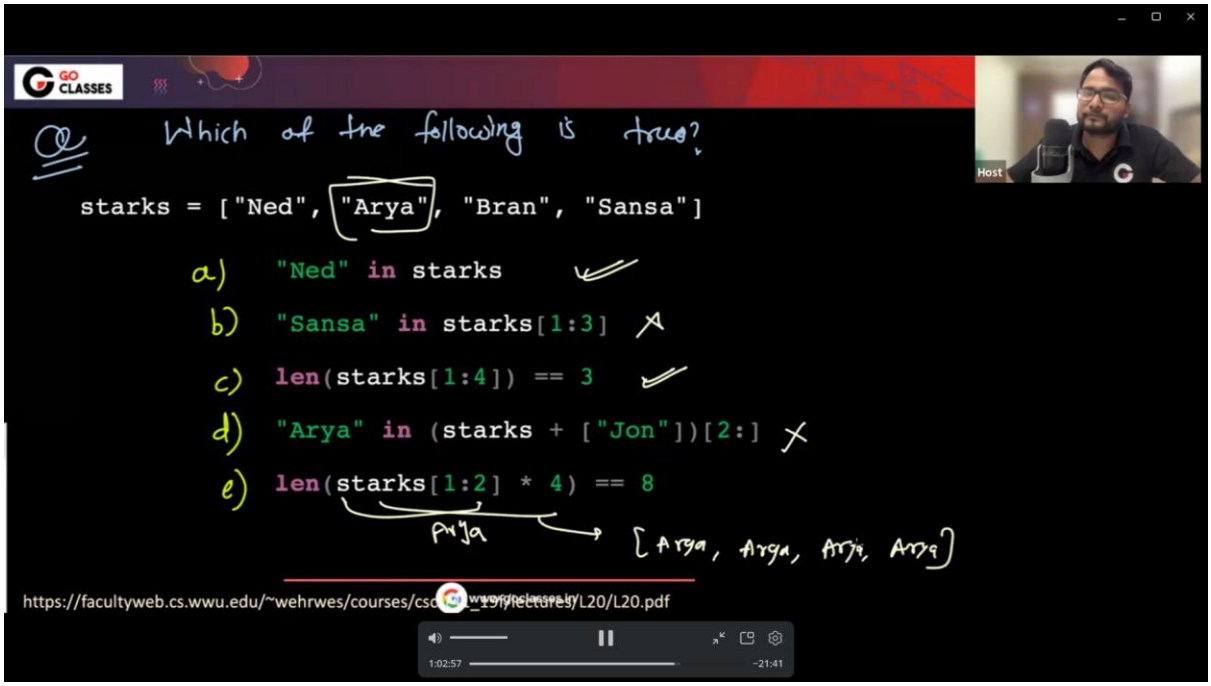
Therefore, `len(st[1:4])` is 3, and the comparison `3 == 3` is `True` [1].

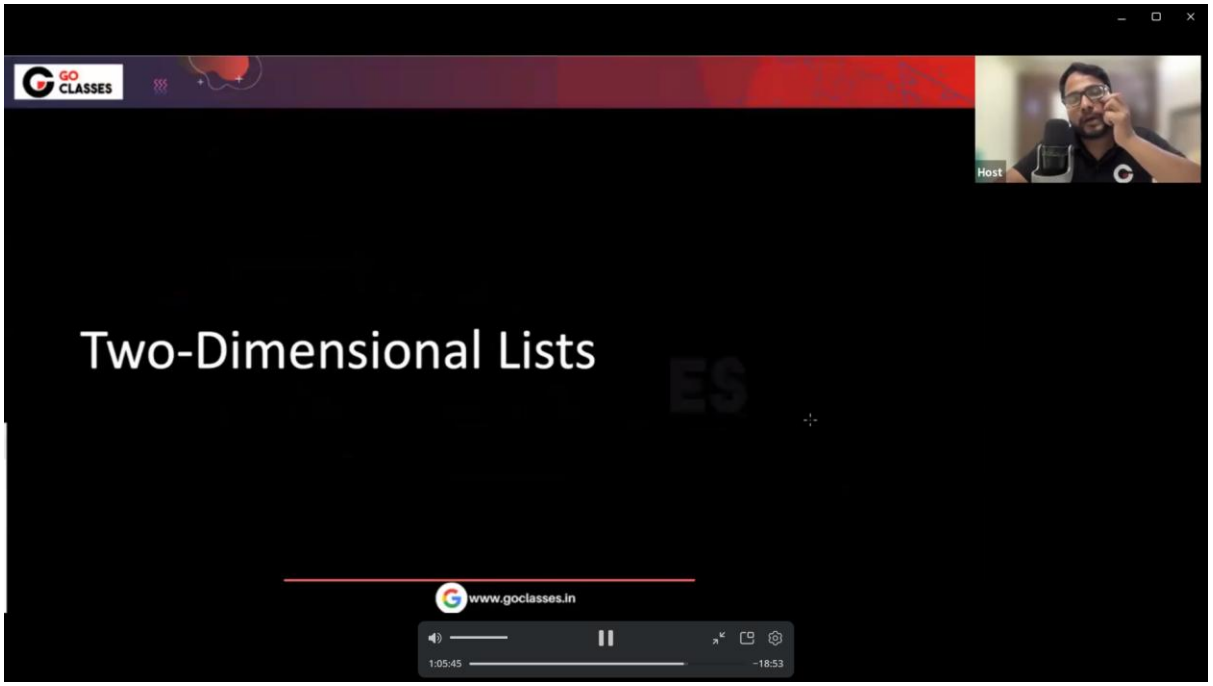
No, `len(st[1:2]*4)` does not equal 10. The result is **4**.

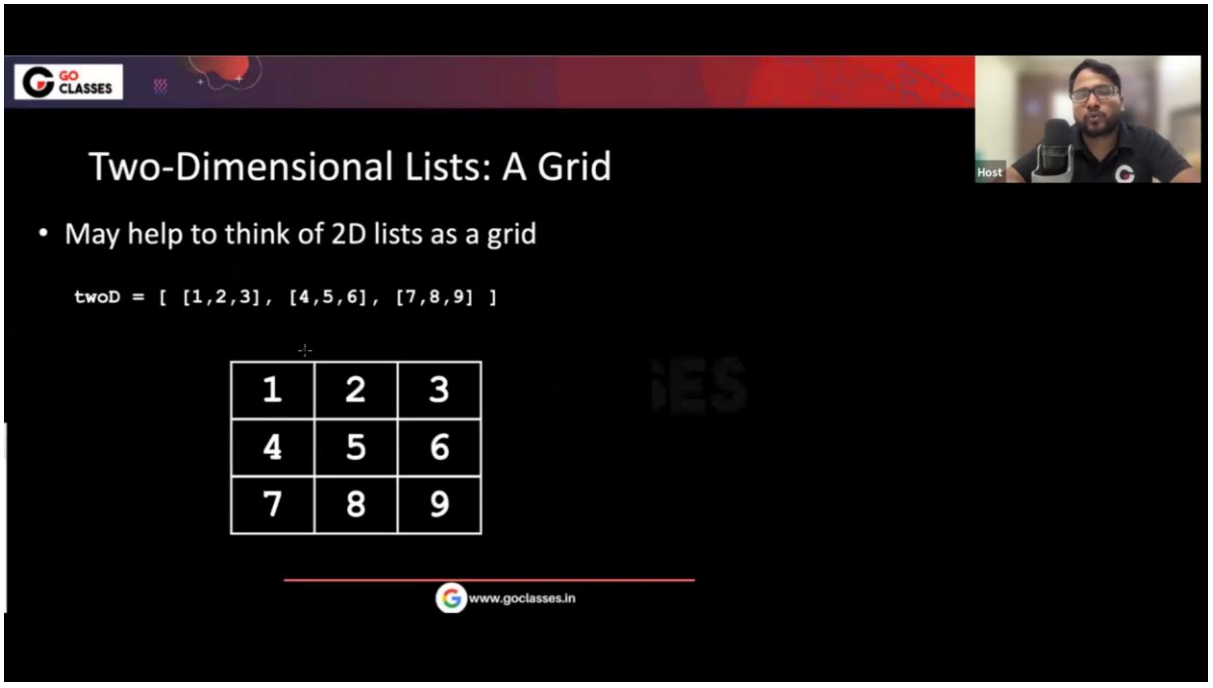
Here is the breakdown:

1. **`st[1:2]`** : This is a list slice that extracts the element at index 1 (which is `'bcc'`) into a new list. The result is `['bcc']`. Note that slicing a list, even for a single element, returns a list, not the element itself.
2. **`st[1:2] * 4`** : This repeats the list `['bcc']` four times through concatenation. The result is `['bcc', 'bcc', 'bcc', 'bcc']`.
3. **`len(['bcc', 'bcc', 'bcc', 'bcc'])`** : The `len()` function returns the number of items in the list, which is 4.

