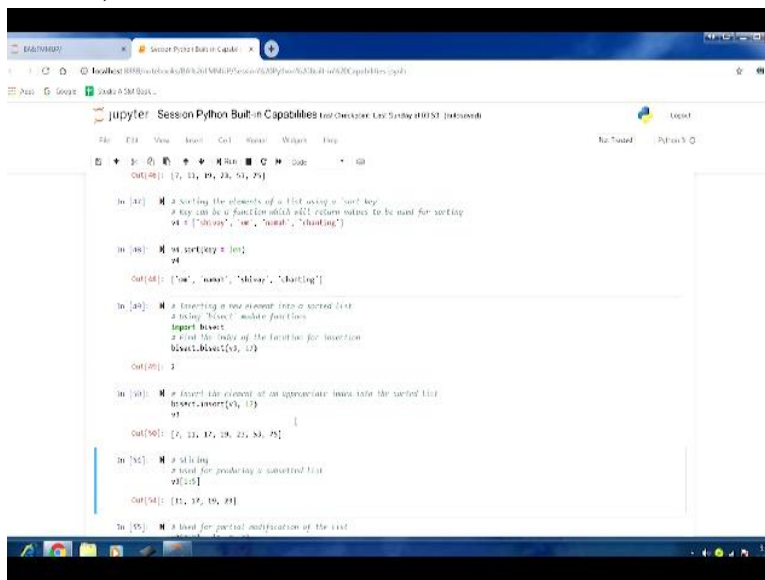


Business Analytics And Text Mining Modeling Using Python
Prof. Gaurav Dixit
Department of Management Studies
Indian Institute of Technology-Roorkee

Lecture-11
Built-in Capabilities of Python-III

Welcome to the course business analytics and text mining modeling using python. So in previous few lectures we have been focusing on the python language.

(Refer Slide Time: 00:37)



```
Out[48]: [7, 11, 15, 23, 31, 35]

In [47]: # a sorting the elements of a list using a sort key
# key can be a function which will return values to be used for sorting
v1 = ["orange", "red", "mango", "chocolate"]

In [48]: # we sort key = len
v1

Out[48]: ['mango', 'orange', 'chocolate', 'red']

In [49]: # a inserting a new element into a sorted list
# using 'insert()' method
# list: Only of the function for insertion
blist.insert(3, 12)

Out[49]: 12

In [50]: # insert the element at an appropriate index into the sorted list
blist.insert(5, 13)
v1

Out[50]: [7, 11, 15, 23, 31, 35, 12, 13]

In [51]: # a slicing
# used for producing a sub-list
v1[1:5]

Out[51]: [11, 15, 23, 31]

In [52]: # a loop for partial modification of the list
```

So, in previous lecture specifically we were discussing lists and various you know methods and various scenarios on how we can work with lists. So, let us pick up from the point where we stopped in the previous lecture. So we started you know our discussion on slicing and how we can perform the same using list data structures in python.

(Video Starts: 00:58)

So, let us start from there sometimes there might be certain scenarios where we would like to subset a list and use that for further coding further processing depending on the scenario. So, how that can be performed slicing briefly we have you know used in previous lectures also. So, now let us discuss this in more detail. So, when slicing we use the bracket operator again. So, let us say we have this you know list V3 that we were using in the previous lecture.

And now within the bracket operator if let us say you know. So, in the output number 50 you can see the values of this V3 list 7 11 17 19 you know 23:53 and 75. So, we have total 7 elements as part of this list. Now if you want to subset this list is starting from index 1 to index you know 5, when we say index 1 it actually means the second element in the list and when we say index 5 it actually means the 6th element in the list.

So, in the V3 it will actually mean that we will start the slicing and will include 11 and then 17 and then 19 and 23. So, if we perform this if you execute this line of code V3 brackets 1-5. So, you can see the output. So, these 4 elements have been sliced out of the existing list V3. So, in a sense we have you know produced a subset at list, now this particular slicing you know mechanism can also be used to perform partial modification of a list.

So, in a sense you know if you look at the code that example a code example that we are using here V3 and we are picking you know these 3 element here indices we are mentioning 0 to 2 right. So, on the right hand side you can see we are assigning these values 1 2 3 and let us run this. So, as you can see in the output we have just executed this assignment. So, you can see the first 3 elements they have been you know added here 1 2 3.

