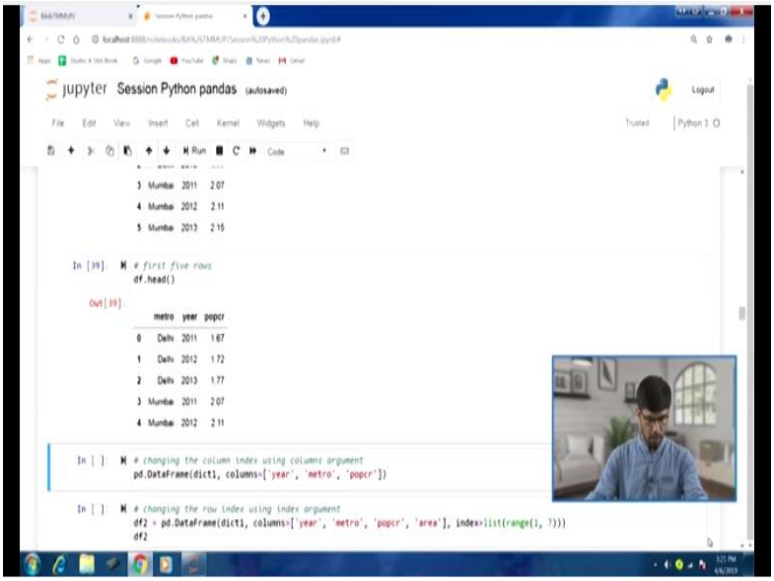
Business Analytics and Text Mining Modeling Using python

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

Lecture-27
Database Using python-Pandas-Part II

Welcome to the course business analytics and text mining modeling using python. So, in previous lecture we had discussed these 2 data structure series and data frame from the pandas package. So, we were specifically discussing the data frame object data frame data structure. So, let us pick up from where you know from the point we stopped in the previous lecture.

(Refer Slide Time: 00:48)



So, we talked about data frame and as we said that sometimes you will be dealing with larger data sets. So, therefore we just want to have a look at first few observation, first few rows, so therefore head method can be really useful in that context. So, we can type a command like df.head and we will have those you know first 5 rows as an output. Now the next thing is you know changing the column index using columns argument.

So, this is you know we can use the data frame function that we have, so just like the index argument that we have to change the you know row index. Similarly we have the columns argument to actually change the columns in you know index. So, columns index as we discussed is about the variables name. So, we can always you know specify our own customized you know ordering for our you know variable names and all that.

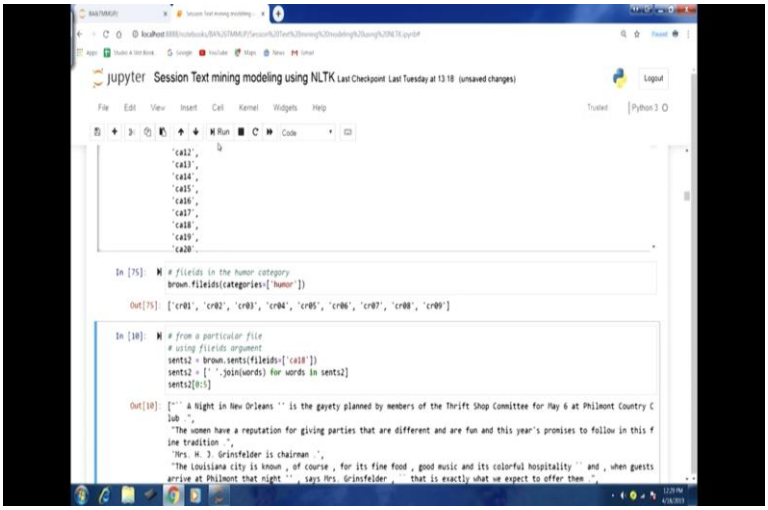
So, if you look at this code pd.data frame dict1, so we are trying to create this you know data frame object from this dict1 and columns you can see year metro and popcr. If you look at the output number 39, so their ordering is metro year and popcr, here the ordering has been changed, so using columns are gone, this is how we can actually do this.
(Video Starts: 02:10)

So, let me run this and you will see the output, here metro and popcr and the ordering has been changed otherwise the where we have use the same dict object. So, let us move forward now let us talk about the you know as we said you know we can also change the row index using the index argument. So, in the next example we have taken both we are changing both columns index using columns argument in the data frame function.