Therefore, `len(st[1:2]*4)` is 4, which is not equal to 10.




GO CLASSES
Host
Two-Dimensional Lists
www.goclasses.in
1:05:45
-18:53


GO CLASSES
Host
Two-Dimensional Lists: A Grid
• May help to think of 2D lists as a grid
twoD = [[1,2,3], [4,5,6], [7,8,9]]
1 2 3
4 5 6
7 8 9
www.goclasses.in

GO CLASSES

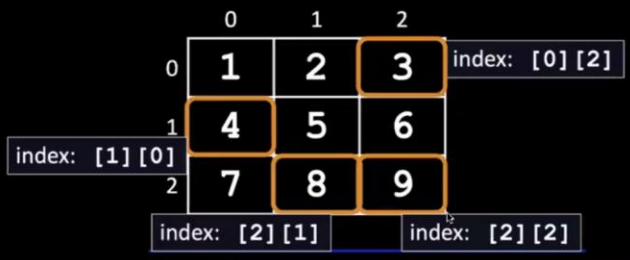

```
In [11]: li = [[1,2,3], [4,5,6], [7,8,9]] #list of lists
         li
Out[11]: [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

In [12]: li[-1][1:]
Out[12]: [8, 9]
```

1:07:07 -17:31

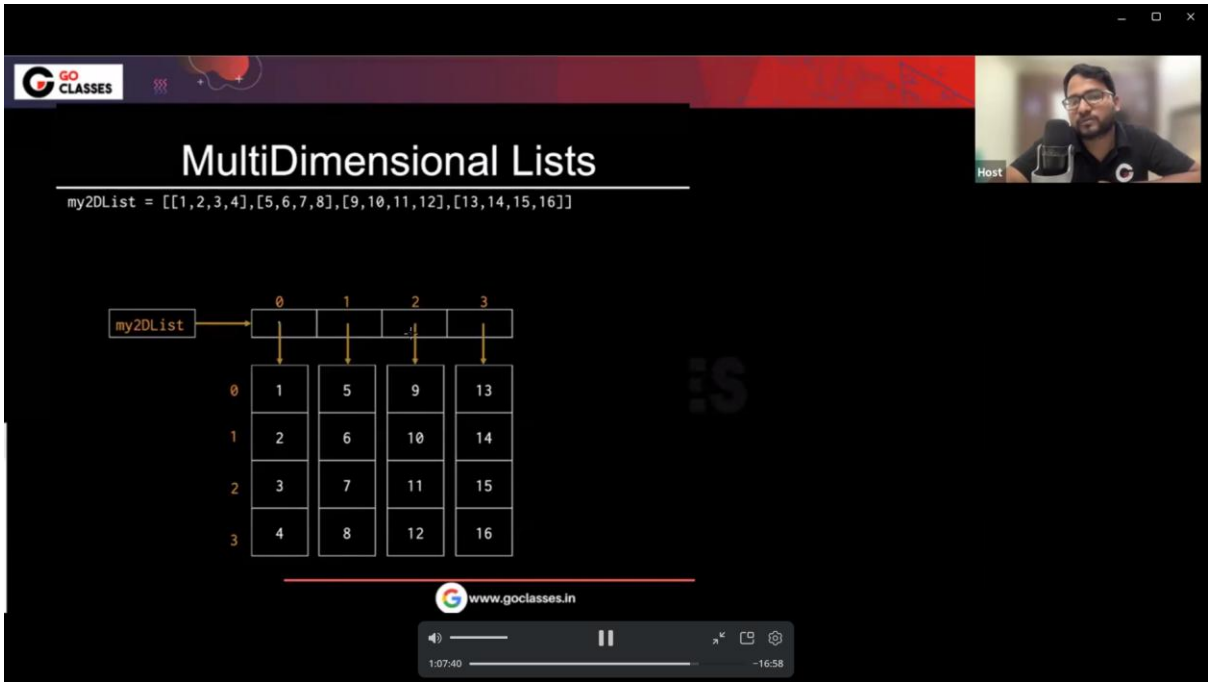
GO CLASSES

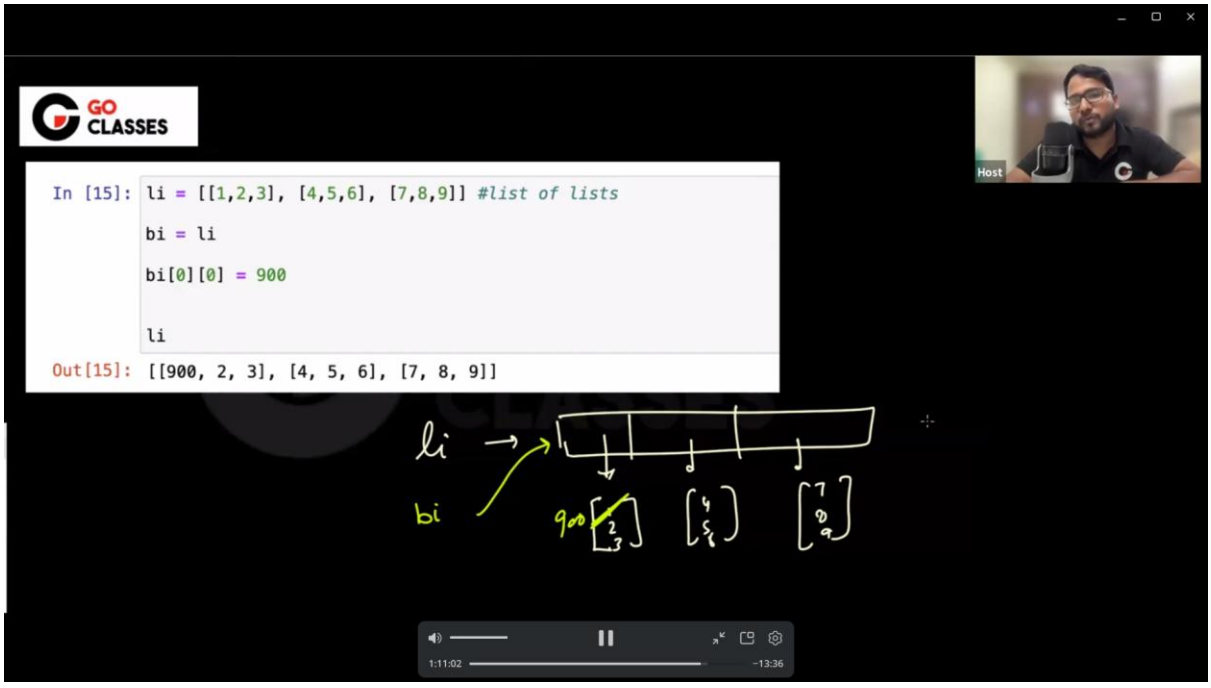
Two-Dimensional Lists: A Grid

- You access an element by the index of its row, then the column
  - Remember – indexing starts at 0!

