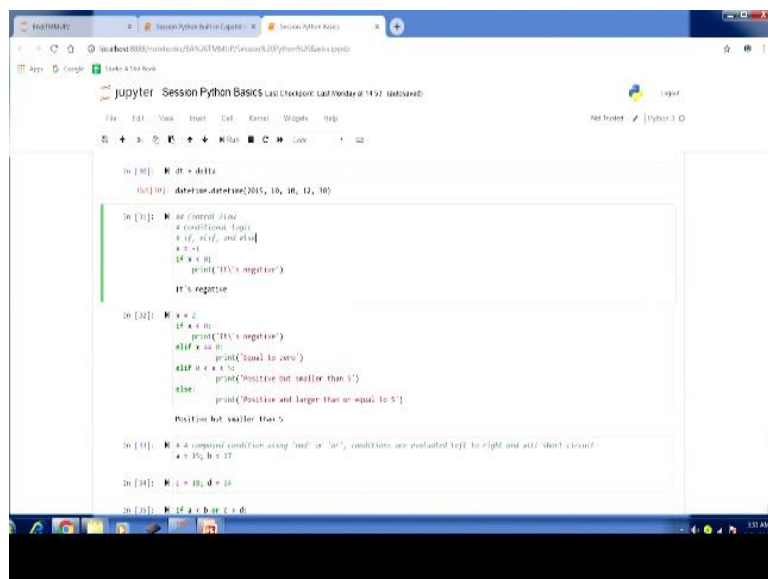


Business Analytics And Text Mining Modeling Using Python
Prof. Gaurav Dixit
Department of Management Studies
Indian Institute of Technology-Roorkee

Lecture-09
Built-in Capabilities of Python-I

Welcome to the course business analytics and text mining modelling using python. So, in previous lecture we were discussing the control flow, and specifically we were discussing loops. So, let us pick up from where we left in the previous lecture.

(Video Starts: 00:41)



```
In [10]: # if-else condition
Out[10]: datetime.datetime(2015, 10, 12, 30)

In [11]: # for loop
a = 10
while a > 0:
    print("It's negative")
    a = a - 1
    print("It's positive")

In [12]: # if-else condition
a = 10
while a > 0:
    print("It's negative")
    if a < 5:
        print("Smaller than 5")
    else:
        print("Positive but smaller than 5")
    a = a - 1
    print("Positive but smaller than 5")

In [13]: # if-else condition using 'and' or 'or'
a = 10
b = 10
if a > 0 and b > 0:
    print("Both are positive")

In [14]: # if-else condition using 'and' or 'or'
a = 10
b = 10
if a > 0 or b > 0:
    print("Both are positive")
```

So, we talked about the for loop. And we also talked about, you know certain scenarios where few more key words like continue to advance you know to the next iteration, you know the break key word to exit from the loop. So, those scenarios also we have gone through few of the common scenarios in the previous lecture. And we talked about the nested you know nested loops and how we can whenever this situation demands we can exit out of the loop, you know out of the you know nested loop and into the outer loop using the break statement.

Now let us move on to the next loop type. That is a while loop. So while loop is slightly different from for loop, in a sense for loop we have, we do the iterations over a collection over a sequence. So we typically define our sequence and we run through all the number of iterations that are allowed by that sequence. So based on that you know for statement, we make our iterations.

However in case of while loop we have a conditional exploration. So till that conditional expression is found to be true, that loop will go on. So in a sense in our following, you know, block that we are executing after that while a conditional expression, we would like to also ensure our existing you know existing scenario there itself. So let us understand the same thing to an example.

So here we have initialised a variable x as 112. And we have I another variable as 0. So in the while loop, I am using the while keyword first, and then the conditional expression for this while loop. So this is x greater than 0. So right now the value in the initialise value of x is 112 and till this value remains greater than 0, this 112 value remains greater than 0, the following block in the while loop is going to be executed.

So if you look at the following block, the first line itself, we have ensure, how will exit out of this loop. So there I have the conditional explanation, if I greater than 200. So whenever the value of variable I becomes greater than this value 200 we are giving the break statement, so we will be able to exit out of the loop. So in the following lines of code, I am increasing the value of I, $I+=x$, and x also I am using this x and this operator, double you know forward slash operator here to continuously decrease the value of x.