And also using index argument we are going to change the row index, so if you look at the columns argument, we have a specified year metro, popcr and we have added one more area. And so what is going to happen is when we add you know this kind of you know another index value here in another level here. So, but this is not going to be present in dict1, key part, so therefore unmatching will happen and one additional row is going to be created index and row corresponding row and with the index value is going to be in corresponding column and with the index value area is going to be created.

Similarly if we look at the index argument we have a list range 1 to 7, so I think in the dict1 object as you can see in the output 40 we have 6 rows. So, range 1 to 7 will give us 6 indices those index values, so we have fine with this. So, let me run this and you can see in the output that we have you know because of the customization of the index, we have an additional column area.

(Refer Slide Time: 03:49)



So, this additional column has been created as a result of index customization, so and the values because the values were not present in a dict1 object that we used to create the data frame object here. So, therefore all nan missing values have been indicated in this case, now let us have a look at the you know the columns index using the columns attribute. So, you can see here, now how do we access you know different columns if which is want to access 1 particular column.

So, how do we access that, so accessing column using dict key like notation that just like we you know discuss there, the same kind of notation we can use in the data frame context as well. So, df2 and within brackets we can indicate the label within single codes popcr, so if our indexes actually you know made of a string values that means the labels. So, we can specify that within single codes and then within the brackets operator and we can access that particular column.

So, if I run this line of code you can see output 43 and you can see the column corresponding to popcr that has been produced in the output. Similarly you know there is another mechanism to actually access these columns. So, we can access column using attribute like notations, so just like we have been using other attribute like index and columns and others similarly column names they also become they can also be treated as an attribute.

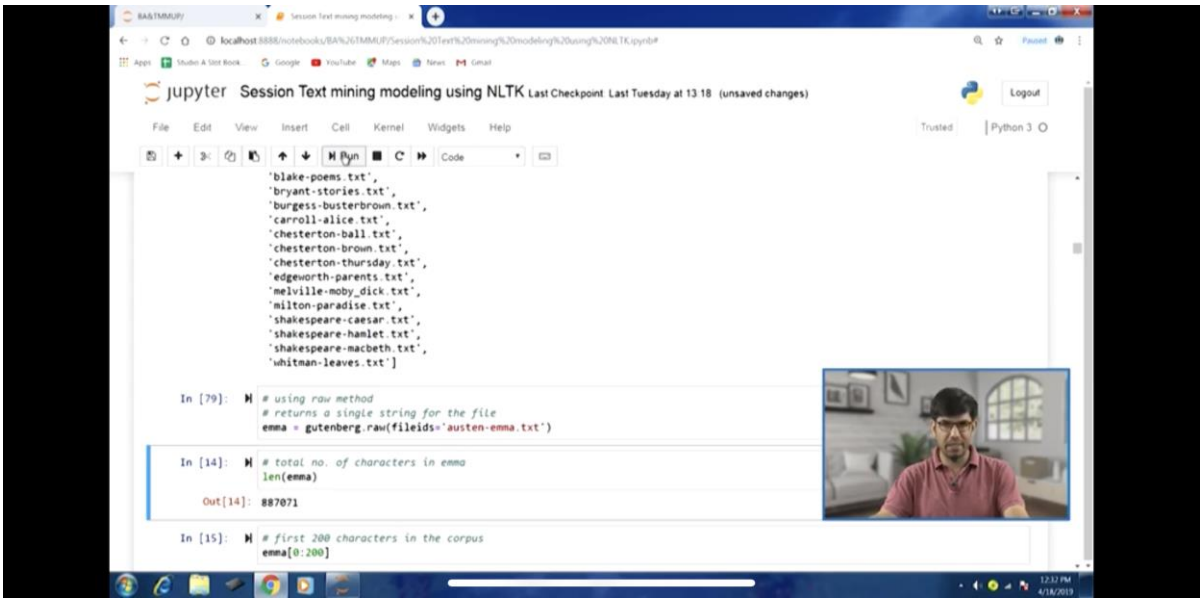
So, we want to access year column, so we can type this code df2.year as you can see next line and you can see this year all the data that was there in the year column that has been produced in the output. Similarly we can retrieve the next another column popcr, so using the attribute like notation df2. popcr.

