REC

Lists []

- ✓Creating Lists,
- ✓Basic List Operations,
- ✓ Indexing and Slicing in Lists,
- ✓Built-In Functions Used on Lists,
- ✓List Methods,
- ✓The del Statement.

REC

Lists []

- ✓Creating Lists,
- ✓Basic List Operations,
- ✓ Indexing and Slicing in Lists,
- ✓Built-In Functions Used on Lists,
- ✓List Methods,
- ✓The del Statement.

REC

Python Collections (Arrays)

- There are four collection data types in the Python programming language:
  - **List:** is a collection which is ordered and changeable. Allows duplicate members.
  - **Tuple** is a collection which is ordered and unchangeable. Allows duplicate members.
  - **Set** is a collection which is unordered and unindexed. No duplicate members.
  - **Dictionary** is a collection which is unordered, changeable and indexed. No duplicate members.

REC

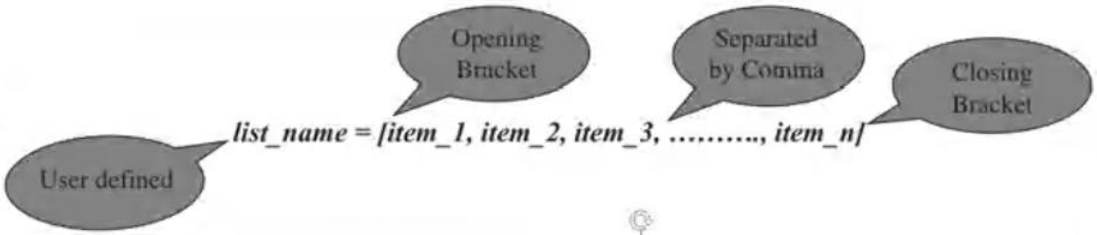
Lists-Introduction

- Lists are one of the most flexible data storage formats in Python because they can have values added, removed, and changed.
- You can think of the list as a container that holds a number of **items**.
- Each element or value that is inside a list is called an **item**. All the items in a list are assigned to a single variable.
- Lists avoid having a separate variable to store each item which is less efficient and more error prone when you have to perform some operations on these items.
- Lists can be simple or nested lists with varying types of values.

REC

Creating Lists

Lists are constructed using square brackets [ ] wherein you can include a list of items separated by commas.




```
1  superstore = ["metro", "tesco", "walmart", "kmart", "carrefour"]
2  superstore
```

```
['metro', 'tesco', 'walmart', 'kmart', 'carrefour']
```

You can create an empty list without any items. The syntax is,

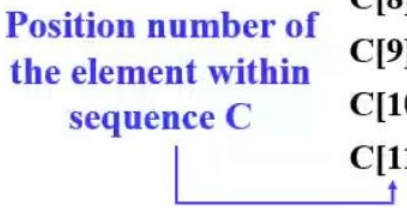
list_name = [ ]

