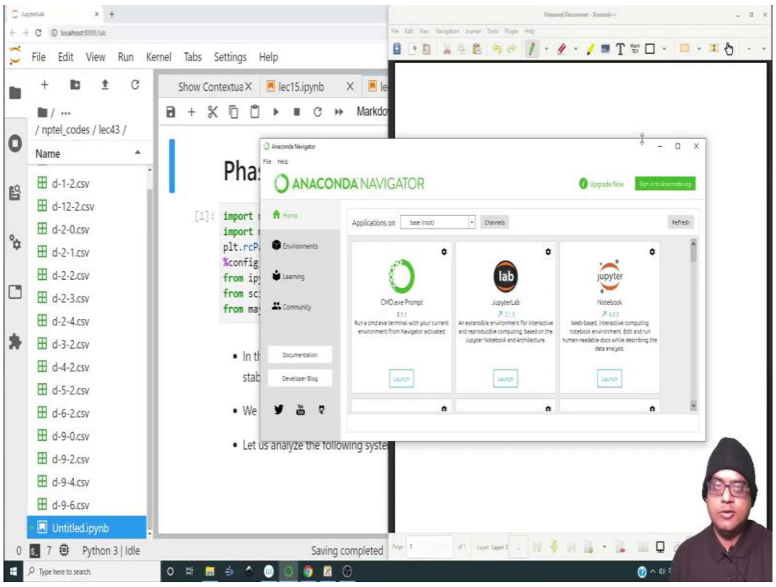
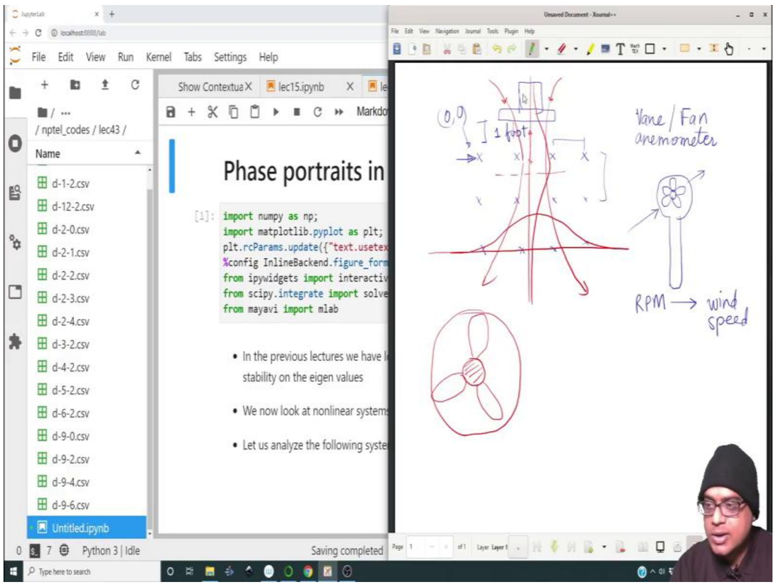
Tools in Scientific Computing
Prof. Aditya Bandopadhyay
Department of Mechanical Engineering
Indian Institute of Technology, Kharagpur

Lecture - 43
Analyzing data files and 2D interpolation

(Refer Slide Time: 00:29)



(Refer Slide Time: 00:38)



Phase portraits in

```
[1]: import numpy as np;
     import matplotlib.pyplot as plt;
     plt.rcParams.update({"text.usetex
     %config InlineBackend.figure_form
     from ipywidgets import interactiv
     from scipy.integrate import solve
     from mayavi import mlab
```

- In the previous lectures we have l stability on the eigen values
- We now look at nonlinear systems
- Let us analyze the following syste

Hi everyone, in this particular lecture we are going to analyze the following problem. So, I set up a table fan or a stand fan at a certain location in my lab something like this this is the top view and I marked out various positions on the floor each of the positions were separated by 1 feet.

The distance between each cross is 1 feet and what I did was I took a vane anemometer or a fan anemometer I do not know. I mean you can call it whatever you want, but basically it is a device like this it has a housing like this and that is it has a fan like this.

