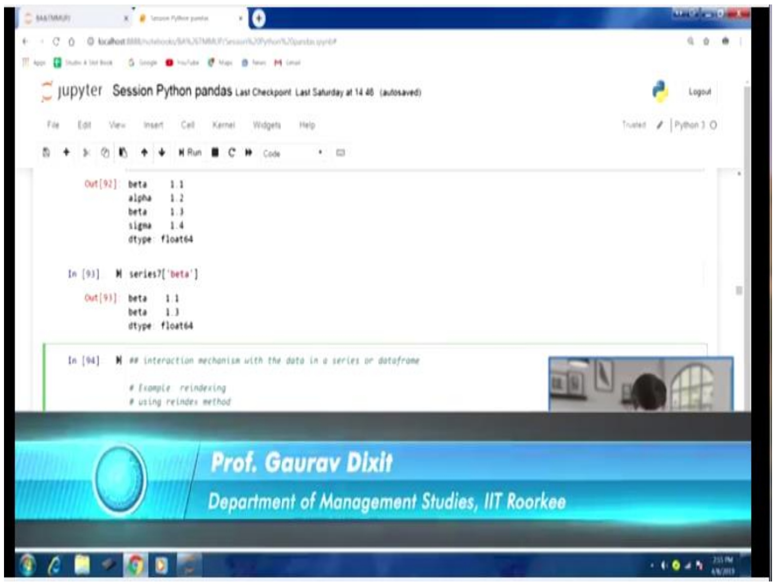
Business Analytics and Text Mining Modeling Using python

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

Lecture-28
Database Using python -Pandas-Part III

Welcome to the course business analytics and text mining modeling using python. So, in previous few lectures we have started about discussion on pandas another important package in the python platform. So, let us pick up from where we started in the previous lecture.

(Refer Slide Time: 00:43)


jupyter Session Python pandas Last Checkpoint Last Saturday at 14 46 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
Prof. Gaurav Dixit
Department of Management Studies, IIT Roorkee

```
Out[92]: beta     1.1
         alpha    1.2
         beta     1.3
         sigma    1.4
         dtype: float64

In [93]:  series7['beta']

Out[93]: beta    1.1
         beta    1.3
         dtype: float64

In [94]:  ## interaction mechanism with the data in a series or dataframe

          # Example  reindexing
          # using reindex method
```

So, we were about to start the interaction mechanism with the data and in a series or data frame, so let us pick up from that. So, let us take the example of reindexing here, so for this to perform the indexing, so we as we have understood that in both series and data frame we have a you know 1 and 2 indexes respectively. So, if we you know want to change that, so the few approaches we have discussed before, one is you know another approach is using reindex method.

So, let us talk about this, so let us take this example of series8 where we are you know initializing or defining this series you know with the 4 elements and you can see the index also.

(Video Starts: 01:28)

So, if you know let me run this, so if we want to reindex this series, you can see series8 and series8.you know reindex. So, in this fashion we can call this method and within the parentheses you can see that we can respecify the new index here. So, you can see earlier one we had just the 4 elements in the index when we passed it in the arguments and index.

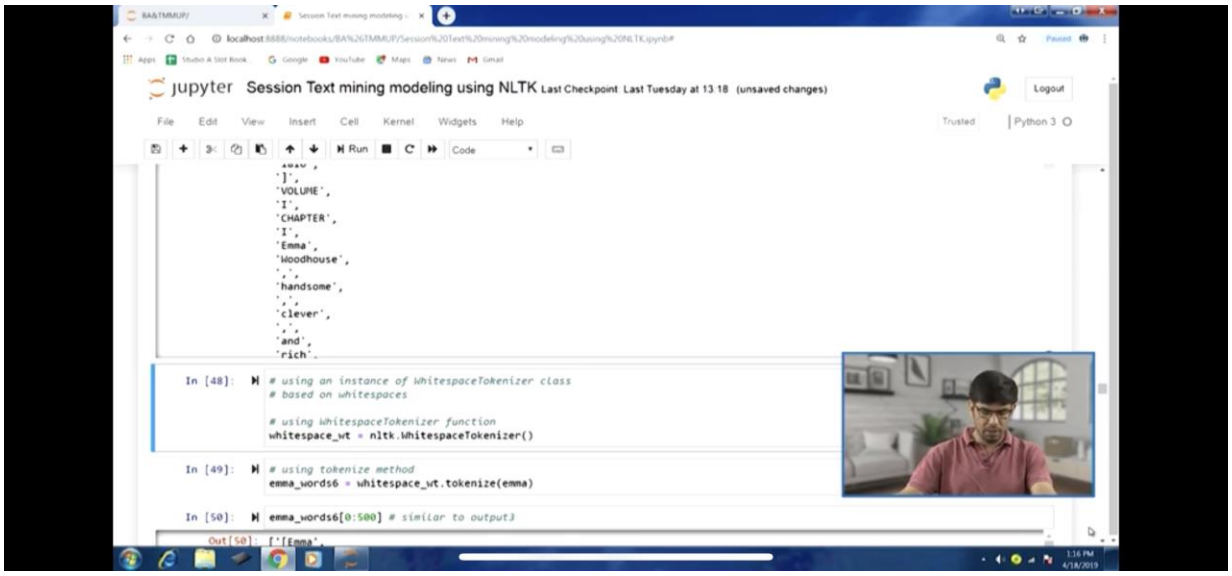
Now we have 5 elements in this index when we are doing when we are calling reindex method here a, b, c, d, e, so e is the additional and also you can see that the order has changed, So, now in reindexing we are using the alphabetical order, so let us run this. So, what will happen is that in the new series9 whatever indexes are matched, so they would be appropriately displayed in the as per the new index you can see a, b, c, d, e.

But where e there was no value element there in the series, so the for that nan is being displayed. So, in this fashion we can use reindex method as well to you know change our you know indexes. Now as you can understand that while we are reindexing as you saw in the previous output 272 that any unmatched in you know indices, they might have nan values.

So, we have certain methods which can help us in refilling those values, so let us talk about some of these methods. So, filling values when reindexing, so one of these method is ffill method. So, how do we use this, so let us take example of this series10, so in this we have these you know 3 elements blue, purple and yellow and the indexes as well. So, if you notice the index you know argument, we have 3 elements with 0, 2, 4.

