Business Analytics And Text Mining Modeling Using Python

Prof. Gaurav Dixit
Department of Management Studies
Indian Institute of Technology-Roorkee

Lecture-06
Python for Analytics-Part III

Welcome to the course business analytics and text mining modeling using python. So, in the previous lecture we were doing exercises in a Jupyter notebook platform for python. So, let us quickly go through some initial lines of code and then we will pick up from where we stopped in the previous lecture.

(Video Starts: 00:44)

So, let us go through some of the initial lines of code. So, I will just run through this so, we have discussed this in previous two lectures. So, I will just walk through all these lines of code and we will move to the next thing. So, line magic, cell magics, magic commands. So, all this we have discussed before. I will just run through this and we will move to the next thing, matplotlib (()) (01:26) few things we discussed.

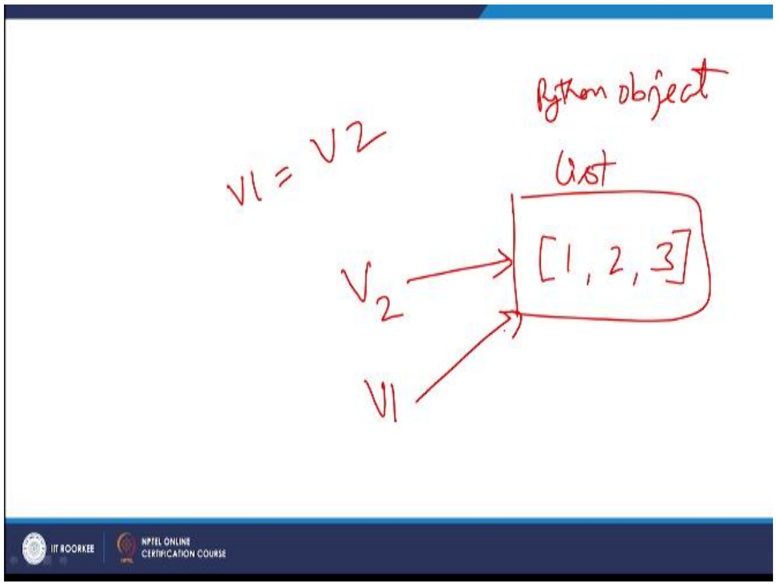
And programming semantics, python object, the () and object method calls something that we discussed in the previous lectures and then we talked about few more things about methods positional and keyword arguments. So, all these things we discussed before and then a variable assignment. So, this was something that we were discussing and let us move forward. So, this part was also discussed.

Now I think we stopped at this point where I was showing I was displaying that within the functional scope if we have you know there we can print the elements, we can print the variables a list and element. If we try to access these objects outside the you know functional scope outside the local namespace, then we will get these not define error. So, I think at this point we stopped, so, we will pick up from this point.

(Video ends: 02:43)

Now we come to something called dynamic references. So, one aspect that we have discussed in the previous lecture also, when we assign a variable to another variable.

(Refer Slide Time: 02:54)

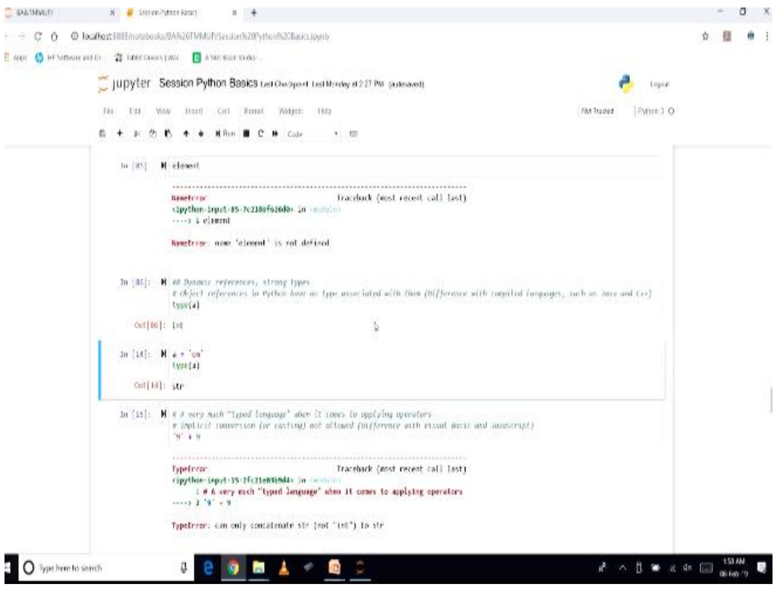


Let us you know like this V1 = V2, so what happens is let us say we have this list and we have these values, these elements within the list. So, what happens that when we make these kind of assignment. So, V2 was let us say you know referring to this object, this is the object python object, you know list object that we have and what happens that when we assign V2 to V1. V1 will also start referring to the same object.