And if you compare this with the you know a V3 output that is output 50 here. So, compare what we have in the output 50 with what we have in the output 52. So, you can see earlier we had you know 7 11 and then 17. Now what we have 1 2 3 and then you know you know 17. So, you can see 1 2 3 have been added there and we have lost 7 and 11. So, this is happening because the end point when we did this sub setting in this you know the code of line number 52 0 to 2.

So, only 0 and 1 you know the end point the start point and the you know up to the end point are you know selected. So, indices 0 and 1 were selected. So, those were meant for these first 2 values of the elements and V3 7 and 11. So, 7 and 11 were replaced by a new list 1 2 3 and that is why we are seeing the list V3 has been modified in that sense. So, the same thing you know thing will learn again using few more examples.

So, let us move on you know that this particular syntax that we talked about where if we are

doing this you know slicing. So, we mentioned starting index and the ending index, sometimes we can omit the you know start index in the top index we will have to you know I mean all the time. So, next example what we are doing is `V3` and then the brackets no start index and then we have colon and then 5 which is the stop index.

So, if I run this execute this you can see `1 2 3 17 19`. So, you can see the index the stop index was 5 that means you know 6th element but we in the output we have just got the 5 element. So, you can see that end point is not included and 2 things we are trying to demonstrate here one is the syntax where we you know can afford to drop the start index in the code and the second thing is in the output you can clearly see that the stop index the element associated with the stop index is not part of the final output.

So, the sub setted list that we you know you know that we generate does not have the element associated with these stop index. Now similarly it might also happen that you know sometimes you know we would like to drop the stop index. So, one of the these 2 indices are required. So, in the next code example what we have `V3` and then we have the start index 6 and colon and then no stop index.

So, what will happen is that would be by default will be taken at you know till the end of the list. Similarly in the previous case when we had you know when we did not mention the start index it was by default was taken from the start of the list. So, in this case end of the list. So starting from the index number 6 to the end of the list if I run this. So, you can see `V3 6` and we got these just you know these 2 elements 53 and 75 right.

So, you can see in the output number you know 50 and compare it bit here. So, you can see that this is what we have got. So, start index is included. So, that is why we have got 53, then till the end of the list that is 75, it is also possible that in you know certain scenarios we would like to use the negative index you know for a certain situation. So, that kind of slicing can also be performed.

So, in the next code example you can see that we are using `V3` and in the start index we have

used the negative value that is -3 and then colon. So, what happens when we do this let us execute this code example first. So, you can see in the output, now compare the output 55 and output 52. So, output 52 is the original list of V3 and you can see that in the line of code number 53 we have mentioned -3.

So, it is you know it is being counted from the end of the list. So, from the end of the list 3 elements are being sliced from the original list. So, that is why we are getting 23 53 and 75. So, in this fashion we can use negative indexing also and from the end of the list those you know that particular operation is going to be performed. Let us understand this thing you know again with one more example. So, in this new example you know in the this code example 59 we have V3.

And the start index is -5, then colon and they stop index is -2. So, again the you know numbering is in a sense you can consider like numbering is starting from the end of the list and you know from the end of the list we go back to the you know we count up to 5 and then. So, in this fashion if we look at the original list that is there in u have 52. So, you can see 17 19 and 23 then will have to stop because they stop index is -2.

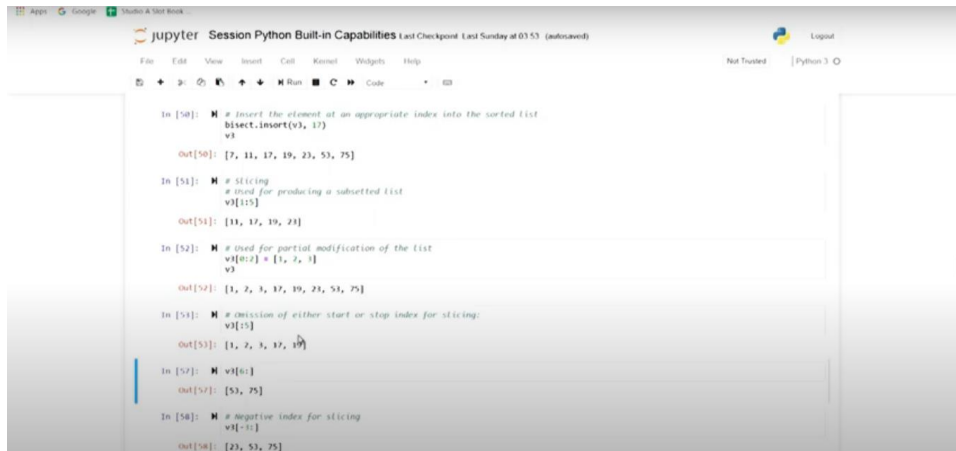
So, let us run this and you can see we have got these 3 elements 17 19 and 23. So, in this fashion the slicing in a particular list can be performed and you know it is a quite useful when a while we are you know processing data which we typically require for most of the analytics problem. Now there might be certain situation where we would like to follow a pattern to perform our slicing something like that can also be performed.