So, you can see at few indices are missing, so we can always fill you know these indices using a film method. So, we will see about that let me first define this series10 you can see 0, 2, 4 and blue, purple, yellow. So, now we can fill this using a film method you can see series10.reindex named 6 and you can see in the method name ffill. And if I run this, now you can see the change has also been change, so range is 6 and because the remaining elements we would be filling up.

(Refer Slide Time: 04:57)



jupyter Session Text mining modeling using NLTK Last Checkpoint: Last Tuesday at 13:18 (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

```
 ']',
 'VOLUME',
 'I',
 'CHAPTER',
 'I',
 'Emma',
 'Woodhouse',
 ',',
 'handsome',
 ',',
 'clever',
 ',',
 'and',
 'rich',
```

```
In [48]: # using an instance of WhitespaceTokenizer class
         # based on whitespaces

         # using WhitespaceTokenizer function
         whitespace_wt = nltk.WhitespaceTokenizer()

In [49]: # using tokenize method
         emma_words6 = whitespace_wt.tokenize(emma)

In [50]: emma_words6[0:500] # similar to output3

Out[50]: ['[Emma',
```

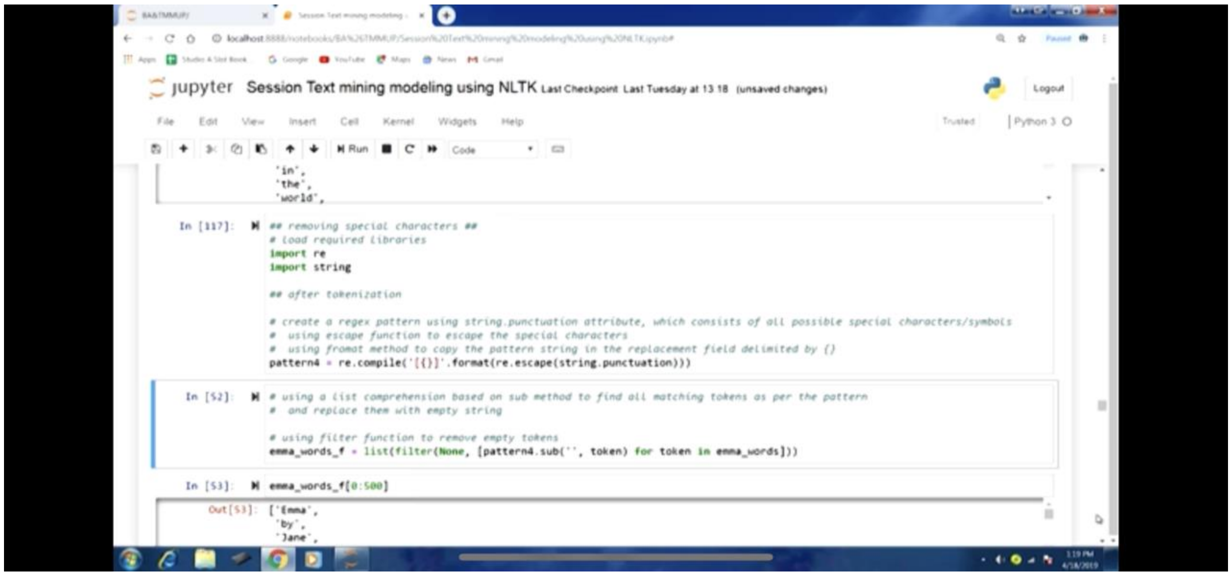
So, if I run this and you have a look at the output you can compare these 2 output 274 and 275 and you can see you know earlier the series10 had 3 element. Here we have in a 6 element, the you know index has been changed, now we have you know 6 you know index values starting from 0 to 5. And the new values they have been filled using a ffilled method, so you can see you know for 0 and 1 which is blue, 2 and 3 purple.

So, this is how the ffill method you know can be use to fill the values and you know reindex the you know series as well. Now let us move forward, now let us talk about reindexing with a data frame object. So, let us take example of this data frame df4, so here you can see that we have these you know we have this we are using a range function to generate the values and the dimensions are 3*3.

So, first dimension 3 element, second dimension 3 elements and index you can see acd and Delhi, Mumbai, Bangalore. So, if I run this you can have a look at the df4 data that we have and if I want to change the index here you can see and the index right now we have acd. So, I can add b also here, so let me use the reindex method here, so we are storing this one also df5=df4.reindex and I have specified in you know the change index also.

So, if I run this we will df5, now if you look at the **d** df5 here then you know row number row with the index value b all the values are nan. Because you know the corresponding you know value elements were not present. Now similarly we can also reindex the column index as well, so for this we will have to specify the columns argument.

So, in the method reindex will have to use this you know a keyword argument columns and then we will have to specify our specific ordering or index that we want. So, you can see now we have change the order we have made it you know alphabetical, so Bangalore comes first then Delhi and Mumbai. If you look at the output 278 then the ordering is slightly different. **(Refer Slide Time: 08:39)**


jupyter Session Text mining modeling using NLTK Last Checkpoint Last Tuesday at 13 18 (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

```
 'in',
 'the',
 'world',

In [117]: ## removing special characters ##
          # load required libraries
          import re
          import string

          ## after tokenization

          # create a regex pattern using string.punctuation attribute, which consists of all possible special characters/symbols
          #  using escape function to escape the special characters
          #  using fromat method to copy the pattern string in the replacement field delimited by {}
          pattern4 = re.compile('[{}]'.format(re.escape(string.punctuation)))

In [52]:  # using a list comprehension based on sub method to find all matching tokens as per the pattern
          #  and replace them with empty string

          # using filter function to remove empty tokens
          emma_words_f = list(filter(None, [pattern4.sub('', token) for token in emma_words]))

In [53]:  emma_words_f[0:500]

Out[53]: ['Emma',
          'by',
          'Jane',
```

So, if I run this you can see the output 279 and the ordering the column index has changed, now let us talk about another way of reindexing. So, this is using loc attribute, so you know we can have this you know we can use this loc attribute df4.loc and in this we can specify these 2 indices here. In this fashion you can see it is like row index first and then followed by the column index.

