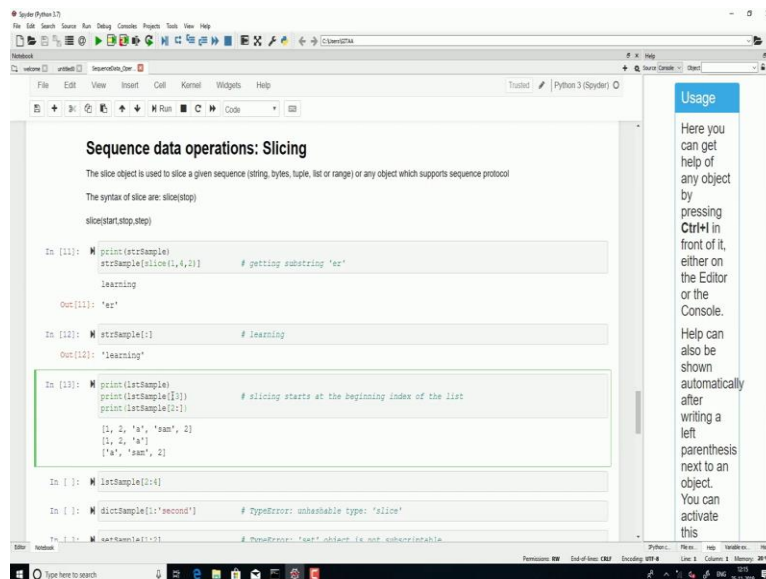


Python for Data Science
Prof. Ragnathan Rengasamy
Department of Computer Science and Engineering
Indian Institute of Technology, Madras

Lecture – 10
Sequence Data Part 3

Hello all welcome to the lecture on the sequence data and in this lecture we are going to look at some of the operations that can be performed on a sequence data.

(Refer Slide Time: 00:15)



Sequence data operations: Slicing

The slice object is used to slice a given sequence (string, bytes, list or range) or any object which supports sequence protocol

The syntax of slice are: slice(stop)

slice(start, stop, step)

```
In [11]: print(strSample)
strSample[slice(1,4,2)] # getting substring "ex"
learning
Out[11]: 'ex'
```

```
In [12]: strSample[1:] # learning
Out[12]: 'learning'
```

```
In [13]: print(listSample)
print(listSample[slice(1,4)]) # slicing starts at the beginning index of the list
[[1, 2, 'a', 'sam', 2]
 [1, 2, 'a', 'a']
 ['a', 'sam', 2]]
```

```
In [ ]: listSample[1:4]
```

```
In [ ]: dictSample[1:'second'] # TypeError: unhashable type: 'slice'
```

```
In [ ]: listSample[1:1:] # RuntimeError: 'list' object is not subscriptable
```

Usage

Here you can get help of any object by pressing **Ctrl+H** in front of it, either on the Editor or the Console. Help can also be shown automatically after writing a left parenthesis next to an object. You can activate this

So, let us get started with slicing. So, the slice object is used to slice a given sequence and the sequence can be string the sequence can be strings tuples, bytes, list or range or any object which supports sequence protocol. And the slice object represents the indices specified by the range function given the start stop and the step value. And the syntax of the slice operation the syntax for the slice object is given here and the syntax for the slice function is given here.

So, you can either use slice of stop or you can use slice of start stop and step. So, basically the slice function takes in three arguments which have the same meaning in both the constructs. So, the first one is start which is a starting integer value where the string which is an integer value which is an integer value which represents the starting from where the slicing of the objects.

So, the first argument is start. So, that is the starting integer where the slicing of the object starts

the next one is stop that is an integer until which the slicing takes place. So the slicing stops at the index $n - 1$ or the minus 1. So, for example I have given the second argument as 4 it means that is the stop value. So, it is going to give me the sequence of values still 3. So, next one is step an integer value which determines the increment between each index for slicing.

So, for example if you are giving 2, so, you will fetch the, so, you will be getting the values in the step size 2. So, by default if you do not give anything it is going to take it as 1 as the increment value but if you are providing any value then it is going to consider that as the increment value. So, let us just get started with the example on string. So, we have already created the string called strsample which has a string called learning. So, now my objective is to get the substring from a given string strsample from the given string learn from the given string learning.