Let us understand this aspect through an example. So, let us you know look at this you know V3 variable once again. So, let us execute this. So, this is our V3 list we have these you know elements 8 elements 1 2 3 17 19 23 53 and 75. So now if we want every second element after start so how we can achieve that. So, these index is going to be like this V3 and in the brackets or operator within the brackets operator we can type double colon and 2. So, it will say that we can pick you know after this start we can pick every second element.

So, you can see if I run this, if I execute this line of code you can see the starting element of the

list V3 has been is the first element in this output 1 and then every second element after start. So we got 3 and then you know 19 and then 53. So, following this pattern alternate index you know alternate element is being produced in the output. Similarly we can take another code example where we would like to pick up the you know every third element after start.

(Refer Slide Time: 06:38)



```
In [50]: # insert the element at an appropriate index into the sorted list
bisect.insort(v3, 17)
Out[50]: [7, 11, 17, 19, 23, 53, 75]

In [51]: # slicing
# used for producing a subsetted list
v3[1:5]
Out[51]: [11, 17, 19, 23]

In [52]: # used for partial modification of the list
v3[0:2] = [1, 2, 3]
v3
Out[52]: [1, 2, 3, 17, 19, 23, 53, 75]

In [53]: # omission of either start or stop index for slicing:
v3[5:]
Out[53]: [17, 19, 23, 53, 75]

In [57]: v3[6]
Out[57]: [53, 75]

In [58]: # negative index for slicing
v3[-1:]
Out[58]: [23, 53, 75]
```

So, the code can be appropriately changed. So, in the V3 and within the brackets we can type double colon 3. So, every third element after the start element is going to be part of this slicing you know sliced you know substitute list. So, you can see 1 is there then you know third element 2 3 17 is there, then 19 23 and 53 is here. So, if I run this you can see this output is there. Now slicing can also be used to achieve certain other you know operations, for example we were seeing a list.

So, that is also possible. So, let us again have a look at the values of this list V3 you can see the same elements. Now if we want to achieve the reversing of this list we can use this particular index in the code line number 66 V3 and within double brackets we can mention double colon and then -1. So, this -1 is indicating that the list is to be reversed. So, if I execute this line of code you can see in the output the whole list has been reversed and we have got 75 53 23 in that reverse order you know of the original list V3.

Because we have talked about that list and tuples they are you know semantically you know they are very similar. So, this reversing is something that we can also perform in the tuple also, this a tuple are immutable objects. So, but the word same thing you know can be performed because it doesn't modify we get a new tuple actually. So, let us take this you know tuple 6. So, let us see.

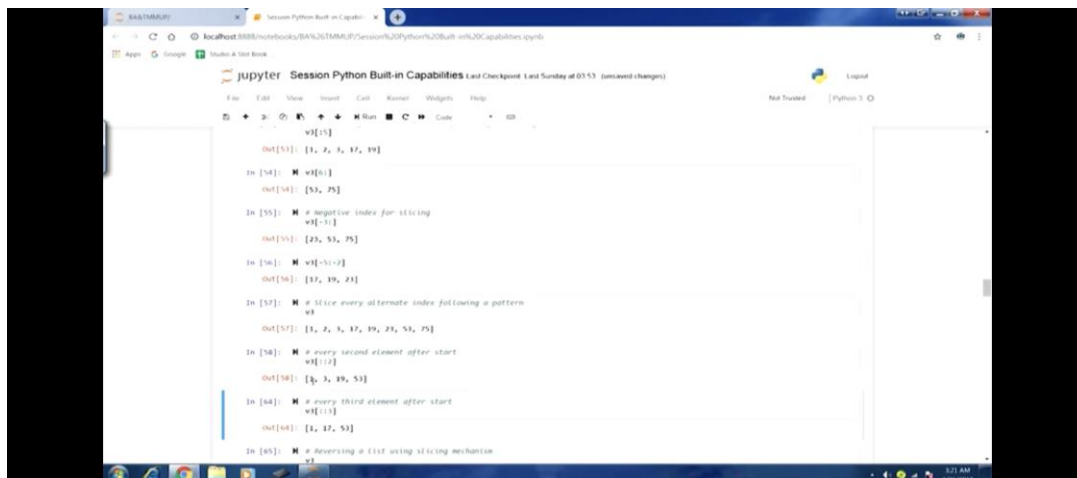
So, these are the elements in this tuple 6 1 2 3 4 5 6 7.