So let us run this. Let me go down here to the while loop block. So if I run this, let us check the value of x. So you can see the 7 and then if I check the value of I you can see 200 and 10. So, off course, you know that is greater than 200. So this conditional expression was executed, so the value of x what is still greater than you know 0. So as per that while keywords conditional expression so that was still evaluating to be true.

But, you know, he had the exact scenario there, where if the value of I greater than 200, we had used the break keyword, so that scenario was achieved, and therefore we exit out of the loop. Now let us move on to the you know next, you know construct that is past. So this particular construct is typically about when we do not want, we might have a particular you know block where you know, based on certain conditions, we do not want anything to be executed.

So this is mainly coming, because you know we use wide spaces in python, in other programming language, we have braces or other things, which in essence, allow us you know, to leave those braces without any statement and therefore do not have, we can manage when we do not have

anything to execute in that particular you know block of code. However, in case of python, because we are using wide spaces we need this kind of, you know, key word pass, to explicitly tell the interpreter that, you know, we would just like to pass from this particular block of code.

So we can understand the same thing through this example. So the first conditional logic block that we have is if x less than 0, so value of x is presently 7. So this would not be true. So if x less than 0, then 10 negative action, else if x is equal to 0, then, you know, no action now to be coded later. So, if the block is you know, if we do not want any action at all, then this will just work fine.

If we have responded in a sense that we would like to add the code later on in the whole code is in the development process. And we might not have focused on this particular, no, we will not like to focus on will not like to code this particular block as of now, then we can use the pass key word here. So if that this else if condition is met, if it is evaluated to be true, then pass keyword would do the job.

And then the next block is else that is the catch all block that we have. So if all ever conditional expression are evaluated to be false, then this is going to be executed and will print this statement positive action. So in this case, since the value of x is 7, which is positive, so will actually be printing this else block. So let us run this, you can see positive action message is actually printed. Now let us move to the next thing.

So next we are going to discuss this range function. So in the previous lecture, you might have seen that we had used this particular range function for iterating over a collection a sequence. So this sometimes can be really useful. So let us understand how this particular function works. So as we did for the for loop before in the previous lecture, this range function is typically used to create an iterator.

So whenever you like to you know, iterate over a sequence. So this particular function can actually be used to create that sequence. So, typically, this function accepts 3 arguments, a start and an step. So where we would like to, you know, indicate, which will, and how we would like to start and then the end part of it and how we would like to step. So, step might also be negative in here.

So, after we done this, it will return an iterator that will yield a sequence of evenly spaced integers. So actually, it is going to return us, you know integers, and evenly spaced integer. So let us run this range 5. So here you can see, I have just passed 1 argument. So this is one way to call this function. So you can see range 0-5. So if I, you know, just pass this, so the start argument will take will default to the value 0.

So the you know iterator would be from 0 to 5 and as we said, the sequence of evenly spaced integers. So, ah this range 0-5 and step value would be 1, so it will default to 1 and the range will start from 0, if you want to, you know confirm the same thing, the next line of code where we can use the list function. So to convert this sequence into a list type and see what is inside of it, so you can see 0 1 2 3 4.

So you can see step value is 1 and we did not indicate the start value which defaults to 0. And you know, so the argument that we had in was 5, so this indicating that we required 5 elements there, that would also be indicated by the you know end agreement as well. So 01234, so, this is the collector, this is the iterator that we have just created. So, range function can be really useful in this sense to create the iterator for you know for loop.

Similarly, I will take you know, another example where we are giving all the argument passing all the arguments. So, you can see the next line 0 start value then 10 and 12 n value and then 1, if I run this you can see 0-9. So, this is the list that we would like to iterate over. Now, the sequences that are typically produced using the range function, they have the start point, so the start point would be included, but the endpoint is not included.

So same thing if you want you know, if you want to understand more details about range function, so I would also like to, you know, tell you in this, you know, help tab that we have in this Jupyter notebook interface. If I click this, and you can see there are many links for, you know, various help that we might require. So, if I just happened to you know because still we are into the python building of, you know, functionality of python.

