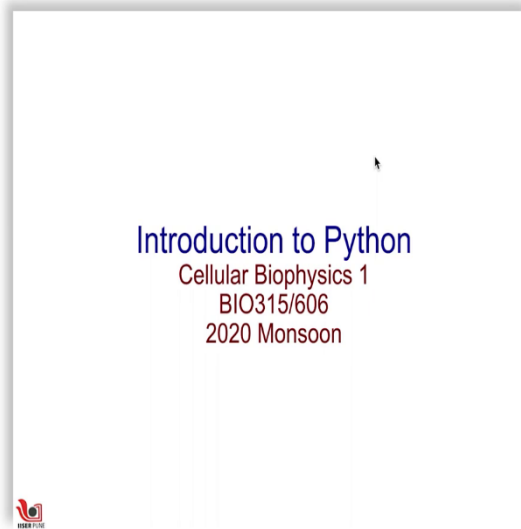


**Cellular Biophysics**  
**Dr. Chaitanya Athale.**  
**Department of Biology**  
**Indian Institute of Science Education and Research, Pune**  
**Lecture 03**  
**Python Programming Part 3**

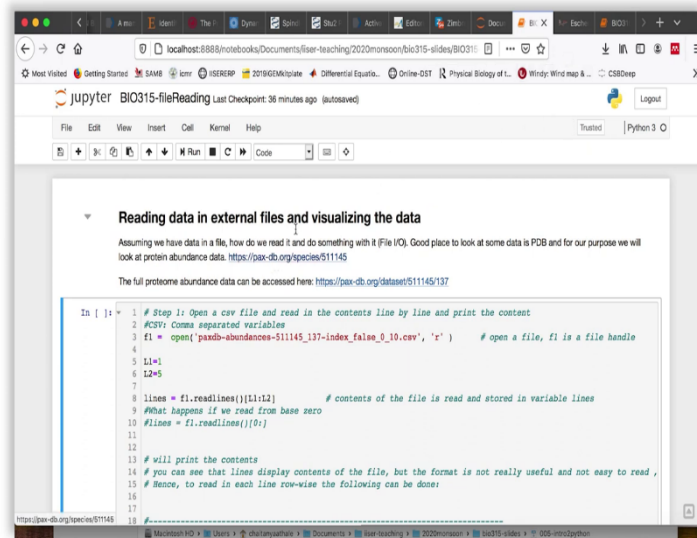
(Refer Slide Time: 00:15)



In the previous class, we talked a little bit about a very basic introduction to Python, and I had mentioned a learn as you go principle, so that our aim is not to become better programmers, which I hope will happen in the course of our activities, but to get the job done, in what is called scientific code. So, in that spirit, I will continue where I left off by adding to the part which relates to the assignment.

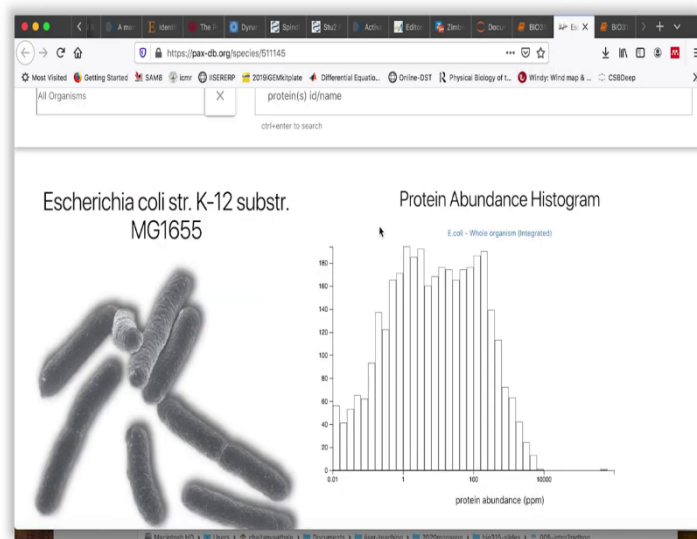
The assignment, if you remember was to plot the histogram of the frequency distribution of proteins and do some more analysis with it that you think is appropriate, so that you get a feel for two things, data input, and data visualization. So, without further ado, let us jump straight to it.

