

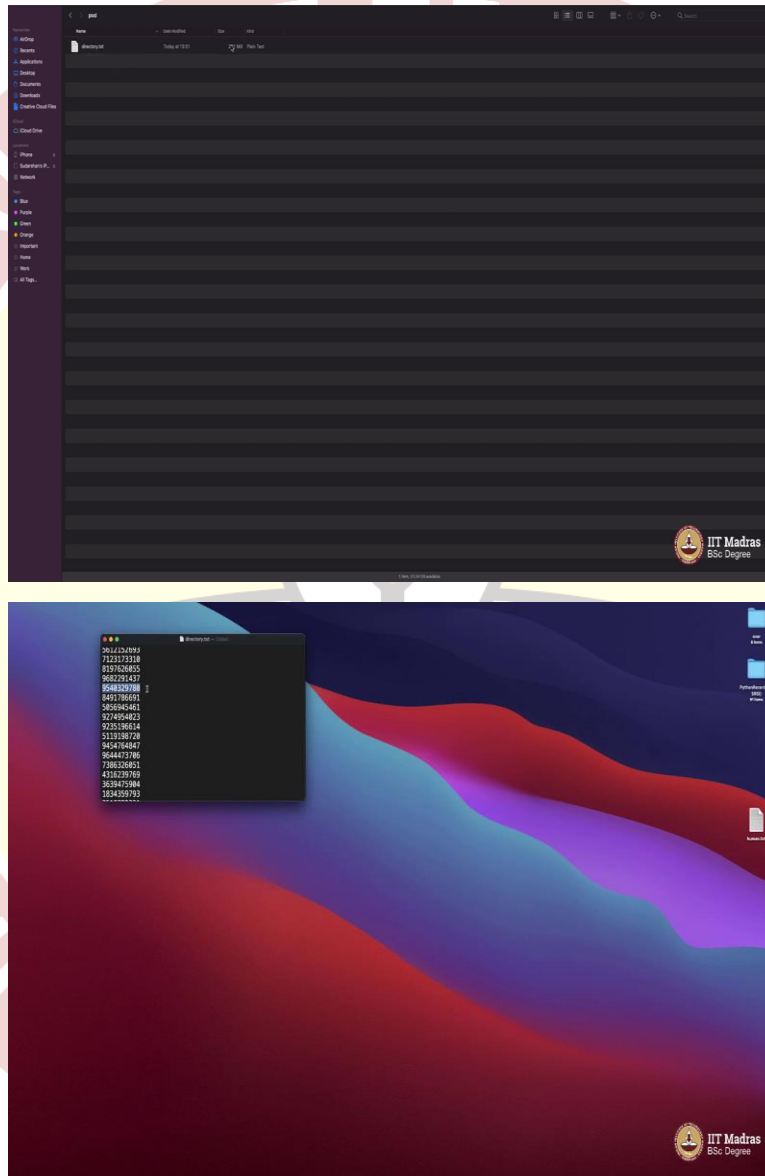


IIT Madras

ONLINE DEGREE

Programming in Python
Professor. Sudarshan Iyengar
Department of Computer Science & Engineering
Indian Institute of Technology, Ropar
Big text file handling

(Refer Slide Time: 0:16)

