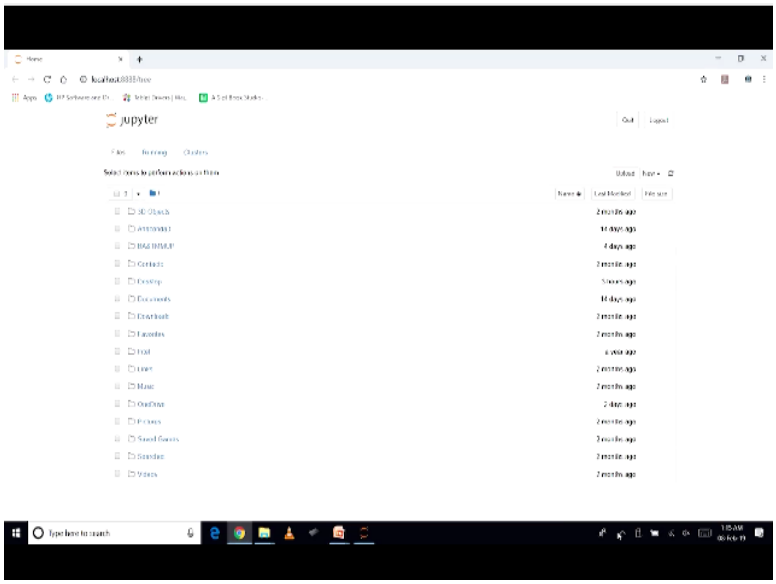
Business Analytics And Text Mining Modeling Using Python
Prof. Gaurav Dixit
Department of Management Studies
Indian Institute of Technology-Roorkee

Lecture-0five
Python for Analytics -Part II

Welcome to the course business analytics and text mining modeling using python. So in previous lecture, we started our discussion and exercises in python. So let us go back to the same thing. So as we understood the installation aspects, how to launch Jupyter notebook, and various other related things with the Jupyter notebook in previous lectures.

(Refer Slide Time: 00:50)



So, let us just get started from where we left in the previous lecture. So let us open this file session python basics.

(Video Starts: 0one:0one)

And in previous lecture, we had gone through some of the basic functionality about python. So, we will do a recap of that, and then move on. So, some of the things that you know, I wanted to make you familiar with is like, you know, loading require library modules, as you can see, in the first line, we talked about, you know, NumPy, and how can we loaded import NumPy as NP as NP just indicating that we can all load any module, any library module with a different name.

So, there is going to be original name like NumPy, and then we can load it using a different name shorter, you know, name like NP. So, after that, you know, typically, we would like to set certain parameters, as we discussed in the previous lecture as well. So, let us just go through this. So, let us run this. Now, we have gone through this first line of python code hello world thing. So I am just going to this variable creation, tab completion.

This aspect we have looked at and then we looked at, you know, further tab completion features, other type completion features where we used to . thing, then to see info about an object, so, how to use this question mark operators, this have gone through in the previous lectures and just running through this and writing a function. So, you know, though, we would be later on doing you know more about this.

But to familiarize you with this, let us go through this whatever we discussed in the previous lecture, if you want to display the source code, for example, this one you can see here, source so, this part we have discussed in the previous lecture. Now if you want to search the namespace to this is how it can be done. Let us move forward. Now we talked about magic commands as such LS magic which will display you the will line magics and cell magic as we have discussed before.

Some of the magic comments we learned to use in the previous lecture. For example, if we want to find out our current working directory, you can see here % the operator and then PWD it will help us print out the current working directory. Then if you want to list all the files and folders, so % LS is the magic command here. So, we can just run this and whatever is present in the working directory that is going to be displayed as you can see here.

Now, if you have a separate file as a Python script, so how we can run that Jupyter notebook, so that also we can see % run and the name of the file. So, you can see this you know line magic, where arguments require for that command to be executed or also passed in line, in the cell magic, the following is the lines of code would also be executed. So that being the difference, so let us run this.

So, once we run this particular piece of code this python script, so there we have defined a function F and a few variables ABC. So we want to just access those variables and find out what are the values stored there, we can immediately see that we had also invoked make a function in that script. So result is the returned you know variable for the returned value to store the returned value.

So that we can again you know, execute and find out. So, all these things we had discussed in more detail, slightly more detail in the previous lecture. Similarly, we want to load a script into a cell. So we can use this magic command, % load and then the name of the script. Here, you can see that this is the output that we had executed in the previous lectures, let us skip this. And let us move forward then, we stopped at this point in the previous lecture.