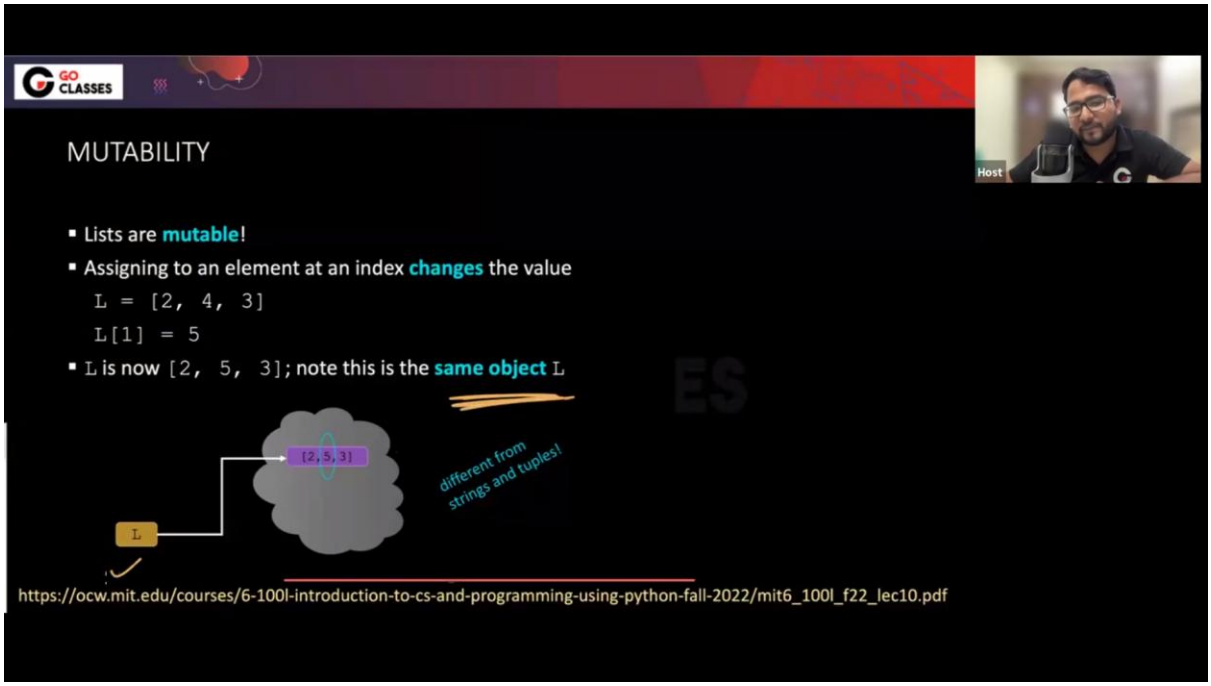


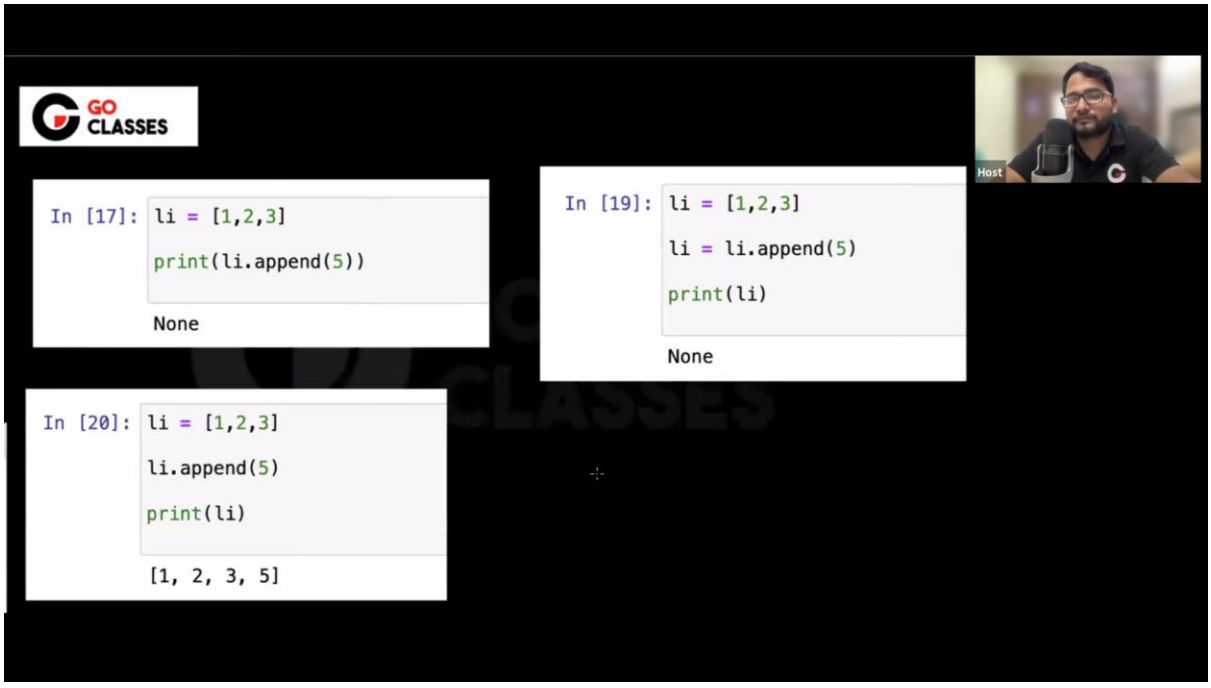
www.goclasses.in





Shallow copy





Because of '=' symbol

GO CLASSES

BIG IDEA

Some functions mutate the list and don't return anything.

https://ocw.mit.edu/courses/6-100l-introduction-to-cs-and-programming-using-python-fall-2022/mit6_100l_f22_lec10.pdf

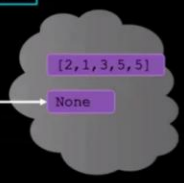
GO CLASSES

OPERATION ON LISTS – append

- **Add** an element to end of list with `L.append(element)`
- **Mutates** the list!

```
L = [2,1,3]
L.append(5)      → L is now [2,1,3,5]
L = L.append(5)
```

Be careful! The append operation does a mutation, but returns the None object as a result.



https://ocw.mit.edu/courses/6-100l-introduction-to-cs-and-programming-using-python-fall-2022/mit6_100l_f22_lec10.pdf

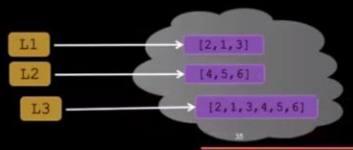
COMBINING LISTS

Remember strings

- **Concatenation**, + operator, creates a **new** list, with copies
- **Mutate** list with `L.extend(some_list)` (copy of `some_list`)

```
L1 = [2,1,3]
L2 = [4,5,6]
L3 = L1 + L2
```

→ L3 is [2,1,3,4,5,6]