So, we can use the slice object to do that. So, on strsample I am using the slice of I am using the slice function inside the square brackets then inside the slice function I am just giving three values one for start stop and the step value. So, here the output I have already printed that is er which is e and r from learning. So, basically we have got that by specifying the starting index as one. So, actually the slicing started from e and it will actually end at the index 4 then it is 2 3 and 4.

So, n will have the index 4 but actually it is going to give you till $n - 1$ right. So, it will stop at r and since I have given the increment value as 2 it is not going to give you all the sequence of sequence of strings from e as it is not going to give you the sequence of strings as e a r rather it is going to give you the sequence in terms of incrementing the strings by two. So, it is just going to give you as a r alone.

So, it is just going to give you the output as e r alone. So, this is how the slicing operator. So, this is how we use the slice function in order to slice the given sequence of elements or given sequence of substrings from the given string or the any sequence object. And the other method is the other method for slicing the elements or the substrings from a given object is using the square bracket. You do not have to use the slice function inside it you can either use the slice function or

the colon operator.

Here I am using just a colon inside the square bracket. So, that it gives me all the substrings from a string learning that is learning. So, just by giving just by just placing colon inside the square brackets fetches me the learning that is the complete set of strings from the main string that is strsample. So, this is how we perform the slicing on the string sequence data. So, basically we use slicing whenever we want to perform any operations on a particular set of elements on your string or on your any or it can be applied on any sequence data.

So, all the values from a list will not get modified based on your calculation or based on your operation only the particular set of elements or substrings will be modified based on the operations that you are performing on it. Similarly we can perform the slicing operations in other in any other sequence data for example the next example to illustrate you is on list. So, lists basically have a default bit of functionality when slicing.

So, here let us just print what is there under the list sample so these are the values these are the elements that is there under the lstsample 1 2 a sam and 2. So, now I am going to perform slicing on the list sample. So, basically if there is no volume before the first colon for example like this I have not given any value before the first colon and I have given value after the colon in that case it needs to start at the beginning index of the list.

It means to start the slicing at the beginning index of the list. So, let us just see what happens when we use colon 3. So, the output is given here. So, this the slicing started from the index 1. So, the slicing started from the beginning of the list that is from 1 itself and the next value is the stop value. So, 0 1 2 3 it is going to give you the sub elements from the list till or before sam. So, we have got the output as 1 2 and a.

Since the index the since the stop index is at 3 it is just going to give you a value it is just going to give you a sequence of elements or sequence of sub elements from your main list which is just before the third index that is 1 2 and a. The other thing to note here if you have not provided a value after the first colon like this then it means to go all the way to the end of the list starting

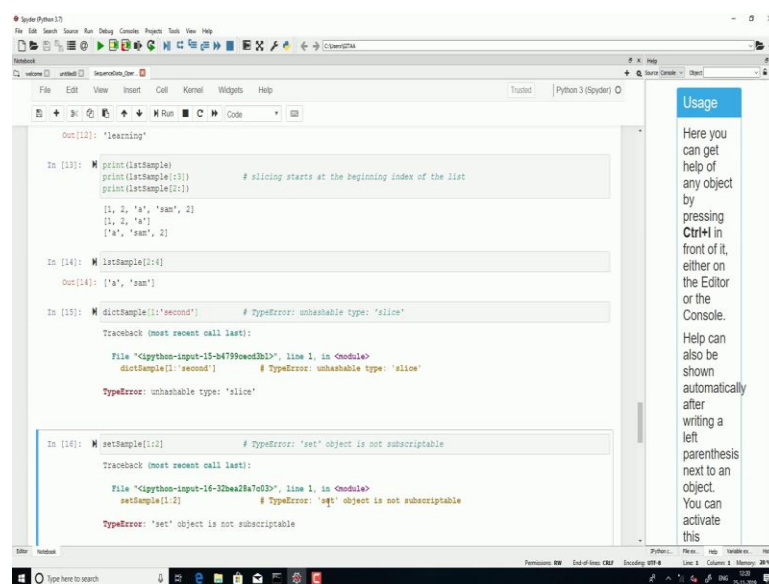
from the index 2. So, I am just saying to slice starting from the index 2 and to go all the way to the end of a list and this basically saves this time.

