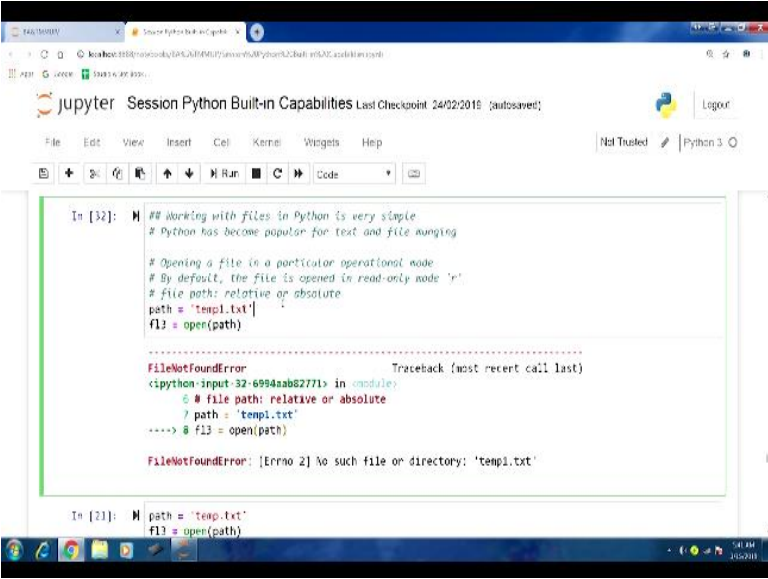


Business Analytics And Text Mining Modeling Using Python
Prof. Gaurav Dixit
Department of Management Studies
Indian Institute of Technology-Roorkee

Lecture-18
Built-in Capabilities of Python-X

Welcome to the course business analytics and text mining modeling using python. So in the previous lecture we started our discussion on file management in the python platform. So we will just quickly go through some other things that we discussed in the previous lecture and then we will pick up from the point where we stopped. So as we said that working with the files in python program is quite simple and python is quite actually popular for you know text processing and files managing

(Refer Slide Time: 00:57)



```
In [32]: ## Working with files in Python is very simple
# Python has become popular for text and file managing

# Opening a file in a particular operational mode
# By default, the file is opened in read-only mode "r"
# file path: relative or absolute
path = 'temp1.txt'
f13 = open(path)

-----
FileNotFoundError: Traceback (most recent call last)
<ipython-input-32-6994aab82771> in <module>
      6 # file path: relative or absolute
      7 path = 'temp1.txt'
----> 8 f13 = open(path)

FileNotFoundError: [Errno 2] No such file or directory: 'temp1.txt'
```

```
In [21]: path = 'temp.txt'
f13 = open(path)
```

And we talked about opening a file which is the first very first thing that we typically do in file management. So there are different operational modes, so if you just use the function open and we just pass on the file name as the argument for example temp1.txt, then by default the file is going to be open in the read-only mode that is R within single quotes. So what I will do this part we have discussed in the previous lecture.

(Video Starts: 01:22)

So I will just run through this, so if the file does not already exist then we get there in this case it

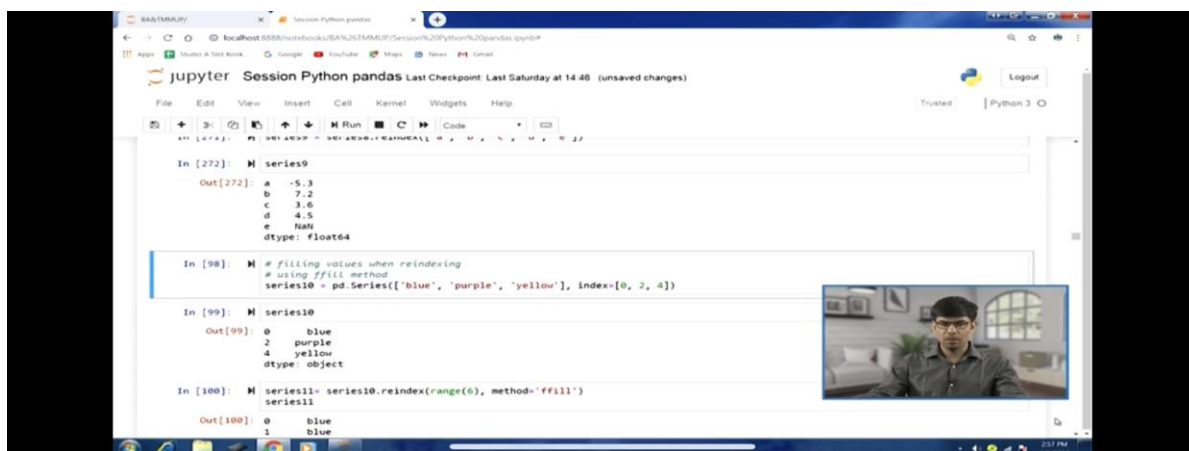
is there. Let us take temp.txt then you know if you want to read the contents of this file. So you can see in this case probably in the previous lead we have already that all the contents that was there in the file. So we got the empty bytes object here and you can see end of the line is already released.

Now closing the file objects `fl3.close` closes a function as we discussed in the previous lecture. If you know iterating over the lines in a file, then again you know if you want to you know iterate over the line and print the line and we will end of line markers by default slash n, so each of those lines are going to be printed in a different line. So that is there then there is another method closed method which will tell us about whether the file is opened or closed.

So I will just run this run through this, then if we want end of line list of lines. So in this case because we have already read these, so the output is slightly different then cleaning up of this file, so we can use the bid block, so where automatically cleaning happens and so if the file is closed then you know will typically get this error io operation on closed file. Then we move on to the next thing.

So I think we were discussing this part in the previous lecture we stopped at this point. So python file modes for writing, so there are 3 different modes that are typically used for writing. So W that is for write only mode, so if there is already a file with the same name that is going to be erased in the sense the data would be lost. Then there is another file mode x and that is also write only mode.

