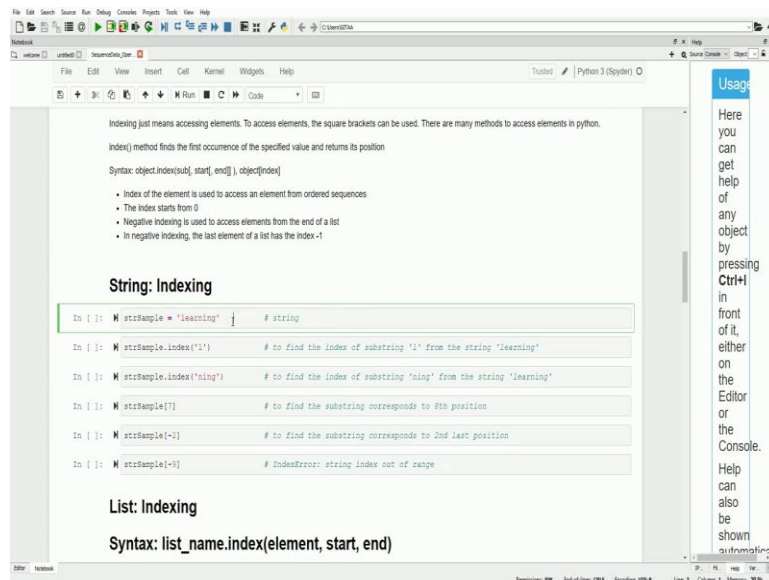


**Python for Data Science**  
**Prof. Ragunathan Rengasamy**  
**Department of Computer Science and Engineering**  
**Indian Institute of Technology, Madras**

**Lecture – 9**  
**Sequence Data Part 2**

Hello all and welcome to the lecture on the sequence data operations. So, in the previous lecture we have seen what sequence data is. And we were also seen how to create sequence data for example we have seen about creation of strings, lists, tuples, dictionary, sets and range. And in this lecture, we are going to see how we can perform some of the operations on sequence data.

**(Refer Slide Time: 00:39)**



For example we are going to see how we can do indexing on a sequence data. So your let us see what indexing means first. So indexing is just means accessing elements from your sequence data. So, to access the elements you can use square brackets. And to access any elements from the given sequence you should be, knowing the index of it because all the elements inside the sequence data will be arranged in a particular order.

And every element will have its own index. So, if you want to access elements from the sequence then it should be, knowing its corresponding index to access it from. So to access elements you can use the square bracket. And there are other methods that can be used access in Python as well. Also, we are going to look at a method called index which can be used to find the

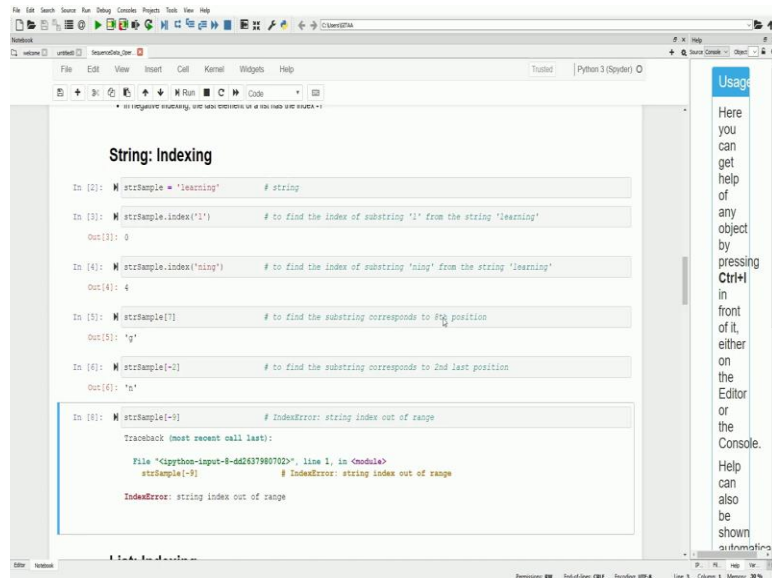
position of a given element.

For example, the index method find the first occurrence of the specified value and it is going to return its position. If an element is occurring multiple times in a sequence data and it is going to give you the index data for the first occurrence alone. So let us see how to use the index method. So you can apply this index method on any object on any sequence data. So for example of given the syntax here to use the index method.