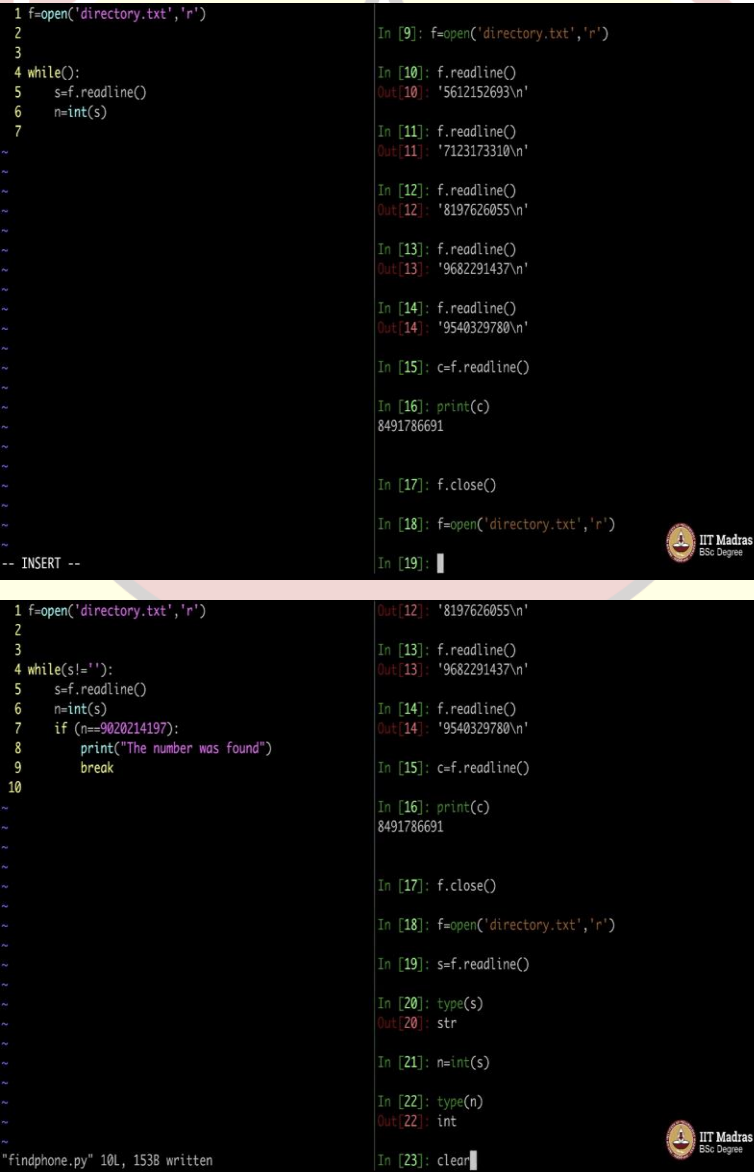


So, here we have a huge file called directory dot txt containing 231 MB of information. Let me double click and see what that contains. It contains these phone numbers, one phone number per line. It is a 10 digit number, as you can see. It is a long list. So, long that, as I am scrolling, the

scroller is not moving, as you can see. It has 230 MB of information. That is going to be quite a lot. So, let me close this.

You see, even closing it takes a long time, see. I have good memory on this computer. But I mean, good RAM and processor, but still it took a long time to even close it. 231 MB, it is not easy if a file is that big, especially a text file. A movie file that big, some of it opens, but a text file that big will actually, as you saw, will sort of get trying on your computer, meaning I mean, it is difficult for the computer to open such a file or close such a file.

(Refer Slide Time: 1:16)



```
1 f=open('directory.txt','r')
2
3
4 while():
5     s=f.readline()
6     n=int(s)
7
~
~
~
In [9]: f=open('directory.txt','r')
In [10]: f.readline()
Out[10]: '5612152693\n'
In [11]: f.readline()
Out[11]: '7123173310\n'
In [12]: f.readline()
Out[12]: '8197626055\n'
In [13]: f.readline()
Out[13]: '9682291437\n'
In [14]: f.readline()
Out[14]: '9540329780\n'
In [15]: c=f.readline()
In [16]: print(c)
8491786691
In [17]: f.close()
In [18]: f=open('directory.txt','r')
In [19]:

-- INSERT --

1 f=open('directory.txt','r')
2
3
4 while(s!=''):
5     s=f.readline()
6     n=int(s)
7     if (n==9020214197):
8         print("The number was found")
9         break
10

~
~
~
Out[12]: '8197626055\n'
In [13]: f.readline()
Out[13]: '9682291437\n'
In [14]: f.readline()
Out[14]: '9540329780\n'
In [15]: c=f.readline()
In [16]: print(c)
8491786691
In [17]: f.close()
In [18]: f=open('directory.txt','r')
In [19]: s=f.readline()
In [20]: type(s)
Out[20]: str
In [21]: n=int(s)
In [22]: type(n)
Out[22]: int
In [23]: clear

"findphone.py" 10L, 153B written
```

So, let me come here, and then try to see my motive is to find a particular number in this big file. So, how do I go about doing that? So, one way is, of course, you can say, `f equals`, what is that, `open` and the file name, `directory dot txt`, `read from it`. And every time you say `f read line`, it will give you the first number in that file. Again, if you do the same thing, it will give you the second number.

What exactly happens here, what happens here is the following. In your particular file, in a particular file, the first number whatever it is will be shown to you, and then it will move on to the second number, and then third number and then fourth number, fifth number and so on. I hope this is clear to you.

So, whatever was in this file, the first claim was this. Again, if you execute it, the second line will come there. Again, if you execute it, the third line will come there. Let me show you. The third line, whatever is there in the file, and then the fourth line, and then the fifth line, and so on. And ultimately it will reach the end. When it reaches the end, you see, if `c equals`, if I say, `read line`, and then `print c`, it will show you this number.

The moment it reaches the end, when there is nothing else to show, then it will result in simply discard. So, that is how you check for the end of the file. When it reaches the end, it will basically not give you the number, but an empty space here. So, we will try coding with this much of information. I believe this should be enough for you to understand what I am going to do next.