(Refer Slide Time: 03:05)



The screenshot shows a Jupyter Notebook interface with the following code and output:

```
In [272]: M series9
Out[272]: a -5.3
          b  7.2
          c  3.6
          d  4.5
          e  NaN
          dtype: float64

In [98]: M # filling values when reindexing
          # using ffill method
          series10 = pd.Series(['blue', 'purple', 'yellow'], index=[0, 2, 4])
In [99]: M series10
Out[99]: 0    blue
         2    purple
         4    yellow
         dtype: object

In [100]: M series11= series10.reindex(range(6), method='ffill')
          series11
Out[100]: 0    blue
          1    blue
```

But if the file with the same name is already existing then this if we open the file using this mode that will fail and then we have the another file mode a this is also for writing. So if file does not exist already that file is going to be created. But if the file is already existing you know any data that we want to write that is going to be appended at the end of the line at the end of the file.

So these are the 3 popular modes. So let us open this file temp.txt in the a mode. So we would be you know appending here and you can see the 9 output that we are seeing here is actually the number of you know characters that are part of this string that you have written. So that is returned. So let us close this file and now we can use another method lead lines that can be used to you know store all the lines that are there in this file.

So if I run this and again lines, so now you can see the appended you know this has been written in the file, so that is the spread, now let us open this file and the in the next file that is w that is right only. So because this file is already there and we have the data appended there. So if I run this again we will lose this data and if I try to read the content here not readable because the file is opened in the right only mode.

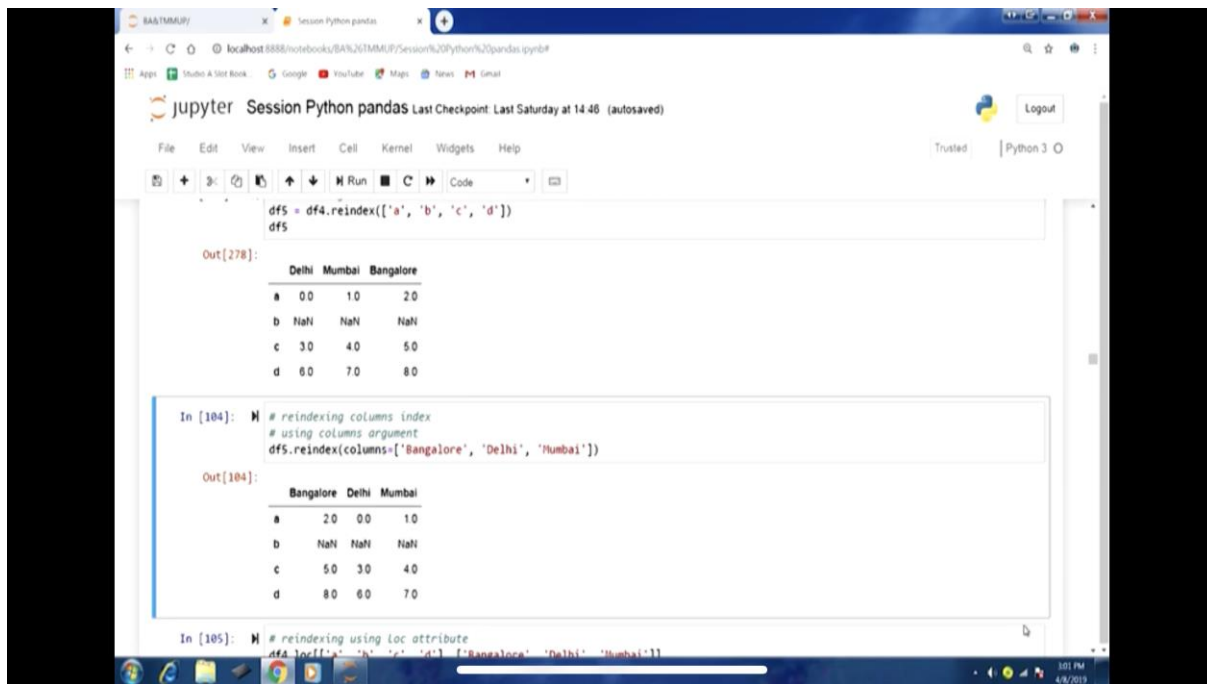
So I will have to close this file object and then again open it for the reading. So you can see and you can see the empty bytes object is written in the output number 65 because we use the you know write mode there. So therefore all the data that was part of the file that was erased out. Now the next file mode is X. So let us take an example for this one also. So we are opening temp.txt in this file mode x.

So if I run this, so this will fail you can see the error number 17 file exists, if the file is already existing this opening of this particular file object will fail, this particular file will fail in the text you know file mode. So now let us move to the you know next example, so here we are opening another file temp1.txt. So if I run this so in this case also file is already existing. So again this will fail.

Now if I open in the append a file mode then this file will open and I can add this line. So now I can add this line and this would be appended at the end of the file. So I am going to write this is

string om namah shivay and if I run this, so you can see in the output we have 16 that is the number of characters you know that have been written there. So what we will do we will run it 1 or 2 more times, so we are just writing few lines in that file.