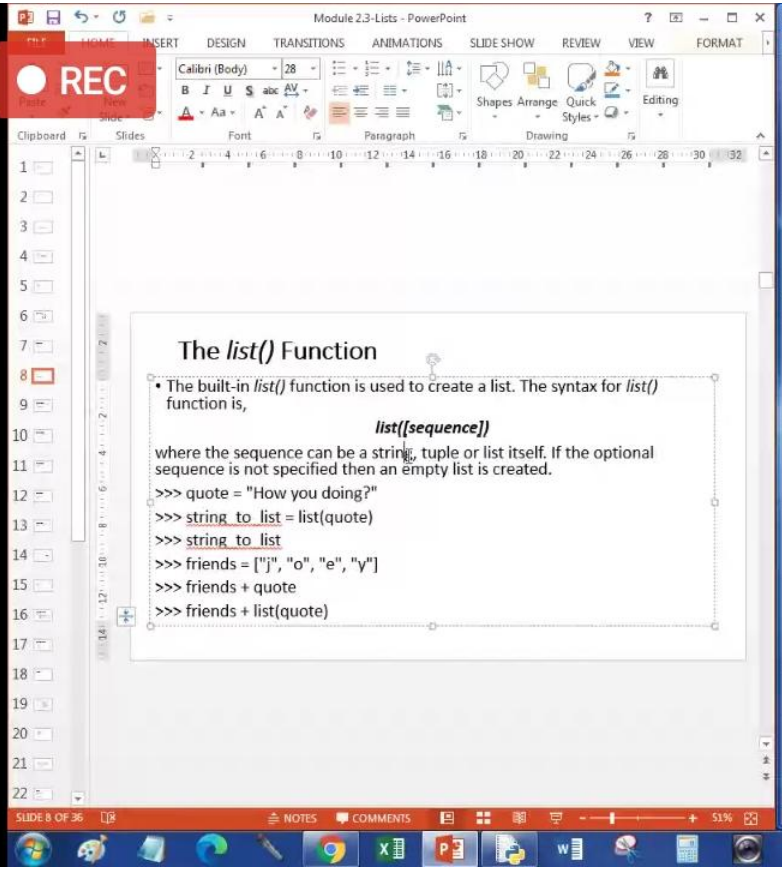
You

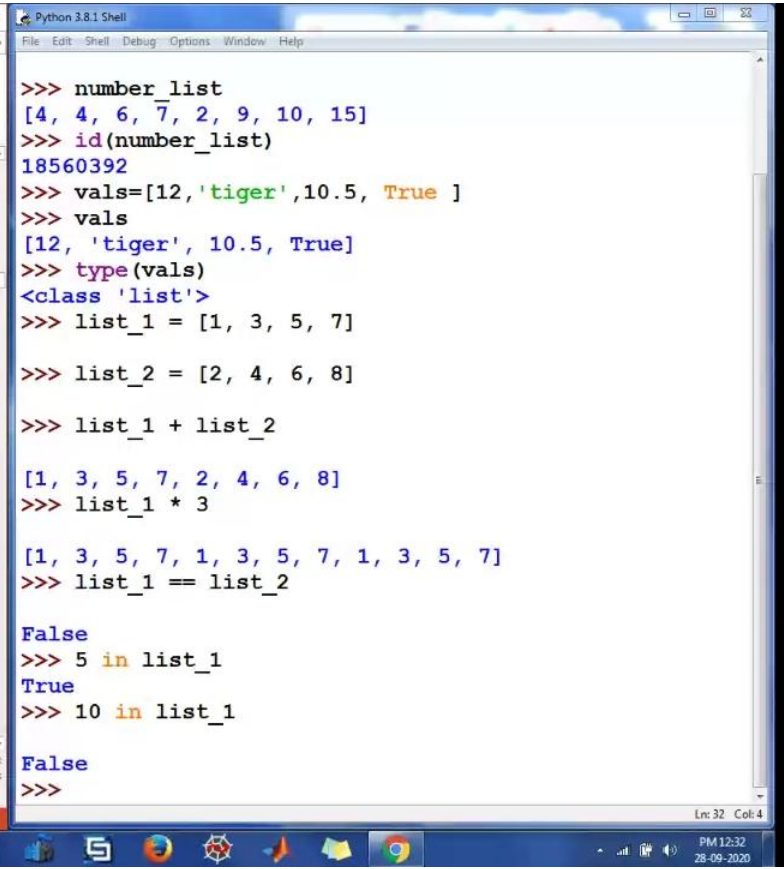
REC

Example of Sequences

C[0]	-45	C[-12]
C[1]	6	C[-11]
C[2]	0	C[-10]
C[3]	72	C[-9]
C[4]	34	C[-8]
C[5]	39	C[-7]
C[6]	98	C[-6]
C[7]	-1345	C[-5]
C[8]	939	C[-4]
C[9]	10	C[-3]
C[10]	40	C[-2]
C[11]	33	C[-1]




Module 2.3-Lists - PowerPoint
The list() Function
• The built-in list() function is used to create a list. The syntax for list() function is,
list([sequence])
where the sequence can be a string, tuple or list itself. If the optional sequence is not specified then an empty list is created.
>>> quote = "How you doing?"
>>> string_to_list = list(quote)
>>> string_to_list
>>> friends = ["j", "o", "e", "y"]
>>> friends + quote
>>> friends + list(quote)
SLIDE 8 OF 36
NOTES
COMMENTS
51%


Python 3.8.1 Shell
File Edit Shell Debug Options Window Help
>>> number_list
[4, 4, 6, 7, 2, 9, 10, 15]
>>> id(number_list)
18560392
>>> vals=[12,'tiger',10.5, True]
>>> vals
[12, 'tiger', 10.5, True]
>>> type(vals)
<class 'list'>
>>> list_1 = [1, 3, 5, 7]
>>> list_2 = [2, 4, 6, 8]
>>> list_1 + list_2
[1, 3, 5, 7, 2, 4, 6, 8]
>>> list_1 * 3
[1, 3, 5, 7, 1, 3, 5, 7, 1, 3, 5, 7]
>>> list_1 == list_2
False
>>> 5 in list_1
True
>>> 10 in list_1
False
>>>
Ln: 32 Col: 4

Module 2.3-Lists - PowerPoint

Indexing and Slicing in Lists

- . For the list *superstore*, the index breakdown is shown below.


```
superstore = ["metro", "tesco", "walmart", "kmart", "carrefour"]
print(superstore[0])
print(superstore[1])
print(superstore[2])
print(superstore[3])
print(superstore[4])
print(superstore[9])
```

```
metro
tesco
walmart
kmart
carrefour

---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-9-3ceed56b913f> in <module>
      5 print(superstore[3])
      6 print(superstore[4])
----> 7 print(superstore[9])

IndexError: list index out of range
```

PP-Monday

People (43) Chat (5)