concatenation creates **new** list with copies

1:19:45 -04:53

List assignment + slicing: Demo

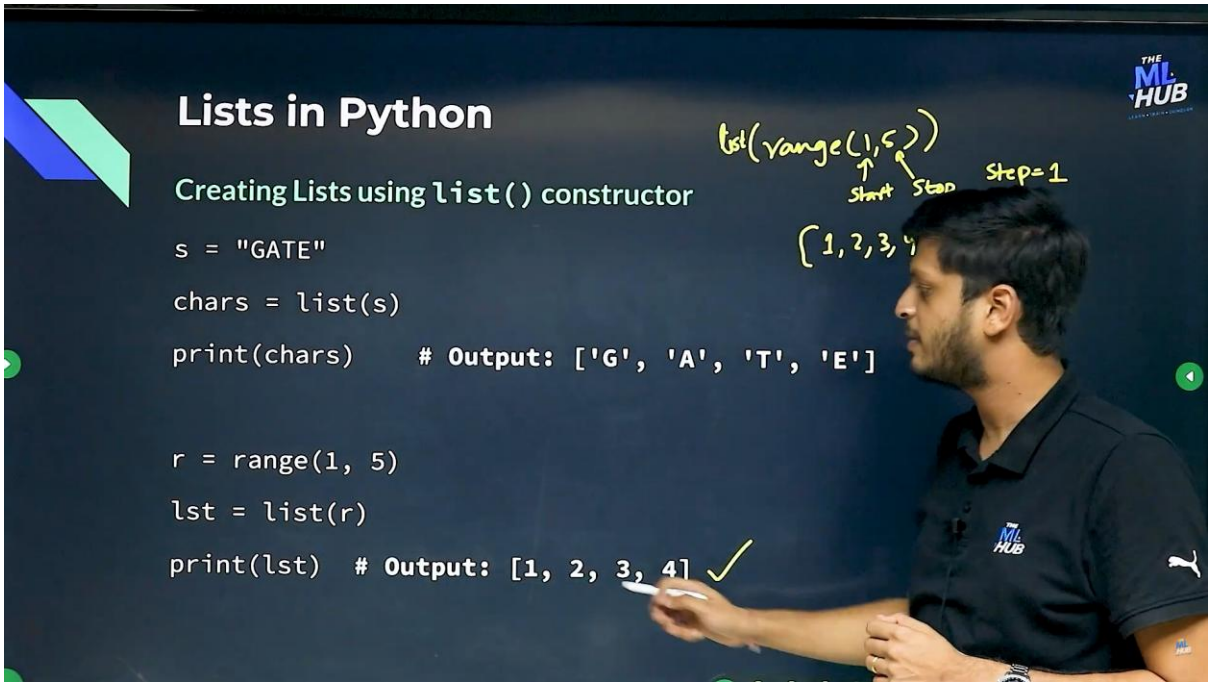
```
a = [5, 6, 7, 8]
a[:2] = [3, 4]
```

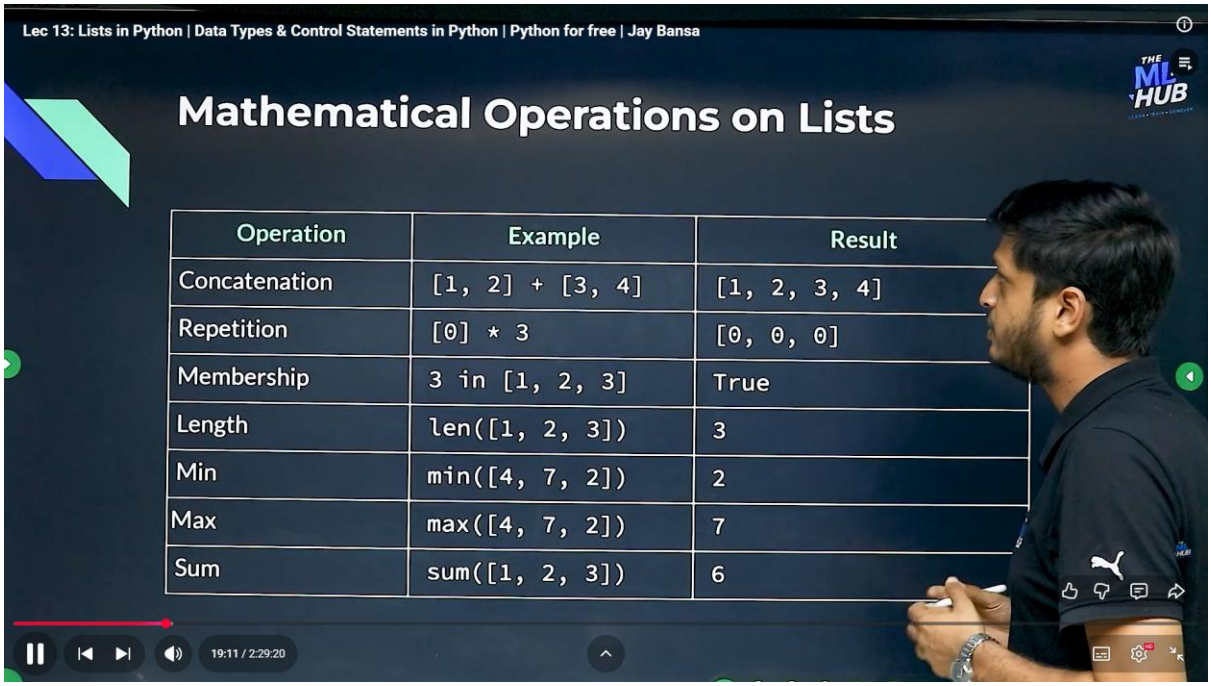
← No one use these things

```
a = [5, 6, 7, 8]
a[:3] = a[1:]
```