So, if I run this you will have the output 45 will have this you know in the output.

Now sometimes why we are selecting columns, so we would like to you know select particular row indices. So, you know more on this will be discussing further later on however few examples I have shown here as well. So, if we want to select few rows you know for a column. So, we can specify like this df2 and within brackets we can specify the column label here, so let us say popcr.

And then another you know brackets because you remember that they are based on arrays and we said this is another way to you know access you know multidimensional array. So, in this fashion you can indicate you can use this column you know operator here 1 to 4. So, we would be selecting in this output those rows are going to be selected and for that column.

(Refer Slide Time: 06:09)



So, if I run this, so you can see for the popcr only the rows corresponding to indices 2, 3, 4 have been selected, so you can see this in the output. Similarly you know a few more examples df2, popcr and just we have just mention this stopping point here. So, you can see year and the output similarly we can use this -3 also, so it will be counted from the end. So, you can see that kind of output has been displayed.

Similarly another example here you have not mention any start, stop points, all the observations have been selected for that column. So, in this fashion we can access you know particular columns also and also if you selected rows as well. Now there is another mechanism that is loc attribute, so for a more

precise you know access of you know rows and columns and values in in data frame and series object we can use these attribute.

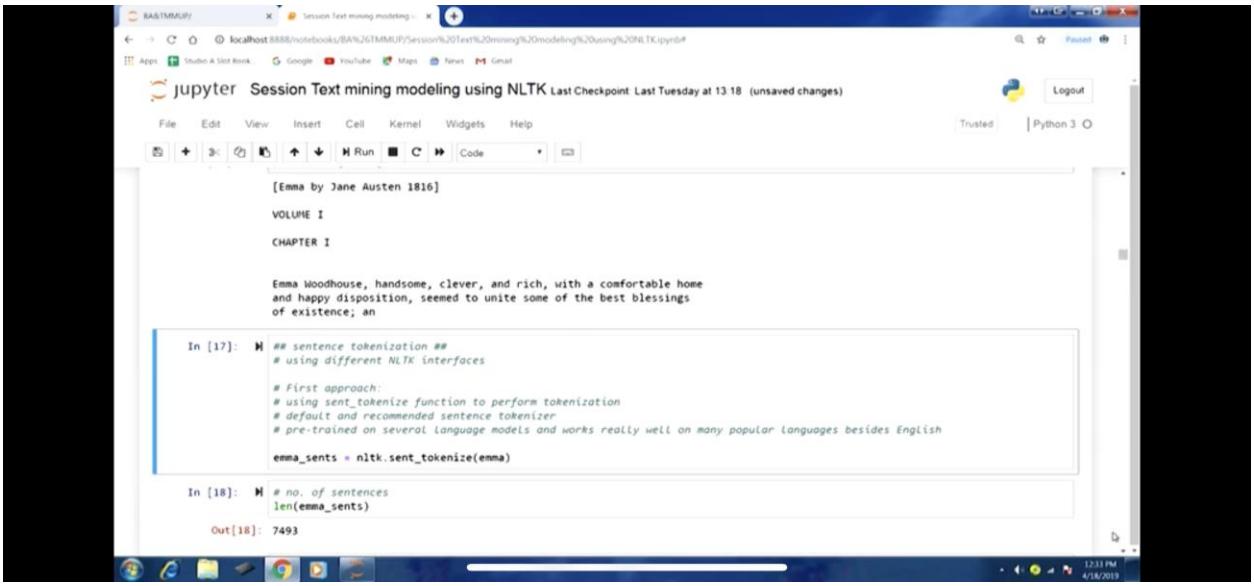
So, in this case we can specify like df2.loc and then we can specify the row, so in this case 1. So, if I run this df2loc1, so first row corresponding to you know index value 1 is has been displayed here. Similarly another example df.loc4, so you can see, so this loc attribute or locate sort of locate attribute is can also be used to access these value. Now let us move forward, let us now talk about another aspect of you know these data structures.