So, that we do not have to manually specify length of a which basically. So, this saves us time. So, that we do not have to manually specify length of len of lstsample in order to specify that is the end of a list or that is to specify the that is to specify or to check how many elements are there in a list. So, in order to get rid of that you can just use a colon after the you can you do not have to specify anything after the first column. So, that it automatically takes all the sub elements till the end of your list. So, this is how we perform the slicing operation on a list.

So, here if you have noted down I have not used the slice function rather I have just used the colon operator to slice the sub elements from the list by just separating by just giving the start stop and the step value in terms of using colon operator. And if you can recall here whatever order that you are giving here inside the slice option inside the slice function the same applies here but instead of using slice function I have just used the square bracket.

And if I give 2 colon 3 colon 1 then it means that starting while the starting index is 2 and the ending index is 3 and the increment value is 1 but you can as well specify without even giving any value just to denote that I need all the values from 2 and I need all the values till 3.

(Refer Slide Time: 09:54)



```
Out[12]: 'learning'

In [13]: print(lstsample)
print(lstsample[:3])
print(lstsample[2:])
# slicing starts at the beginning index of the list
[[1, 2, 'a', 'sam', 2],
 [1, 2, 'a'],
 ['a', 'sam', 2]]

In [14]: lstsample[2:4]
Out[14]: ['a', 'sam']

In [15]: dictsample[1:'second']
# TypeError: unhashable type: 'slice'

Traceback (most recent call last):
  File "C:\python-input-15-84799ced3b1>", line 1, in <module>
    dictsample[1:'second']
# TypeError: unhashable type: 'slice'

TypeError: unhashable type: 'slice'

In [16]: setsample[1:2]
# TypeError: 'set' object is not subscriptable

Traceback (most recent call last):
  File "C:\python-input-16-32bea2fa7cd3>", line 1, in <module>
    setsample[1:2]
# TypeError: 'set' object is not subscriptable

TypeError: 'set' object is not subscriptable
```

Usage

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 automatically after writing a left parenthesis next to an object. You can activate this

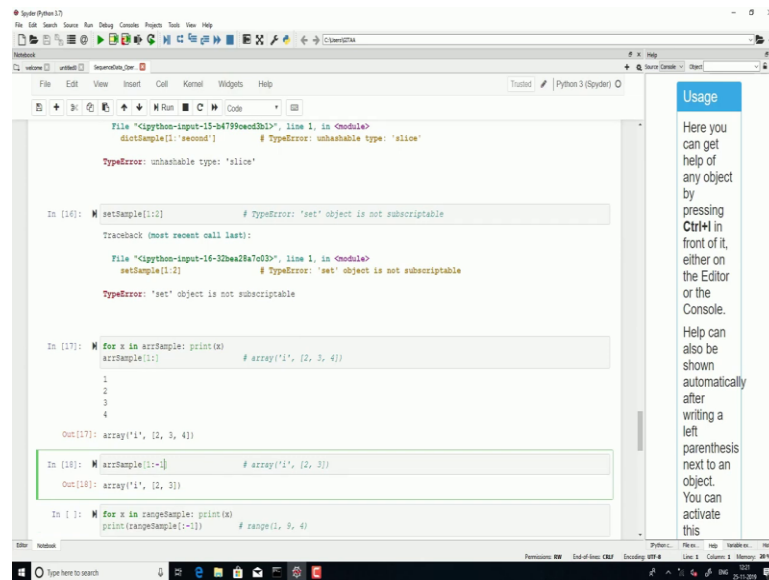
For example this is the start value and the stop value let us check what is the output? So, the slicing operation is being done from the index 2 of the list that is 0 1 2 from a until 4 3 and 4 and it is not going to consider the 4 as the index but it just considered till n minus 1. So, that will be 3. So, it is going to give you a and sam as the output. So, I hope this is clear. So, let us move on to performing the slicing on the other sequential data.

So, we have also created a dictionary sample right so let us just perform slicing on it but it gives you an error saying type error and hashable type that is slice. So, if you can recall the python dictionary object provides a key value indexing facility and the values in the dictionary are indexed by keys and they are not held in any order in that case we will not be able to perform slicing because it requires the index to slice the given sequence of values or given sequence of elements.