So, this is slightly different from other programming languages where a new copy is created. However, if you are going to use list () you know then of course a new object is going to be created. But if we are going to use this assignment operator = operator which is an assignment operator. Then you know both the variables are going to be referring to the same object list in this case.

And you know any changes that you would do using one of these you know variables that changes would also be reflected in the other one, because of them both of them referring to the same object.

(Refer Slide Time: 04:07)



```
In [85]:  element

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-85-7c2180f626d0> in <module>
----> 1 element

NameError: name 'element' is not defined

In [86]:  ## Dynamic references, strong types
          # Object references in Python have no type associated with them (Difference with compiled languages, such as Java and C++)
          type(a)

Out[86]: int

In [14]:  a = 'om'
          type(a)

Out[14]: str

In [15]:  # A very much "typed language" when it comes to applying operators
          # Implicit conversion (or casting) not allowed (Difference with visual basic and javascript)
          '9' + 9

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-15-2fc21e03b9d4> in <module>
      1 # A very much "typed language" when it comes to applying operators
----> 2 '9' + 9

TypeError: can only concatenate str (not "int") to str
```

Now let us talk about you know some other aspects about you know python language like dynamic references and strong types. So, object references that we talked about just now they have no type associated with them. So, this is a slight difference when it comes to you know comparison of python with other compiled language such as Java, C + +. There if you create a variable of integer type or string type or float type you cannot use that variable you cannot assign any variable of other type to this variable.

So, that kind of thing cannot be done, however, in python this kind of dynamic referencing is possible.

(Video Starts: zerofour:fiveseven)

If I look at the you know that this variable a that we have created earlier, integer variable that we had created earlier, if I look at the type of this variable if I just run this you can see in the output integer is the type of this variable. However, if I assign a string to this variable you can see in the next line I am doing a = you know assigning this string om to a and I am also looking at type of a.

If I run this you would see the type of a has changed, it has now become you know STR that is for string. So, dynamic referencing takes place in python. So, a variable which was you know integer type if we directly assign some other value to belonging to a different type to this variable, then it is type will also change as you can see, type a has changed from integer to string.

This is as I said is different from what we learn in Java in other compiled programming language C + + and others.

However this is not to say that python is not a typed language, it has strong types, but they are very much you know they are visible when it comes to applying operators. So, whenever we are trying to apply operators with one variable, two variables depending on the operator, depending on the kind of operator that we are applying and then we can see that any implicit conversion you know or you know casting that might be allowed in certain programming language is not allowed in python.

So certain programming language like visual basic and JavaScript you know if you do a operation like you know where you are adding a string with the value nine you can see in this next example here nine is being added with you know numeric value nine. So string value nine, the string character nine is being added with the you know numeric value 9. So, in some programming language like visual basic you know costing will happen, coercion will happen.

And nine will be a string value would be equals to the numeric value and we will get eighteen probably and in JavaScript it might happen the other way around where the numeric value nine might be coerce to a string value and we might get 99 as a string output. So, those kind of things can happen in visual basic in JavaScript, but python is a typed language and this particular operation is not allowed.

So if I run this I will get this error, you can see can only concatenate str to str, so python is a typed language but when it comes to applying operator, when it comes to you know applying you know certain functionality certain processing certain computation, but when it comes to assignment and referencing then that is dynamic in nature. So, that is slight difference you know from Java and C+ + .

Let us move forward to explain few other things about you know python being a you know type language is that, a another example that we have is where we are assigning 5 you know as a value to this variable a and we look at the type of a. So, this will be in integer type int type and c you know that we had previously created the value of c is seven point five and we look at the type of c. So, I think it is you know float.

And if we apply a operator here we are dividing a by c and let us look at the value. So, you can see we are getting 0.66 which is a float value. So, we can see there is something very obvious, so that kind of you know that kind of conversion will happen. But otherwise in most cases you know this casting or conversion does not happen in python. So, this was just to show you an example where conversion is possible.

Because that operation is possible so, the operation is very obvious that if a integer value can be divided by a float value or vice versa then that will happen, otherwise it is not allowed. Similarly another example I have given here I am written this print statement. So, how this you know how this code actually works out that we will be discussing over the you know in the in the coming lectures as well.