So, if I just click python reference, and here in the search box, I can type you know, range, and I will get the details. So, whenever we are interested in finding out more detail about a particular you know, about a particular you know function, we can always search. So, you can see range,

and you can see in the balances python clause built in types, if I just click here. So, you can see all the detail about this particular function can see range start, stop and step.

So, you can see you start you can see it clearly mentions here that it defaults to 0. So, this particular exercise related to finding help you know related to finding python reference and doing so that whenever you require more detail about a function which we have been discussing, and in coming lectures also will be talking about many functions, many built in types many other things.

So, you can always go to the appropriate reference and you know, just type in that particular you know, word and find out the details. So, you can see start defaults to 0. And then we are the you know, step thing you know step argument right, so all these details are here, few examples are also given here. So, in this fashion, you can see the step argument you can see in this step argument.

You can see if it is not specified then it will be 1, if the parameter was not supplied the same thing I was mentioning here, you can see the start 0 if the parameter was not supplied. So in this fashion, we can always find out more detail about any function. So as we said that a step value could be negative also. So in this example, I am giving the start well as 4, the end value is -4 and the step is -1.

So, if I run this, this is the you know the iterator that we get, so you can see here start point 4 is included, but if you see the end point -4 is not included. So, that is another point to note. So whenever you are indicating your endpoint. So, you know a value less than or greater than depending on this step, you know, value would be you know, part of the collection would be part of the iterator. Now, another usefulness of range could be that you know rather typical usefulness typically ranges used as an index for iterating through sequences.

So now, in this case, something similar example we had used before. So in this example, you can see, I am initialising, a sequence here, this is a list of 12345, 5 elements, and then we have a for loop here. So for x in range and length sequence, so here you can see that we have first capture the length of the sequence. So it has 5 elements. So in a sense we are just, you know, passing on just to one argument here.

So it is actually range 5 in effect, and the X will iterate over this range 5. So when we say the range 5 essentially we have created an iterator, created a collector, iterator and the x value will iterate over it, and our variable V5 is going to be assigned those values in each of the iterations of the for loop. So if I run this you can see the value of V5 variable comes out to be 5. So because the last value in the sequence you can see I am passing on x as an index within the for loop first line, of code, sequence in brackets x.

So in essence, I am using the you know range, the iterator that I have created, that I am using as an index, so that any other sequence, I can use that index and work on that particular sequence for whatever you know computations, whatever assignment and other things manipulation, I might require. So, in this fashion typically range is used as an index. Now, there might be certain situations where we might be dealing with large number of iterations.

So, you know, if we do not use range function, then will end up creating some sort of you know, list or sequence and that would require memory, you know, especially that you know, large number of iterations are required, but instead of we use the range function, then memory use would be optimised. So, for this we have given one example here, so, we have any slide this variable sum as 0 and then we have a for loop here.

So, this is for x in range, and you can see a 100000, so 100000 iteration that we have in this piece of code, and then you can see here using model operator here, so we have the conditional expression of x percentage to equal to equal to 0 or x percentage 3 zero equal to equal to 0 then all those values they are going to be summed up. So, if I run this so, in this last you know, in a large number of iterations, and you can see very higher value for sum.

And if I run the value of x, you can see, so, those many 99,999 that is the value of x. So, whenever we are dealing with this situation memory use can also be optimised using range function. Let us move on to the next thing that is ternary exploration. So, ternary exploration is especially useful whenever we are looking for a single line or exploration in place of a value producing if else block.

So, we might have you know, you know depending on typically we use if else block whenever, we would like to control the flow of execution, whenever we have some you know, conditional expiration we which we would like to evaluate which we would like to check and then execute

the following block then execute the following lines of code. However, in some situation, you know, our if you know if else blocks might be producing certain value.

And some situation we might we would like to assign those value to you know, some other variable, record that value in some other variable. So, just to condense the code, we would like to use a single line or expression especially in you know, value producing we have a value producing if else block. So, this ternary expression can be really useful in those situations. So, let us take an example.

So, here how typically we would write if x less than 0 negative action. So, in a sense, this is a value of a string type and else positive actions, if I run this, so, to take an example if you know, in this case x value is greater than 0, so, if x less than 0 the negative action would not be executed, the else block is going to be executed that is positive action, the same thing because here we are, you know, in a sense, value of string type is being produced.\

The same thing can be written in a ternary expression format, where we are writing like this. So negative action if x less than 0, so again if x less than 0, so, I can write in this fashion, else positive action. So, sometimes I would like to record this value. So, this whole ternary expression can be assigned to some variable. So we can say like we V7=, and this ternary exploration or whatever well is there it will be recorded.