So you can give the substring inside the index method. We will see an example on how to use the index method as well. You can also use square brackets to access the elements from a given sequence of data and you can give the index inside the square bracket. So it is going to give you the element corresponding to the given index. And as we know that index of an element is used access an element from ordered sequences. So in Python generally the indexing starts from 0. So, for example if you want access elements in the order sequence, then you can use the index starting from zeroes.

The first element has the index 0 but in a case where you want to access elements in reverse order, for example, if you want to access elements from the end of a list then you can look for negative indexing where the negative indexing starts from -1. For example, the last element in an ordered sequence to have the index -1 correspondingly the second last element will have the index -2. So let us see how to index and element from a string.

**(Refer Slide Time: 03:21)**



The screenshot shows a Jupyter Notebook window with a title bar and menu options. The notebook content is titled "String: Indexing" and contains several code cells. The first cell defines a string variable `strSample = 'learning'`. The second cell uses `strSample.index('l')` to find the index of the character 'l', returning 0. The third cell uses `strSample.index('ning')` to find the index of the substring 'ning', returning 4. The fourth cell uses square brackets `strSample[7]` to access the character at index 7, returning 'g'. The fifth cell uses negative indexing `strSample[-2]` to access the second-to-last character, also returning 'g'. The sixth cell attempts `strSample[-9]`, which results in an `IndexError: string index out of range`. A traceback is shown for this error, indicating it occurred in the file `ipython-input-8-d4d637980702d.py` at line 1. On the right side of the notebook, there is a "Usage" sidebar with text explaining that help can be accessed by pressing `Ctrl+I` in the editor or the console.

```
String: Indexing

In [2]: strSample = 'learning' # string

In [3]: strSample.index('l') # to find the index of substring 'l' from the string 'learning'
Out[3]: 0

In [4]: strSample.index('ning') # to find the index of substring 'ning' from the string 'learning'
Out[4]: 4

In [5]: strSample[7] # to find the substring corresponds to 8th position
Out[5]: 'g'

In [6]: strSample[-2] # to find the substring corresponds to 2nd last position
Out[6]: 'g'

In [8]: strSample[-9] # IndexError: string index out of range
Traceback (most recent call last):
  File "ipython-input-8-d4d637980702d.py", line 1, in <module>
    strSample[-9] # IndexError: string index out of range
  IndexError: string index out of range
```

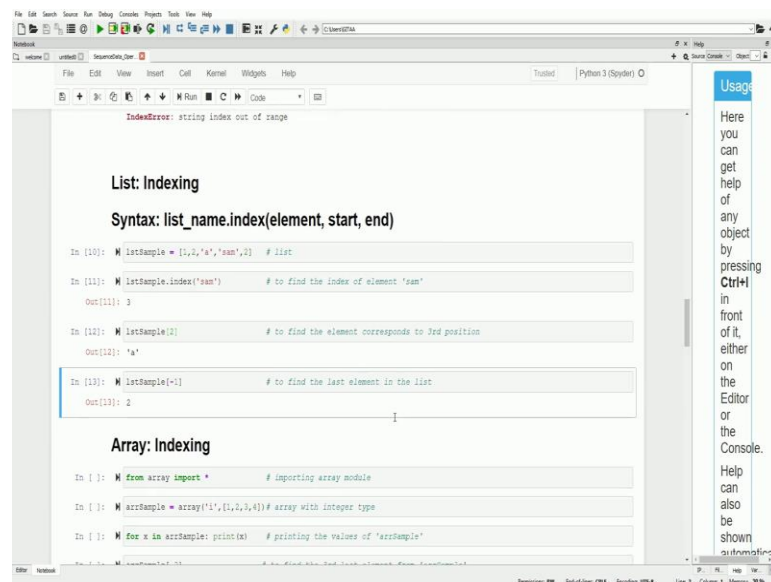
Basically, we have already created a string `strsample` which has a string as `learning`. So in the previous lecture we have already seen how to create a string. So I am just using the same example, but I am going to take you through an example on how to index or how to access elements from the particular string. As I have already used `strsample` string. So now I am trying to use the index method to find the index of the substring `l` from the string.

So, the output 0 since the index of the `l` is 0. For example, if you are trying to give sequence of values inside the function `index`, then it is going to give you the index of the first letter of your substring. So, your `n i n g` is a substring from the original string `learning` and I have given that as a sequence here, but the output is 4. The 4 corresponds to the index of the value `n` here. If you look at `n` in the `learning` string index corresponding to `n` is 4 so that is the output that we are getting here.