To slice the sequence of values or sequence of keys from your dictionary, so, slice. So, we cannot perform slicing on a dictionary because it is a non it is a container for non it is a container for holding non sequential data. So, in that case we will not be able to apply that onto a set as well. So, let us just see what happens when we apply that onto a set. So, it also throws an error saying set object is not subscriptable.

As we know that set is also a container to hold non-sequential data. So, all the elements inside the set are not indexed or not indexed by any indices rather they are not they are not held in any order. So, we cannot perform the slicing operation on a set also because even set is also a container for holding the non-sequential data in that case all the elements inside the set will not have any index for index corresponding to each element. So, we can perform the slicing in index as well.

(Refer Slide Time: 12:37)



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

```
File "C:\python-input-15-b4799cedb3d>", line 1, in <module>
dictSample[1]['second']
# TypeError: unhashable type: 'slice'

TypeError: unhashable type: 'slice'

In [16]: # dictSample[1:2] # TypeError: 'set' object is not subscriptable

Traceback (most recent call last):
File "C:\python-input-16-32ba28a7e03d>", line 1, in <module>
dictSample[1:2]
# TypeError: 'set' object is not subscriptable

TypeError: 'set' object is not subscriptable

In [17]: # for x in arrSample: print(x)
arrSample[1:] # array('i', [2, 3, 4])

1
2
3
4

Out[17]: array('i', [2, 3, 4])

In [18]: # arrSample[1:-1] # array('i', [2, 3])
arrSample[1:]
Out[18]: array('i', [2, 3])

In [ ]: # for x in rangeSample: print(x)
print(rangeSample[1:-1]) # range(1, 5, 4)
```

On the right side of the notebook, there is a 'Usage' sidebar with the following text:

Usage

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 automatically after writing a left parenthesis next to an object. You can activate this

So, the next one is slicing on the array. So, let us just print what we have it under the array sample which has the values 1 2 3 and 4. So, if you want to basically access or if you want to fetch or slice the values inside your array then you can just perform the same slicing operation on it I am just given one colon. So, that it gives me the output gives giving us all the values from the first index that is 2 3 and 4 because the value because the index corresponding to the value 2 is 1.

You can also slice it by giving the reverse index or the negative index I have given access the values from starting from the index 1 and the end index should be minus 1 that is the second last element of your array. So, let us just see what is the output? So, in that case you will just have 2 and 3 because you've excluded the first element that corresponds to the index 0 and the last value because we have given minus 1.

(Refer Slide Time: 13:53)

The screenshot shows the Spyder Python IDE interface. The main editor displays the following code:

```

File "C:\python-input-16-32ba28a70d3d", line 1, in <module>
    arrSample[1:2]
TypeError: 'set' object is not subscriptable

In [17]: for x in arrSample: print(x)
         arrSample[1:]
         # array('i', [2, 3, 4])
1
2
3
4

Out[17]: array('i', [2, 3, 4])

In [18]: arrSample[1:-1]
         # array('i', [2, 3])
Out[18]: array('i', [2, 3])

In [19]: for x in rangeSample: print(x)
         print(rangeSample[-1])
         # range(1, 9, 4)
1
5
9
range(1, 9, 4)

```

Below the code, the text "Sequence data operations: Concatenation" is visible. On the right side, a "Usage" panel provides information on how to access help for objects.

Similarly you can perform the same mono range as well since I have given range sample and inside the square bracket I have given colon and minus 1. So, basically you will be fetching or you will be slicing all the values from the first but you do not you will not be able to but then you will just exclude the last element of your array last element of the range then it gives you only 1, 9, 4. So, now we have seen how to basically perform the slicing operation on the sequential data and.