And if I use the same you know you know kind of syntax here for you were seeing the tuple tup 6 and within the brackets I have double colon -1 and if I run this you can see the tuple has been reversed. So, now let us move to these some of the important some of the useful sequence function that you know in certain scenarios they are very useful. So, let us start with this function called enumerate.

So, this function this particular you know function enumerate is actually a generator. So, it will actually materialize whenever we are using it with you know a list or for loop we will understand that with an examples, we will also talk about few other functions which also materialize when whenever they are used with list or for loop. So, some of these functions are quite useful in certain scenarios. So, we will you know learn them through some of the code examples.

So when this can be useful. So, can be used to track the current iteration number. So, many times you know there are situation where you would like to track I know which particular iteration is running. So, first let us go through the typical approach that we adopt. So, let us take this example of sequence. So, in this we have these values 7 2 6 4 9 and then we have a variable I 0 in each slice at 0.

(Refer Slide Time: 15:27)



```
v[15]
Out[15]: [1, 2, 3, 17, 19]

In [14]: w[0]
Out[14]: [53, 75]

In [15]: # negative index for slicing
w[-1]
Out[15]: [29, 53, 75]

In [16]: w[-1:-2]
Out[16]: [17, 19, 21]

In [17]: # slice every alternate index following a pattern
w
Out[17]: [1, 2, 3, 17, 19, 21, 53, 75]

In [18]: # every second element after start
w[1::2]
Out[18]: [2, 3, 19, 53]

In [19]: # every third element after start
w[2::3]
Out[19]: [3, 17, 53]

In [20]: # Reversing a list using slicing mechanism
w
```

And this is actually is we are going to use to you know keep track of the iteration number in the for loop. So, in we have very simple for loop here just to demonstrate this you know this scenario So, for X and sequence and X simple computation X is being assigned $x+I$ and then I is being incremented you know by 1. So, if I run this you can see the final value of I is 5 because you know 5 number of iterations have been performed at any particular point of time we can print this within the loop also.

And we can find out which iteration is running. So, in a sense I can be used to track the iteration number, if you look at the value of x so this 13. Now the same thing can be achieved much easily using enumerate function. So, we will have to just type fewer lines of code here and more structured manner. So, here you can see the loop goes like this. So, enumerate function. So, when we pass on this list sequence to enumerate function in the for loop because this is being used in the follow-up it will return a sequence of tuples. Sequence of tuples in the sense I'm running this loop like for Ix x in enumerate.

So, it will the tuple will have the index and the same order the value in the this particular sequence. So, if I run this particular for loop you can see and I, the value of I is now 4 because the again indexing is you know 0 indexing is used. So, it will start from 0 1 2 3 4. So, after 5 iteration the value of I is going to be 4. So, the tuples would be 0 and you know the value of you know x and the sequence list is going to be 7 0 7.