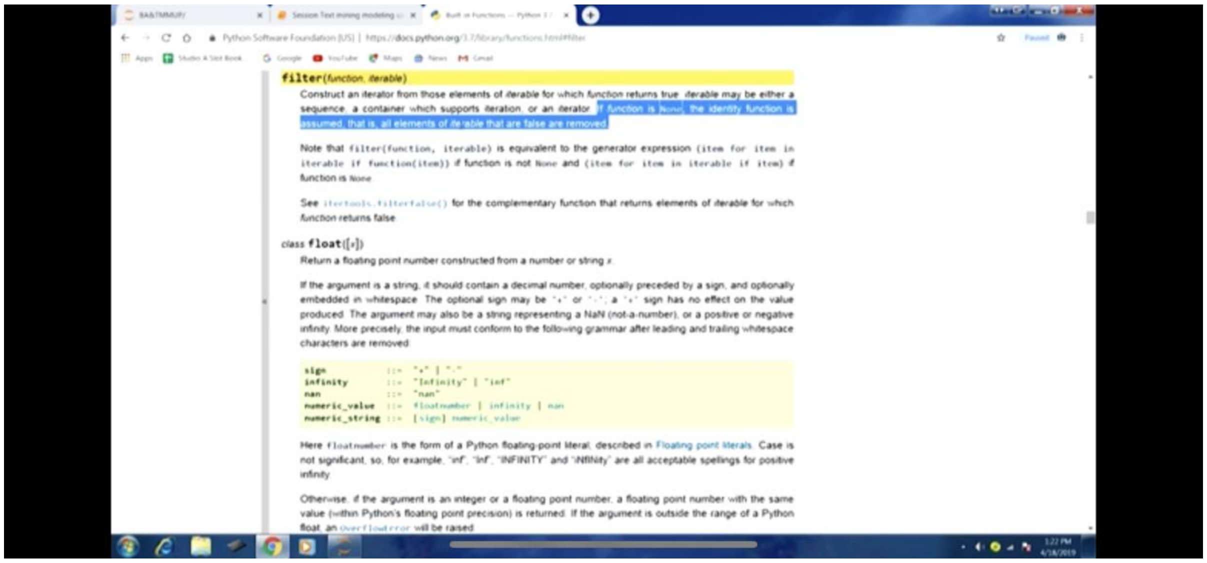
So, row index we have specified a, b, c, d and column index Bangalore, Delhi, Mumbai, so whatever we did using the reindex method we can do the same thing using loc attribute but slightly different syntax here. So, if I run this you know you can have a look at the output here, so the output is same as the previous reindexing that we did using reindex method however we are also getting a warning here that future warning.

So, in future in this mechanism of .loc is not going to be you know available. So, you can see that you can read the warning, so right now it is working but in future this support might be taking out. Now let us talk about another aspect here that is about dropping cases or row sometimes you would like to you know delete certain columns, certain variables that we might not find relevant or certain rows you know for you know based on our analytics project we might like to get rid of certain rows it could be due to missing values as well.

So, let us take the example of df5, so in this you can see the you know row with the row index b is having all the nan values. So, we can drop this because of the missing values, so for this we can use the drop method here. So, we can call df5.drop and we can pass on the row index here which is you know within single codes we can specify b and you would see in the output that this particular row is gone.

So, we can use this method in this fashion to drop certain rows or columns, so similarly we want to drop 2 rows multiple rows. Then again within a we can use a list and we can pass on this row indices there and the drop method will do the job here. So, let me run this and you can see we are left with just 2 rows, now let us talk about the another scenario dropping columns or variables.

(Refer Slide Time: 11:24)


https://docs.python.org/3.7/library/functions.html#filter

filter(*function, iterable*)

Construct an iterator from those elements of *iterable* for which *function* returns true. *iterable* may be either a sequence, a container which supports iteration, or an iterator. If *function* is `None`, the identity function is assumed, that is, all elements of *iterable* that are false are removed.

Note that `filter(function, iterable)` is equivalent to the generator expression `(item for item in iterable if function(item))` if function is not `None` and `(item for item in iterable if item)` if function is `None`.

See `itertools.filterfalse()` for the complementary function that returns elements of *iterable* for which *function* returns false.

class **float**([*x*])

Return a floating point number constructed from a number or string *x*.

If the argument is a string, it should contain a decimal number, optionally preceded by a sign, and optionally embedded in whitespace. The optional sign may be `'+'` or `'-'`; a `'+'` sign has no effect on the value produced. The argument may also be a string representing a NaN (not-a-number), or a positive or negative infinity. More precisely, the input must conform to the following grammar after leading and trailing whitespace characters are removed:

```
sign           ::=  "+" | "-"
infinity       ::=  "Infinity" | "inf"
nan            ::=  "nan"
numeric_value  ::=  floatnumber | infinity | nan
numeric_string ::=  [sign] numeric_value
```

Here `floatnumber` is the form of a Python floating-point literal, described in Floating point literals. Case is not significant, so, for example, "inf", "Inf", "INFINITY" and "iNfINity" are all acceptable spellings for positive infinity.

Otherwise, if the argument is an integer or a floating point number, a floating point number with the same value (within Python's floating point precision) is returned. If the argument is outside the range of a Python float, an `OverflowError` will be raised.

So, let us df5 again, so this is df5 and if you want to you know drop let us say Bangalore here. So, apart from indicating the apart from the first argument you know passing the value of the indices that we want drop that is Bangalore in this case. We also have to specify the axes, so axes=1 to indicate that we want to you know drop a column which is having this index column index as Bangalore.

So, if I run this you can see in the output Delhi and Mumbai these 2 columns are present but Bangalore is gone. Let us move forward, so now in this case we are dropping multiple columns here, so we are dropping Delhi and Mumbai and will be left with Bangalore as you can see in the output. Now apart from the drop method we can also do a inplace version of you know drop method.