(Refer Slide Time: 01:15)



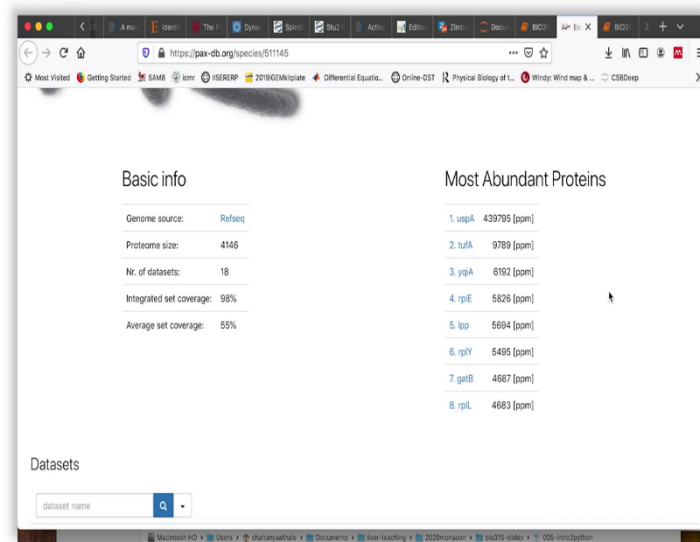
The screenshot shows a Jupyter Notebook window titled "jupyter BIO315-fileReading". The browser address bar shows the local host path. The notebook content includes a markdown cell with the heading "Reading data in external files and visualizing the data" and a code cell with the following Python code:

```
In [ ]: 1 # Step 1: Open a csv file and read in the contents line by line and print the content
2 #CSV: Comma separated variables
3 f1 = open('paxdb-abundances-511145_137-index_false_0_10.csv', 'r') # open a file, f1 is a file handle
4
5 l1=1
6 l2=5
7
8 lines = f1.readlines()[l1:l2] # contents of the file is read and stored in variable lines
9 #What happens if we read from base zero
10 #lines = f1.readlines()[0:]
11
12
13 # will print the contents
14 # you can see that lines display contents of the file, but the format is not really useful and not easy to read ,
15 # hence, to read in each line row-wise the following can be done
16
17
18
```



So, I am back in my environment where I have text or otherwise called markdown, and I have code which is in fact Python code, this is basically Python 3 as you can perhaps see here. And I am going to run the text, so that the text is nicely formatted. What I am doing here is to read in protein abundance data, which is taken from this website, which is nothing but eco-like k-12 substrain mg1655 protein abundance.

(Refer Slide Time: 01:56)



The screenshot shows a web browser window with the URL <https://pan-ctb.org/species/511145>. The page displays proteomics data for a specific species. It is divided into two main sections: 'Basic info' and 'Most Abundant Proteins'. Below these, there is a 'Datasets' section with a search bar.

Basic info	
Genome source:	RefSeq
Proteome size:	4146
Nr. of datasets:	18
Integrated set coverage:	98%
Average set coverage:	55%

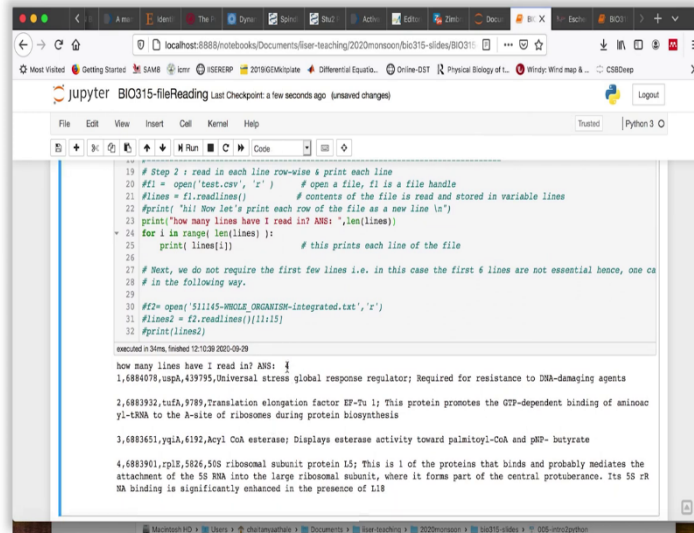
Most Abundant Proteins	
1. uspA	439795 [ppm]
2. tufA	9789 [ppm]
3. yqjA	6192 [ppm]
4. rpiE	5826 [ppm]
5. lpp	5694 [ppm]
6. rpiY	5495 [ppm]
7. galB	4887 [ppm]
8. rpiL	4683 [ppm]

Below the 'Most Abundant Proteins' table, there is a 'Datasets' section with a search bar labeled 'dataset name' and a search icon.

In other words, given the proteome size, that is to say 4,146 proteomes, from 18 data sets, with 98 percent set coverage and average that coverage of 55 percent, integrated versus average. We have a whole bunch of proteins whose abundances in parts per million are given. So all right, so, what can we do with this data?

we can effectively read that file in, and this is the file that has already been provided to you. And I am going to read just lines 1 to 5. We will call it the start line and end line, and then read in the lines. Now, you remember we talked about this in Python earlier, we have a so-called base 0 referencing system. So why do I start with 1? We will see. I hope I can answer this question to you at the end of this section.