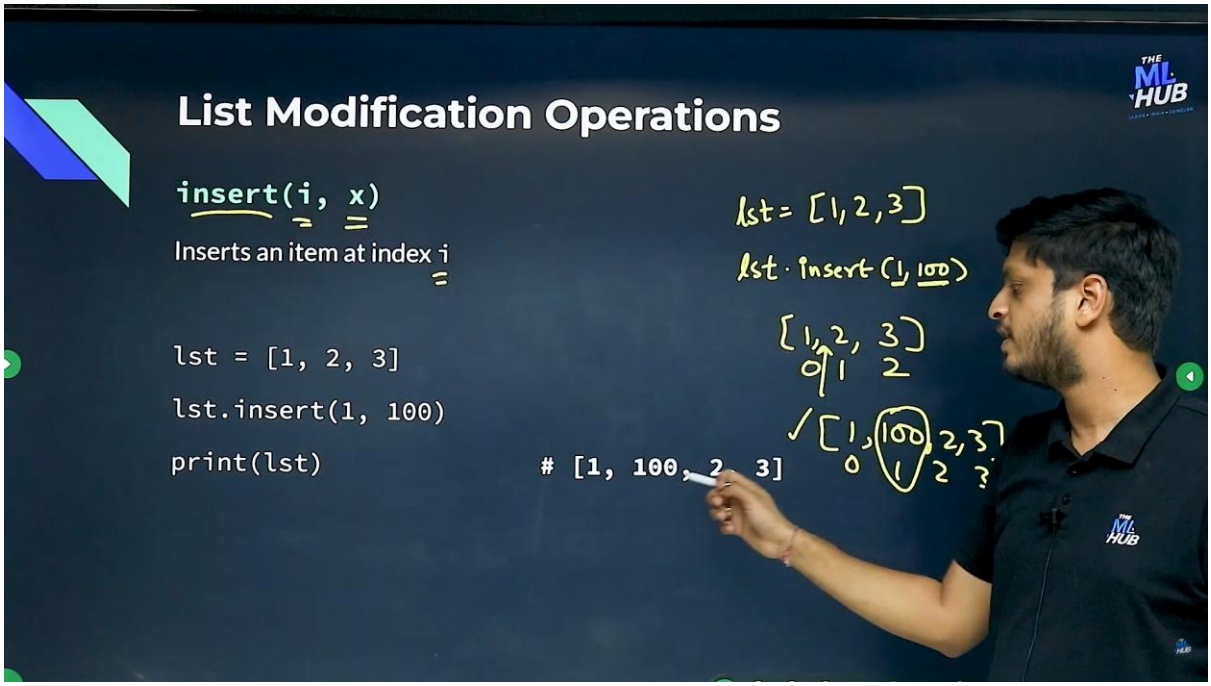
```
a = [5, 6, 7, 8]
a[:2] = a[1:]
```

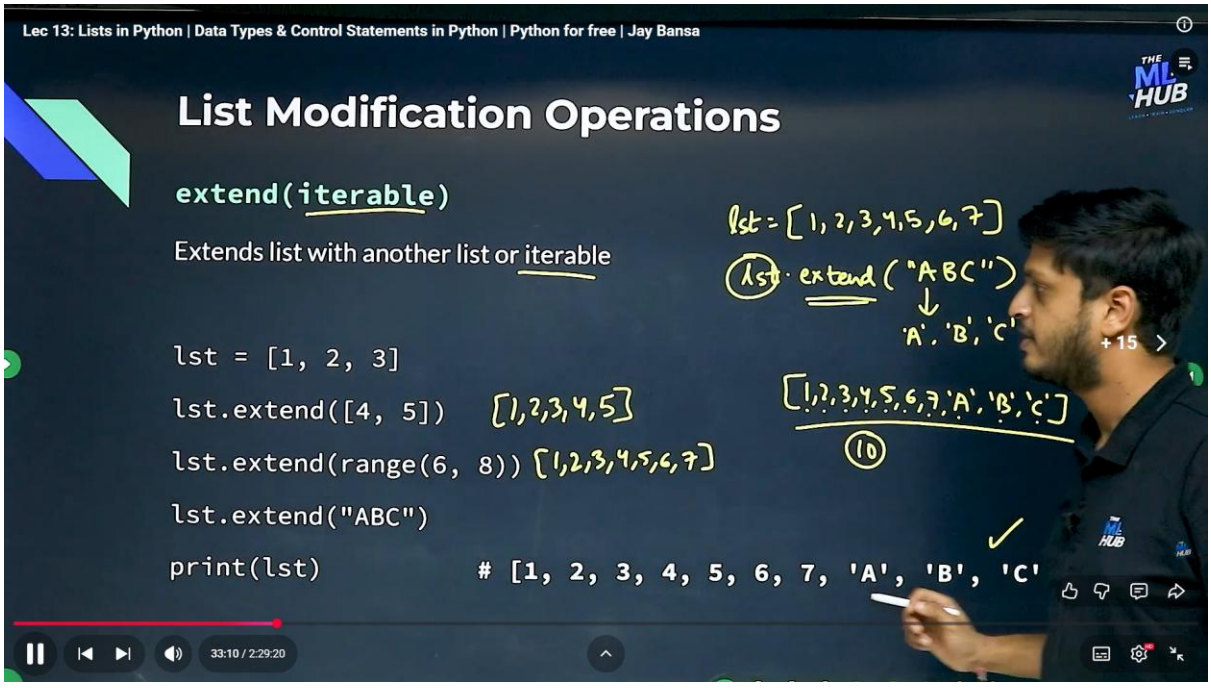

1:22:10 -02:28

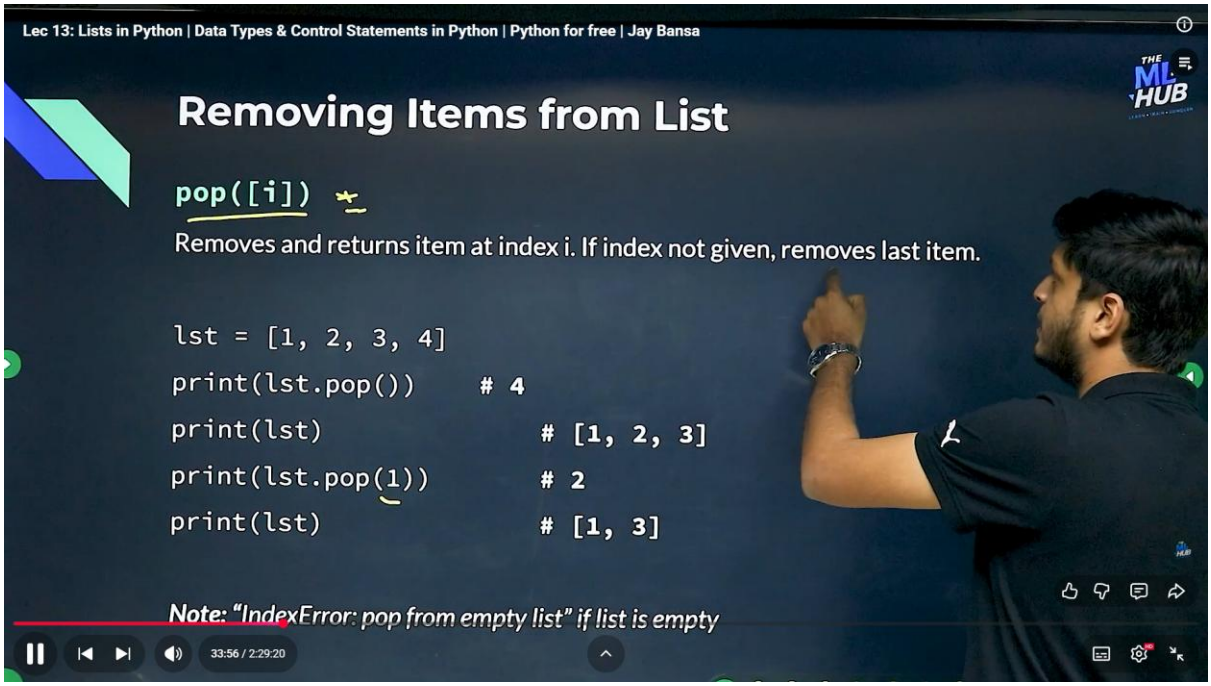


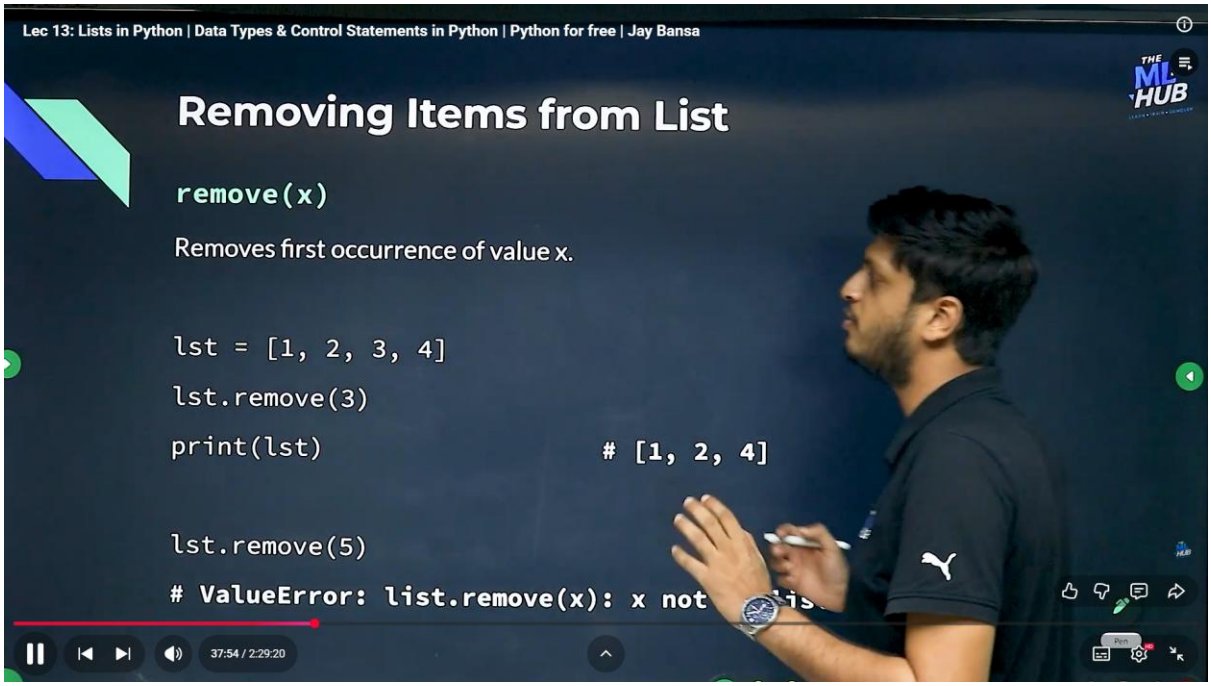


Operation	Example	Result
Concatenation	[1, 2] + [3, 4]	[1, 2, 3, 4]
Repetition	[0] * 3	[0, 0, 0]
Membership	3 in [1, 2, 3]	True
Length	len([1, 2, 3])	3
Min	min([4, 7, 2])	2
Max	max([4, 7, 2])	7
Sum	sum([1, 2, 3])	6


List Modification Operations
insert(i, x)
Inserts an item at index i
lst = [1, 2, 3]
lst.insert(1, 100)
print(lst)
# [1, 100, 2, 3]
lst = [1, 2, 3]
lst · insert (1, 100)
[1, 2, 3]
0 1 2
[1, 100, 2, 3]
0 1 2 3


Lec 13: Lists in Python | Data Types & Control Statements in Python | Python for free | Jay Bansa
List Modification Operations
extend(iterable)
Extends list with another list or iterable
lst = [1, 2, 3]
lst.extend([4, 5])
lst.extend(range(6, 8))
lst.extend("ABC")
print(lst)
# [1, 2, 3, 4, 5, 6, 7, 'A', 'B', 'C']
lst = [1, 2, 3, 4, 5, 6, 7]
lst · extend ("ABC")
'A', 'B', 'C'
[1, 2, 3, 4, 5]
[1, 2, 3, 4, 5, 6, 7]
[1, 2, 3, 4, 5, 6, 7, 'A', 'B', 'C']
10
+15
33:10 / 2:29:20


Lec 13: Lists in Python | Data Types & Control Statements in Python | Python for free | Jay Bansa
Removing Items from List
pop([i])
Removes and returns item at index i. If index not given, removes last item.
lst = [1, 2, 3, 4]
print(lst.pop()) # 4
print(lst) # [1, 2, 3]
print(lst.pop(1)) # 2
print(lst) # [1, 3]
Note: "IndexError: pop from empty list" if list is empty
33:56 / 2:29:20


Lec 13: Lists in Python | Data Types & Control Statements in Python | Python for free | Jay Bansa
Removing Items from List
remove(x)
Removes first occurrence of value x.
lst = [1, 2, 3, 4]
lst.remove(3)
print(lst) # [1, 2, 4]
lst.remove(5)
# ValueError: list.remove(x): x not
37:54 / 2:29:20

Removing Items from List

`clear()`

Removes all elements from list.

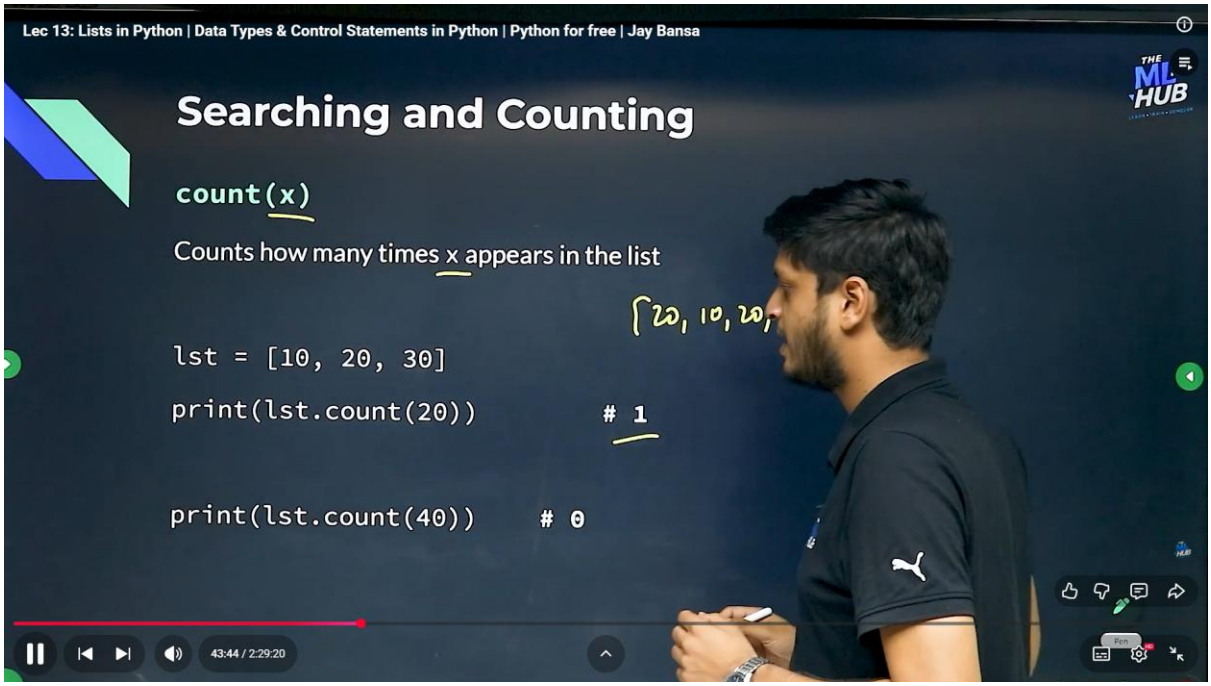
```
lst = [1, 2, 3, 4]
lst.clear()
print(lst)                # [
```