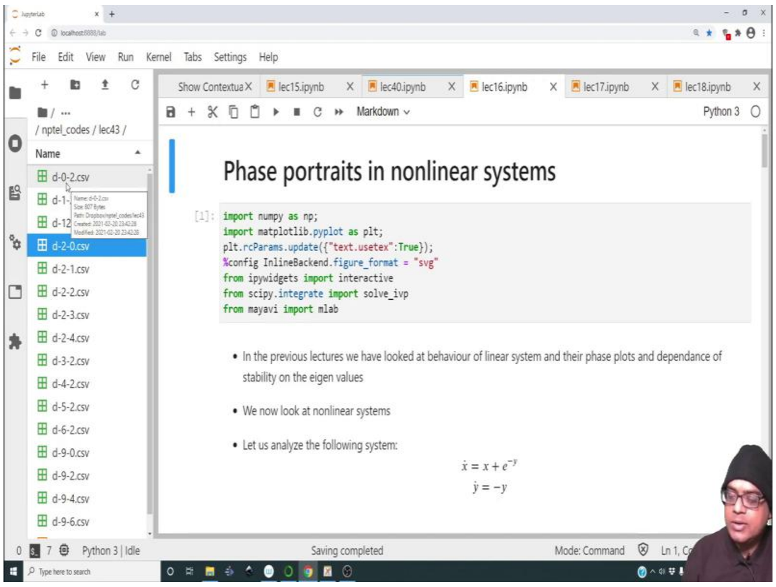
And when wind blows like this the fan starts to rotate and the microcontroller inside is able to transduce the RPM of the fan to the wind speed ok. So, the fan is turned on and then I stand at various positions in the hallway and I denote this position where the fan does not rotate at all (0, 0). So, this distance is also 1 foot and so, we will call this as the origin I mean we do not mind we can call this as the origin and we have a bunch of points.

So, I went ahead and collected data for all these kind of marked locations ok. So, what do you expect? I mean first things first, we expect the fan to draw air like this and push it out. So, because a typical fan has a casing like this and this point is closed ok, so something like this.

So, immediate to the vicinity there will be I mean of course, there will be a velocity, but it will not be high at some distance it will be high and then it will be something like this. So, at this at some location we expect a maximum velocity and if I look at the velocity field at this transect I should have a variation of velocity like this ok.

And along the actual location as well I should maybe I have something like this. Well let us see whether we are able to achieve all these kind of variations and let us try to reconstruct the 2 dimensional image this entire problem. So, let us begin and just to give you some context all the data files are saved like this.

(Refer Slide Time: 03:58)



Phase portraits in nonlinear systems

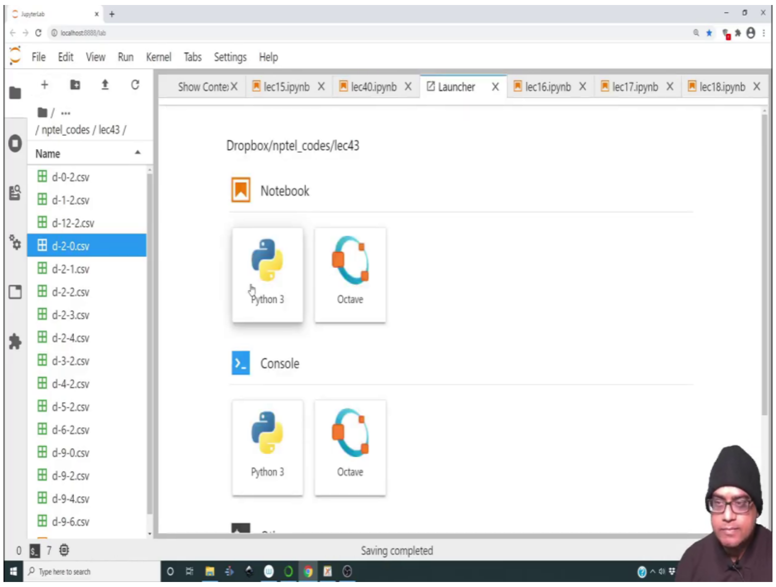
```
import numpy as np;
import matplotlib.pyplot as plt;
plt.rcParams.update({"text.usetex":True});
%config InlineBackend.figure_format = "svg"
from ipywidgets import interactive
from scipy.integrate import solve_ivp
from mayavi import mlab
```