So, sometimes, there are situations where ternary expression could be really useful. So, this is the end of session on python basics. So, what are the basics that we wanted to cover, so that we have been able to cover here, now, some of these things, we have not gone into more detail. Now will, you know, in the coming lectures, however in the coming lectures will, you know, go into more detail about many aspects that we have covered in brief in this particular you know, session.

Now, we will move on to our in our next session, which is on built in capabilities in python. So, in this particular session will like to focus on , so in coming lectures will focus more on the, you know, built in, you know, data structures first. So, typically, these tuples, list, decks, and, you know, sets. So, these are the built in data structure that we have in python. So, let us start, so, we will start with tuple.

So, what is the tuple, so this is a fixed length, immutable sequence of python object. So, we have talked about the mutable and immutable objects in python in our one of the previous lectures. So, this particular tuple and they will be talked about that tuple is an example of an immutable object. So, certain situation would make tuple quite useful. So how this is created, so we use a comma separated sequence.

And whenever we are giving a number of values, whether integer code or any string or any other type in a comma separated form that will actually define a you know, define or initialise a tuple. So look at the example. So first tuple you know, that we are creating here, tup, and you can see 1, 5, 9. So if I run this I will have my, you know, tuple created here, you can see if I run top, I will get 1, 5, 9, this is the tuple.

So within the parentheses, I will have all 3 elements. Similarly, so this is one way, so there, when we created this first variable tup, we did not use any parentheses, but in the semantics of, you know, tuple, whenever you are trying to print tuple, you know, then it will be always with the parenthesis, we can all the time, you can use parenthesis also in the creation of tuple itself. The next example is about this tüp 1, you can see I am using parentheses, and then 2, 6, 10.

So if I run this, we get 2, 6, 10. Now, we can have nested tuple also, so tuple of tuples. So we have another example here and top. So in this case, we have 1,5,9 as a 1 tuple, then comma and then second tuple within parenthesis 2, 6, 10. So this would be our nested tuple , if I run this, you can see you know, within the parenthesis we have, you know, 2 tuples as an element of the, you know final tuple.

Now, conversion of, you know, other python objects to tuple so just like we talked about the, you know, basic types, load int and bool you know those types. So we have those functions to convert any other type to you know, that particular type. Here also, we have the tuple function. So in this example, if you can see here, I am passing on a list here, so the element of list are 1, 0-1.

So if I pass on this list to this, you know, tuple function, I will get this, particular list converted into a tuple type. So, if I run this, you can see on the list 1 0 -1, which was passed on using the you know, brackets, you know, now as converted into a tuple, and it is described in parentheses. Similarly, you know, I can, you know, tuple can also be used to create tuple function can also be used to create a tuple from a string.

So, in the next example, if I pass tuple om om kar and have by pass this as an argument, if I run this, so, you can see in the output, I will have you now tuple created of you know, the characters which are part of this string if I run this you can see all the characters o then m then space and o then m k a r in this fashion, this tuple has been created. So, from a string also, we are able to create a tuple.

Now, if we are interested in accessing and working with the, you know, individual elements of a tuple, so, let us talk about that. So, accessing tuple elements. So for this just like we have been doing before also, in the previous lectures, we have been typically using list and we have been accessing those list elements also using the, you know, using the brackets operator. So, here also for accessing tuple elements, we use the same operator bracket operator.

So, here as you might have understood by now that most of the sequences including list, which we have used in the previous lecture, they are 0 indexed. So whenever we are trying to access a list, the indexing starts from 0, so the list has 5 elements, the indexing will start from 0 1 2 3 4. So the 4 index will actually cover the 5th element. So here, top 4, if I want to access the 6th element, I will give index as 5 because it 0 index.

If I run this you can see I got K here for you know, 6 element of top 4 so I have given the indexes 5 for 6 element, if you just count in the previous output, you can see o then m then space 3, and then another, you know, 6 characters k. Now, we talked about that tuple are immutable objects, for immutable python objects, so can they be modified. So let us first you know, create a tuple, so we are passing on a list, and will have this top 5.