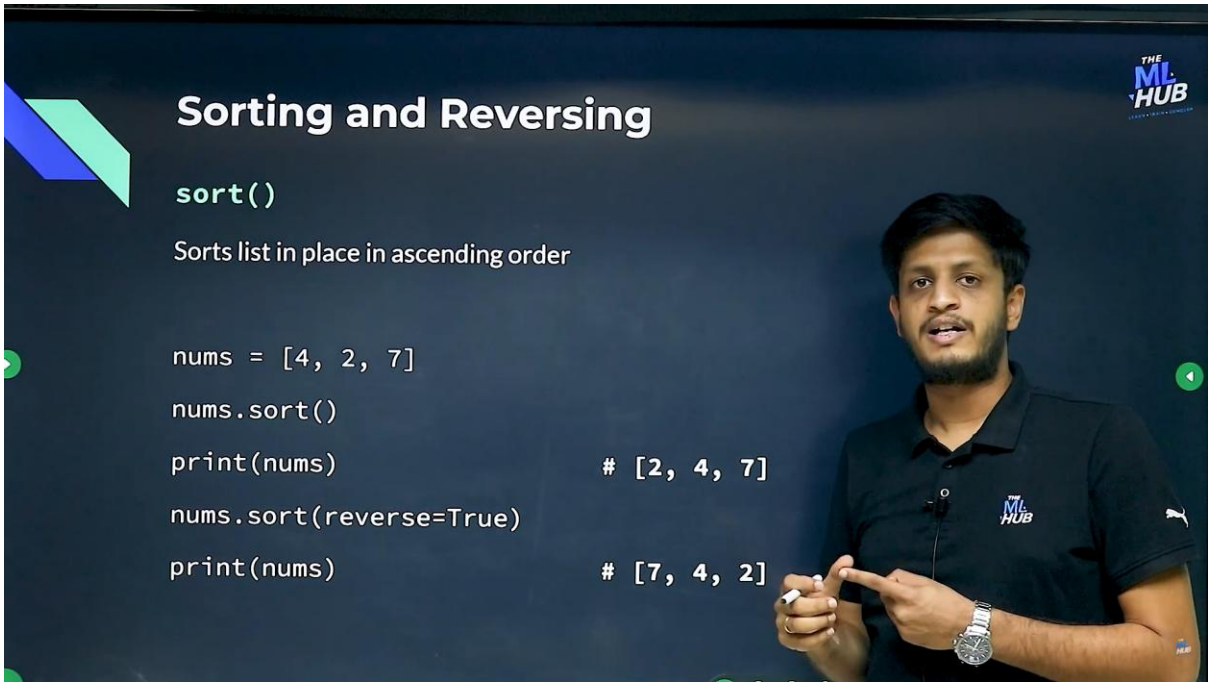
Searching and Counting

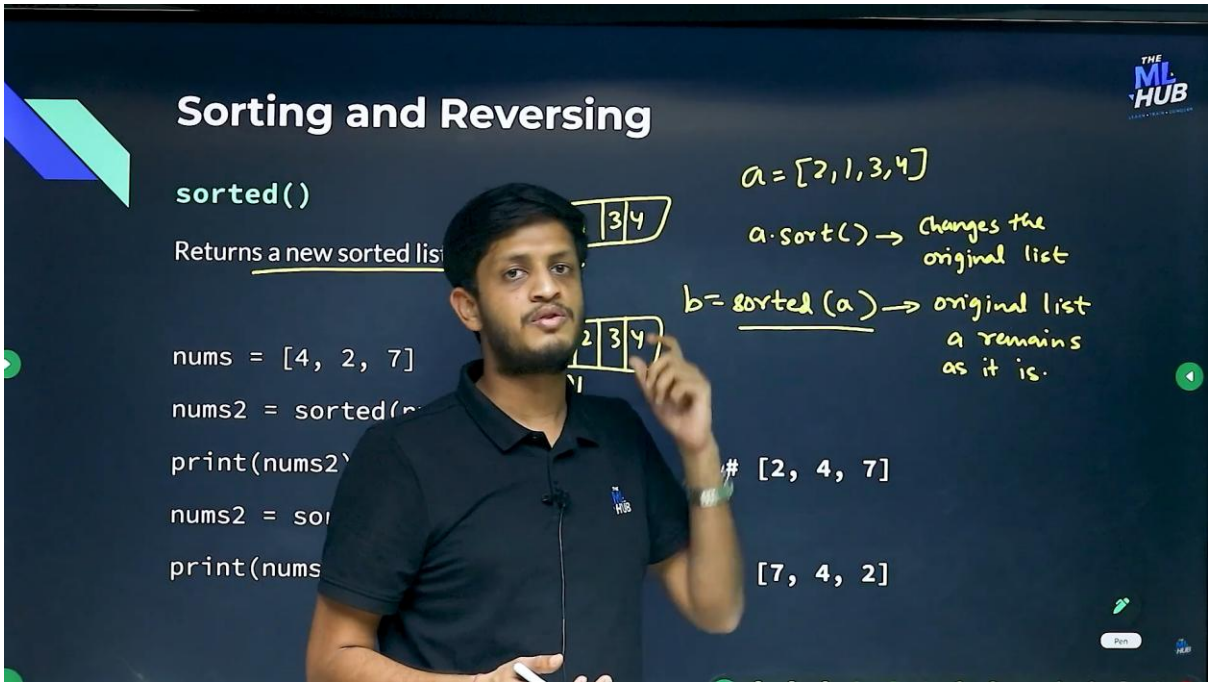
`index(x)`

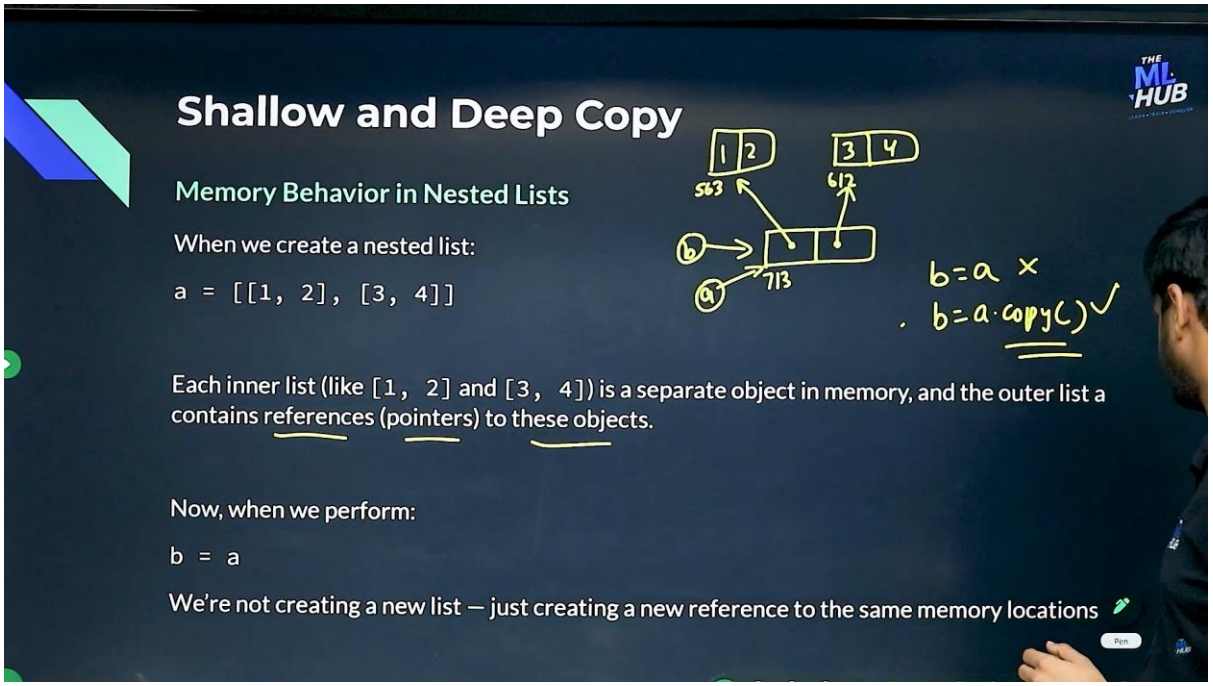
Returns index of first occurrence of value x