So, we will focus on data processing and transformation aspect of how processing and transformation can be performed using these series and data frame objects. So, we will take few examples, so let us first start with the value assignment, so let us pick area column. So, in this case you can see using the brackets and specifying the label area we will be able to access the column and on the right hand side we have just mentioned one values.

So that is going to be broadcasted across all rows. And all the rows and across all rows all the area volume values will have 6000 as the value. So, if I run this and you can see in the output the area which earlier had nan values. Now you can see that for all the rows we have 6,000 as the area value. Now sometimes we might be required to populate values, we might be just you know manufacturing certain data set for certain demonstration certain you know implementation.

So, all for some other purposes also, so sometime these function can be really useful, these mechanisms can be really useful. So, let us pick again area column here and you can see you can use this np.arrange function and let us say starting value 6000 and ending value 6600 and this step is 100. So, in this fashion instead of having just you know the earlier when we did perform the value assignment the same value was assign to all the rows.

(Refer Slide Time: 00:48)



In this fashion also we can populate we can use the arrange function and populate equally distributed values in a certain range. So, if I run this one and let us have a look at the output, so in the df2 you can see that along the area column in the output number 55, we can see first row we have this 6000 and second row 6100 then 6200 and so on, so in this fashion we would be able to populate the values. Now we can also use series to populate values, so as we have discuss before that if there are any unmatch indices, so they are going to be treated as missing values.

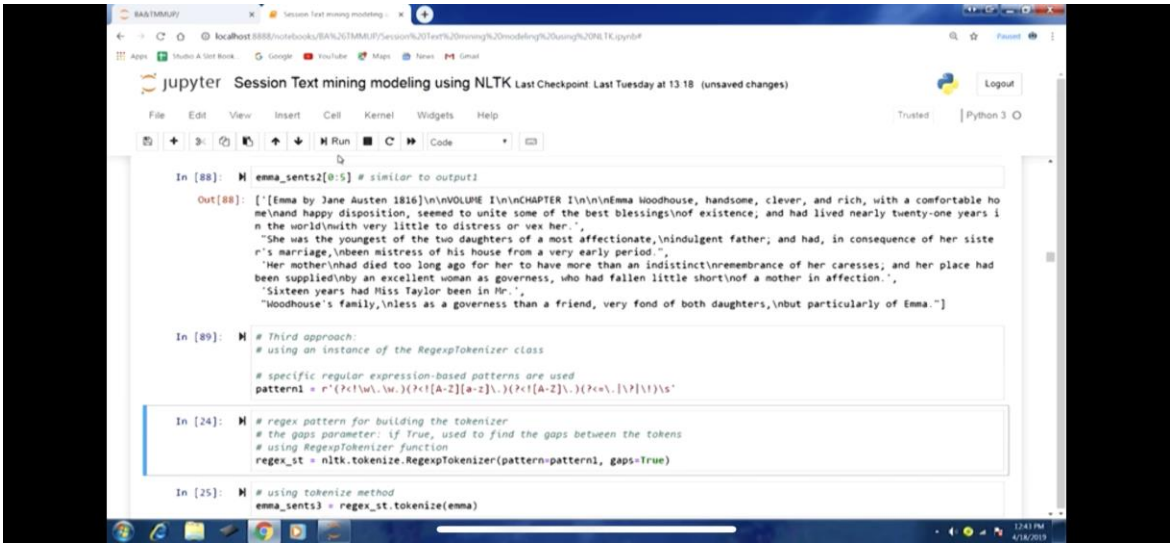
So, let us take you know few examples, so let us say we have this seriesval and we are going to create it using this pd.series function, we have 3 values and 3 indices. So, you can see indices, if you look at the indices we are mention 2, 4, 6. So, the remaining you know indices we are going to you know populate. So, and then the next line of code you can have a look they up df2popcr. So, we are picking up the popcr column here and we are assigning this val series object there.