Let everyone send messages

1DS19EC425 RAHUL KUMAR 12:17 PM
Good afternoon ma'am

Pramath V 12:35 PM
Yes ma'am

1DS18EC070-Puneetha Ramya 12:35 PM
Yes ma'am

1DS18EC082 __Shamaa kindre 12:35 PM
Yes maam

1DS19EC422 Nithin TM 12:35 PM
Yes mam

Send a message to everyone

You

REC

Module 2.3-Lists - PowerPoint

FILE HOME INSERT DESIGN TRANSITIONS ANIMATIONS SLIDE SHOW REVIEW VIEW FORMA

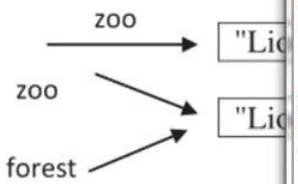
Calibri (Body) 26

Clipboard Slides Font Paragraph Drawing

Modifying Items in Lists

- When you assign an existing list variable to a (=) on lists does not make a new copy. Instead variable names point to the same list in mem

```
>>> zoo = ["Lion", "Tiger", "Zebra"]
>>> forest = zoo
>>> id(forest)
>>> id(zoo)
>>>zoo[0] = "Fox“
>>> zoo
>>> forest
>>> type(zoo)
```



SLIDE 13 OF 36 ENGLISH (INDIA) NOTES COMMENTS

Python 3.8.1 Shell

File Edit Shell Debug Options Window Help

```
TypeError: can only concatenate list (not "str") to list
>>> friends + list(quote)

['j', 'o', 'e', 'y', 'H', 'o', 'w', ' ', 'y', 'o', 'u', '
', 'd', 'o', 'i', 'n', 'g', '?']
>>> hoovu=['mallige', 'sampige', 'kanaka', 'kanagala']
>>> hoovu[2]
'kanaka'
>>> id(hoovu)
12101000
>>> hoovu[2]='rose'
>>> hoovu
['mallige', 'sampige', 'rose', 'kanagala']
>>> id(hoovu)
12101000
>>> hoovu= flowers
Traceback (most recent call last):
  File "<pyshell#25>", line 1, in <module>
    hoovu= flowers
NameError: name 'flowers' is not defined
>>> hoovu=flowers
Traceback (most recent call last):
  File "<pyshell#26>", line 1, in <module>
    hoovu=flowers
NameError: name 'flowers' is not defined
>>> zoo = ["Lion", "Tiger", "Zebra"]

>>> flowers=hoovu
>>> flowers
```

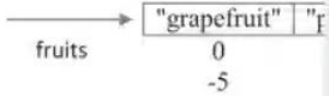
Ln: 71 Col: 11

PM 12:42 28-09-2020

REC

Slicing in Lists

- >>> fruits = ["grapefruit", "pineapple", "blu
- >>> fruits[1:3]
- >>> fruits[:3]
- >>> fruits[2:]
- >>> fruits[1:4:2]
- >>> fruits[:]
- >>> fruits[::2]
- >>> fruits[-3:-1]




```
mallige', 'sampige', 'rose', 'kanagala']
> id(hoovu)
101000
> hoovu= flowers
aceback (most recent call last):
File "<pyshell#25>", line 1, in <module>
  hoovu= flowers
meError: name 'flowers' is not defined
> hoovu=flowers
aceback (most recent call last):
File "<pyshell#26>", line 1, in <module>
  hoovu=flowers
meError: name 'flowers' is not defined
> zoo = ["Lion", "Tiger", "Zebra"]

> flowers=hoovu
> flowers
mallige', 'sampige', 'rose', 'kanagala']
> id(flowers)
101000
> flowers[0]='kamala'
> flowers
kamala', 'sampige', 'rose', 'kanagala'
> hoovu
```

You

REC

Built-In Functions Used on Lists

Built-In Functions Used on Lists

Built-In Functions	Description
len()	The *len()* function returns the numbers of items in a list.
sum()	The *sum()* function returns the sum of numbers in the list.
any()	The *any()* function returns *True* if any of the Boolean values in the list is *True*.
all()	The *all()* function returns *True* if all the Boolean values in the list are *True*, else returns *False*.
sorted()	The *sorted()* function returns a modified copy of the list while leaving the original list untouched.

```
owers=hoo
owers
ige', 'sa
(flowers)
00
owers[0]=
owers
la', 'sar
ovu
la', 'sar
uits = ['
ango", "k

uits[1:3]
apple',
uits[1:4]
apple',
```

REC

Built-In Functions Used on Lists

```
>>> len(lakes)
>>> numbers = [1, 2, 3, 4, 5]
>>> sum(numbers)
>>> max(numbers)
>>> min(numbers)
>>> any([1, 1, 0, 0, 1, 0])
>>> any([ 0, 0, 0, 0])
>>> any([ 0, 1, 0, 0])
>>> all([1, 1, 1, 1])
>>> lakes_sorted_new = sorted(lakes)
```

REC

List Methods

- The list size changes dynamically whenever you add or remove the items and there is no need for you to manage it yourself.
- dir(list)

REC

Populating Lists with Items

```
>>> cities = ["oslo", "delhi", "washington", "london", "seattle", "paris", "washington"]
>>> cities.count('seattle')
>>> cities.count('washington')
>>> cities.index('london')
>>> cities.reverse()
>>> cities
>>> cities.append('brussels')
>>> cities
>>> cities.sort()
>>> cities.pop()
>>> cities.pop(2)
>>> more_cities = ["brussels", "copenhagen"]
>>> cities.extend(more_cities)
>>> cities
>>> cities.remove("brussels")
>>> cities
```