(Refer Slide Time: 03:02)



```
19 # Step 2 : read in each line row-wise & print each line
20 #f1 = open('test.csv', 'r') # open a file, f1 is a file handle
21 #lines = f1.readlines() # contents of the file is read and stored in variable lines
22 #print( "hi Now let's print each row of the file as a new line \n")
23 print("how many lines have I read in? ANS: ",len(lines))
24 for i in range( len(lines) ):
25     print( lines[i] ) # this prints each line of the file
26
27 # Next, we do not require the first few lines i.e. in this case the first 6 lines are not essential hence, one ca
28 # in the following way.
29
30 #f2= open( '511145-WMGLE_ORGANISM-Integrated.txt', 'r')
31 #lines2 = f2.readlines()[1:115]
32 #print(lines2)

executed in 34ms, finished 12:10:29.2025-09-29
how many lines have I read in? ANS: 4
1,6884078,uspA,439795,Universal stress global response regulator; Required for resistance to DNA-damaging agents
2,6883932,tufA,9789,Translation elongation factor EF-Tu 1; This protein promotes the GTP-dependent binding of aminoac
yl-tRNA to the A-site of ribosomes during protein biosynthesis
3,6883651,yqiA,6192,Acyl CoA esterase; Displays esterase activity toward palmitoyl-CoA and pnt- butyrate
4,6883901,rplE,5626,50S ribosomal subunit protein L5; This is 1 of the proteins that binds and probably mediates the
attachment of the 5S rRNA into the large ribosomal subunit, where it forms part of the central protuberance. Its 5S r
NA binding is significantly enhanced in the presence of L18
```

So, once we have opened the file. That is what this open does, I mean it is Python. So the language is fairly simple. You want to open a file, you call the file name, and say open. That R indicates that it is in a read-only format. Once I have read in the file, I want to take the information in each line or rows, if you consider the matrix nomenclature, and put it into some data container.

So, this text is just simply to say hi, let us print each row of the file as a new line, and go through those lines that we read in here, which is called by a function called read lines, very convenient. Remember, this is f1 dot, meaning to say in the file object, read lines. Be careful of the syntax, modify it if you like to make sure you understand how it works.

If I now say print lines of index i, inside a for loop which says i in the range of length of lines, then essentially it is going through every line that I have read. In order to know how many lines there are in the first place that I have read in it, is very easy if you remember your commands from earlier.

How many lines? That is what the question is, right, how many lines have I read in? Answer, so, so there we go, oh I am so sorry, what did I miss here, good, four lines. Okay, that is the answer. So, the four lines are uspA, tufA, yqiA, and rplE that is what I have read in. These are the abundances. And now, I want to do something with it.

And that do something, remember the aim is the histogram is to split them, and store the data. You know, this is now a string effectively. It is, or you could say, a line, that line has comma separated variables, CSV, remember that is the type of the file that we are using, CSV here.

(Refer Slide Time: 05:25)

```

20 print('ORF values:\n',orf)
21 print('Abundance values:\n', abundance)
22 f.close() # Bye, the task is done! so we close the file!
23
24
25
executed in 47ms, finished 12:12:14 2020-09-29
Mo: 1, PID: 6884078, ORF: uspA, Abundance: 439795
Mo: 2, PID: 6883932, ORF: tsfA, Abundance: 9789
Mo: 3, PID: 6883651, ORF: ygiA, Abundance: 6192
Mo: 4, PID: 6883901, ORF: rplB, Abundance: 5826
Protein ID values:
[6884078, 6883932, 6883651, 6883901]
ORF values:
['uspA', 'tsfA', 'ygiA', 'rplB']
Abundance values:
[439795, 9789, 6192, 5826]

Do something with the values
Protein abundances should be visualized. What do you suggest? Abundance vs. PID? What do we learn from such graphical visualization?

In [ ]: 1 import numpy as np
        2 import matplotlib.pyplot as plt
        3
        4 f.close() # Bye, the task is done! so we close the file!

```

So, I go through the file, I go through the lines, and now I split the line. So now for every element in each line, I split it, yes, so I have a list of lines, I now split those individual components of each line. According to where the comma is placed, store that value and append it to three containers that I have created here, empty containers, buckets.

Protein ID, you are familiar with this, abundance, and ORF, Open Reading Frame, genes in other words, gene names with three letter words. If it is all good, and it has worked, then it should probably print me, for every row, the serial number, the P ID, the ORF, and the abundance. Let us run it and see if it does that. Sure enough, and it has stored it in containers for P ID, ORF, and abundance.