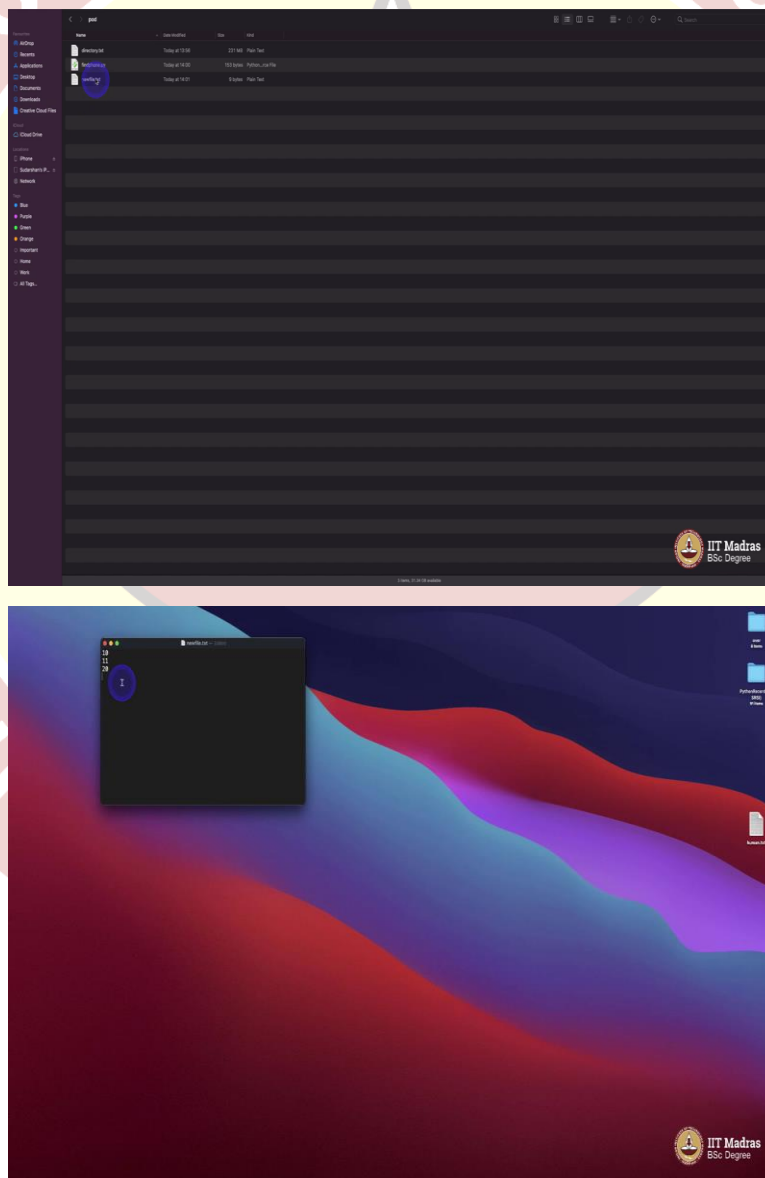
I am going to say `f equals`, let me just come here and then close the file, come here, `f equals` what, `open directory dot txt` and then `read from it`. Let me call this Python file as `find phone dot py`. So far, so good. So, what I do is I go ahead and I will write within a while loop something, some condition, let us see what is the condition is. What I do is I will first consider `s is equal to f read line`, and I want that to be a number. In fact, you can retain it as a string itself. But as you know this will be a string here, let me just illustrate that, `f equals open directory`, `s equals f read line`, type of `s` will be a string.

When you say `n equals int of s`, it becomes type of `n` is integer. I want my phone numbers to be integers. For some reason I am comfortable with it. In fact, you can retain them a string itself, but for my comfort, I am, it is easy on my mind. So, I will convert them to integers. And what I will

do is I will say if n is equal to the number that I am in search of n 920214197, print, the number was found.

And then I do not want to continue any further, because I need to break the while loop, I will say break, otherwise, keep continuing this. Keep continuing until s is not equal to empty. This is what happens when the file goes to the end. Once it reaches the end, it becomes equal to this thing. Whenever it is not equal to this thing, it should keep running. Let me illustrate this with a small example of what is happening here. Let me clear the screen.