But if you want to find the substring corresponds to the given position then you can use a square brackets. So I am using the square brackets followed by the string name `strsample` and given the index as 7. So, it is going to give me an element which corresponds to the seventh index, which is nothing but `g`. For example, if you want to find the substring that corresponds to the second last position. For example I am accessing the substring from the end of a string then I can use negative indexing.

So I have given -2 and output is n because the second last element of the string is n. And if I try to give -9 then it is going to throw me error same index error string index out of range because if you want to access the last element then the index corresponding to the last element is only -8 you do not have any element which corresponds to -9 at all. If you are trying to give an index which is out of range then it is going to throw you an error.

**(Refer Slide Time: 05:37)**



The screenshot shows a Jupyter Notebook interface with the following content:

```
File Edit View Insert Cell Kernel Widgets Help
In [10]: list_sample = [1,2,'a','sam',3] # list
In [11]: list_sample.index('sam') # to find the index of element 'sam'
Out[11]: 3
In [12]: list_sample[3] # to find the element corresponds to 3rd position
Out[12]: 'a'
In [13]: list_sample[-1] # to find the last element in the list
Out[13]: 3
```

**List: Indexing**

**Syntax: list\_name.index(element, start, end)**

**Array: Indexing**

```
In [ ]: from array import * # importing array module
In [ ]: arr_sample = array('i',[1,2,3,4]) # array with integer type
In [ ]: for x in arr_sample: print(x) # printing the values of 'arr_sample'
```

So here we see how to find the index of a substring using the index method and we have also seen how to find the substring corresponds to the given position that is given index. And we have also seen how to find the substring corresponds to the given index using square brackets. Similarly you can apply this onto any sequence data. For example, we can apply this list and other sequence data. So let us see a few examples on how we can use this on other sequence data.

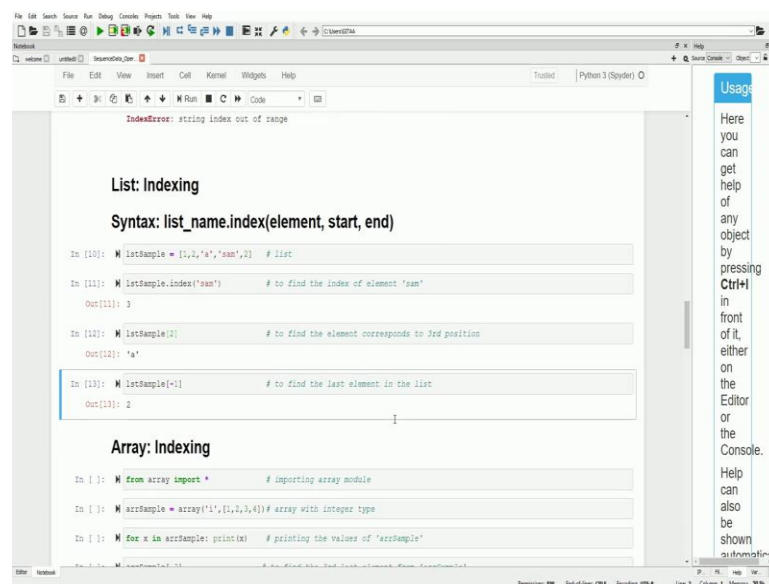
So here I am just going to illustrate you the indexing part using the list sequenced data. So syntax is given here you can use the list name and you can use the index function. And inside the function we can just give the element for which you want to find the position. So I have already created a list call list listsample which has few elements to it. It has both numbers and strings and to find the index of the element sam I can use the index method.

Inside the method I can just give the element sam. So that gives me an output three because index corresponds to sam is 3. The index corresponds to 1 is 0 and the index corresponds 2 is 1

and index corresponds to a is 2 and index corresponds to sam is 3. But if you are going to find the element corresponding to a given index, then you can get that using square brackets. So let us just try that I am using lissample of two inside the square bracket.

So it is going to give me an element which response to the index 2 which is nothing but a here. Similarly you can access any elements from the end of a list as well using the negative indexing. So if I give lissample of -1 is going to give me the last element of the list lissample. So which is nothing but 2.