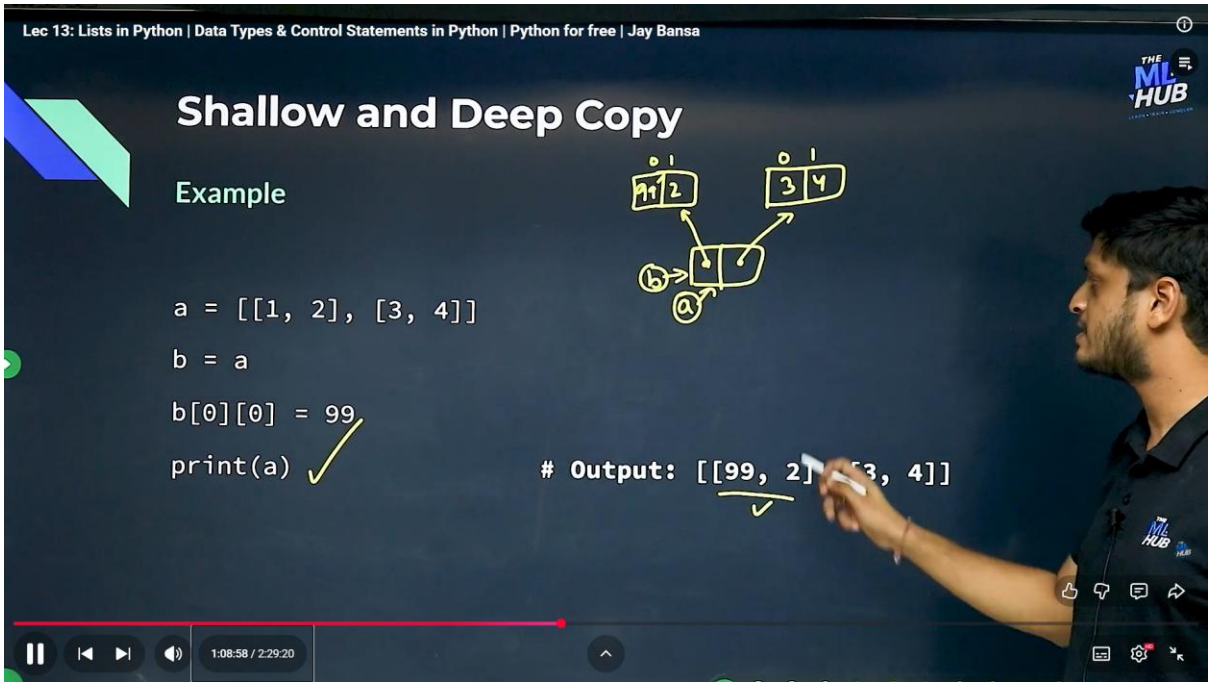
```
lst = [10, 20, 30]
print(lst.index(20))    # 1

print(lst.index(40))
# ValueError: 40 is not in list
```

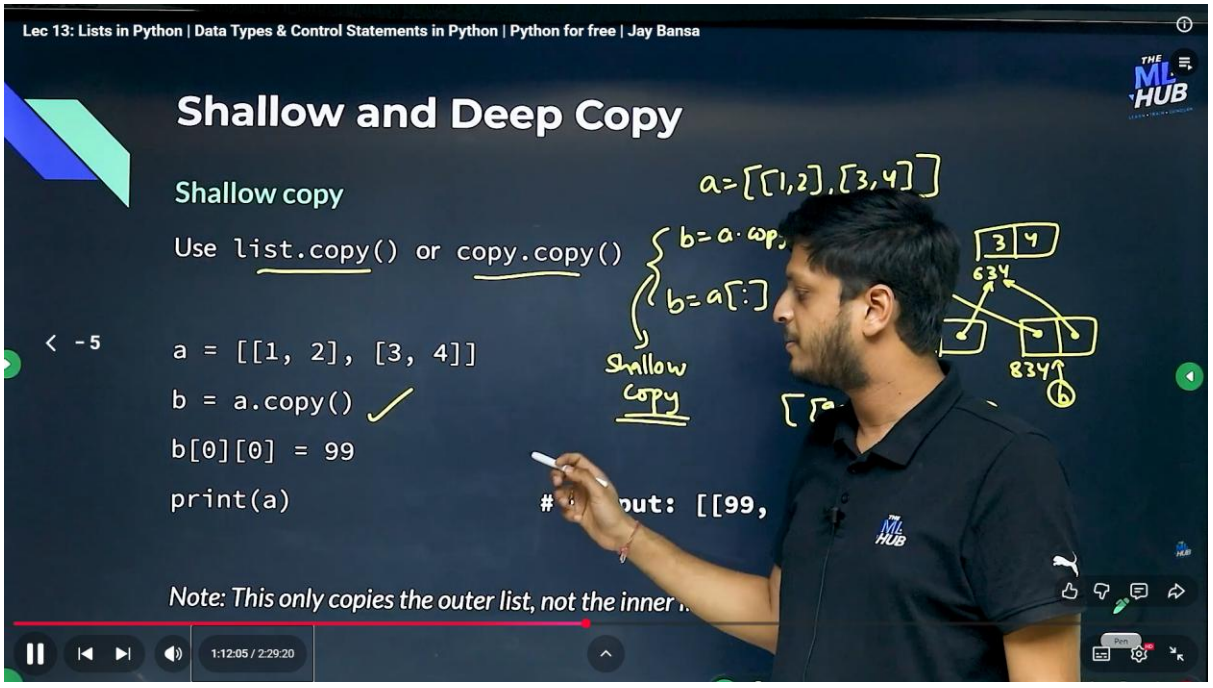

Lec 13: Lists in Python | Data Types & Control Statements in Python | Python for free | Jay Bansa
Searching and Counting
count(x)
Counts how many times x appears in the list
lst = [10, 20, 30]
print(lst.count(20)) # 1
print(lst.count(40)) # 0
43:44 / 2:29:20

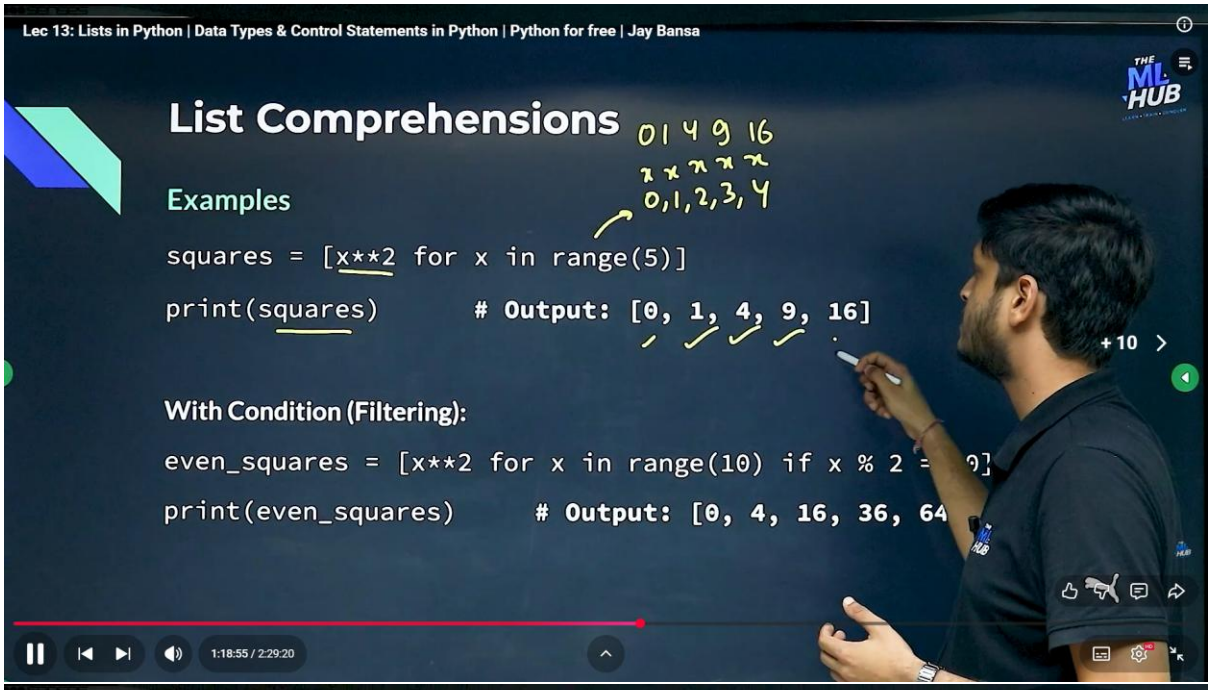

Sorting and Reversing
sort()
Sorts list in place in ascending order
nums = [4, 2, 7]
nums.sort()
print(nums) # [2, 4, 7]
nums.sort(reverse=True)
print(nums) # [7, 4, 2]