So for this particular column we are populating using the series object . And in the series object as you can see we can always specify the index, so let me run this. And if I look at the output you can see here that popcr because we had a specified just 3 indices, so 2, 4, 6 and we had assign that this is object for this column. So, the remaining values they are missing values nan and those corresponding to these 3 indices 2, 4, 6 they have the values from the series object.

So, you can see in this fashion we can populate the data frame object using a series object. Now we might also have you know sometime we would like to add certain column and also we would like to you know populate some values also, so that can also be done. So, you can see here in this example what we are doing we are assigning values to a non-existing column let us say northern. So, in essence we are saying out of the metros that we have which are in the northern part of India. So, if this column is right now not existing.

So this is going to be created you can have a look at the code. So, on the left hand side we have df2 northern and our right hand side we have a conditional statement which will produce a boolean sequence. So, df2.metro==Delhi, so as you know that from the metro column that Delhi is there and Delhi happens to be in the northern part. So, using this you know expression will have the you know boolean output there for wherever Delhi is present it will you know for northern value for that would become true which is the you know correct output.

(Refer Slide Time: 10:45)



And for the you know other rows where we have met Mumbai as metro it would be false. So, using this one line of code we would be creating a non-existing column and we would also be able to populate the values also. So, if I run this and have a look at the df2 and you can see, in that another column has been added at the end not northern and you can see or Delhi it is true**.** All the rows you know that are having the Delhi as metro it is true northern column and for the

Mumbai it is false.

So, in this fashion you can see how we were able to create a non-existing column and populate as well. Now sometimes you would like to remove a column, so for that you know like before we can use del keyword. So, in this case one we have just now created northern column here, so we can use del keyword and df2 and within brackets we can pass on the name of the column, label of the column, so that would delete this column.

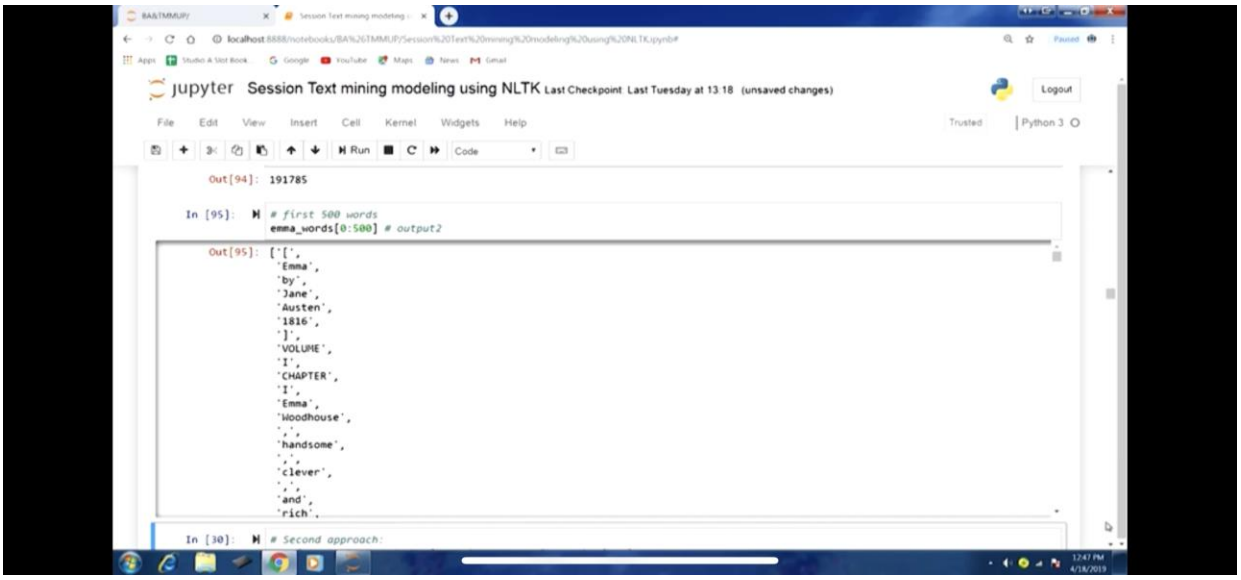
So, if I run this and let us have a look at the columns index you can see just for 4 names are there here metro popcr area and the northern is actually deleted. Now let us take another example in this one we are going to be using nested dict of dicts to create a data frame. So, how this is going to work out, so what we will do outer dict keys in this case we will use as column indices, you can have a look at the example also there and the inner keys will use as a row indices.