Now, the next thing that we would like to discuss is that about integration of data visualization tools in Python, and you know, in Jupyter notebook platform. So, the, we talked about particular library, a module that is Matplotlib, so how that can be integrated, how the different functions and methods available in that particular library module can be accessed here.

So we need to use this magic command here, % Matplotlib, and this space in line. So this is slightly different from what you might have to execute in ipython cell. But this will work for Jupyter notebooks. So let us execute this. So once we run this, now another thing that we can do is that there are different, you know, within that module, there are going to be different, you know, modules as well.

So that we can import for example, Mattplotlip . pyplot, so that we can import with a, you know, different name PLT. So, let us run this one also. So within this model, there are going to be a number of functions, so that we can access so one of them is like PLT . plot, and where we can pass on this argument, which is another is again a function and NP.random . pandon and we are passing fifty.

So we would like to create fifty random, you know, fifty random values, and then . comes on, then we would like to, you know, compute the cumulative some of those values. And because

we are passing all these values into the PLT . plot function, so those are going to be plotted. So in one goal, this particular piece of goal is going to be executed. So let us run this. So you can see this plot is generated.

And internally what I was explaining you, if you want to, you know, break it down to smaller steps. So this was also something that I have done here, you can see I am calling here NP.random . randn and fifty values I want fifty random values I want, so if I just run this, I will get these fifty values. If I want the cumulative same of you know cumulative pattern of these values. So again, I can run in this fashion, how the syntax is working, you know.

And the specific language semantics and programming constructs in python, we will be discussing, as we, you know, go along this lecture and coming lectures as well. So as you can see, that what I have done the first, you can see fifty random values have been generated, and then the cumulative sum of those fifty values, as you can see in the output. So, that same thing was actually plotted there.

So, let us move forward. Now, we will start our discussion on the essential programming concept, programming constructs that are available in the python, and the language mechanics as well. So let us start, so first thing is about how the code is to be written, and how Python is going to be different from something that you might have learned in other programming languages.

In terms of, you know, different types of language that we have talked about the compiled programming language, and interpreted programming language, languages. So Python is the interpret programming language and typically, you might be more familiar with the compiled programming language. So there is going to be, you know a slight differences when it comes to Python.

So first thing indentation there in this programming language, python, we do not use braces rather we use wide space, which includes tabs and spaces. So these wide space is actually used to a structure the code instead of basis as is typically done in compiled programming languages.

So a small piece of code that I have written to you know, to explain the same thing you can see for loop is written there,

And you can see that the first statement for x array here and that is very clearly stopped at you know, colon is being used to you know, denote the start of an indented code block. And then we have the conditional statement, if and again, you have the colon, because an another indented law code law is going to be started and then we have else and then colon and then another intended code log is going to be starting.

So, colon is used to denote the start of an indented code work and wide spaces are to use the structure the code instead of basis. So, you can see, when we look at line for x in array and the column, then the next line is after one tab that is 4 spaces. Similarly, when we you know when we have another column at the end of statement, then again one more tab is used to intend the next upcoming block.

So, in this fashion indentation is done and braces are not used as you might be familiar with in other programming language. So, let us run this. So first we are creating this list named as array here and then pivot is another variable, and then we have less and greater to empty list here and then we are running a loop where for every object that is present in this list named array , and then we run, then we compare that particular you know value that element x from the pivot word value.

But is less than that, then we append that particular element in the less list, otherwise, we append that element in the greater list. So, let us run this block. So, you can see less, after that, we come to the next one, let us find out the value of less that is one, two, three, four and the value of greater you know list that is five, six, seven, eight, nine. So, the starting you know list was having all these nine values, and because all the values which are less than five have gone to less list.

You know all the values which are = or greater than five have gone to the greater list. So, this example one may lead to you know, tell you how the structuring of the code, how indentation is

done, just to display that particular aspect of python. Now, another important difference you know, with other languages mainly compiled programming language is that a semi colon uses of semi colon.

So, in this case, in python, semi colon is actually used to separate multiple statements, which might be present in the single line, instead of termination of a statement in many, you know, compile programming language semicolon is used to terminate this statement here, that is being used as separator. So, you can see, we have this example, this line of code, where we have a is being assigned the value of five, so a = 5, then semi colon to separate it from the next line next statement, which is written in the same line, that is b = 6.