(Refer Slide Time: 14:37)

The screenshot shows the Spyder Python IDE interface. The main editor displays the following code:

```

Sequence data operations: Concatenation

Syntax: ',', '+', '+='

In [21]: print(strSample, 'python') # learning python
         print(strSample)
         newString = strSample + 'python'
         print(newString)
learning python
learning
('learning', 'python')

In [22]: print(lstSample)
         lstSample.append('py')
         # [1, 2, 'a', 'sam', 2, 'py']
[1, 2, 'a', 'sam', 2]

Out[22]: [1, 2, 'a', 'sam', 2, 'py']

In [23]: print(arrSample)
         arrSample[50:60]
         # TypeError: can only append array (not "list") to array
array('i', [1, 2, 3, 4])

Traceback (most recent call last):
  File "C:\python-input-25-89db90cb43d", line 2, in <module>
    arrSample[50:60]
TypeError: can only append array (not "list") to array

```

Below the code, the text "Sequence data operations: Concatenation" is visible. On the right side, a "Usage" panel provides information on how to access help for objects.

Now we will learn some of the most fundamental sequence data operation that is concatenation and multiplication. So, this will be helpful when you are looping over your looping over elements in your containers. So, now we will learn some of the most fundamental sequence data

operations that is concatenation and multiplication which can be applied on to any sequential data. So, we are going to do concatenation using the plus operator.

And there are a few ways of doing this depending on what you are trying to achieve and the simplest and the most common method is to use the plus symbol to add multiple strings together or to add multiple elements together it can be applied onto any sequential data as well. And simply place a plus between as many as simply place a plus between as many strings as you want to join together. So, let us see an example to do that. So, for example, so, we know what is there under strsample it just has a word learningt.

So, I am going to concatenate two strings together using the plus operator here. So, I am going to concatenate strsample with another string that is python but before python I have given a space inside single quote. So, it is going to have a space in between learning and python. So, that is where given a space. So, this is just to join this is just to join two strings together. So, let us just see the output. So, the first line is the output of the print strsample and in the next line I am just checking whether it has been updated in the strsample or not because I have added or have joined a new string to the existing string.

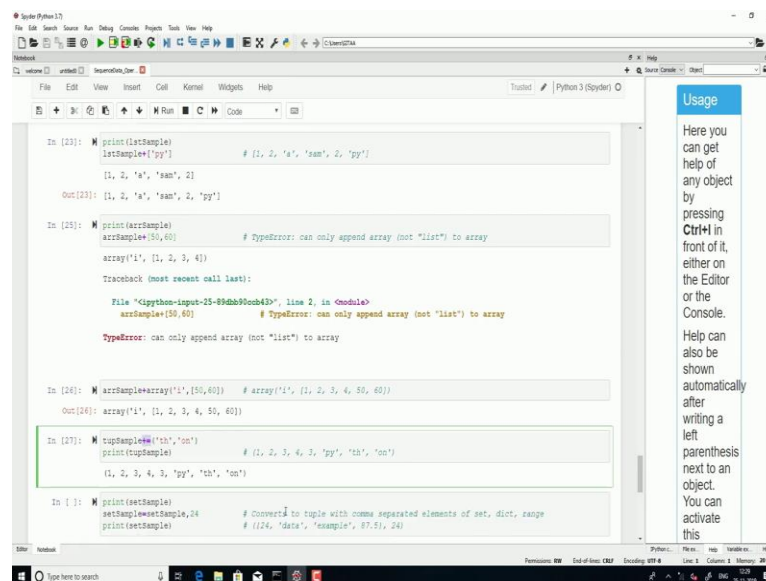
So, let me just check. So, let me just print strsample and see what is there but unfortunately it has only learning in the python does not get updated to the existing strsample that is why because remember string that is why because remember strings are immutable and if you concatenate or repeat a string stored in a variable you will have to assign the new string to another variable in order to keep it. So, let us just do that. So, whatever operation that I am performing here I am just updating it to a new variable. So, that it gets updated.

So, now if you print the new string then it will have learning, python as specified in the concatenation. So, this is how we do concatenation on a string. So, this will be very handy when you want to when you are performing when you are looping through the objects in your data and if you want to perform some of the operations like concatenation or multiplication this will be very handy. So, next let us see the concatenation on list.

So, this is the original list that we have 1 2 a sam 2 and in the next line I am concatenating the list to the existing list lstsample and the list contains a string that is p y I am giving that inside the square bracket because you will be able to concatenate to list. So, if you can see here the py list has been added as has been added as an element to the existing list that is lstsample. So, we do not have a separate list which contains a py string rather it has been added or gets updated in the original list itself or gets updated as a last element in the original list itself.