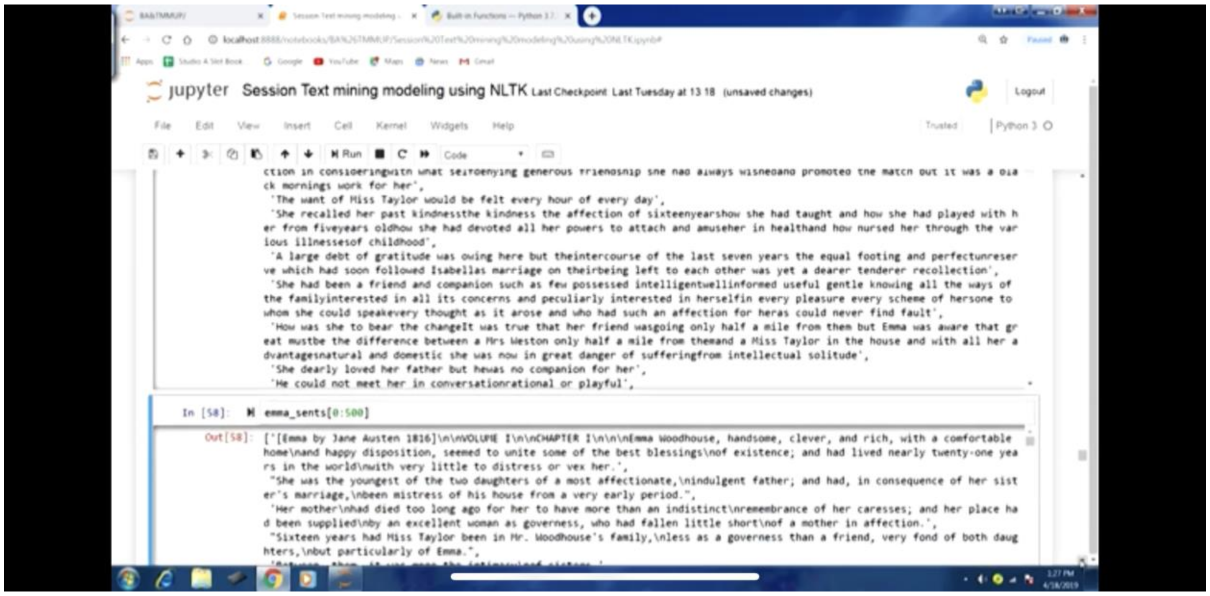
So, for this you know we have an you know inplace argument there, so let us again take the example of df5 and in the next thing that we are going to do is we are going to drop this row with b index. And you can see we are using inplace argument as true, so in the you know df5 data frame itself this particular you know column is going to be drop, so d5 data frame is going to be modified, this is inplace version using the inplace argument here.

So, if I run this you can see df5 and 1 particular row is gone, now let us focus on some other aspects of python which is about accessing, selection and filtering. So, let us start with a series, so let us take the example of the series11 and which we had earlier created. So, here we have the 6 elements you know blue blue, purple, purple and yellow yellow and let us take another series, series7.

So, here we had slightly different indices beta, alpha, beta sigma and the values 1.1, 1.3 and so on. So, if you want to access 1 row as you would accept that series11 and within brackets we can specify the index here. So, I can say series11 and again specify 3, so in this case it is an integer index, so it I can just specify that and you can see you have got the purple as output.

Now in case of string index here series7 we will have to specify within single codes. So, series7 within brackets single codes and sigma and we would be able to access the value which is in this case 1.3. Now we want to access a range of rows. So, for that we can specify using colon operator, so in this case you know example series11 within brackets we are saying 3:5. So, starting from 3 to you know stop point 5.

(Refer Slide Time: 14:48)


jupyter Session Text mining modeling using NLTK Last Checkpoint Last Tuesday at 13 18 (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

```
ction in consideringwith what selfdenying generous friendship she had always wishedand promoted the match but it was a bla
ck mornings work for her',
 'The want of Miss Taylor would be felt every hour of every day',
 'She recalled her past kindnessthe kindness the affection of sixteenyearshow she had taught and how she had played with h
er from fiveyears oldhow she had devoted all her powers to attach and amuseher in healthand how nursed her through the var
ious illnessesof childhood',
 'A large debt of gratitude was owing here but theintercourse of the last seven years the equal footing and perfectunreser
ve which had soon followed Isabellas marriage on theirbeing left to each other was yet a dearer tenderer recollection',
 'She had been a friend and companion such as few possessed intelligentwellinformed useful gentle knowing all the ways of
the familyinterested in all its concerns and peculiarly interested in herselfin every pleasure every scheme of hersone to
whom she could speakevery thought as it arose and who had such an affection for heras could never find fault',
 'How was she to bear the changeIt was true that her friend wasgoing only half a mile from them but Emma was aware that gr
eat mustbe the difference between a Mrs Weston only half a mile from themand a Miss Taylor in the house and with all her a
dvantagesnatural and domestic she was now in great danger of sufferingfrom intellectual solitude',
 'She dearly loved her father but hewas no companion for her',
 'He could not meet her in conversationrational or playful',

In [58]:  emma_sents[0:500]

Out[58]: ['[Emma by Jane Austen 1816]\n\nVOLUME I\n\nCHAPTER I\n\n\nEmma Woodhouse, handsome, clever, and rich, with a comfortable
         home\nand happy disposition, seemed to unite some of the best blessings\nof existence; and had lived nearly twenty-one yea
         rs in the world\nwith very little to distress or vex her.',
          "She was the youngest of the two daughters of a most affectionate,\nindulgent father; and had, in consequence of her sist
         er's marriage,\nbeen mistress of his house from a very early period.",
          'Her mother\nhad died too long ago for her to have more than an indistinct\nremembrance of her caresses; and her place ha
         d been supplied\nby an excellent woman as governess, who had fallen little short\nof a mother in affection.',
          "Sixteen years had Miss Taylor been in Mr. Woodhouse's family,\nless as a governess than a friend, very fond of both daug
         hters,\nbut particularly of Emma.",
```

So 2 rows are going to be displayed in the output here. So, if I run this you can see row index 3 and row index you know 4 those rows have been displayed in the output. Now we want to select few rows, so that can also be done, so for this within the brackets operator we can pass on a list of the selected rows that we would like to display.