So, this is the tuple that we have first element is a string om then we have a list just a second element, then we have a bool as you know the third element. So if I want to change, I know top 5 and a second element that is, you know, that is bool type, if I want to change it from false to true if I run this, I will get an error you can see here. So something similar kind of exercise that we have done in the previous lecture also, where we were discussing mutable and immutable objects.

So here again just to you know, display again, that tuple is an immutable object. So, you can see we are getting a type error that means tuple object does not support item assignment. So, any

element that is part of the tuple we cannot directly change it. So, the value false that we had as the you know third element indexed as you know, index with the value 2. So that we cannot change, then how do we you know, modify is any kind of modification possible immutable objects of you know immutable type like tuple.

So, tuple might have tuple will have a number of elements. Now, some of those elements might be of mutable type. So, some of those elements can be because of them being mutable type they can be modified you know in place. So we can in the sense we can use methods. And if the you know, that element is mutable, then we can modify. So in this case, you can see top 5 1, so that is second element in the top 5 tuple that we have.

And then dot append so we are calling this you know function because the second element is actually a list. So we would like to add another element in that list, which is a list of a tuple top 5. So if I run this, you can see the whole tuple. And you can see the second element 0, 1 so 0, 1 now it is having 0 1 and 2. So in this fashion, we have immutable element we can modify. Now let us move on to the next aspect concatenations.

So can we you know, concatenate 2 tuples. So yes that is possible, so we can use plus operator. So let us use this example top 5+top4 if I run this, you can see I got a long tuple here, where top 5 and 4 , all the elements of top 5 and top 4 have been concatenated . So they are coming one after another. As you can see, all the elements are top 5 they are coming first, and it is followed by all the elements of top 4.

So in this fashion we can concatenate 2 strings 2 tuples. Now, there is another concatenation that we can perform, which actually involves repetition of certain elements. So for example, if we have this tuple 3 that we have seen earlier, and if I just multiplied with 3, then all the elements that are there in top 3, they are going to be repeated thrice. So let us run this and find out what is happening.

As you can see, in the top 3, we had 1 0-1. And this is, you know, another kind of repetition or concatenation we can do can see then we have another tuple concatenating to it 1, 0 -1 then 1, 0 -1. So in this fashion we can concatenate. Now, let us move to the next aspect of tuple. unpacking tuple. So in this case, you can see on the next line that is, on the left hand side of the assignment operator, we have ABC, which is a, b, c it is another tuple.

And on the right hand side, we have the tup 1, so tup1 has certain values. So in a sense, when we are using assignment operator, the in a sense unpacking is happening, where the individual elements of the tuple are going to be assigned to the individual elements of the tuple which is on the right hand side. So the values which are on the right hand side element values, which are on the right hand side tuple are going to be assigned to the you know the elements that are on the left hand side tuple.

So if I run this, so this unpacking will happen. And if I just want to access the 1 element, let us say c, you can see the value 10 is there. Similarly, nested tuple also, we can do the same thing. So here another example z y z p q r and end tup we had earlier created. So if I run this and if I look at the value of q here if I run this can see 6, so when the nested tuple also this unpacking works quite well.

So, this can help us in certain scenario where we can easily go if we want to assign certain values, you know, we can easily go through that. Now, there are certain other situation where, you know tuples can be really useful for example, swapping, you know, variable names. So here typically, if you have to swap variable names, it will have to write 3 lines of code, as we typically do in programming language, right.

So x=y and the next statement y=z and then z= so in that fashion, we will have to write our code and do a swapping of variable names. However with the help of tuple this can be done through just one statement you can see. So, we have this if you focus on the right hand side tuple so we have these 3 elements a b c, if I want to change your c with a variable name, then you can see on the left hand side, I have changed the tuple c b A

Now the a will become you know c and c will become a if I just run this and if I want to find out the value of c, you can see it has changed. So earlier it was 10 and now it is 2 which was actually the value of a element. So this is about tuple.

(Video Ends: 30:42)

So at this point would like to stop here and we will continue our discussion on tuple and the next lecture, thank you.

Keywords: Elements, Assignment operator, nested tuple, functions, magic commands, concatenation, range function.