And let us see an example on how we can concatenate the arrays let us see an example whether we will be able to concatenate two different sequence data for example let us try an example to concatenate the list and the arrays together. So, first let us see what is there under the array that it has basically values one two three and four which are of integer data type and by using a plus operator I am going to concatenate a list together I am going to concatenate a list to the existing array. So, let us see whether that happens or not.

(Refer Slide Time: 19:51)



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

```
In [23]: print(lstsampl  
lstsampl+['py']  
# (1, 2, 'a', 'sam', 2, 'py')  
  
Out[23]: (1, 2, 'a', 'sam', 2, 'py')  
  
In [25]: print(arrsampl  
arrsampl[50,60]  
# TypeError: can only append array (not "list") to array  
  
array('i', [1, 2, 3, 4])  
  
Traceback (most recent call last):  
File "C:\python-input-25-89dbb90cb43D", line 2, in <module>  
    arrsampl[50,60]  
# TypeError: can only append array (not "list") to array  
  
TypeError: can only append array (not "list") to array  
  
In [26]: arrsampl+array('i',[50,60]) # array('i', [1, 2, 3, 4, 50, 60])  
  
Out[26]: array('i', [1, 2, 3, 4, 50, 60])  
  
In [27]: lstsampl+('ch','on')  
print(tuple(sampl  
(1, 2, 3, 4, 3, 'py', 'ch', 'on')  
  
In [ ]: print(setsampl  
setsampl+setsampl,24  
print(setsampl  
# Convert to tuple with comma separated elements of set, dict, range  
# ((24, 'data', 'example', 87.5), 24)
```

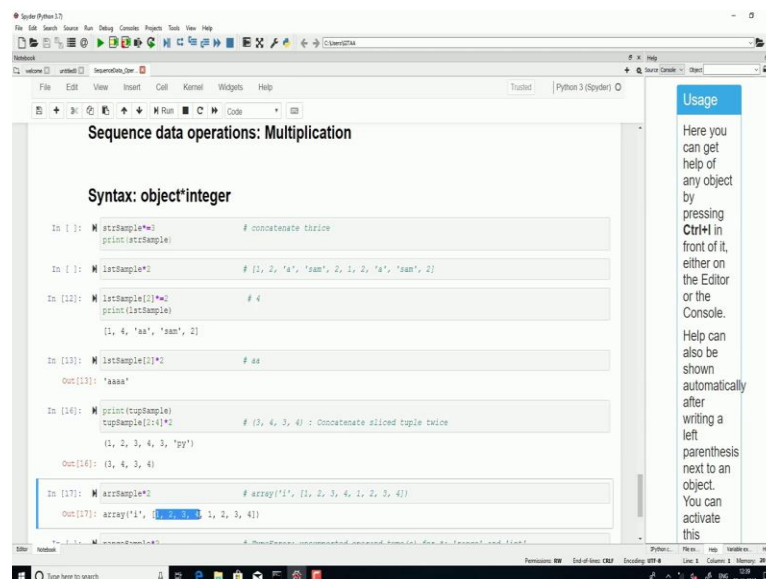
A traceback error is shown for the code in cell [25], indicating a `TypeError: can only append array (not "list") to array`. The error message is repeated twice. The code in cell [26] successfully concatenates an array to another array, and the code in cell [27] concatenates a list to a list.

So, it basically throws you what traceback errors it basically throws you a type error saying that it can only append array to list it cannot append array to a. So, the type error is throwing basically it saying can only append array to array not list to array. So, we because we are trying to concatenate or we are trying to append the list to an existing array that is not possible. So, what we have to do here means if you want to concatenate to an array.

If you want to concatenate a sequence data to an array then that sequence data should be of the same kind that is array. So, what we can do here is we can use the same values but just put them as array basically using the array function and keeping the data type as integer and giving the values as array and giving the values as list inside the array. Array sample will get concatenated with the given values that is 50 and 60 since you created it as array and then concatenated.