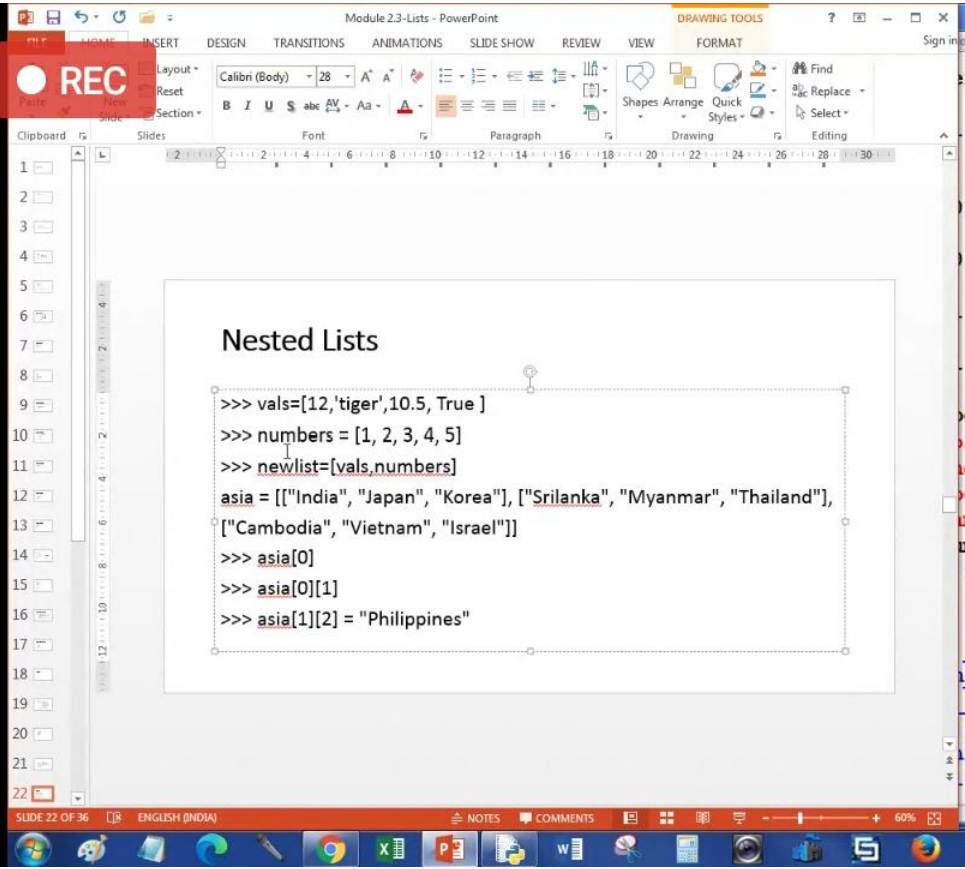
REC

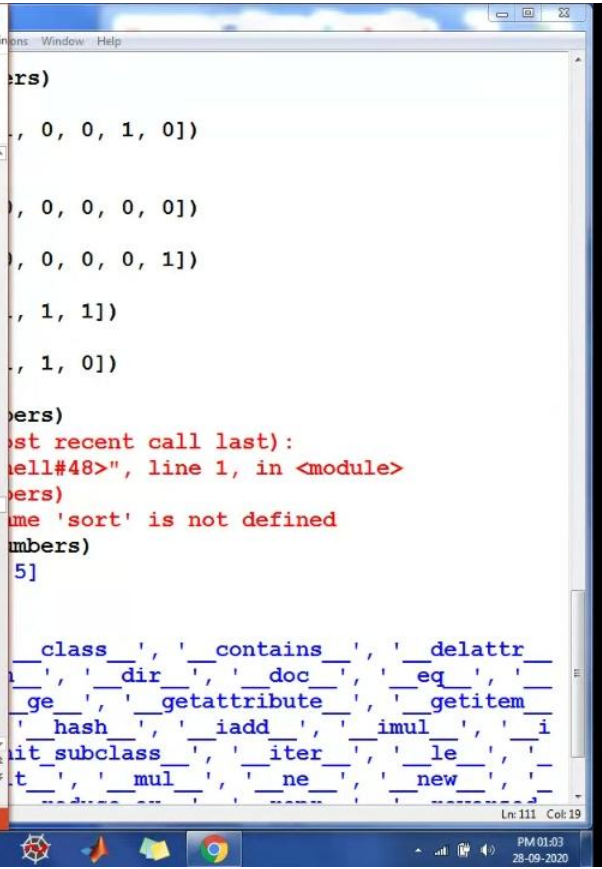
Various List Methods

List Methods	Syntax	Description
append()	list.append(item)	The *append()* method adds a single item to the end of the list. This method does not return new list and it just modifies the original.
count()	list.count(item)	The *count()* method counts the number of times the item has occurred in the list and returns it.
insert()	list.insert(index, item)	The *insert()* method inserts the item at the given index, shifting items to the right.
extend()	list.extend(list2)	The *extend()* method adds the items in list2 to the end of the list.
index()	list.index(item)	The *index()* method searches for the given item from the start of the list and returns its index. If the value appears more than once, you will get the index of the first one. If the item is not present in the list then *ValueError* is thrown by this method.
remove()	list.remove(item)	The *remove()* method searches for the first instance of the given item in the list and removes it. If the item is not present in the list then *ValueError* is thrown by this method.
sort()	list.sort()	The *sort()* method sorts the items *in place* in the list. This method modifies the original list and it does not return a new list.
reverse()	list.reverse()	The *reverse()* method reverses the items *in place* in the list. This method modifies the original list and it does not return a new list.
pop()	list.pop([index])	The pop() method removes and returns the item at the given index. This method returns the rightmost item if the index is omitted.