**(Refer Slide Time: 07:37)**



The screenshot shows a Jupyter Notebook interface with a code editor and an output area. The code editor contains the following text:

```
IndexError: string index out of range
```

**List: Indexing**

**Syntax: list\_name.index(element, start, end)**

```
In [10]: lissample = [1,2,'a','sam',2] # list
In [11]: lissample.index('sam') # to find the index of element 'sam'
Out[11]: 3

In [12]: lissample[2] # to find the element corresponds to 3rd position
Out[12]: 'a'

In [13]: lissample[-1] # to find the last element in the list
Out[13]: 2
```

**Array: Indexing**

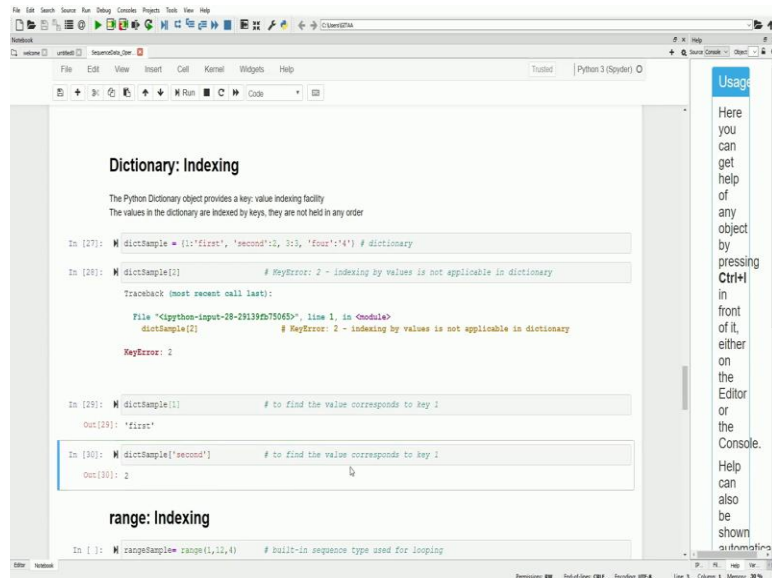
```
In [ ]: from array import * # importing array module
In [ ]: arrsample = array('i',[1,2,3,4]) # array with integer type
In [ ]: for x in arrsample: print(x) # printing the values of 'arrsample'
```

The output area shows the results of the code execution: 3, 'a', and 2.

Similarly we can perform the indexing on an array. So we have already seen how to create an array. So let us just use the same example. So I print out the values of array. The array sample has values 1, 2, 3 and 4. So, here I just have an example to show you how you can perform the negative index on this array. I am accessing the third last element of the array so it is going to give me the value 2 which is nothing but the third last element of the array.

And similarly we can perform that on a tuple. So you can use the index method to find the position of the given elements. So let me just use py so the index of py is 5. So here I am just trying to find the element which responds to the second index which is 3 here. So, let us see how the indexing works in set.

**(Refer Slide Time: 08:47)**



The screenshot shows a Jupyter Notebook interface with a code cell containing the following text:

```
Dictionary: Indexing

The Python Dictionary object provides a key-value indexing facility.
The values in the dictionary are indexed by keys, they are not held in any order.

In [27]: dictSample = {'first': 1, 'second': 2, 3: 1, 'four': 4} # dictionary

In [28]: dictSample[2] # KeyError: 2 - indexing by values is not applicable in dictionary

Traceback (most recent call last):
  File "C:\python-input-28-29139db75065", line 1, in <module>
    dictSample[2] # KeyError: 2 - indexing by values is not applicable in dictionary
KeyError: 2

In [29]: dictSample[1] # to find the value corresponds to key 1
Out[29]: 'first'

In [30]: dictSample['second'] # to find the value corresponds to key 1
Out[30]: 2

range: Indexing

In [ ]: rangeSample = range(1,11,4) # Built-in sequence type used for looping
```

On the right side of the notebook, there is a vertical sidebar with a blue header that says "Usage:" and some text that is partially visible: "Here you can get help of any object by pressing Ctrl+I in front of it, either on the Editor or the Console. Help can also be shown asynchronously."

So we were already created a set sample which has example 24, 87.5 data 24 and data as element to it. I am trying to use an index call 4 which gives me an error saying set object is not subscriptable which means that you cannot perform the indexing operation set. As we know that set is container to hold non sequential data. You will not be able to access the elements inside the set using indexing.