So, if you look at the this nested this dict object that we are talking about you focus on the right-hand part, we have Delhi as the you know key. And then in the value part this is another dict and there we have year and population combination, so 2011 1.6, 2014 1.9. So, you can see whatever you know data that we have been using till now while creating these data frame objects, it can be presented in this fashion also as a that's nested dict objects.

So, we can create this dict2 object and now we can use that to create a data frame, so let me run this and we dict. Now we will pass on the dict2, so what will happen inner dict keys they are going to be combine and you know they are going to be sorted to form the index. So, if I run this tf3 I will look at the output, so now you can see that in the Delhi part of the dict2 object that you can see we had just 2 years 2011 and 14.

And the Mumbai part we had 3 years 2011, 12 and 13, so of course the indices the you know inner dict key is they will have become the row indices and of course many of them were not matching. So, all of them is union sort of union of indices has happened and all those rows for those corresponding indices are present in the output. And we have these 2 columns Delhi and Mumbai and wherever we did not have the value for whichever year nan has been introduced.

(Refer Slide Time: 21:10)



So, you can see wherever we had the matching data, so the automatic alignment of indices and value that has been you can see how that link is working out. So, union of indices happens and wherever the values not present nan has been filled, so you can see in the output. So, this is how we can create the another example of creating data frame object. Now because as we have discussed the data frame they are like 2d arrays, so let us also focus on transposing a data frame.

So, essentially it is about swapping rows and columns, so sometimes this is also going to be useful, so if p just transpose df3.t. So, you can see in the output, so you can see, now the row indices they have become column indices and the column indices have become row indices. If you just compare the output number 63 and output number 64 you can see the indices have been reversed.