Note: Replace the word "list" mentioned in the syntax with your *actual list name* in your code.

You


Module 2.3-Lists - PowerPoint
DRAWING TOOLS
Nested Lists
>>> vals=[12,'tiger',10.5, True]
>>> numbers = [1, 2, 3, 4, 5]
>>> newlist=[vals,numbers]
asia = [["India", "Japan", "Korea"], ["Srilanka", "Myanmar", "Thailand"],
["Cambodia", "Vietnam", "Israel"]]
>>> asia[0]
>>> asia[0][1]
>>> asia[1][2] = "Philippines"
SLIDE 22 OF 36
ENGLISH (INDIA)
NOTES
COMMENTS
60%

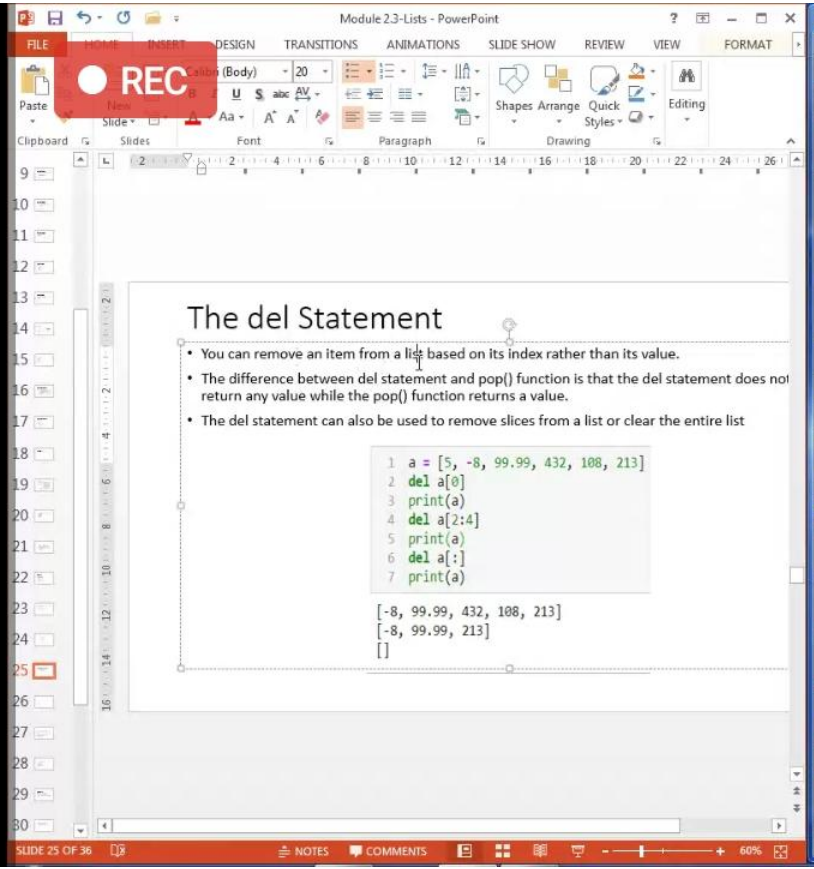

, 0, 0, 1, 0])
, 0, 0, 0, 0])
, 0, 0, 0, 1])
, 1, 1])
, 1, 0])
st recent call last):
me 'sort' is not defined
__class__', '__contains__', '__delattr__
__', '__dir__', '__doc__', '__eq__', '__
ge__', '__getattribute__', '__getitem__
__hash__', '__iadd__', '__imul__', '__i
it_subclass__', '__iter__', '__le__', '__
t__', '__mul__', '__ne__', '__new__', '__
Ln: 111 Col: 19