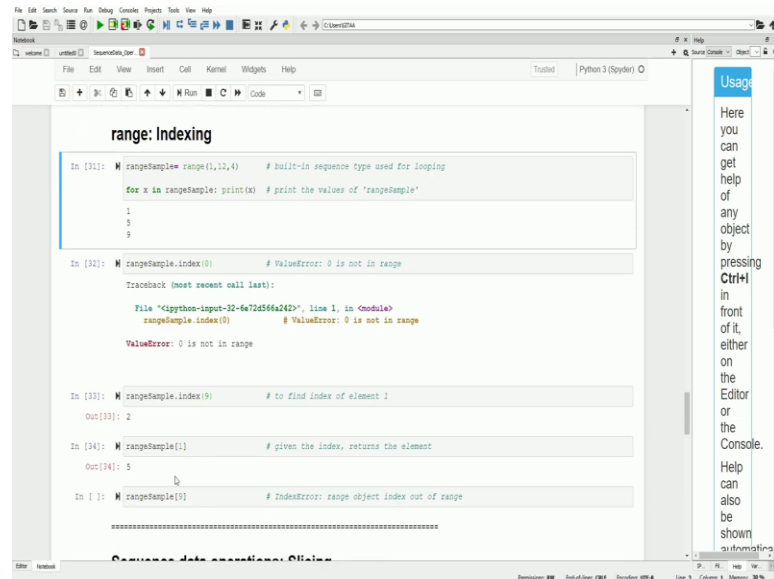
So next we are going to see how we can perform indexing on a dictionary. So, Python dictionary object provides a key-value pair facility. So the values in the dictionary are indexed by keys and they are not held in any order. So, if you want to access the values from the dictionary then you can access them by using their keys. So we have dictionary called dict sample which we have already created which has keys as one second 3 and 4 and it also has a key value corresponding to it.

For example first, 2, 3 and 4 so now let us see what happens when you are trying to access an element using an index I am using index say for example that is 2 it gives you the key error that is 2 as we know that values in a dictionary are indexed by keys. So you will be able to access them using keys not using the index. So now I am using key inside my square braces. So I am going to get a value corresponding to the key that is 1.

So the value corresponding to the key one is first and as we know that he can may be a string or

integer value. So key can either have string as well as a integer. So here I have a key called second so I will be able to access the value corresponding to the key second. So I am getting output as 2. So, the value corresponding to key second is 2.

**(Refer Slide Time: 11:07)**



The screenshot shows a Jupyter Notebook interface with a code editor and a console. The code editor contains the following text:

```
range: Indexing

In [31]: M rangeSample= range(1,12,4) # built-in sequence type used for looping
          for x in rangeSample: print(x) # print the values of "rangeSample"
          1
          5
          9

In [32]: M rangeSample.index(0) # ValueError: 0 is not in range
          Traceback (most recent call last):
            File "C:\python-input-32-6e7d566a242d", line 1, in <module>
              rangeSample.index(0)
              # ValueError: 0 is not in range
          ValueError: 0 is not in range

In [33]: M rangeSample.index(9) # to find index of element 1
          Out[33]: 2

In [34]: M rangeSample[1] # given the index, returns the element
          Out[34]: 5

In [ ]: M rangeSample[9] # IndexError: range object index out of range

=====
Source data examples: Slides
```

The console on the right shows the output of the code, including the error messages for the invalid index operations.

So the last one example which is the last one is to illustrate the indexing using range. So we have already created range sample which contains a value is 1, 5 and 9. So, you should be using an appropriate value which is already there in your range if you are trying to give you a value which is not in the range then it is going to throw you a value error. We are trying to check the index corresponding to the value zero.

But in our case the range sample does not have the value 0 at all because it has only the values 1, 5, 9 in that case it throws a value error. So it says zero is not in range. But if you are trying to find the position of the given element that is 9 here you will be able to do that because we already have a range value that is 9 and it gives you the position of that particular element that is 2 and we can also get the elements given the index.

So, given the index as one, so it give the output as 5 because the index corresponding to the value one is zero and the index corresponding to the value 5 is 1. But if you are trying to give the index which is out of range then it is also going to throw you an error saying index error because you have an error in the given index that is why the error says it is index error and saying range

object index out of range. Because you do not have any element corresponding to the index 9 that is, why the error says index error.

So in this lecture we will see how to get the elements given an index. And also in this lecture we have seen how to get the elements given the index inside the square bracket and we also seen how to use the index method to find the position of the given element. We have seen the indexing in all the sequential and non sequential data and in that case we have seen how to perform an indexing in a sequential data.

And we will not be able to perform indexing on a non sequential data since the values inside the non sequential data not held in any particular order. So we cannot access them using indexing and the when dictionary is concerned, even though it is a container for non sequential data, you will be still be able to access the values inside your dictionary using keys the values in a dictionary are indexed by keys not by any index.

So, in the upcoming lectures we will be seeing more operations that we can perform on the sequence data. Thank you.