And then against semi colon, that is to separate these two statements with the another statement that is also written in the same line of code that is c = 7.5. So we run this block, so all the variables objects are going to be created, and these values are going to be assigned. Now, let us move forward, let us discuss a bit about a python object. So as far as any number, a string data structure, function, module, pretty much anything that we create in python environment that is considered by the python interpreter as an object.

And it is like, you know, that object is like, you know, its own box and you know, referred as python object and it could be anything starting from number is string, as I said, data structure, function class and all those things, and each object is going to have its own associated type. So, it could be integer or strings, those types are also going to be there and internal data. So it will have its own type and also internal data.

When it comes to you know, creating comments section in python programming language typically # mark as you can see here, the pound sign here, this is typically used to add comments, and also sometimes he would like to exclude certain code or blocks, we might have written for certain experimentation purposes, and we might not you know, like to execute that part of the code.

So, we would like to comment that out. So, they are also # could be used. So, typically comments that are preceding a particular line of code, they are typically written to for explanation purposes to explain what we are going to record next, how we plan to code next few lines, next few statements to explain that part. And typically, when we use a comment after a line of code, that is typically used for debugging purposes.

So, these are main, you know, main, you know, in a way practices that are being followed in terms of you know, creating comments section, that is common across programming language, same is the case in python. So, here you can see one line of code I have written print these to this line, and then after that, you know, that the statement we have this # symbol status report, that is mainly for you know, debugging purposes.

So, this is why I have out here, and to explain, you know, all this that I am you know, discussing, I have a number of you know, comments section before this line of code. So, that is typically for as I said for explanatory purposes, so, let us run this. Now, next thing that we would like to discuss is function and object method calls. So, here you know let us understand function call when it is being when the function is being called without any arguments or you know no return value is going to be there.

So, no written value assignment is required. So how that is done. So, you can see the we use this keyword def DEF to define a function and the name of the function and then the parentheses to indicate that this is a function. And then again, this is also you know, indeed, y a colon which will again as we have discussed will indicate the start of the intended code of you know, a block of code.

So, you can see DEF for the defining G function, you know, parentheses and colon, and then we have just one line of code that is print function with no arguments, and then we are calling this function G. So, we run this code will get this as our printed output. Now, if we want to make a function call, which requires us to pass on the arguments, and we might also have a written value which then will be assigned. So, for example, we have this function f that we have defined previously.

So, we pass on these arguments the A, B, C and that whatever under functionality is present in this function definition f definition that is going to be processed, that is going to be executed and the return value is going to be assigned to the result variable. So, let us run this and you can see, the next line was result itself. So, we got in the output the 1.46 the value that is there, let us move forward.

Now, there is another case, there is another thing called methods, which is function attached with objects. So, function which are attached with objects are typically referred as methods in python, and they will have access to objects internal data. So, any processing, any computation, any steps that computational stuff that you want to perform, that you want to execute, the methods will have access to that data and it will be easily accessed and done.

So, for example, we had created a less list there, so, if you want to call this method append, so, we can just write like this less . append, there are going to be other methods available, which can also be used, and which can also work on this particular type less, which is a list and in this case, what we are doing is we are adding this pivot value into this list. So, we are calling like this less . append and pivot.

So, earlier as you might remember that less particular this particular list less had just four elements one, two, three, four, now in the output, you would see that another element will get added that is five because pivot value was five and there is another aspect related to arguments that is know there are two types, positional arguments and the keyword arguments. So, whenever we are defining a function, so each argument has its position clearly embarked.

So first argument, so whatever we want to pass and its first argument is to be you know, written fast, is to be type fast and then second argument and then third argument. Then there are keyword arguments where the key word is used and a value is assigned, and that value is going to be the value for that keyword inside the steps that are going to be executed within the functional scope.

So, these keyword arguments might not have a fixed position. So, they are to be identified by function with their name. So, you can see here we are writing a function H. So again, to define the function as we need def H and first argument is positional argument, which is x, then second argument is a positional argument Y and then third argument is positional argument z. So all these three arguments they have to be in that order.