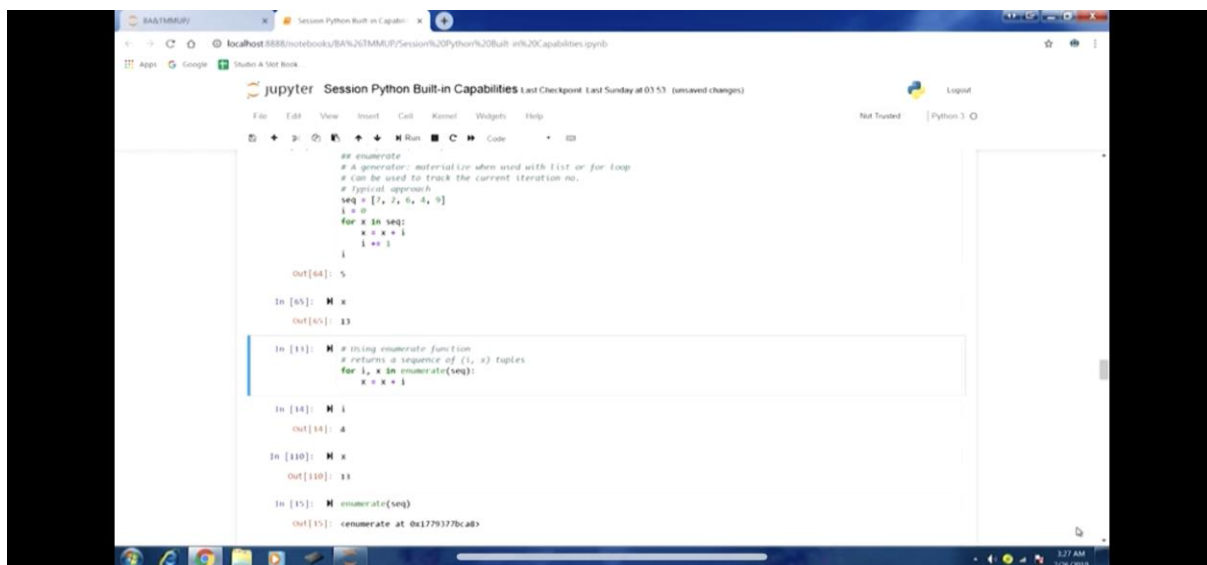
Then 1 2, then 2 6 and 3 4 and then 4 9. So, these are the tuples that are going to be written by enumerated function and you know accordingly the loop will run and if you look at the value of x there is no change the value comes out to be you know 13. Now if we just execute enumerate sequence to find out the output what is happening. So, we do not get any output here, but if you used it with list we will get what we are I was talking about that the enumerator is returning tuples.

So, you can see in the output number 7 0 7 1 2 2 6 3 4 and 4 9. So, enumerate function is returning these tuples and then this you know in the far loop I,x that is also in a sense forming a tuple. So, the iteration it just like we talked before that easily this iteration can be performed

using this function. Now this particular numerate function can also be useful in certain other scenarios, for example when we want to map the indices you know with the values.

So, the same thing you know in the previous output number 70 you can see in a sense and mapping has been done between indices and the element values. So, the same thing can be done will in a more structured manner So, let us take this example this list V4. So, if I run this you can see this V4 is a list of these string elements string values ohm namah shivay chanting So, these are the 4 elements.

(Refer Slide Time: 20:02)



```

# enumerate
# a generator: materialize when used with list or for loop
# can be used to track the current iteration no.
# typical approach
seq = [7, 7, 6, 4, 5]
i = 0
for x in seq:
    x = x + 1
    i += 1

Out[64]: 5

In [65]: x
Out[65]: 13

In [13]: # using enumerate function
# returns a sequence of (i, x) tuples
for i, x in enumerate(seq):
    x = x + 1

In [14]: i
Out[14]: 4

In [110]: x
Out[110]: 13

In [15]: enumerate(seq)
Out[15]: <enumerate at 0x179377bcad>

```

And now I am introducing another data structure in the python, a very important data structure that is dict and this dict variable that I have mapping and with the you know we are defining or in slicing and predict here and with the curly braces more detail on it we will discuss you know in the in the coming lectures. So, once we initialize this. So, dict is a sense a collection and this can really be used to perform this kind of mapping.

So, let us go through this example just to demonstrate how enumerate can be really useful in this mapping situation. So, for i, x and enumerate V4. So, when we say enumerate V4 it will again return us the sequence of tuples and then easily the for loop is going to be iterated and you can see we are just creating this mapping. So, within the for loop the only line of code that we have

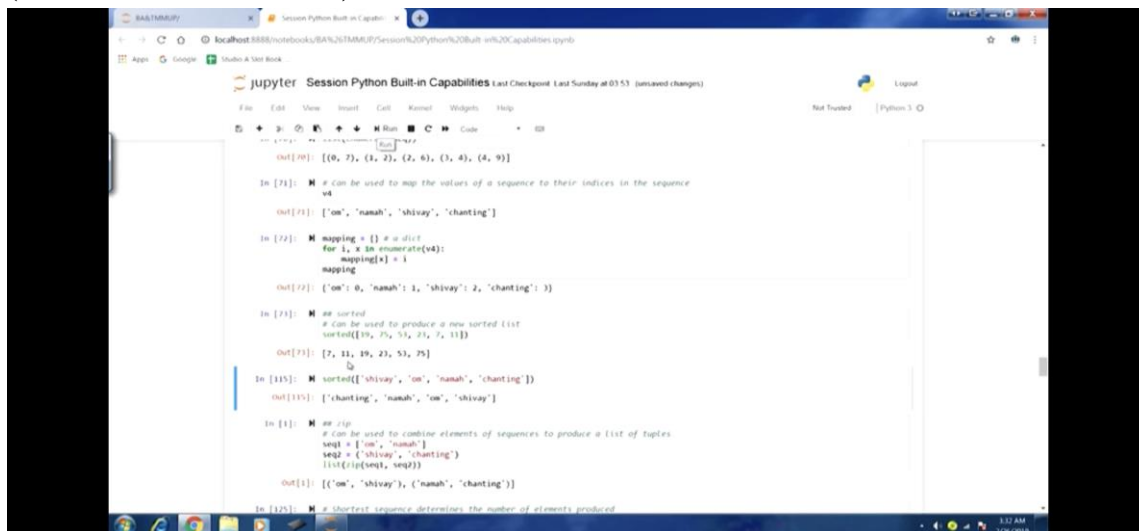
yes mapping x with I.