(Refer Slide Time: 06:26)



The screenshot shows a Jupyter Notebook interface with two code cells and their corresponding outputs. The first cell reindexes a DataFrame by columns, and the second cell reindexes it by a specific column index.

```
df5 = df4.reindex(['a', 'b', 'c', 'd'])
df5
```

Out[278]:

	Delhi	Mumbai	Bangalore
a	0.0	1.0	2.0
b	NaN	NaN	NaN
c	3.0	4.0	5.0
d	6.0	7.0	8.0

```
In [104]: # reindexing columns index
# using columns argument
df5.reindex(columns=['Bangalore', 'Delhi', 'Mumbai'])
```

Out[104]:

	Bangalore	Delhi	Mumbai
a	2.0	0.0	1.0
b	NaN	NaN	NaN
c	5.0	3.0	4.0
d	8.0	6.0	7.0

```
In [105]: # reindexing using loc attribute
df4.loc[['a', 'b', 'c', 'd'], ['Bangalore', 'Delhi', 'Mumbai']]
```

So now let us close this file and if we open this for you know you can open this for reading here. Now let us talk about the methods which are typically used in the read modes. So there are 3 methods that could be useful lead seek and tell. Let us understand few more aspects about reading data files. So by default bytes from a file are decoded to unicode. So that is the you know default file encoding. So depending on the platform this is the fault file encoding can actually change.

So however unicode remains the you know one of the common default file encoding and this is actually a text mode for files. So by default we get by default the default mode is text mode and in that case the file you know the bytes from a file are they are due according to the you know to unicode and what constitutes a character that is actually they do not mind the kind of for you know encoding that we might be using.

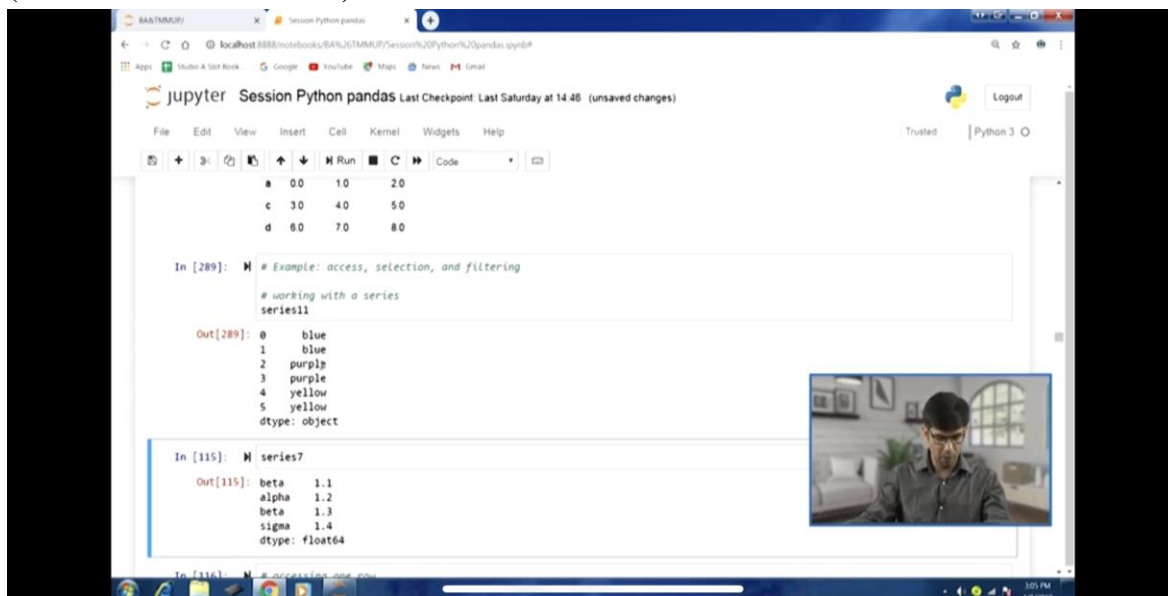
So utf-8 is the one of the most popular you know Unicode format that we have. So let us

understand some of these aspects through examples. So let us take read method first, so we have talked a little bit about read method before also, but we will use different examples here. So read method it will advance the file handles position by the number of you know bytes that number of bytes that have been read.

So first thing is whenever we apply read method you know this file version will be advanced. So that is why in the previous example that we have seen and in the previous lecture in this lecture as well as in the previous lecture once we you know call this read method then you know data is going to be dead. However the file position will also change. So if whenever we have called that read method again we have got the empty bytes object.

Because we have been calling we have been reading all the data in a file till now, we have not specified the number of characters that we would like to read rather all the characters we have been reading. So whenever we have called that my read method again we have got the empty path object, because file handles position will be changed by the number of bytes that. So if you read all the bytes it will lease at the end of the file and the empty parts object would be returned.

(Refer Slide Time: 11:04)



```
File Edit View Insert Cell Kernel Widgets Help
+ - 2: Run C Code
a 00 10 20
c 30 40 50
d 60 70 80

In [289]: # Example: access, selection, and filtering
# working with a series
series11

Out[289]: 0    blue
1    blue
2    purple
3    purple
4    yellow
5    yellow
dtype: object

In [115]: # series7

Out[115]: beta    1.1
alpha    1.2
beta    1.3
sigma    1.4
dtype: float64
```

So in this case you can see we are using this example fl6.read and 9 so we would like to the argument that we are passing is having this value 9, that means we would like to read 9

characters here. So if I done this you can see I have got this string om namah and then space total if you count the characters here then it comes out to be you know 9. Now there is another file mode apart for the default file mode is text mode and where the bytes are typically in you know encoding you know using the unicode format.