(Refer Slide Time: 06:31)

```

Abundance values:
[439795, 9789, 6192, 5826]

Do something with the values
Protein abundances should be visualized. What do you suggest? Abundance vs. PID? What do we learn from such graphical visualization?

In [ ]: 1 import numpy as np
        2 import matplotlib.pyplot as plt
        3
        4 f.close() # Bye, the task is done! so we close the file!
        5
        6 plt.hist(abundance)#,bins='auto') ##### IS A HISTOGRAM
        7 plt.title("Histogram of protein abundances")
        8 plt.xlabel("Protein Abundance")
        9 plt.ylabel("Frequency")
        10 plt.show()
        11
        12 plt.bar(orf,abundance)#, '-or')
        13 plt.xlabel("ORF same")
        14 plt.ylabel("Abundance")
        15 plt.show()
        16

executed in 234ms, finished 11:31:37 2020-09-29

In [ ]: 1

```

Now, I want to visualize it. So, what do you suggest? What kind of visualization is useful to do? Should we plot the abundance versus P ID? Should we plot a ranked list? What should we do? So, the simplest thing that we always like to do when we have statistics and we are trying to make sense of what is going on, and what is the underlying phenomenon, is to plot a frequency histogram, when we have data like this, when we have such rich data.

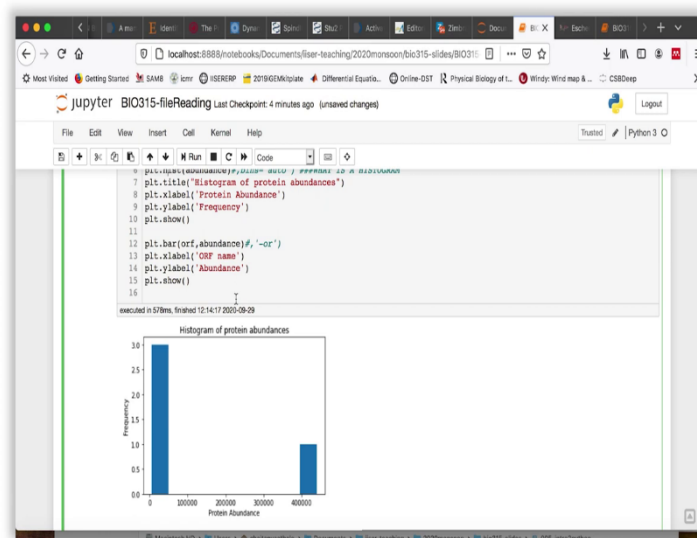
That frequency histogram in Python and many of the scripting higher numerical scripting languages is inbuilt as a function called hist. For the future, I would like you to find out and tell me, how you would build a histogram in the first place? In this case, I have used the command hist, I have fed the list which is that which contains abundance values, they are just four of them, and I label the axes.

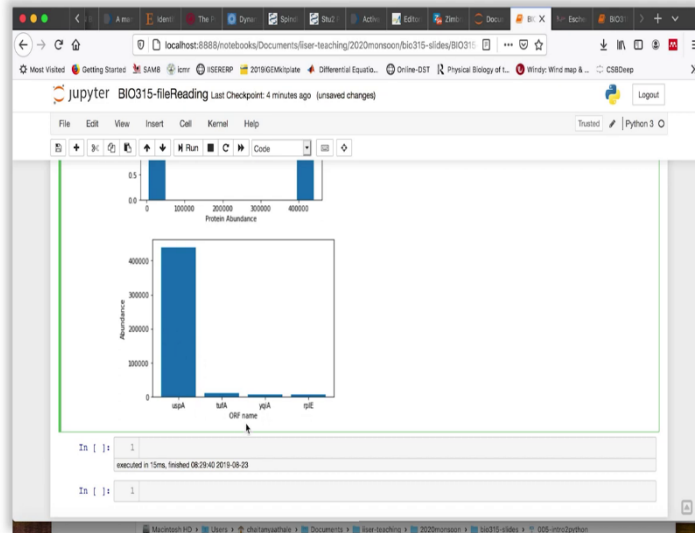
Please remember a graph without axes labels is artwork. You are, maybe in your free time you are artists, but you are training to be scientists. It is your duty to label your axis, to say what is on the x-axis, and what is on the y-axis, because otherwise besides you nobody else knows what is going on.

Your guides, bosses, and maybe one day you will become famous institute leaders, directors, lab heads, CEOs, I do not know, whatever you do, you will find this clarity of presenting your data very useful, you may or may not thank me for it, but other people will. Thank you, that is to say, so please label your axes.

Sure enough, when I run the command, hist abundance, I get a histogram, does not look very interesting, I want you to discuss with me next time, why it does not look interesting?

(Refer Slide Time: 08:36)





And I have also done the other thing, I have plotted a bar plot of each gene name, and its abundance, and you already see a trend of sorts it, kind of looks like it is already sorted. This is related to the fact that *uspA* is the most abundant protein in *E.coli*. So, that is it for the purpose of data reading in and data visualization, a very, very basic introduction. When I return with the next video, we are going to talk about how to put all this together in terms of simulations. Thank you very much.