So, whatever we are passing as x or y or z, they have to be passed in those positions position one, two and three, then we have keyword arguments W, which is you can see equal operator is being used, it is being assigned the value of nine, so w = nine, and then we have another keyword argument t it is being assigned a string o so, these five arguments that we are using to define this function, and within the body of this function, we have just one line of code, where we are painting this first t you know which is a string type object.

And then we are doing this computation in the next object that is x plus y plus z, first three arguments, which are positional argument and some of those three arguments are is being divided by W the whole key word, you know, based argument. Now we can call this function like this, we got a calling edge and within the parentheses, we have A, B, C, first the argument which are positional argument.

They have to be specified in that order itself, then W=five and T= you know, we are passing a different value for W and we are passing a different you know string for t , if we do not pass these values, and the default values are going to be taken, so let us execute this. So you can see omkar is printed and a value is also computed. Now in the next example you can see, I am calling this particular function differently.

Here you can see you know, calling H and within parentheses, I have A, B and C, and then I have you know reverse the order of T keyword argument and W keyword argument and you we will still get the same output, because other things remain same, let us move forward. So, now let us discuss variable assignments and argument passing.

So, how valuable assignment takes place in python, and how argument are to be passed, so some aspects related to these two things. So, whenever you know, we create a variable and you know that variable, you know might be assigned to another variable for these wherever will start referring to the same object. So, in R this is one difference, where you know, once we do not create a different copy of a variable.

When a variable is assigned to another variable a new copy is not created, rather both the variables start referring to the same objects. So, in a sense, variables are just named for objects within a particular name space. So whatever scope of namespaces there in that they are just different, you know names for the same object. So, for example, we had earlier created V1 and V2 and now if we run this block, so we are assigning V1 to V2.

And if we just you know, run V2 and you can see this having this these values one, two, three, which were originally created with V1, and now if you want to test this particular you know, does this particular aspect, again so, this is another way. So, what we can do is we can add another element append another element in this list. So, what we are doing is V1 or append we are adding element with value four.

So let us run this and instead of printing V1, we are painting V2 here. So, we print V2. So, let us see, and you can see one, two, three, ,four that value which was actually appended with V1 is also you know part of V2 because both of them are referring to the same object, let us move forward. Now another important aspect of you know passing objects to arguments as arguments to a function that we like to discuss.

So, we are passing objects as argument to a function within the functional scope, new variables are created, and those new variables are actually referring to the original objects, which are being passed as arguments. However, because they are being created in the functional scope. So, as the functional call concludes, those you know, available, those new variables are going to be destroyed.

Now, if we make any changes in the new objects, which are created within the functional scope, or let us say local namespace, then those changes will not be reflected in the parent scope. So, in the parent scope we are we might be passing certain objects as argument to the function, there would not be any changes, no changes would be reflected there, because they are in the parent scope, parent name is space.

And whatever gets created within the local names space within the functionally scope, that is, you know, processes their computations happened there, and it can be accessed there. And once the call you know terminates, then those objects are also destroyed. The same thing we are trying to display here. So, we again we are creating a function, so def and then append underscore element.

So in this we are passing a list and the element that we want to append to that list. And again for that list, we are calling the associated method that is you know and then append and we are passing element as the argument again and to display this you know, once this appending happens, we are printing this list, so, that we will get to know that list is updated within the functional scope.

The local variable is updated within the functional scope, and we can also you know, print the value as well. And after you know, once you know, the call will happen to this function. And once you know in the parent scope, we can make a call to this function and there also we can print the you know, variables defined in the in the parent scope and see how this is happening.

So, let us the first you know create this, see here the definition would be created and these two statements a list and element would be printed within the local when they were there in the local name space of this function, and we are also making the call. So, all this will happen here, you can see in a list we have one, two, three, four five, now five has been added into the list and then element is also printed that is five.

Now we are calling append element also. So that is a V1, V1 we are adding five. So because of this call this printing happened, the previous printing happen, let us look at what has happened

in this V1. So we just run this, you can see this is also you know one, two, three, four, five. However, if we want to access a list here, and we run this, you can see we will get this name error, because a list was destroyed it was created within the functional scope.

And later on, you know it was you know destroyed. So, we get this error that a list is not defined. Same is the case with element if we run this again. So outside the functional scope, this also does not exist. And you can see the error that is there. You know, element is not defined.
(Video Ends: 29:20)

So at this point, we would like to stop here, and we will continue our discussion on python in the next lecture, thank you.
Keywords: magic commands, arguments, NumPy, parent scope, Python.