However if we look at just this particular you know line of code we have paint and then we have a string and then we have and we are calling certain method format where we are using type a and type c and you can see when we I run this line of code I get this message that a is class int and c is class float. So, in this case it is not working out. So, clearly we can see python is a type language and we are you know you know certain obvious scenarios when the conversion is feasible or acceptable then it happens otherwise not.

Few other things you know that we can do in python is using this () is instance. So, using this () is instance we can find out whether a particular object is an instance of a particular type. So, lets you know do this example a we are checking whether it is integer or not. So, I am just calling isinstance (), first argument a is the object which I want to check and I want to check whether it is integer or not.

So, I will get true or false as a return value, so, let us run this. So, this because a was integer as we saw earlier. So, I got true as a result. Similarly I can also, use this () is . instance and find out whether a is one of the you know integer float, because certain operations might work out so, it could be either integer of float. So, I might require in certain scenarios I might require that kind of check.

So, this is something also we can do here, now similarly another example that we can do with this object c here also I am checking whether c is int or float and because c is float as we know so this is coming out to be true. Now let us discuss few other aspects about python object. So, objects in python they typically have stored attributes and also associated methods. So, when we say attributes these are other python objects which are stored inside the object that we are referring to.

And then by methods we need main functions which are associated with that particular object you know regular object type more precisely and it those methods will of course have access to the objects internal data. So, every object will have these stored attributes and associated methods which are actually functions having access to internal data. So, let us do an example, so, I am you know creating this you know string object here a with om space omkar.

So, if I run this and you know if I want to find out what other methods can be available for this object, then I will actually have to I will have to type . after this I will just to display the same thing to you, I will just remove this . here. So, as you can see I have you know if I just type a and . and then tap. So, I will get a list of associated methods that can access the internal data of the object a and can be applied.

So, this is just to show you different methods that are available for this kind of object type which is actually string. Now the same thing we can do by name also, so we can access attributes and methods by name. For this we will have to use this get attribute (), get attr (), where first argument is a which is the object and name of the (). So, in this case let us say split.

So, if we run this will get the output as you can see the previously generated output also, that () STR . split is available where we have two arguments, one is the separator which is by default null and then we have the max split which is by default -1, that means all the strings which are separated by none, they could be split. So, these are the default values these are you know keyword arguments.

Then if you want to call you know this is split () on the object a, which we have previously created on that home . ohm space omkar. So, if I just run this I will get this output where these you know they have been this string listing string has been splited and we have a first element as om, now it is the list first element to be omkar and the second element being ohm car. So, in this fashion we can access the stored attributes and the associated methods.

And use them as per our requirement. Now there might be certain scenarios where we might be expected to write () that can accept multiple kinds of input. So, we might be dealing with the objects and the our scenario might require to check whether that particular object is of a particular type and if not you know as per our desire type then convert it into one and then process.

That kind of scenario is typically encounter in especially in the data science context, let us understand how this can be done. So, writing functions that can accept multiple kinds of input. So, except any kind of you know case whether it is a list top all or array or even an iterate. So, what we typically do is we check if an object is of certain type if not then convert. So, this is the code that we can use as an example.

So, here what I am doing is I am checking whether the object a is list or not. So, the code is if not is instance is the () to check this thing. So, if not is . instance a list if is not an instance of a list type then converter that is a = list a, where I am calling the list () and converting into the list type. So, in this fashion when we would like to write () to accept multiple kinds of input these kind of conversion can be done.

So, let us run this, if I run this then let us find out earlier as we remember that a was a string type we had checked before. Now let us run this type a and you will see we will get it lists now, this particular object is of list type. Now next thing that we would like to discuss is you know importing a module. So, we have talked about library module and how to import in the very first line of this progress script.

And you know if we have a you know certain test script and certain functions are defined there we might have created. So, one thing is importing library files, then other thing is importing you know user defined user created files. So, we are touching on that for aspect when we might have created our own files, we might have written our own functions and certain variables that we would like to access.

So, how to you know do that in python platform. So, any file with . py extension which is expected to contain python code is considered a module and we can import that using import keyword. So, what we have to do is we have to just type import keyword and the name of the file which is expected to be a python file. So, import test our script the file that we have previously also used. So, we can import this one.

So, let us run this, now once we have imported we can use this particular file you know as an object as a module object. So, we have created a module object and where you can see test . script in that file as we know the let me open this file for easy reference. So, this is the file if I click this you can see this is the code that I am referring to. So, here you can see I have defined a () f which is just you know taking three arguments X Y Z and returning X + Y and Y divided by J.

And the three variables a b c having these values five, six and seven.five and I am also calling this () within this. So, result will store the return value of this () call. So, this is what I have in this test script. So, here in the module object that I have just created by importing. So, test or escape now . I will have access to this () and I can pass my own arguments let us say four, five, six and given value is going to be stored.