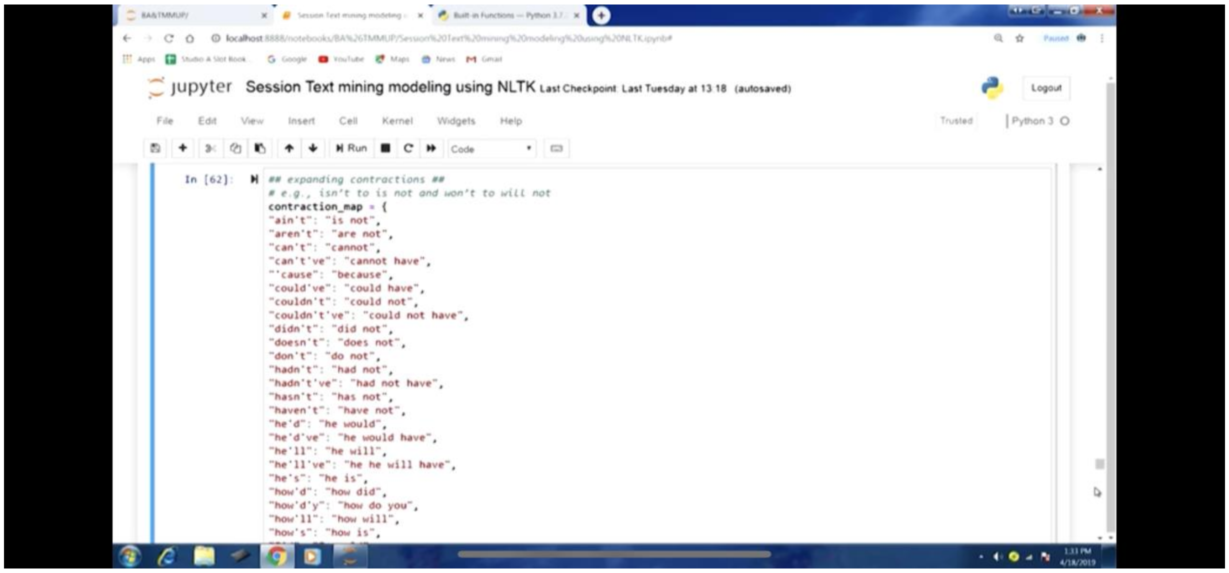
So, for example series7 and within double brackets we are passing a list again using brackets sigma and alpha. So, we are selecting these 2 rows, so if I run this you can see in the output only 2 rows have been displayed. Similarly at the series11 also we can do the same thing where we are using the integer index row index. So, we are selecting you know row index 1 and 5, so if I run those you know rows are going to be displayed here.

Now filtering, so we can have certain logic and based on that logic we can filter certain number of rows or columns. So, let us take an example in series7 in we can specify within brackets you know and expression which is going to returned you know kind of Boolean values. So, that we will be selecting those particular rows only in a sense we would be performing filtering.

So, if we are looking for you know rows where the series value is less than 1.3, so we can you know put that expression within double brackets and only those rows are going to be displayed. So, if I run this code here you can see the adjust 1 value beta which was less than 1.3 you know has been display. Let us move forward next aspect is slicing, so you know slicing is very much similar to you know like we have discuss in other context.

However in this case end point is inclusive, so earlier you know whenever we have use slicing the end point was you know non inclusive. So, when we do slicing on series on data frame, so that is inclusive, so let us say series7 and within bracket we can pass on like this alpha and in colon and sigma. So, those rows are going to be sliced and split, so if I run this, so you would get that output number 297 you know starting from alpha, beta and sigma those 3 rows which were part of the slicing they have been displayed.

(Refer Slide Time: 16:15)



```
In [62]:  ## expanding contractions ##
          # e.g., isn't to is not and won't to will not
          contraction_map = {
          "ain't": "is not",
          "aren't": "are not",
          "can't": "cannot",
          "can't've": "cannot have",
          "'cause": "because",
          "could've": "could have",
          "couldn't": "could not",
          "couldn't've": "could not have",
          "didn't": "did not",
          "doesn't": "does not",
          "don't": "do not",
          "hadn't": "had not",
          "hadn't've": "had not have",
          "hasn't": "has not",
          "haven't": "have not",
          "he'd": "he would",
          "he'd've": "he would have",
          "he'll": "he will",
          "he'll've": "he he will have",
          "he's": "he is",
          "how'd": "how did",
          "how'd'y": "how do you",
          "how'll": "how will",
          "how's": "how is",
```

However since you know in the series7 we are we do not have unique index values we have certain rather can be defined as non unique you know index. So, in this case the selection might not works if we you know have not to select the particular slicing where this non unique indices are involved. So, let us say series7 beta to sigma, so if you remember we have 2 betas in this if you look at the previous output let me go up.

And let me go up further yes here output number 290 you can see beta then alpha then beta then sigma. Now here you have 2 betas, so it would be difficult for the python interpretative infer which beta we are talking about. So, in that case because of non unique you know indexing the slicing might not work because inferences difficult for the python interpreter.