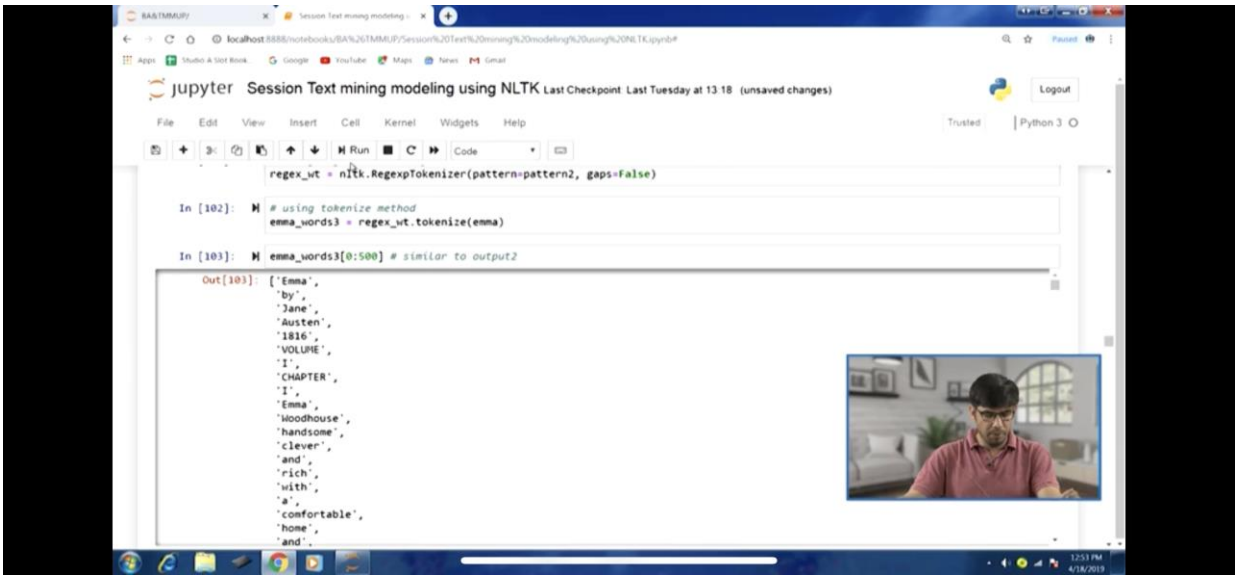
So, transpose t attribute has been you know can be really useful sometimes, so here the you know variables have changed and the name for row indices have also changed. We can also use a dict of series to create a data frame, so as we have done you know before also but here we are you know working slightly differently. So, in this you can see we have Delhi as the key and in the value part we have a series object here.

So, you can see from the df3 data frame itself we have selected a particular you know series Delhi column here and we have selected certain rows also. And similarly for Mumbai part of the you know dict in the series part, in the value part we have the series where we have picked up Mumbai column and you know certain number of rows as well. So, you can see in this fashion also we can structure our dict object.

So, you can see from the you know data frame that we have df3 from that itself, we are able to create a dict2 object and later on we can you know transform it. So, all these examples are in a you know certain kind of transformation which might be you know useful. So, if I run this and have a look at the dict2 object, so you can see and we can use this to create a data frame, so next line is that.

So, using the data frame a function and you can have a look at the output that how you know for those rows. So, wherever the matching happened, so for that you know we have the values and wherever we still have with the nan you know indicating the missing values. Let us move forward, now let us talk about the name attributes in a data frame, so earlier we talked about in the series context.

(Refer Slide Time: 23:47)



Now let us discuss this attribute in the data frame context, so df3 let us take this example. Now in this you know index we can change the name of the index here using the name attribute, so df3.index.name we can keep it here. Because in the index we have those year values, so this is how we can change, so quite similar to what we did in the context of series. So, df3.columns.name, so here we have 2 indices, so for both of them we can change the name here, so df3.column.start name.

So, we can have a look at the output, now in the output you can see that for the columns name we have metro there. So, you can see in the very first row it is look like a header then we have metro then Delhi, Mumbai. And then for the row indices you have year mention there and then so index obviously along that column. So, in this fashion we can use the name attribute as well to define our data frame.

Let us look at few more things about values attribute in a data frame, so when data frame columns have same d type. So, if we look at the df3 values, so they will look like something like this in this case you know all the values that were there in this data frame df3 they were you know it is safety of the columns. They were of same data types, same dtype, now when these columns they might be having sometimes you know one might be having string values another

might be having integer values or floating values.

So, different dtypes, so in that case for example we have the state this data frame df2, so if I run this you can see every column is having a through different dtypes have been used. So, if we look at df2.values here, so you can see in the array here and how those values are going to be, so how this values attribute is you know going to display those output. Now let us move forward, now let us talk about few more points about index objects and series and data frame.