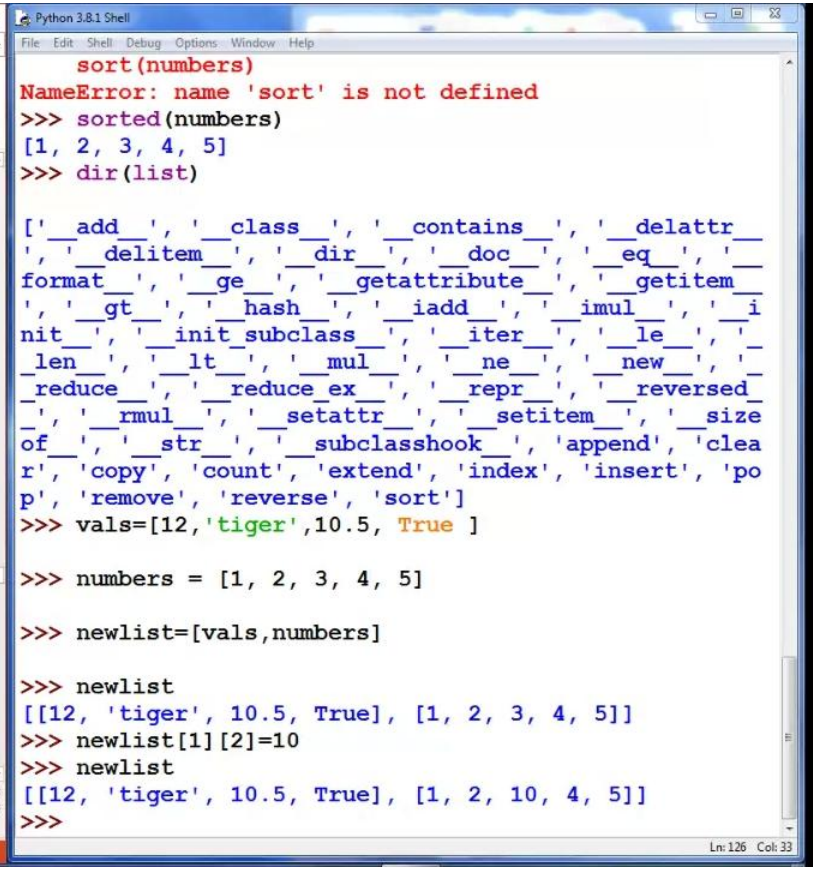
A two-dimensional list

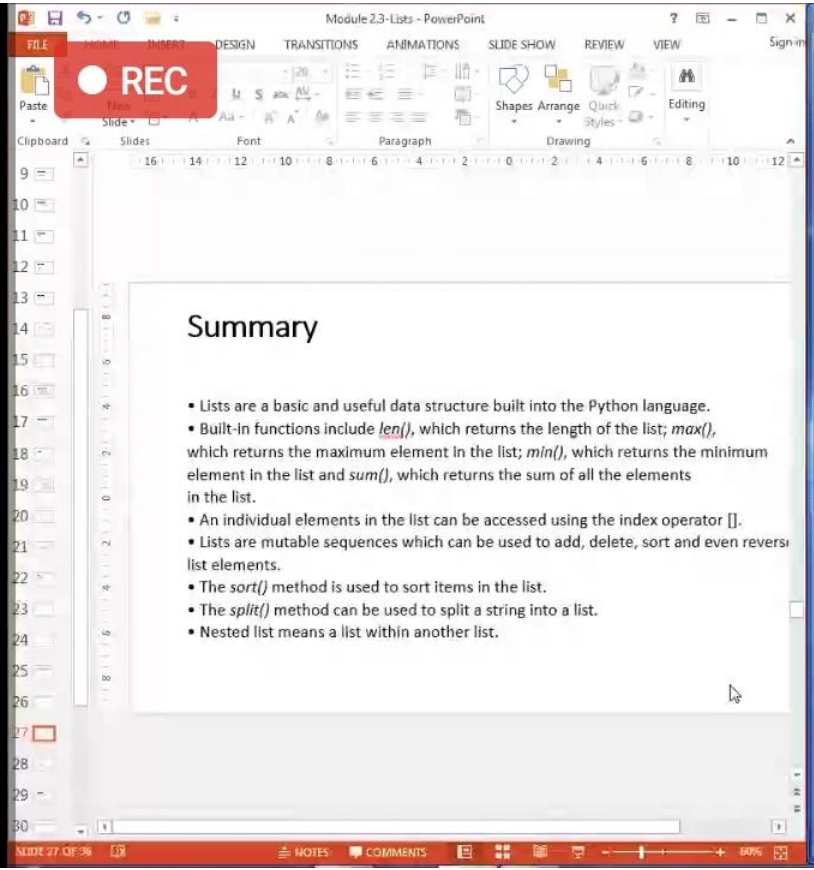
	Column 0	Column 1
Row 0	'Joe'	'Kim'
Row 1	'Sam'	'Sue'
Row 2	'Kelly'	'Chris'