So, indices are being mapped with this you know these values. So, if I run this you can see in the output this is the output that you are getting is actually this data structure type dict d i c t and you can see a colon separator is there in the first you know element that is there in the dict and you know the first we have ohm and then colon and the 0. So, in a sense this is a key value kind of a round combination we will discuss in order to in more detail later.

So, you can see the next mapping is number 1 then see y2 and chanting 3. So, you can see enumerate can be really useful to perform this kind of mapping. So, let us move on. So, next useful function now is sorted. So, now this can be used to produce a you know a new sorted list. So, let us say we have this list 19 75 53 23 7 and 11 and we can use this sort and function and the output would be a sorted list.

So, sometimes this function can also be useful you can see in the output. Similarly if we take another code example here we are passing a list of string elements and you can see **shivayya** om namah and chanting and if we use the sorted function you will see the output would be based on the you know first letter the alphabetical sorting has been performed here. So, you can chanting gone first namah and om and shivay. So, first later of you know these string values. So, the list you know has been sorted in that particular fashion.

(Refer Slide Time: 23:00)



```
Out[10]: [(0, 7), (1, 22), (2, 6), (3, 4), (4, 9)]

In [7]: # x can be used to map the values of a sequence to their indices in the sequence
        # via
Out[7]: ['om', 'namah', 'shivay', 'chanting']

In [2]: # mapping = {} # = dict
        # for i, x in enumerate(v4):
        #     mapping[i] = i
        # mapping
Out[2]: {'om': 0, 'namah': 1, 'shivay': 2, 'chanting': 3}

In [3]: # # sorted
        # # can be used to produce a new sorted list
        # sorted([19, 75, 53, 23, 7, 11])
Out[3]: [7, 11, 19, 23, 53, 75]

In [15]: # sorted(['shivay', 'om', 'namah', 'chanting'])
Out[15]: ['chanting', 'namah', 'om', 'shivay']

In [1]: # # zip
        # # can be used to combine elements of sequences to produce a list of tuples
        seq1 = ['om', 'namah']
        seq2 = ['shivay', 'chanting']
        list(zip(seq1, seq2))
Out[1]: [('om', 'shivay'), ('namah', 'chanting')]

In [15]: # # shortest sequence determines the number of elements produced
```

So, sometimes this sorted function can be really useful. Now let us talk about another useful function that is zip. So, we talked about concatenating tuples and you know lists in this lecture in previous lecture as well. Now let us take you know a one list and one tuple and we will see how zip can be you know used to you know combine elements. So, concatenation was one thing now what here we are doing we are combining elements in a sense we are creating element pairs.

So, in the sequence 1 we have 2 elements ohm namah and in sequence 1 we have 2 elements om namah and sequence 2 we have again just 2 elements shivay and chanting. Now when we use the zip function main idea is to combine first element of sequence more with the first elements of all element of sequence 2. Similarly second element of sequence 1 with the second element of sequence 2 so on and so forth.

So this kind of situation is required many times. So, we can use zip function in this. So, again let us run this block of code and you can see from sequence 1 and sequence 2 using zip function where we are passing these 2 sequences as arguments and return you know value of this is a function from that we are getting a list. So, you can see this list is made of 2 tuples.

So, first tuple of is has combine first element from sequence 1 and first element from sequence 2 ohm shivay in 1 tuple and the second element we have combine second element of sequence 1 and second element of sequence 2, you can see namah chanting. So, in this fashion zip can be used to combine elements sequentially. Now it might so happen that these sequences might have you know different number of elements within them.