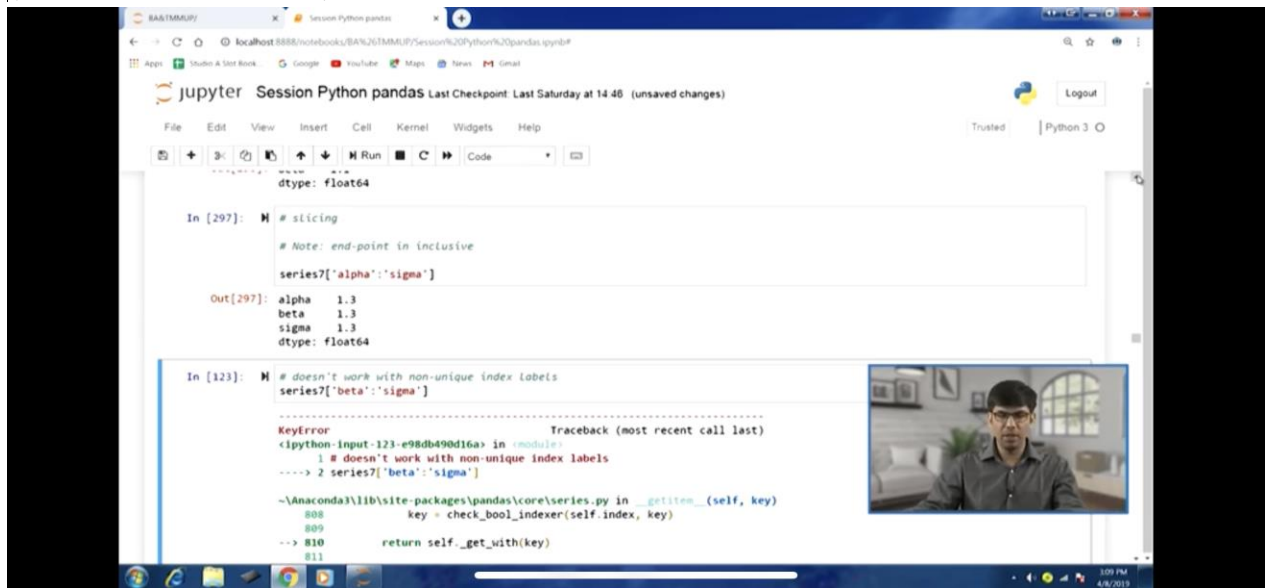
So then another file mode is the binary mode. So this is typically used to read raw bytes. So whenever you know we want to actually you know read the data the way it is written in the raw form then we can use this binary mode form b is used and the read method will actually returns the exact number of bytes that have been read if the file is opened before you know open in the binary mode then exact number of bytes in this particular file mode is returned.

If you remember what I said for you know for the text mode that the number of bytes you know the file position moves by the number of bytes read. So for a given number of characters that we might want to read in the text mode the number of bytes might be more than that you know more than the number of characters that we might have read. So therefore there the file position you know will change slightly differently, might change slightly differently depending on the you know encoding that we might have used.

However in the binary mode it is actually the exact number of bytes, so when we open the file in the binary mode let us take an example fl7 and we are opening this file temp.txt, rb. So we are you know a pair we have appended this b in the read mode so now this is read we are opening the file for the reading but in the binding mode. Now will we are asking to read 9 bytes not 9 characters because the file is going to be open in the binary mode.

So when we call this method fl7.read and past this argument 9. This is actually about reading 9 bytes and that too in the raw form. So if I run this in the output you can see the output is up you know is prefixed with be b that is to indicate that the data that is coming after that b is in the you know binary more in the in the draw form. So you can see b om namah. So in this case there is no problem and you know because the encoding is such that the characters that we have printed there is no problem the output is very similar to what we had in the text mode.

(Refer Slide Time15:00)



```
dtype: float64

In [297]: # slicing
          # Note: end-point in inclusive
          series7["alpha": "sigma"]

Out[297]: alpha 1.3
         beta 1.3
         sigma 1.3
         dtype: float64

In [123]: # doesn't work with non-unique index labels
          series7["beta": "sigma"]

KeyError                                Traceback (most recent call last)
<ipython-input-123-e98db490d16a> in <module>
      1 # doesn't work with non-unique index labels
----> 2 series7["beta": "sigma"]

~\Anaconda3\lib\site-packages\pandas\core\series.py in _getitem_(self, key)
    808     key = check_bool_indexer(self.index, key)
    809
--> 810     return self._get_with(key)
    811
```

So one difference is the mode default mode is text mode another mode is binary mode, so in the text mode the file cohesion might be you know changes differently because we would be reading number of characters, in the binary mode you would be reading number of bytes. So file cohesion changes accordingly. Now let us talk about the you know another method that we have here tell method. So what this method does this method gives the current position of the file handle. So this is typically an integer value.

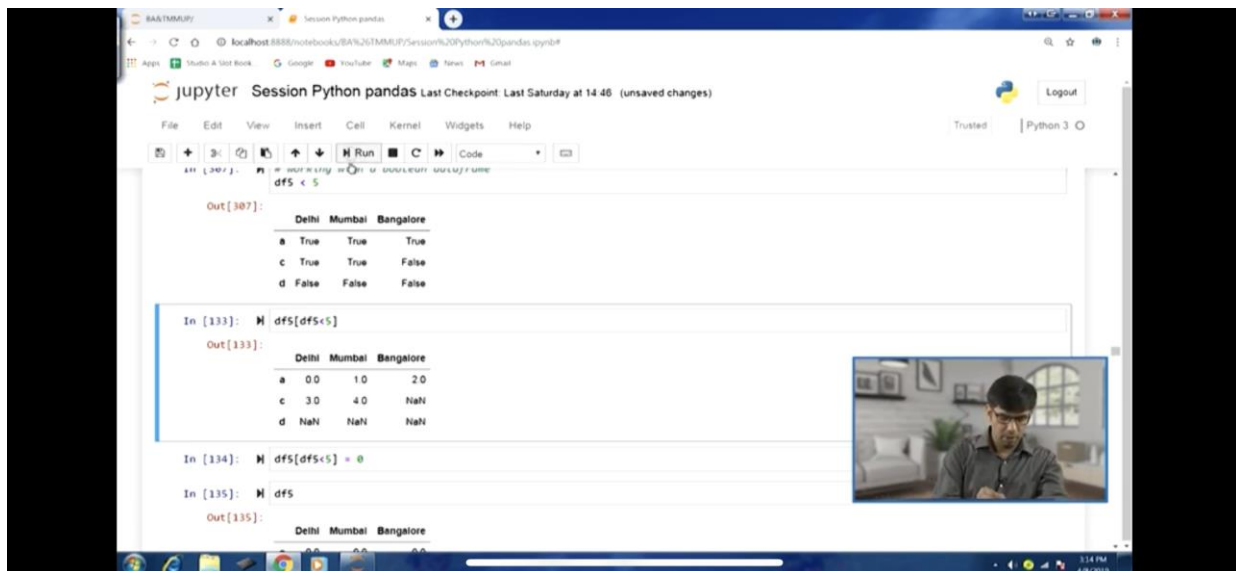
So if I look at the fl6 file object that he had created, so if I call this tell method so let us find out what is the current position of the file and for this file object. So if I run this you can see 9 because the you know previous you know operation that we had done using this file object fl6.read9 we had read 9 characters there and the current position of the file handle is 9. So probably 2 you know you know in code those 9 characters we requires just 9 bytes.