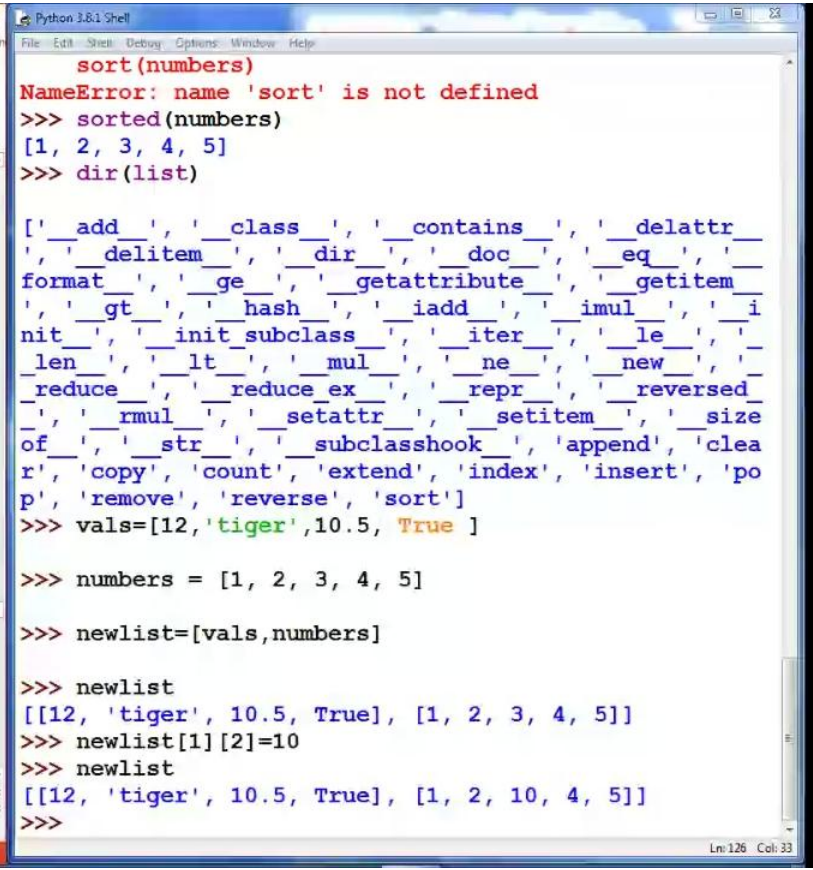
Subscripts for each element of the `scores` list

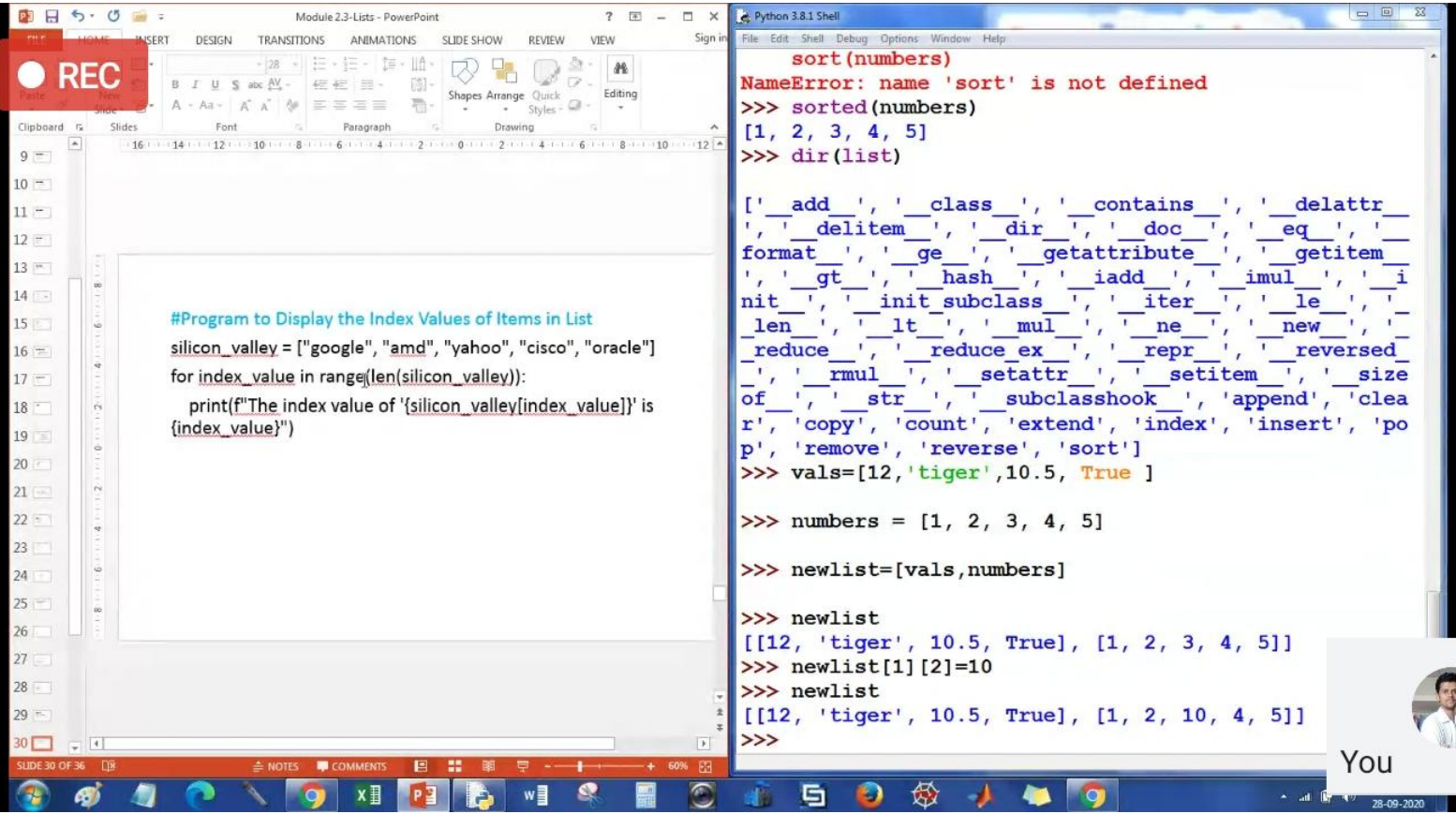
	Column 0	Column 1	Column 2
Row 0	scores[0][0]	scores[0][1]	scores[0][2]
Row 1	scores[1][0]	scores[1][1]	scores[1][2]
Row 2	scores[2][0]	scores[2][1]	scores[2][2]