So, the next one is concatenation on concatenating two tuples together. So, here if you can see here I have just used plus and equal to then it means that I am going to add a new tuple to the existing tuple, tupsample and that is automatic glue and that gets updated in the tupsample itself. So, the tupsample basically had 1, 2, 3, 4 and py and I have added th, on separately as a and I have added a new tuple having two strings th and on. So, and that get add and that gets added to the existing tupsample and that gets updated on the existing tuple sample because I have given plus and equal to.

(Refer Slide Time: 21:58)



```
Sequence data operations: Multiplication

Syntax: object*integer

In [1]: # concatenate thrice
print(tupsample)

In [2]: # [1, 2, 'a', 'sam', 2, 1, 2, 'a', 'sam', 2]
tupsample*2

In [12]: # tupsample[1]*4
print(tupsample)

In [13]: # tupsample[1]*2
Out[13]: 'aaaa'

In [14]: # print(tupsample)
tupsample[1:4]*2
# (3, 4, 3, 4) : Concatenate sliced tuple twice
(1, 2, 3, 4, 3, 'py')

Out[16]: (3, 4, 3, 4)

In [17]: # tupsample*2
Out[17]: array('i', [1, 2, 3, 4, 1, 2, 3, 4])
```

The next one is concatenation on set. So, basically here I am I already have a set sample which has values 24 data example and 87.5 as values of set and I am doing concatenation here. I am just going to convert to tuple with comma separated elements of set and dictionary. So, I am just concatenating a value to the existing set sample by just placing a comma after set sample and then updating it to the set sample. So, that that value will get updated in the actual set sample.

So, if you can see here I have run multiple times. So, each and every time you run it is going to update it in the original set sample. So, similarly you can perform the operation called multiplication on any sequential data for example you multiplication in the sense if you are applying the multiplication on a string then it means that you are going to multiply a string you are going to repeat you are going to repeat a set of strings by just giving a number.

Multiplication in the sense if you have strings and it has some if you have a string then if you are doing some multiplication on it then if you are performing multiplication operation on it then it is going to repeat the values to the number of times that you've given as a multiplier. So, let us see what happens. So, I have a strsample here I have given asterix equal to 3. So, I have strsample asterisk equal to 3 it means that I am going to repeat whatever is there under strsample and then I am going to update it to the existing str sample itself. So, let us just run.

So, learning is what is there and under. So, learning is a string under strsample since I have given tree asterix 3 it has repeated the string thrice and it gets updated in the strsample and that is why while printing the strsample you got the output as learning, learning and learning. But what happens when you perform it on a list it also has both numbers and strings to it but in this case since it has string to it but in this case it is just going to repeat the element of your list twice since you have given asterix and 2.

And you can as well repeat a particular element to the number of times by accessing them using the index. So, here I am just going to repeat a value corresponding to the first index twice that is 4 but you will be able to for example but if you can print the list sample and see but if you can print the list sample and see the value is not repeated twice because you have not updated that there. So, if you just give equal to symbol here and then execute it.

So, basically the python does not do explicit conversion for example if you are performing an operation which has string then it is not going to multiply a number to a specified to it is not going to multiply a number with a given value rather it is just going to repeat the values to the number of times that you have given. But if you are trying to access a particular value which is of integer and then you are performing a multiplication operation on it and then in that case it is

going to multiply that particular value to the given number of times.

Let us see an example here we have a value that is 2 that corresponds to the first index and I am just trying to multiply that with the value 2 and also trying to update it in the original list in that case it is going to give me an output saying 8. So, now let us see what will be the output. So, in this case originally you had 1, 2, a sam and 2 and since I have given lstsample of 1 asterix equal to 2 that particular value has been multiplied with the given value that is 2.

So, 2 times 2 will give you a value called 4 and that gets dated in your original list. So, this is what happens when you are performing a multiplication operation on an integer or a float but whereas if you have any string value and if you are trying to perform some multiplication operation on it for example we have a value called a right. So, 0 1 2, 2 is the index corresponding to it. So, let us just try what happens. So, it is going to repeat that particular element to the number of times that you have given here since I have given 2 the value gets repeated twice.