So that is why this is the position now, this is an integer value, so whenever we call this tell method we get the current position and integer value. Now similarly for fl17 also if I run this. So fl17 this file object was open in the binary mode and there also we had read 9 bytes, there. So the current position for this one also you know if you look at the output number 79. So the current position of file under in this case also is 9.

Because there was you know we the way encoding is there the number of characters and the number of bytes required to encode those characters are equal. So that is why you know we are getting this you know same number as output. However this might not be the case if we change the text encoding. Now sometimes you might be required to check the default encoding that is being used.

So in the open function that we have, so if you just you know go to the python reference. So as we have said before also sometimes it might be really useful to you know refer to the main pages manual pages of these function. So let us take an example here open if I try to find out here more details about the open. So you can see in this we have this python function built-in function. So this is what we are using here.

(Refer Slide Time: 19:30)



The screenshot shows a Jupyter Notebook interface with the following content:

```
df5 < 5
```

Out[107]:

	Delhi	Mumbai	Bangalore
a	True	True	True
c	True	True	False
d	False	False	False

```
In [133]: df5[df5<5]
```

Out[133]:

	Delhi	Mumbai	Bangalore
a	0.0	1.0	2.0
c	3.0	4.0	NaN
d	NaN	NaN	NaN

```
In [134]: df5[df5<5] = 0
```

```
In [135]: df5
```

Out[135]:

	Delhi	Mumbai	Bangalore
--	-------	--------	-----------

So you can see there are various arguments that can be passed on to this function you can see file then mode then buffering the encoding, you can see the encoding also there by default it is none right. So that means it is the system's encoding that is going to be used. If you want more detail about encoding you can scroll down and find out where they have the discussion on encoding in this main page. So you can see here you can see here.

Encoding is the name of the encoding used to decode or encoding the file this should only be

used in text mode. So this particular argument is only to be used in the text mode because it is in the text mode that we would like to use different types of encoding and in the binary mode we just want to the data and the raw form. So therefore this argument is not applicable there. However if the value encoding is argument has not been used and so the default value is none.

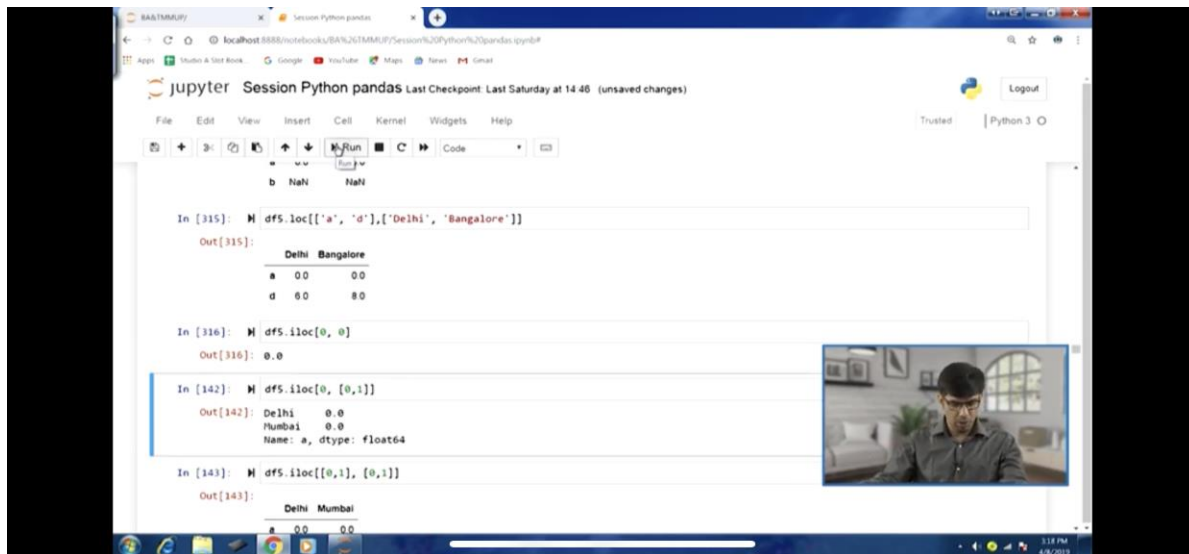
So their whole systems you know platforms default encoding is going to be used in this case. So let us go back, so let us find out what is the default encoding that we have here. So for this we will use the sys module that is that helped us in terms of to you know understanding the default you know system settings as well. So let us run this import sys and then sys start, get default encoding. If I run this you can see utf-8.

So it is the you know the you this unicode utf-8 format you know you in the most common you know unicode format is being used here. Now let us talk about the another method seek. So what this method you know does changes the file position to the indicated byte so an integer value. So we can pass on an integer value, so that entire value would actually be indicating a byte where we would like to move the file position.