(Refer Slide Time: 24:56)

```
seq1 = ['om', 'shivay', 'namah', 'chaitanyam']

In [1]: # sorted
# Can be used to produce a new sorted list
sorted([19, 75, 55, 23, 7, 11])

Out[1]: [7, 11, 19, 23, 55, 75]

In [2]: # sorted(['shivay', 'om', 'namah', 'chaitanyam'])
sorted(['shivay', 'om', 'namah', 'chaitanyam'])

Out[2]: ['chaitanyam', 'namah', 'om', 'shivay']

In [3]: # zip
# Can be used to combine elements of sequences to produce a list of tuples
seq1 = ['om', 'namah']
seq2 = ['shivay', 'chaitanyam']
list(zip(seq1, seq2))

Out[3]: [('om', 'shivay'), ('namah', 'chaitanyam')]

In [4]: # Shortest sequence determines the number of elements produced
seq1 = ['om']
list(zip(seq1, seq2, seq3))

Out[4]: [('om', 'shivay', True)]

In [5]: # Can be used to simultaneously iterate over multiple sequences
for i, (a, b) in enumerate(zip(seq1, seq2)):
    print('({0}): {1}, {2}'.format(i, a, b))

0: om, shivay
1: namah, chaitanyam

In [6]: # unzipping the sequence
```

So, in that case the sorted sequences, sequence having the least number of elements that will determine the you know final number of elements combine elements to be produced. So, let us take another example. So, here we are you know creating another sequence 3 which is having just 1 element that is you know a bool element, bool value true and again if we combine these 3 sequences sequence 1 sequence 2 sequence 3.

Let us run this, you can see in the output we have got just one tuple just one element which has combined the first element of each of these 3 sequences. So, we have got om shivay and true and that is the you know first element over just 1 tuple element in that list. Now let us move on, now we can use zip function along with enumerate function to simultaneously iterate over multiple sequences.

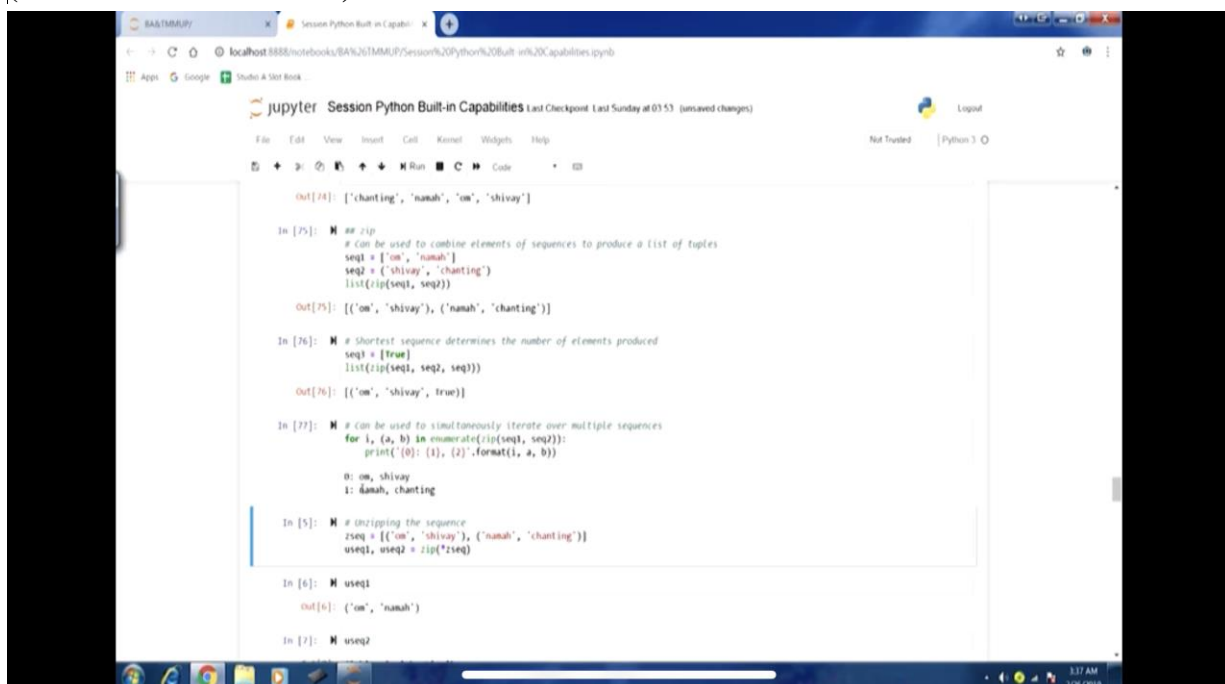
So, zip function in a sense will give us this you know combined elements and enumerate is any way as we have learned it can help us to iterate over multiple I know values. So, let us take this example here we are running a for loop here in this code example and we want to simultaneously iterate over multiple sequences. So, you can see first we have I that is for the you know index and then we have a tuple a,b and then we have the iterator here, we have enumerate and we are zipping these 2 signature sequence 1 and sequence 2.