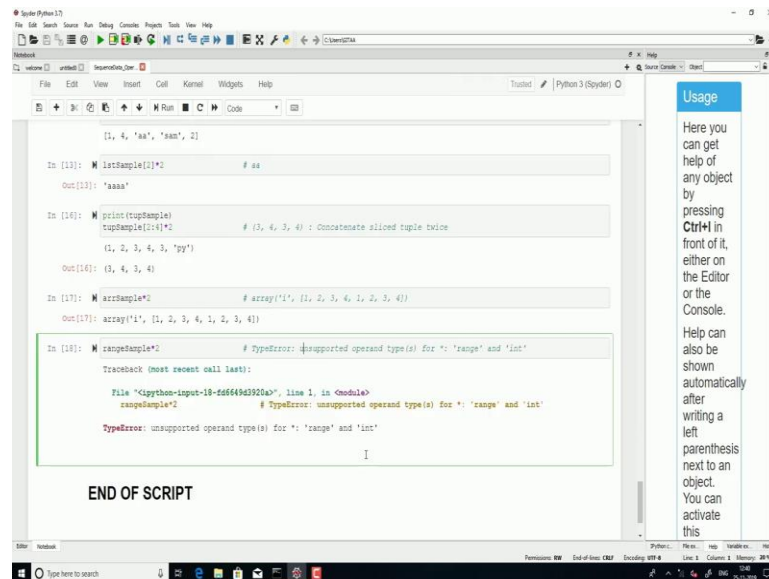
So, this is what happens when you do multiplication operation on any kind of sequential data that you have yeah the same example I have repeated in the below cell. So, already it is double a and if you are trying to multiply that with the number two then it is going to repeat it twice again. Let us see what happens when you do the concatenation. Let us see what happens when you do the multiplication on a tuple. So, base initially it had. So, let us just print what is there in the tuple sample.

So, here what I am doing here is. So, what I am doing here is I am going to slice a particular set of elements from the tuple and then going to do and then perform the multiplication operation. So, before that let us just print what is there in the tuple sample. So, what is there in your tuple sample list 1, 2, 3, 4, 3 and py and 2 colon 4 is nothing but 3 and 4. So, 2 colon is 2 colon 4 slices the elements that is 3 and 4 and I am going to multiply that twice in that case it is just going to multiply them it is just going to repeat that particular element twice and the next thing is performing the multiplication on array.

So, it also basically multiplies it also basically repeats the given array twice since you have given

asterix 2.

(Refer Slide Time: 31:36)



```
[1, 4, 'aa', 'aaa', 2]

In [13]: listSample[2]*2 # aa
Out[13]: 'aaaa'

In [14]: print(tupleSample)
tupleSample[2:4]*2 # (3, 4, 3, 4) : Concatenate sliced tuple twice
(1, 2, 3, 4, 3, 'py')
Out[14]: (3, 4, 3, 4)

In [17]: arraySample*2 # array('i', [1, 2, 3, 4, 1, 2, 3, 4])
Out[17]: array('i', [1, 2, 3, 4, 1, 2, 3, 4])

In [18]: rangeSample*2 # TypeError: unsupported operand type(s) for *: 'range' and 'int'
Traceback (most recent call last):
  File "C:\python-input-10-Ed6649d392ba", line 1, in <module>
    rangeSample*2
    ^^^^^^^^^^^^^
TypeError: unsupported operand type(s) for *: 'range' and 'int'
```

END OF SCRIPT

And the next one is range sample we are just multiplying the range sample by two it throws an error saying type error and supported operand type for asterisk range and I arrange and in integer because you will not be able to perform the multiplication operation on a range because using range you will be able to only create a sequence of values given range. And we have come to the end of the script and in this lecture we have seen in continuation with the previous lecture in this lecture we have seen how to perform some of the slicing operations on any sequential data.

And we have also seen what happens when we do slicing on a container which stands for holding non-sequential data and followed by that we have seen how to perform some of the operations like concatenation and multiplication on any sequential data. And we have also seen some of the checks for example what happens when we try to multiply two numbers and what happens when we do the multiplication on a string

And in the upcoming lectures we will be looking at sequence data methods where we will be concentrating more on what we can do with the sequential data and what are the operations or what are the methods that you can. So, methods so, in the in the upcoming lectures we will be looking at how we can use the sequential data to perform any operations. So, we are wrapping up. So, yeah in the upcoming lectures we will be seeing about sequential data methods that is

nothing but how to apply the methods or any sequential data, thank you.