The seek method can actually be used to perform this particular task. So let us take this file object fl6 and dot seek method and we pass on the value 5. So the file position is going to be moved to this particular byte number. So if I run this and the output is 5 here. And now if I try to read you know from this file object fl6. read1. So I would like to read just 1 character here you can see am m able to read m.

So earlier output if you go back and see what was the earlier output that we have so you know om this string is om namah. So if you see that you know 5th character is M actually. So that is why when we I know change the file position using the seek method. And when we read the first you know just one character from there then it was M. Now let us close these 2 file objects. Now let us talk about few more scenarios here.

(Refer Slide Time: 23:22)



So there are 2 more useful method write lines and you know read lines. So these 2 methods especially the real lines method we have used before also. Now I will discuss them in more detail here. So write lines method this is actually used to write the path sequence of a string to the file. So if we have a certain sequence of strings, so this particular method write lines can be used in to write all those lines in one go.

And those **who** lines sequence of strings can be written in the file. So let us take an example. So what we will do we will use this temp./2.txt, there we already have you know certain data there and if I you know we are using the with block here. So you can open this temp3.txt in the right mode. So whatever is there whatever data is there in the temp3.txt that will be raised and this fl8 object is going to be created.

And now we are using this method fla. you know write lines and here you can see I am using a comprehension here. So x for x in open path 1 if length greater than 1. So if there are multiple lines there in the file and you know there might be certain lines that you know certain lines with more characters there. So only the lines which are having more than 1 character those are going to be filtered and then written here.

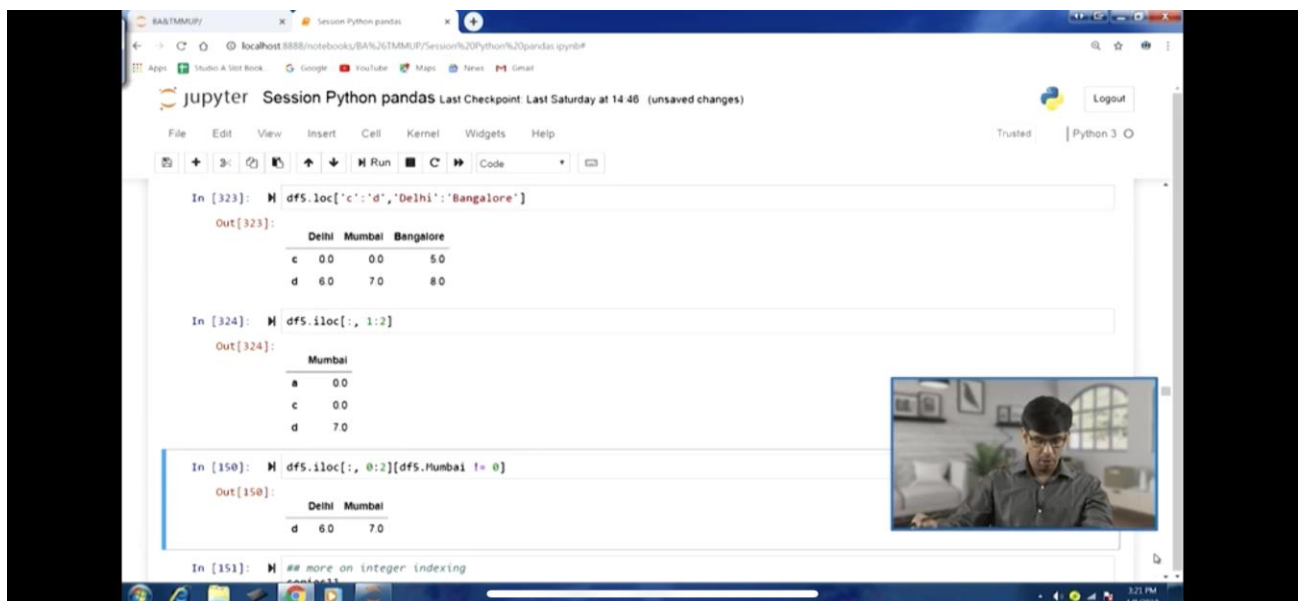
So if I run this and again I open this file and this time this temp3.file I am using you know a

readlines method here. So here if I run this so you can see and you know lines so all the lines that are there and this in this fl8 object that we have just written though it would be stored in the lines variable. So you can see here all these you know all these lines. So if there was any lines you that previously when we had created this file we had the empty bytes object.

So that has been filtered out from here and you can see the main idea is to demonstrate how read lines method can be used so in one go we have been able to read all the lines here/ So it is like a for loop is running we are iterating over all the lines that are there similarly when we use write lines there also a similar kind of thing we can use. Let us move to the next aspect here bytes and unicode with files.

So let us talk about a few details about you know these aspects so unicode format we have already talked about the default this is typically the default you know text encoding that is you know used in the most platform and there also utf-8 is the more popular and if we talk about utf-8 this is a variable-length unicode you know encoding. So in this when we are requesting some number of characters from the file like we did in the few examples before.

(Refer Slide Time: 27:09)



```
In [323]: df5.loc['c':'d', 'Dehi':'Bangalore']
Out[323]:
   Dehi  Mumbai  Bangalore
c    0.0    0.0    5.0
d    6.0    7.0    8.0

In [324]: df5.iloc[:, 1:2]
Out[324]:
   Mumbai
a    0.0
c    0.0
d    7.0

In [150]: df5.iloc[:, 0:2][df5.Mumbai != 0]
Out[150]:
   Dehi  Mumbai
d    6.0    7.0

In [151]: ## more on integer indexing
```

So this is typically done in the default text mode as we have discussed, then the python what will python do it will lead enough bytes from the file to decode that many characters. So previously

we had called this method read and we said that the number of characters at that we want to read so we can pass on that as an argument. So but depending on the encoding that might have been used the python will try to read those many number of bytes that are required to read the specified number of characters.