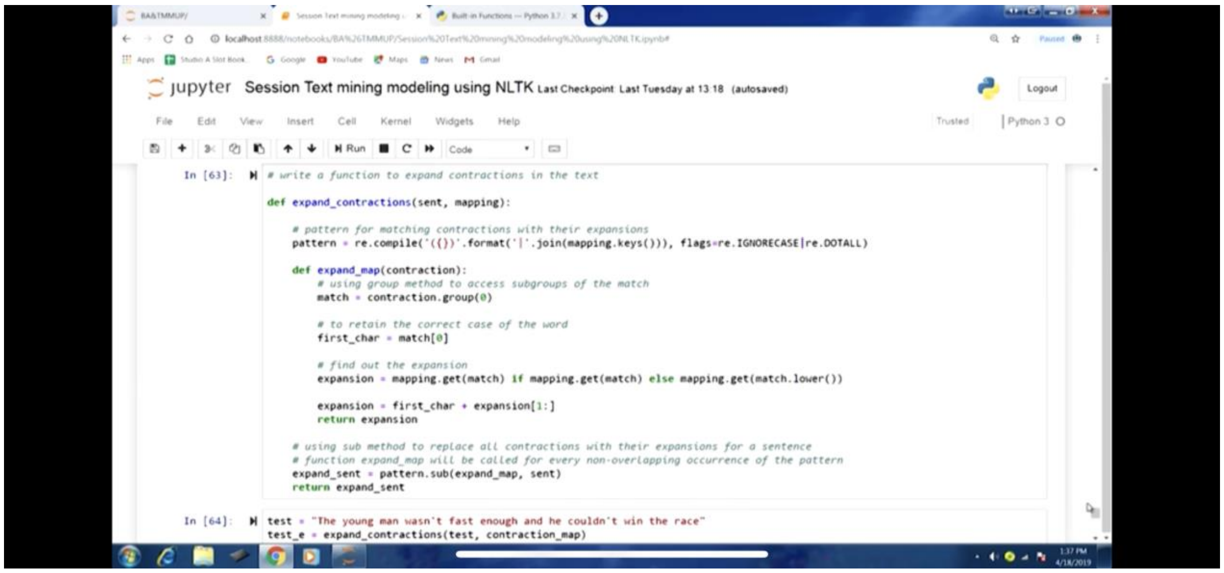
So, if we specify series7 beta:sigma then it would not be able to infer which beta we are talking about. So, if I run this will run into this error here and you can see you know cannot get in the key error is cannot get left slice bound for non unique label beta. Because beta is non unique 2 instances of this the same row index operation, so which is not able to resolve. Now let us take another example let us take another slicing series7 alpha: sigma, so here you can see this time it worked.

So, in this case we have done an assignment, so all the rows which were part of the slicing and they will be assigned this value 1.3. So, if I check it again you can see series7 and those rows starting from alpha to sigma all inclusive. So, those have been assign this value 1.3, now let us take example of data frame and we will talk about some of these things again.

So, df5 let us take this particular data frame, now how do we access a particular column here, so within the double brackets we can specify the column index, so df5 within brackets Delhi in single codes. So, in this fashion we can access a particular column, we can also access multiple column we can pass on a list of column indices. So, next example df5 within brackets they are passing list Delhi and Bangalore and will get a list of indices here.

List of you know column indices and therefore those columns are selected in this output as you can see in the output number 303. Now slicing rows in a data frame, so again within the brackets we can specify we can use the colon operator also for example :2. So, we did not specify these you know starting point, so you know those particular you know indices are going to be selected.

(Refer Slide Time: 21:45)


jupyter Session Text mining modeling using NLTK Last Checkpoint Last Tuesday at 13:18 (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

```
In [63]:  # write a function to expand contractions in the text

          def expand_contractions(sent, mapping):

              # pattern for matching contractions with their expansions
              pattern = re.compile('({})'.format('|'.join(mapping.keys())), flags=re.IGNORECASE|re.DOTALL)

              def expand_map(contraction):
                  # using group method to access subgroups of the match
                  match = contraction.group(0)

                  # to retain the correct case of the word
                  first_char = match[0]

                  # find out the expansion
                  expansion = mapping.get(match) if mapping.get(match) else mapping.get(match.lower())

                  expansion = first_char + expansion[1:]
                  return expansion

              # using sub method to replace all contractions with their expansions for a sentence
              # function expand_map will be called for every non-overlapping occurrence of the pattern
              expand_sent = pattern.sub(expand_map, sent)
              return expand_sent

In [64]:  test = "The young man wasn't fast enough and he couldn't win the race"
          test_e = expand_contractions(test, contraction_map)
```

So, if I run this you can see the output those 2 rows have been selected here, now let us move forward . Another example df5 1 to 2, so in this case if I run , so you can see that you know the slicing in this case the row indexes are actually the string you know string values ac acd I think. And however we are using the integer values here, so these are the corresponding numbers and accordingly we are getting the output here.

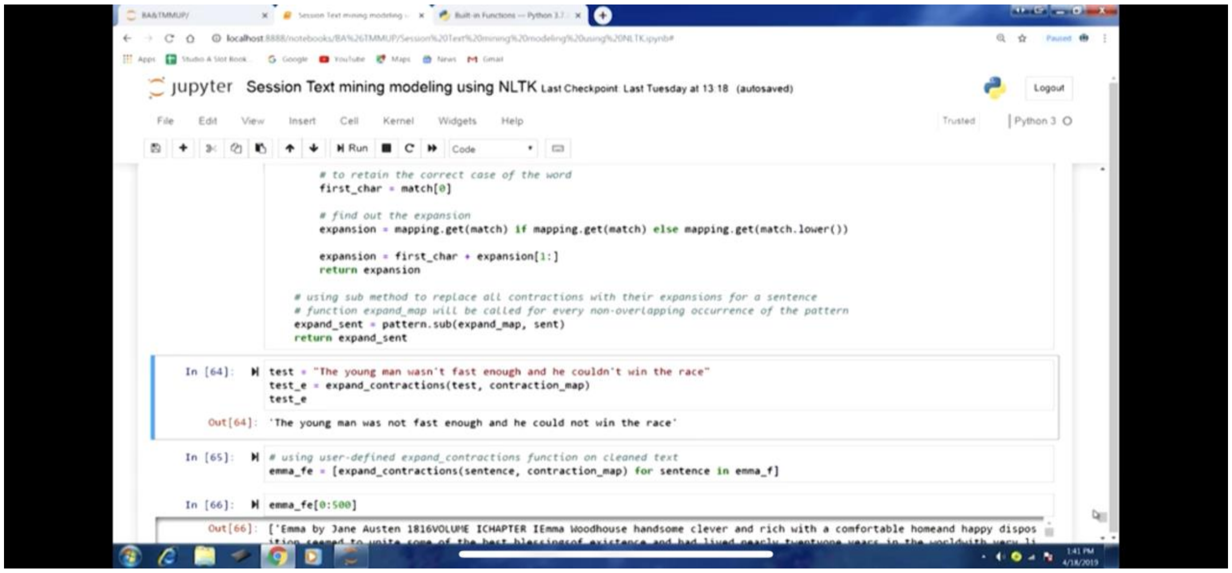
Similarly just like we talked about series and filtering certain rows, here also in data frame also we can perform the same thing. So, let us say in the Bangalore column we would like to just you know filter rows which are having values greater than 5. So, we can specify like this df5 within brackets df5 Bangalore greater than 5. So, in this fashion only those rows which are having this particular column value greater than 5 they are going to be in the output as you can see just 1 row.