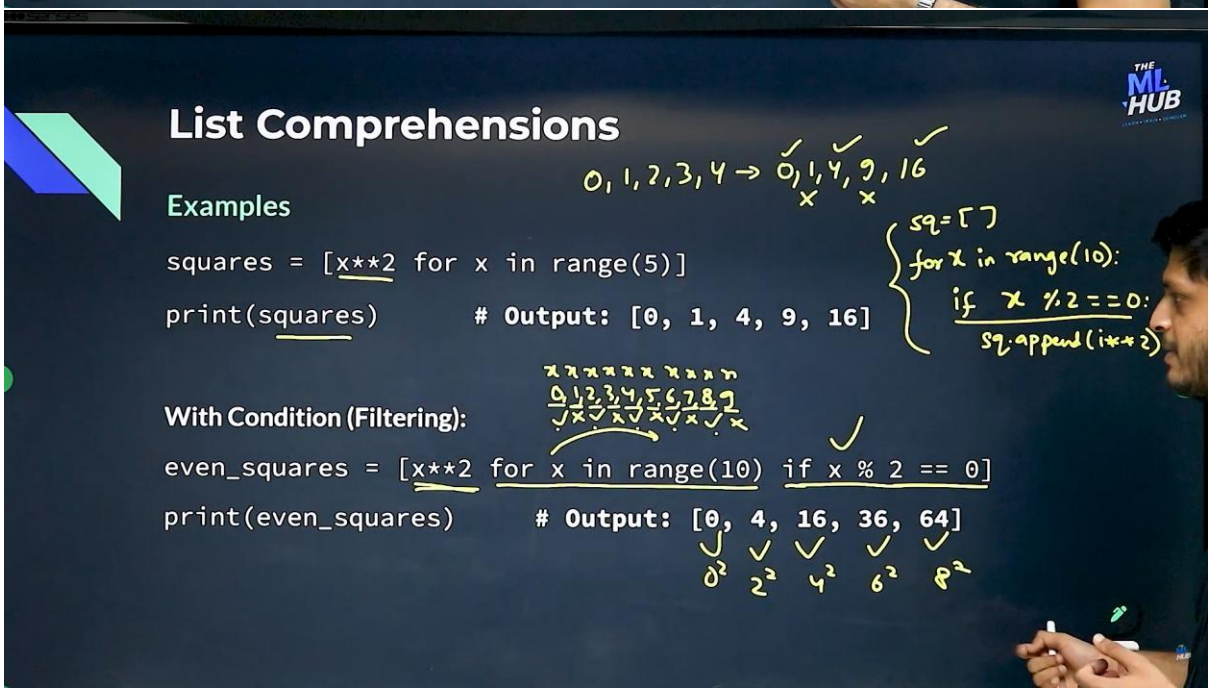

Sorting and Reversing
sorted()
Returns a new sorted lis
nums = [4, 2, 7]
nums2 = sorted(
print(nums2
nums2 = so
print(nums
a = [2,1,3,4]
a.sort() → changes the original list
b = sorted(a) → original list a remains as it is.
# [2, 4, 7]
[7, 4, 2]

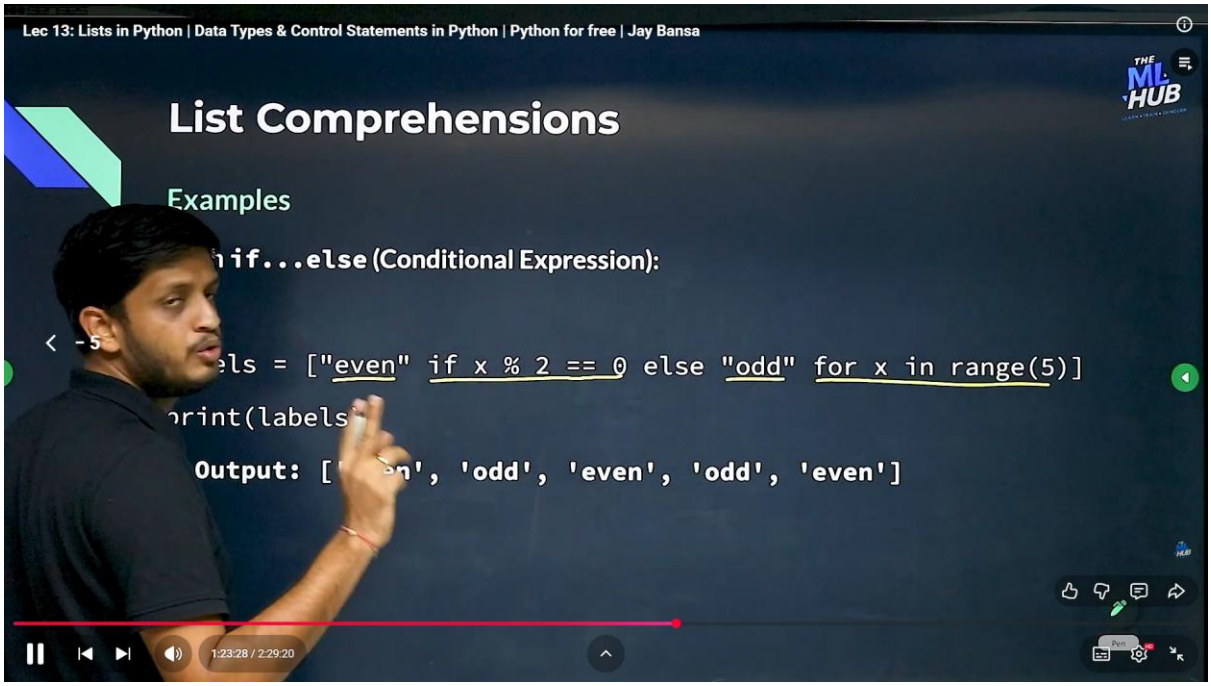

Shallow and Deep Copy
Memory Behavior in Nested Lists
When we create a nested list:
a = [[1, 2], [3, 4]]
563
617
713
b = a ×
b = a.copy() ✓
Each inner list (like [1, 2] and [3, 4]) is a separate object in memory, and the outer list a contains references (pointers) to these objects.
Now, when we perform:
b = a
We're not creating a new list — just creating a new reference to the same memory locations


Lec 13: Lists in Python | Data Types & Control Statements in Python | Python for free | Jay Bansa
Shallow and Deep Copy
Example
a = [[1, 2], [3, 4]]
b = a
b[0][0] = 99
print(a)
# Output: [[99, 2]
3, 4]]
1:08:58 / 2:29:20


Lec 13: Lists in Python | Data Types & Control Statements in Python | Python for free | Jay Bansa
Shallow and Deep Copy
Shallow copy
Use list.copy() or copy.copy()
a = [[1, 2], [3, 4]]
b = a.copy()
b[0][0] = 99
print(a)
Note: This only copies the outer list, not the inner
1:12:05 / 2:29:20


Lec 13: Lists in Python | Data Types & Control Statements in Python | Python for free | Jay Bansa
List Comprehensions
Examples
squares = [x**2 for x in range(5)]
print(squares) # Output: [0, 1, 4, 9, 16]
With Condition (Filtering):
even_squares = [x**2 for x in range(10) if x % 2 =
print(even_squares) # Output: [0, 4, 16, 36, 64
1:18:55 / 2:29:20


List Comprehensions
Examples
squares = [x**2 for x in range(5)]
print(squares) # Output: [0, 1, 4, 9, 16]
With Condition (Filtering):
even_squares = [x**2 for x in range(10) if x % 2 == 0]
print(even_squares) # Output: [0, 4, 16, 36, 64]


Lec 13: Lists in Python | Data Types & Control Statements in Python | Python for free | Jay Bansa
List Comprehensions
Examples
if...else (Conditional Expression):
els = ["even" if x % 2 == 0 else "odd" for x in range(5)]
print(labels
Output: ['
n', 'odd', 'even', 'odd', 'even']
1:23:28 / 2:29:20