(Refer Slide Time: 5:20)



```
1 f=open('directory.txt','r')
2
3
4 while(s!=''):
5     s=f.readline()
6     n=int(s)
7     if (n==9020214197):
8         print("The number was found")
9         break
10
~
~
~
In [24]: g=open('newfile.txt','w')
In [25]: g.write('10\n')
Out[25]: 3
In [26]: g.write('11\n')
Out[26]: 3
In [27]: g.write('20\n')
Out[27]: 3
In [28]: g.close()
In [29]: !ls
directory.txt  findphone.py  newfile.txt
In [30]: !cat newfile.txt
10
11
20
In [31]: g=open('newfile.txt','r')
In [32]: s=g.readline()
In [33]:
"findphone.py" 10L, 153B written

1 f=open('directory.txt','r')
2
3 flag=0
4
5 while(s!=''):
6     s=f.readline()
7     n=int(s)
8     if (n==9020214197):
9         print("The number was found")
10        flag=1
11        break
12 if (flag==0):
13     print("The number was NOTTTTT found")
14
~
~
~
In [31]: g=open('newfile.txt','r')
In [32]: s=g.readline()
In [33]: print(s)
10
In [34]: s=g.readline()
In [35]: print(s)
11
In [36]: s=g.readline()
In [37]: print(s)
20
In [38]: s=g.readline()
In [39]: print(s)
In [40]: s==''
Out[40]: True
In [41]:
```

Let me create a new file name, let us say, g equals open new file dot txt in write mode, g write let us say 10 next line and then again let us say g write 11 let us say next line, g write let us say 20 next line and let me close g. And if I see g will be created, new file will be created here. New file dot txt is here, as you can see, which comprises of these numbers 10, 11 and 20. These are the bytes written. That will be there in new file, cat new file dot txt.

Again, this is the linux way of showing the content of the file. In DOS, I think it is type, in Windows, I think it is type. Instead of cat, you say type here. Do not worry about these things. You can even ignore this particular line. All I am saying is you can read what is there in the file. You can even go to the directory and then open it here. What is that new file and you will see 10,

Now, if I want to, again, read from it. Let us say, I want to open the file, new file dot txt in read mode, I say s equals g read line. You see what happened here, the same thing here. And I will say print s, it gives me 10. S is in fact 10. Again, if I say read line and print, s is 11, 10, 11 and the next one was 20, it is 20. Now, again, when I say s equals g of read line, and then if I print s nothing comes. This empty thing is called this cat.

(Refer Slide Time: 8:06)

```
1 f=open('directory.txt','r')
2
3 flag=0
4
5 s=''
6
7 while(s!=''):
8     s=f.readline()
9     n=int(s)
10    if (n==9020214198):
11        print("The number was found")
12    flag=1
13    break
14 if (flag==0):
15    print("The number was NOTTTT found")
16
```

```
In [41]: run findphone.py
-----
NameError                                Traceback (most recent call last)
~/pod/findphone.py in <module>
      3 flag=0
      4
----> 5 while(s!=''):
      6     s=f.readline()
      7     n=int(s)

NameError: name 's' is not defined


In [42]: run findphone.py
The number was found

In [43]: run findphone.py
-----
ValueError                                Traceback (most recent call last)
~/pod/findphone.py in <module>
      7 while(s!=''):
      8     s=f.readline()
----> 9     n=int(s)
     10     if (n==9020214198):
     11         print("The number was found")

ValueError: invalid literal for int() with base 10: ''

In [44]:
```

"findphone.py" 16L, 239R written

 JIT Madras
BSc Degree

number here? Let me input instead of 7, I will input let us say, 8. This number is not there. I have verified it. So, it takes its own time, because it is a long file, you see.

Probably the number was found initially itself. That is why it said invalid literal for int with base 10. I think there is a mistake here. Whenever this becomes null, I think the `n equals int` does not work. So, we should say, if `s` is not equal to null, only then you do all these things. The entire thing should be only if `s` is not equal to null. This is a small fix here. So, what I am trying to do here is, so the moment `s` becomes `f read line`, assuming it reaches the end of the file, `n equals int` of `s` does not make sense. So, think about it. You will get to know.

See, now you all have matured to Python, I mean, quite mature in Python. I do not have to explain every single thing for a longer time. As and always I type like, the number was not found, number spelling is wrong, I am sorry, that should not matter. Number spelling was wrong, now it is fixed. So, assuming just once again, it must say the number was not found. Little fix here I am sure I can rerecord this entire video by not making that mistake, but then again it is a frequently occurring mistake, whatever happened here. Whenever you try to convert something to integer, when that is null, this error comes. I wanted you to see this error.

Good that I made that mistake unknowingly. Now, that you saw the error, and I also fixed it with a `f loop`. Go through it, go through it slowly, code with me, make mistakes, as I code, the same mistakes that I did, I committed and fix it the way I fixed it and you will understand what is what. I hope this was clear. All we did was we found whether a number was present or not. This int conversion is actually not required. You can simply compare two strings, but then I tried doing it like this. It is fun to do it like this somehow. If it is a phone number, I want to see it as an integer. I mean, I am crazy.

So, there is one thing that I would like to end the discussion with. In the next video, we will see that what if it was a very, very big file. I mean, so big, not 250, 230 MB, but in terms of GB if the file is big, can we really process such a file? Can we handle such a file? Let us see it in the next video.