Working with a boolean data frame, so if I use a expression like this df5<5, so this will give me a boolean data frame where all the you know values they will indicate whether the value was less than 5 or not using boolean you know values true or false. So, if I run this you can see in the output a boolean data frame is there. Similarly using this boolean data frame we can use this as an you know as indices.

So, we can pass on this in the df5 within brackets and you know we will get the output here. So, wherever true was present those values are going to be you know and so on and wherever false is there, so we will get nan. Now we can modify these values also based on boolean expression. So, wherever false was there we can assign a value 0, in a sense we are refilling the data frame using you know boolean expression.

So, if I specify df5 within brackets df5 less than 5, so wherever the you know value was you know not less than 5. So, all those you know elements are going to be assign this value of 0. So, if I run this let us have a look at the output and you can see, so in this case rather wherever the value was less than 5 wherever the value was true those values are going to be assign 0, so rather it would be that kind of output.

(Refer Slide Time: 25:56)



jupyter Session Text mining modeling using NLTK Last Checkpoint: Last Tuesday at 13:18 (autosaved)

```
        # to retain the correct case of the word
        first_char = match[0]

        # find out the expansion
        expansion = mapping.get(match) if mapping.get(match) else mapping.get(match.lower())

        expansion = first_char + expansion[1:]
        return expansion

    # using sub method to replace all contractions with their expansions for a sentence
    # function expand_map will be called for every non-overlapping occurrence of the pattern
    expand_sent = pattern.sub(expand_map, sent)
    return expand_sent

In [64]:  test = "The young man wasn't fast enough and he couldn't win the race"
          test_e = expand_contractions(test, contraction_map)
          test_e

Out[64]: 'The young man was not fast enough and he could not win the race'

In [65]:  # using user-defined expand_contractions function on cleaned text
          emma_fe = [expand_contractions(sentence, contraction_map) for sentence in emma_f]

In [66]:  emma_fe[0:500]

Out[66]: ['Emma by Jane Austen 1816VOLUME ICHAPTER IEmma Woodhouse handsome clever and rich with a comfortable homeand happy dispos
```

Now let us talk about 2 important indexing operator loc and iloc till now we have you know talked about various approaches for indexing reindexing. However they might not lead to you know precise indexing because of the way you know indexing happens there. So, to achieve more precise indexing you know we have these 2 operators loc and iloc, so loc is use with access labels and iloc use with the integer values corresponding to access labels.

So, let us take the example of a this data frame df5 once again and you can see the in syntax here to use the operator loc we can pass it like df5.loc just like an attribute. And within brackets we can specify the row and columns, so you can see clearly row a and column Delhi we can specify. So, we use loc at you know operator previously also in an example, so here in this case you can see how we can access a particular element of a data frame. So, we can specify row and column index and particular element can be access in this fashion similarly we can you know we can select multiple columns as well.

So, let us say in the next example we have picking you know row a but 2 columns Delhi and Bangalore. So, we have passing that as a list in the column index part, so you can see those 2 are there in the output. Similarly this is another way if we want to select multiple rows and multiple columns you can see the code here, 1 example here a and b these 2 rows we would like to pick and Delhi and Bangalore these 2 columns.

So, if I run this and you can see the output however you can see like before this time also we have got a future warning because probably this kind of access is not going to be allowed in future. Let us move forward let us take another example where we are using loc and again we are specifying the you know indices in index values for row and column both, so if I run for this example you can see output again.

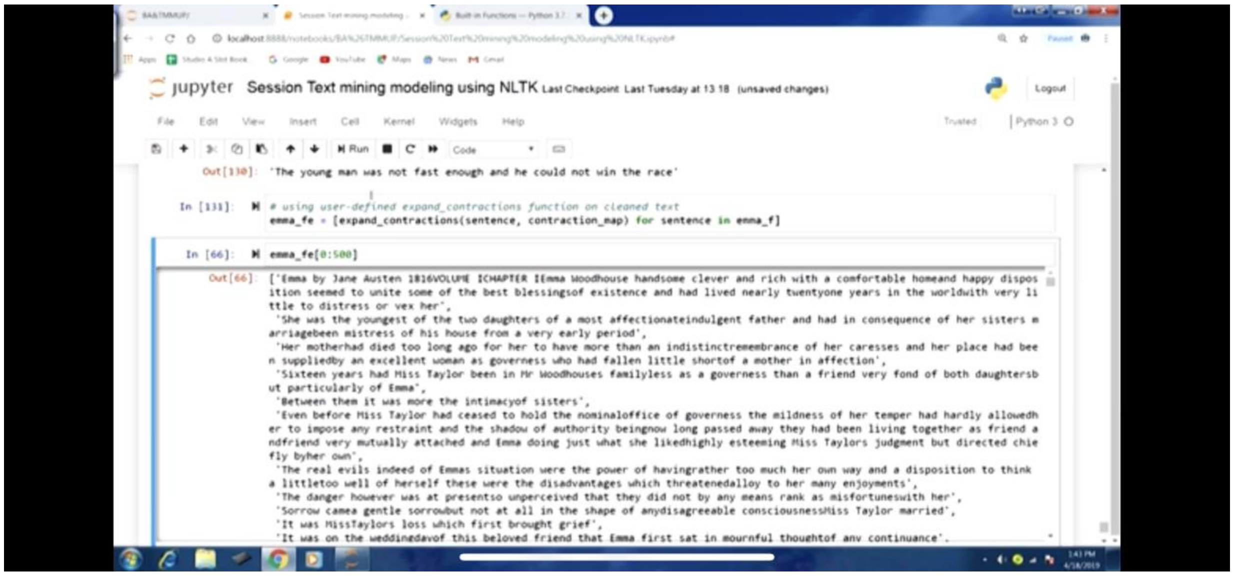
Now let us talk about the second operator that is iloc this is the operator where we are using integer values corresponding integer values . So, here we can specify a syntax like this we can use that data frame df5 and .iloc to specify that we are going to use integer numbers here. And then within brackets 0, 0, 0 first 0 is for the row and second one is for the column.