So, in this fashion we can import module and access you know define variables and functions. So, let us run this and you see that we will get this 1.5 where you know internalized we have seen the code for four + five nine divided by six which comes out to be one point five. Similarly we can access other objects you know which are part of this you know module object right now. So, for example test this script . a.

So, what we have to do we have to just typed as a script and then . and then now you know tab completion and we will get a list of item if I just do this. So, it will take so, in this module object whatever is part of this if I just type here test tab you can see all the options in the in a drop down menu is clearly visible. So, I can select any one and then and will have the output. So, in this fashion we can access any object that is defined in that particular file.

There is another way to perform the same thing where we can use the from keyword, so from and then the name of the file python script from test script and then again import. So, if we do not want to import everything and that could be considered a good practice to not import you know everything. So, from testscript we want to just import these you know objects A, B, C and F.

So, this way also, we can you know do this for process. So, let us run this, now again we can call this now because the () is available with us. So, in this fashion we can call, now we do not have to type like test script . and so, that we do not have to do and now we have this () and will be they will do execute. Let us move forward. So, as we have discussed before also, module and module objects they can also be imported with different names.

Just like we did in the very first line of you know this particular script, we were importing the library modules. So, here also this you know python file testscript, we can import with the different name that we have to write like this just like we did for library modules import test script as TS. So, this would be imported So, if we run this so, it will be available. Similarly there is another way to you know do the same exercise.

And with the similar results we can use from keywords of from test script and import what we require. So, from test strip import a as capital A and b as capital B. So, this is something that this is another way to perform this. Now let us move towards how operators are used how we work with different you know object types and how we apply operators in python. So, let us start that part of the discussion so, let us start with the binary operators.

So, add, so how this is done, so, in as we have discussed about certain important aspects of python, that python being interpreted languages and not just that many analysts, many programmers consider python as you know something you know executable pseudo code. So, which is very clearly you know visible from here if we want to add two values 18+48.5 let us say these are the two values if I just run this, I will get the output.

So, in this easily you know just like a executable pseudo code I can go about writing these lines of code, subtract eighteen minus 18+48.5 if I run this I will get the answer. If I want to do comparisons where I want to find out whether eighteen and this particular value greater than or = this fourtyeight point five I done this, I will get my output not false. Similarly if I want to check whether two variables, two object, two variables are referring to the same object or not.

So, one of the ways to do this is using keywords so, we have this each keyword. So, we have two variables A and B and want to find out whether they are referring to the same object or not. So we have to just type A is B, and if I run this I will get false, because both of them are referring to different objects. Similarly V1 is V2. So, if I run this I will find out and as you remember that V1 and V2 they were referring to the same object.

So we get the answer as true, now we have another keyword visit is not. So, in this case we can find out whether two objects you know are different so B is not C. So, because that is the case we will get true, then we have equals whether two operator they are equal or not so, because V1 and V2 they are referring to the same object. So, we can say V1 = V2 so, we can run this and we will get the answer and as one of the boolean value.

So, you can see true, because both of them were referring to the same. Similarly there is a none object so there is just one instance of none. So, we want to find out whether a particular variable that we might be using whether that is none or not. So, we can say B is none and using you know its keyword we can find out and if I run this it is clearly false because we had a value as we know. Similarly if I want to do multiplication.

So, a and asterisk operator to perform the multiplication a multiplied by b, I will get this value divided. So, this forward slash a slash b and I will get the value if I want to do flow divided well you know I would like to you know take a floor of the output of a divided by b. So, in this fashion double forward slash I can use and I will get the flow divided, you can see you know this is zero, because the actual value divided value is zero point eight three.

If I want to do certain other operations mathematical operations like raised to the power a raised to the power B. So, in this case if I run this you can see I get this, if I want to do certain other mathematical operations like you know bitwise and especially both are integers or for if they are boolean types then we will get true if both are true. So, a and b so, depending on the type that a or might be will get the output.

So, in this case both of them are integers, we get the bitwise and the value comes out before, then bitwise R in case both are boolean's then true, if either is true so, that is the kind of output that we will get. So, in this case both integer we get this and similarly you know bitwise exclusive R. So, if they are boolean then for Boolean true if either is true but not both. So, we use this operator and we will get the output.
(Video Ends: 26:45)

So at this point we would like to stop here. And we will continue our discussion with some more mathematical operators and other functionalities that are available in python platform, thank you.

Keywords: testscript, python, functions, bitwise, Boolean, dynamic referencing, compiled programming language, interpreted language.