Module 2.3-Lists - PowerPoint
REC
The del Statement
• You can remove an item from a list based on its index rather than its value.
• The difference between del statement and pop() function is that the del statement does not return any value while the pop() function returns a value.
• The del statement can also be used to remove slices from a list or clear the entire list
a = [5, -8, 99.99, 432, 108, 213]
del a[0]
print(a)
del a[2:4]
print(a)
del a[:]
print(a)
[-8, 99.99, 432, 108, 213]
[-8, 99.99, 213]
[]


Python 3.8.1 Shell
sort(numbers)
NameError: name 'sort' is not defined
>>> sorted(numbers)
[1, 2, 3, 4, 5]
>>> dir(list)
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__gt__', '__hash__', '__iadd__', '__imul__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__reversed__', '__rmul__', '__setattr__', '__setitem__', '__sizeof__', '__str__', '__subclasshook__', 'append', 'clear', 'copy', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse', 'sort']
>>> vals=[12,'tiger',10.5, True]
>>> numbers = [1, 2, 3, 4, 5]
>>> newlist=[vals,numbers]
>>> newlist
[[12, 'tiger', 10.5, True], [1, 2, 3, 4, 5]]
>>> newlist[1][2]=10
>>> newlist
[[12, 'tiger', 10.5, True], [1, 2, 10, 4, 5]]
>>>


Module 2.3-Lists - PowerPoint
REC
Summary
• Lists are a basic and useful data structure built into the Python language.
• Built-in functions include len(), which returns the length of the list; max(), which returns the maximum element in the list; min(), which returns the minimum element in the list and sum(), which returns the sum of all the elements in the list.
• An individual elements in the list can be accessed using the index operator [].
• Lists are mutable sequences which can be used to add, delete, sort and even reverse list elements.
• The sort() method is used to sort items in the list.
• The split() method can be used to split a string into a list.
• Nested list means a list within another list.


Python 3.8.1 Shell
sort(numbers)
NameError: name 'sort' is not defined
>>> sorted(numbers)
[1, 2, 3, 4, 5]
>>> dir(list)
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__gt__', '__hash__', '__iadd__', '__imul__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__reversed__', '__rmul__', '__setattr__', '__setitem__', '__sizeof__', '__str__', '__subclasshook__', 'append', 'clear', 'copy', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse', 'sort']
>>> vals=[12,'tiger',10.5, True]
>>> numbers = [1, 2, 3, 4, 5]
>>> newlist=[vals,numbers]
>>> newlist
[[12, 'tiger', 10.5, True], [1, 2, 3, 4, 5]]
>>> newlist[1][2]=10
>>> newlist
[[12, 'tiger', 10.5, True], [1, 2, 10, 4, 5]]
>>>
Ln: 126 Col: 33


Module 2.3-Lists - PowerPoint
#Program to Display the Index Values of Items in List
silicon_valley = ["google", "amd", "yahoo", "cisco", "oracle"]
for index_value in range(len(silicon_valley)):
print(f"The index value of '{silicon_valley[index_value]}' is {index_value}")
Python 3.8.1 Shell
sort(numbers)
NameError: name 'sort' is not defined
>>> sorted(numbers)
[1, 2, 3, 4, 5]
>>> dir(list)
['__add__', '__class__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__gt__', '__hash__', '__iadd__', '__imul__', '__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__reversed__', '__rmul__', '__setattr__', '__setitem__', '__sizeof__', '__str__', '__subclasshook__', 'append', 'clear', 'copy', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse', 'sort']
>>> vals=[12,'tiger',10.5, True]
>>> numbers = [1, 2, 3, 4, 5]
>>> newlist=[vals,numbers]
>>> newlist
[[12, 'tiger', 10.5, True], [1, 2, 3, 4, 5]]
>>> newlist[1][2]=10
>>> newlist
[[12, 'tiger', 10.5, True], [1, 2, 10, 4, 5]]
>>>
REC
You

REC

```
#Input Five Integers (+ve and –ve). Find the Sum of Negative Numbers,
#Positive Numbers and Print Them. Also, Find the Average of All the Numbers
#and Numbers Above Average

def find_sum(list_items):
    positive_sum = 0
    negative_sum =0
    for item in list_items:
        if item > 0:positive_sum = positive_sum + item
        else:negative_sum = negative_sum + item
    average = (positive_sum + negative_sum) / len(list_items)
    print(f"Sum of Positive numbers in list is {positive_sum}", )
    print(f"Sum of Negative numbers in list is {negative_sum}")
    print(f"Average of item numbers in list is {average}")
    print("Items above average are")
    for item in list_items:
        if item > average:print(item)

find_sum([-1, -2, -3, 4.2, 5, 6, -3, 0.5])
```