So that is the idea that we would like to demonstrate here if possible. So we have the with block here and in this with block we are opening this file as this f18 object and then we are calling this read method and just 9 characters we want to read from this file and then these 9 characters are going to be stored in this chars variable here. So if I run this and if I just run this chars again access what is there you can see om namah.

So and then this space is also there failing space is there, so 9 characters we have read from this file, so you know depend because in this case the default encoding is utf-8 so therefore to be able to read 9 characters we require python is required to reads just the 9 bytes, but it might so happen that for some other you know text encoding a more number of bytes might be required to be able to read 9 characters.

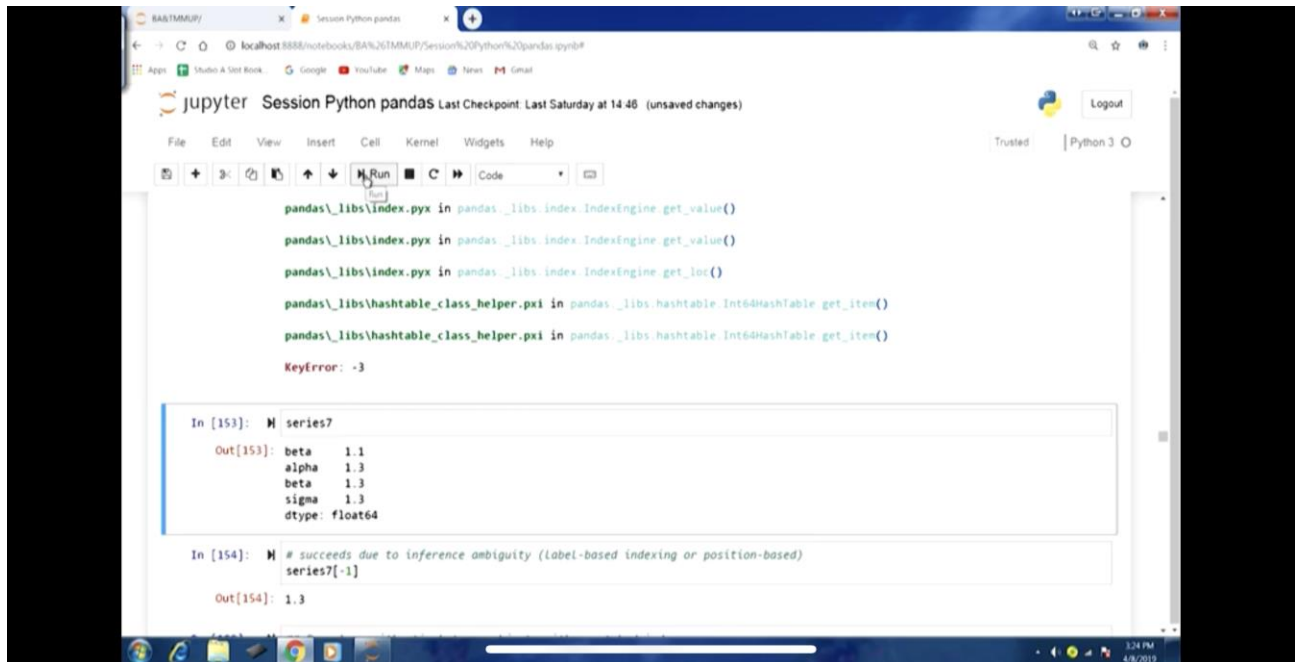
So in that case some of the bytes will increase, so it depends on the text encoding. Now as we have discussed when we use the read method that when we are requesting when we have opened the file in the binary mode and we are requesting certain number of bytes to be read, then in the binary mode python is going to python is going to read the exact number of bytes as we have discussed before.

So let us have this block with open path 1 and the binary mode rb we have indicated as f18: and then raw we are initializing this variable raw and we are reading 9 bytes here f18.read 9 bytes here. So all these you know 9 bytes are going to be read and they are going to be stored in the raw variable here. So if I run this block of code, so we will have raw. So let us execute this one and find out what is there.

So as we have seen before you can see in the output is the whole number of bytes that have been read that is prefixed with the b to indicate that we are into the finally mode here and om namah

and the 9 character. So in this case because of the you know characters that are being used here 9 bytes are actually enough to be able to you know whether we read 9 bytes or whether we read 9 characters.

(Refer Slide Time: 29:32)



The screenshot shows a Jupyter Notebook interface with the following content:

```
pandas\_libs\index.pyx in pandas\_libs.index.IndexEngine.get_value()
pandas\_libs\index.pyx in pandas\_libs.index.IndexEngine.get_value()
pandas\_libs\index.pyx in pandas\_libs.index.IndexEngine.get_loc()
pandas\_libs\hashtable_class_helper.pxi in pandas\_libs.hashtable.Int64HashTable.get_item()
pandas\_libs\hashtable_class_helper.pxi in pandas\_libs.hashtable.Int64HashTable.get_item()
KeyError: -3
```

```
In [153]: series7
Out[153]: beta    1.1
          alpha   1.3
          beta    1.3
          sigma   1.3
          dtype: float64
```

```
In [154]: # succeeds due to inference ambiguity (Label-based indexing or position-based)
          series7[-1]
Out[154]: 1.3
```

That means whether we are into the text mode or whether we are into the binary mode because the encoding is such the characters that we are using are such that to constitute them we just require 9 bytes. So in this case whether we have opened the file in the text mode or we have opened the file in the binary mode the output is same. However if we use a character maybe in some other you know language Latin or something or in some other you know Western languages or Indian languages.

And for that you know some of those characters if more number of bytes are you know required to constitute that character, then in that case you would see clear difference in the output in the sense that you know if we say nine bytes then we would not get the whole string and if we say 9 characters then we will get the you know full string but probably python you know will have to read more bytes.