So, one thing is that these are immutable you know array like objects because these are indices, so we are not to suppose to modify them you know. So, of course we customize them, so that flexibility is there customization of indices however we are not supposed to you know modify, so they are immutable array like objects. So, they also store certain metadata apart from the you know index values the access label that we have.

So, to understand this we will take this example here, so in this we have sequence of labels a, b, c which are going to be use while creating a series or data frame. And you would see that they are going to be internally convert into index objects. So, we have been talking about you know indexes to row and column indexes for data frame and just the index for the series. Now we are talking about the index objects how they are created in the python environment.

So, if I run this one and if we look at the series5.index you can see index and the output number 76 and you can see in the dtype it is saying object right. So, but however you can see in the output itself they are very similar to array like structure. Now in this certain operation we can perform with the index object let us talk about the series5 index. So, in this case we have storing ind1 series 5.index and then we are using this subset call an operator we are sub setting this.

So, ind1 and 1 colon, so starting point we are mention, so a is gone and we are left with bc. So, in this fashion we can work with the index objects as well and select particular indices similarly let us take another example ind1, 0 to 2. Now a few more things that we want to discuss is one thing is immutability. So let us say ind1 and 1. So, that is the index, so if I try to change it and try to make it assign it a different label let us say d.

So let us see whether this will work or not. So, if I run this an error has been generated here, you can see index does not support mutable operation. So, because the index is already there index has been created and the associated number is there for accessing that index. So, that is not allowed that, so every label is associated with a number that is not going to be allowed for modification. So, however as we have discussed we can always customize our index.

So let us take another example. In this case we are going to create an index using index function, so this is one function that can we actually use just to create index. So, earlier we have been customizing index using our you know series function and index argument or you know attribute, index attribute or you know columns attribute or columns argument. Now in this case we are going to use the so, pd.index function and within that we are calling np.arrange function and we are going to create this index.

So, let me run this you can see this is the index, so you can see the index with 3 values 0, 1, 2. Now while creating a series object let us series6, we can use this index here, so pd.series and we can run this and we will have the series object with these values. Now from this index we can you know find out series6 we can do whether we can find out whether series6.index is same as ind2. So, if I run this will come out to be true and another aspect about index object is that the display set like behavior.

So, let us take 1 example we will take this df3 data frame and here let us look at the df3.columns here Delhi, Mumbai. Now you can see in the set data structure you know it was suppose to have unique values and we can always find out whether a particular values present in that set or not. So, in this case also indexes all is not supposed to be modified, so here also certain behavior is similar to set data structure.

So, you can see we can check something like this whether Mumbai is present in the columns index or not. So, within single quotes Mumbai in keyword and df3.column, so we can check the presence or absence of a certain label in you know columns or row index. So, you can see similarly for the row index 2014 whether it is present in the df3 row index or not, so you can see

set like behavior is there and also we have discussed the immutability aspect as well.

Now 1 difference that index object have from the set data structures set object is there index they can have duplicate elements. So, if you look at the you know next line of code that we have calling pd.index function. And we have a sequence of values to create this syntax and you can see beta is coming twice beta then alpha then beta and sigma. So, this is not allowed in set because it is supposed to be you know sequence of unique you know elements.

But in the case of index the behavior is quite similar to set but duplicate elements are allowed. So, if I run this, I will create this ind3 index you can see this index object and the duplicate values are allowed. So, using this kind of index we can create a series object, so next example is about we are calling pd.series function. Then a list of values and then we are using this ind3 index here.

So I can run this and you can see that 2 rows are having the same index level beta and beta. So, try to access rows with the beta you know label, so if I call series7 and within brackets single quote beta. So, you can see all the rows which are having that beta label they are going to be selected, so they are going to be displayed. So, this is how the output is you know different how the behavior of index object is different from the set object.

(Video Starts: 29:49)

Now we will stop at this point and in the next lecture we will start our discussion on some of the interaction mechanism you know with a data in a series or data frame, so we will discuss those aspects, thank you.

Keywords: Pandas, Pseudo random function, data frame, data structure, classifiers, database, python.