So, what will happen we will have these you know the output as zipped sequence in this will get tuple in a sense list of tuple we will be iterating over this. So, in the only line of code that we have within for loop we have the print statement and you can see it has been formatted in the in a sense that we want we would like to print these values of a and b in all the iterations. So, let us run this.

So, you can see first iteration we have 0 then om and a value will be om and then shivay. So, please refer to the output that we have in the 75. So, here you can see we have 2 tuples. So, a and b has been assigned the first tuple, then in the next iteration a and b would be assigned the next tuple that is namah chanting. So, that is why in the first output you get 0 colon and then om namah shivay and then next output 1 and then namah chanting.

So, in this fashion in a sense we have been able to simultaneously iterate over multiple sequences. Now it is also possible to unzip the sequence. So, we might have a list of tuples where the tuple might have been achieved by combining sequences. So, let us take this example where we have z sequence where this is a list and list of tuples. So, in the first of all we have 2 elements and in the second tuple also we have 2 elements. Now we can use this zip function again.

(Refer Slide Time: 27:33)



```
Out[74]: ['chanting', 'namah', 'om', 'shivay']

In [75]: # zip
# can be used to combine elements of sequences to produce a list of tuples
seq1 = ['om', 'namah']
seq2 = ['shivay', 'chanting']
list(zip(seq1, seq2))

Out[75]: [('om', 'shivay'), ('namah', 'chanting')]

In [76]: # Shortest sequence determines the number of elements produced
seq3 = [True]
list(zip(seq1, seq2, seq3))

Out[76]: [('om', 'shivay', True)]

In [77]: # can be used to simultaneously iterate over multiple sequences
for i, (a, b) in enumerate(zip(seq1, seq2)):
    print('{0}: {1}, {2}'.format(i, a, b))

0: om, shivay
1: namah, chanting

In [5]: # unzipping the sequence
zseq = [('om', 'shivay'), ('namah', 'chanting')]
useq1, useq2 = zip(*zseq)

In [6]: # useq1
Out[6]: ('om', 'namah')

In [7]: # useq2
```

This index is slightly different, now in the zip function as an argument we are passing you know we are using asterisk operator, asterisk z z s e q. So, this is the you know the combined you know zipped sequence that we have and using this asterisk operator we can indicate that we would like to unzip. So, because each tuple is having 2 elements. So, we can again in the unzipped sequence we can assign like this in the left hand side you can see u cq1 and you scq 2 to indicate the unzipped sequence.

If I run this and let us check the values of first unzipped sequence u sq1 om namah and then you can see the second unzip sequence that first element om and namah they had been you know club and the original sequence has been restored and in this you know second sequence also the same thing shivay chanting have been clubbed and the original sequence 2 has been restores, in this fashion we can unzip a particular given sequence as well. Now let us move to the next useful function that is reversed. So, this is also in a way generator.

So typically useful when we are using it with the list or for loop. So, again this can also be used to iterate over elements you know of a sequence in reversed order. So, as the name is listing. So, you know whatever sequence that we might have we can iterate in a reverse order. So, let us run this reverse range 10 if we want to produce this sequence wc you can see we do not get you know any meaningful output here.

But if I use it with the list you will see that typically if you apply the instant it will get the output from you 0 to 9 ,but here because we are using the reverse function you will see that the reversed output from 9 8 7 6 this reverse sequence has been generated. So, the next you know with this we conclude our discussion on the list and you know tuple and lists. Now we come to the next important data structure that is used in python that is dict.

(Video Ends: 28:07)

So, we have used this before also in this lecture and this is considered to be the most important one and this is also called hash map or associative array and we can define this as a you know flexible sized mutable collection. If you remember that previously when we talked about the

mutable and immutable objects we mentioned you know dict as you know the immutable objects. So, this is a flexible sized mutable collection of key-value pairs as you have seen in this lecture itself you know with the colon we combine key value. So, we get key value pairs where a keys and values are python objects. So, at this point we would like to stop here and we will continue our discussion on dict in the next lecture, thank you.

Keywords: text mining, python, enumerate(), slicing, dict(), Prediction, Classification, Attributes, Regression.