So that is the idea that we would like we wanted to demonstrate. Now let us move into the next

aspect here that is about decoding the bytes to a string object. Now this particular thing is possible, if each of the encoded unicode characters is fully formed given a particular text encoding. So depending on a you know text encoding that is in use you know. So it is possible that you know bytes can be decoded into a string object.

However it will be possible only if and the encoded unicode characters are fully formed that means you know if a particular character requires more than one byte and the number of bytes that we want to decode it could not cover that character then this popular thing would fail, but if you know we were able to form that character all the bytes that were required to you know constitute that character, they were part of the bytes that we were decoding then this will succeed.

So the method that we typically call is decode, so raw is the object that we have read all those you know 9 bytes and then we calling decode method and utf-8 is the text encoding that we are using here. So if I run this you will get om namah here. Because whatever we had in raw that was already as per in the you know compiled with the utf-8 format, so we got all the characters here.

Now if I try to access raw and just up to you know I take a subset here and I just you know focus on 5 you know 5 bytes here and decode. Then let us see you can see because in this case there is no problem we do not have any character which requires more number of bytes more than 1 byte. So therefore no problem we got om na. So we have got 5 characters here. Similarly next one raw4 a decode you can see 4 characters no problem raw3 no issues 3 characters om then space.

Then let us take another example here what we will do we will you know create a file temp4.txt and in this case you know we will light a different string which will probably hopefully will require more number of bytes to constitute the character. So you can see we are typing we are you know writing new base here. So you can see I in this name is you know coming from you know different language.

So maybe it will require more number of bytes, so let us run this block of code path2 temp4.txt then we have this with block and we would like to write this new base there, then let us open this

file in the binary mode for reading. So we will have raw2 as our variable here let us print this so you can see in the output number 97 here, you can see b is prefixed to indicate that this is in you know binary mode we are in binary mode.

And then you can see the output here n and a then for i you can see different kind of you know characters that are there. So if you can see 1 2 3 4 bytes probably 4 bytes are required to constitute the character you know i.. you know that is there. So this is the output, now if I move to you know if I just you know decode 2 bytes here we get any no problem, if I try to decode 3 bytes, now if I try to decode 3 bytes we are running into that character i you know double.

So if I run this you can see I got this error unexpected end of data, because so far i.. we require 4 bytes, however we just try to decode 3 you know bytes that means only 1 byte that is constituting the i.. is part of this you know data that we want to decode. So therefore unexpected end of data that is the error. So this is what I wanted to demonstrate that time this can happen.

Now let us take the full data here and try to decode, then you can see now after we run into this kind of problem where our file position is in middle of some character where you know middle of some character which requires more than 1 byte, then we might get these kind of error invalid continuation byte because python is not able to continue from there. Because it is lying between somewhere.

Now we have raw2 and now we are changing the encoding here, so you can see I have changed the encoding from utf-8 to windows-1250. So let us see if we use this encoding what is the output. So if I run this you can see the output is slightly changed, so maybe in this text encoding that I have just used those you know those bytes those raw bytes they constitute a different character.

So in this case you can see the apostrophe kind of character has been constituted here using this particular text encoding. Now if I change the text encoding to windows1252 in the next line if I run this then you can see we got this new B here appropriately. So this is working here. Now when we are using this text encoding Windows1252 and again we do the same thing we try to

access 3 bytes here you know. So in a sense here we do not face any problem.

Now let us move into the another aspect sometimes we might be required to convert from 1 text encoding to another. So let us take an example here, so here we can use 2 with blocks inner and outer and we can easily run through you know convert the data from 1 mode to another mode . So we have temp.txt and then we are opening this path to this that we have you know created before as a fl8 object, then another with block open temp path and xt mode.

And we are encoding in 2 different you know so default encoding will be in the utf-8 and the new encoding would be Windows1258 and then what we are doing is you know all the lines from the you know 1 encoding they are going to be read and then we would be writing that into the new encoding. So we will be creating new object fl line, so there we would have written the you know lines in this new encoding. So let us run this, this is how we can do this.

So in this case we have run into some error, so let me resolve this, so we have opened in to open this in the xt format, so that we can change here. So let us fix this error. So what we will do , so if we do not want to change the code, so what we will do I will just delete this file from here it was deleted then we would not have any problem. So let us go back and if I run the same code again then it is ok.

And now what we will do we will read whatever we have written and we have written in different encoding. So first we read you know you know line by line from the you know every line in the original encoding and then written in the another encoding. So let us run this you can see now new B. So encoding has been changed, so we can convert you know we can convert file from one text encoding to another. Now let us focus you know another aspect.

So this is rather a you know caution note, this is related to the seek method that we have discussed and this particular aspect is mainly applicable to the non binary file modes. So what happens is that if the file position as we have seen in previous example file position might fall into in the middle of bytes defining character like we did in the i. character we ran into that middle of the bytes kind of thing.

So when such a thing happened you know so as per a you know so the what might happen subsequent reads that might we might perform they might result in an error. So in this case you can see let us take an example with open temp.5.txt in the right mode and we will now you know write this particular you know string value here. So let us run this and so now this example might work or might not work depending on the text encoding that we are dealing with here.

So let us open this file for reading and you can see this string has been written. Now if I try to do because this caution note is this for class it is ready to seek method. So let me seek into this particular file position 4, so if I run this fl8.seek4 then if I try to read whether I run into the problem or not. So in this case there is a you know no problem, because if the file position falls into the middle of the bytes only then this situation will occur.

Otherwise it will just go smooth so in this case it was not the problem, however the previous example that we have gone through where we are having new base and i.. factor. So there you had seen that we read of you know that particular line failed.

(Video Ends: 36:45)

So that concludes our built-in capability aspects of python language and now we I would like to stop here and in the next lecture we will talk about the next module numpy, thank you.

Keywords: Numpy, Python, encoding, decoding, prediction, Classification etc.