So, in this case we would be effectively like accessing 1 cell just like an excel, so you can see 0, 0. If you want to select 2 columns, so you can see the syntax here example code example df5.iloc within brackets in the row index part 0 and for the column index part we were passing a list of 2 value 0, 1. So, therefore we have selecting 2 columns here, so if I run this you would end up with you know this kind of result here.

Let us take a slightly more you know advance version of this iloc, so in this case what we are doing df5.iloc we are picking 2 rows and 2 column. So, in the row index part we are passing a list of row indices 0, 1 and in the column index part we are passing a list of column indices 0, 1. So, you can see if I run this the output those 2 rows and columns have been you know displayed in the output.

Now another example in this case df5.iloc, so I am just picking you know 1, so in this case if I pass on like this we would be just you know displaying 1 row if we do not specify the column part. Then we would be eventually accessing the rows, so corresponding to the row index 1 the particular row has been displayed in the output. Let us take another example of iloc here slightly you know more number of rows have been selected.

(Refer Slide Time: 29:55)


jupyter Session Text mining modeling using NLTK Last Checkpoint: Last Tuesday at 13:18 (unsaved changes)

```
Out[130]: 'The young man was not fast enough and he could not win the race'

In [131]: # using user-defined expand_contractions function on cleaned text
          emma_fe = [expand_contractions(sentence, contraction_map) for sentence in emma_f]

In [66]:  emma_fe[0:500]

Out[66]:  ['Emma by Jane Austen 1816VOLUME ICHAPTER IEmma Woodhouse handsome clever and rich with a comfortable homeand happy dispos
          ition seemed to unite some of the best blessingsof existence and had lived nearly twentyone years in the worldwith very li
          ttle to distress or vex her',
           'She was the youngest of the two daughters of a most affectionateindulgent father and had in consequence of her sisters m
          arriagebeen mistress of his house from a very early period',
           'Her motherhad died too long ago for her to have more than an indistinctremembrance of her caresses and her place had bee
          n suppliedby an excellent woman as governess who had fallen little shortof a mother in affection',
           'Sixteen years had Miss Taylor been in Mr Woodhouses familyless as a governess than a friend very fond of both daughtersb
          ut particularly of Emma',
           'Between them it was more the intimacyof sisters',
           'Even before Miss Taylor had ceased to hold the nominaloffice of governess the mildness of her temper had hardly allowedh
          er to impose any restraint and the shadow of authority beingnow long passed away they had been living together as friend a
          ndfriend very mutually attached and Emma doing just what she likedhighly esteeming Miss Taylors judgment but directed chie
          fly byher own',
           'The real evils indeed of Emmas situation were the power of havingrather too much her own way and a disposition to think
          a littletoo well of herself these were the disadvantages which threatenedalloy to her many enjoyments',
           'The danger however was at presentso unperceived that they did not by any means rank as misfortuneswith her',
           'Sorrow camea gentle sorrowbut not at all in the shape of anydisagreeable consciousnessMiss Taylor married',
           'It was MissTaylors loss which first brought grief',
           'It was on the weddingdayof this beloved friend that Emma first sat in mournful thoughtof any continuance',
```

So, if I run this you can see in the output 2 rows e and d and 3 columns and you can see ordering has also you know in a sense we can change the ordering also. So, the way we are accessing and the output also will indicate that ordering is also changed. Now we can also perform slicing with you know using loc and iloc operator here. So, df5.loc and I can specify like this column you know c , so all those columns up to this index.

And then you know selecting all those rows up to this row index c and then column I am selecting daily. So, in this fashion slicing can be perform, so if I run this you can see a and c, c inclusive these 2 rows have been selected and for the column Delhi. Let us take another example for slicing with the loc, so df5.loc and within brackets we can again we are doing row slicing c:d.

So, all those rows are going to be sliced for the column Delhi, so if I run this we will get those rows c and d for the particular column Delhi. And let us take another example, so in this case we are performing both row and column slicing. So, in the row side we have the same thing c:d and on the column side we have you know Delhi:Bangalore.

So, those 2 you know those particular you know columns are going to be selected, so Delhi, :Bangalore selects all 3 columns that we have and the 2 rows c:d, so if I run this you can see the output here. We can also perform bare slicing, so as you would remember from our discussion you know from the previous lectures which is indicate colon here that means bare slicing all the rows or all the columns are going to be selected.

In this particular example it is happening for the rows side and then on the column side we are selecting 1:2. So, you know we were just 1 column Mumbai is going to be selected in this case. Now let us move further df5 another example of bare slicing you can see apart from the slicing that we are performing we can also you know filter it out.

So, in this case df5.iloc bare slicing and 0:2, so 0 to 2 those columns are going to be selected. And all the rows are going to be selected and on top of that we are performing this you know this filtering df5.Mumbai not equal to 0. So, only the rows where the value is not 0 are going to be you know should be only those you know should be displayed.

So, if I run this you can see in the output only the row d has been displayed in the output. Now let us talk about few more points on integer indexing here, so let us take the example of series11 and next is in this case you can see that here in this case you see series11. If you many values are actually repeating here, so in this case if I type something like this series11-3 .

So, here label base indexing or position based rich indexing is to be use that inference ambiguity is there. So, because of this is going to fail here, so because it cannot determine you know how this -3 is to be performed. Because if you look at the row index that we have in series11 starts it to 0 index, so we have a row index value as 3 also.

And now if we start from the end side that means 5 side then the value will come out to be you know different. So, therefore that ambiguity is created. So, if I run this series11-3 then will run into an error. So, any kind of code example where certain ambiguity is created for inference ambiguity is created for the interpreter, so it will run into an error in that case.

Now let us take an example of series7, so in this case you see that in the row index we are you know non unique indices. Now in this case if I pass on series7-1, so here you would see that if I run this it is succeeding here in this case 1.3, because you know of this you know inference ambiguity issues. Now next thing that we would like to talk about is the arithmetic between objects with unmatch indexes.

So, if we apply you know arithmetic operations on series or data frames and those objects how because of the certain indexes that might not match how these arithmetic operation will work out or other operation will work out. So, we will talk about those aspects in the next lecture let us stop here, thank you.

(Video Ends: 30:31)

Keywords: Arithmetic operations, Indexing, Slicing, Arguments.