- In the previous lectures we have looked at behaviour of linear system and their phase plots and dependance of stability on the eigen values
- We now look at nonlinear systems
- Let us analyze the following system:

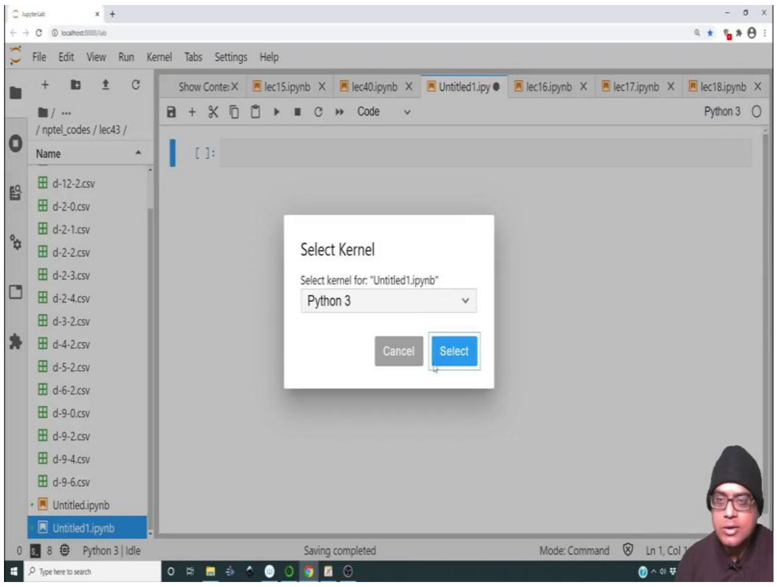
$$\dot{x} = x + e^{-y}$$
$$\dot{y} = -y$$

So, d-0-2 so, I saved the data files with this particular name. So, d-0-2 means the row number is 0 and the column number is 2 so, the row number actually corresponds to a y location the column number corresponds to x location ok. So, we have all these kind of data sets

(Refer Slide Time: 04:27)



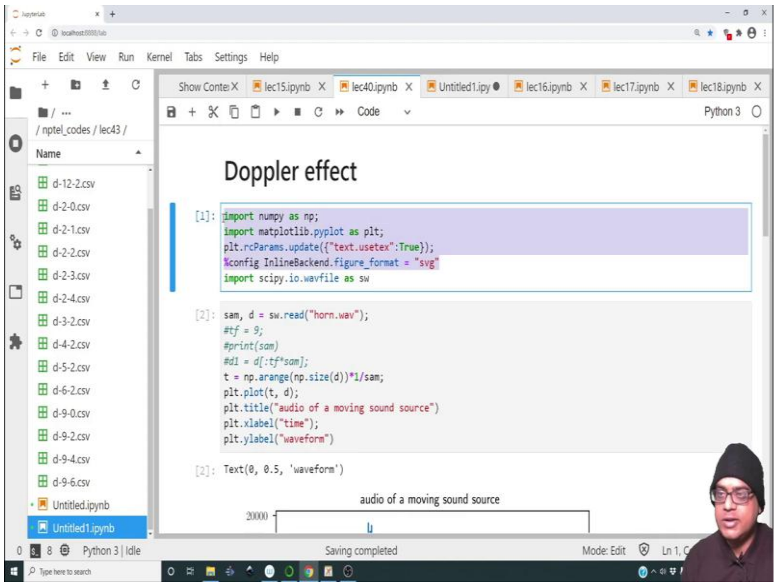
(Refer Slide Time: 04:30)


Select Kernel

Select kernel for: "Untitled1.ipynb"

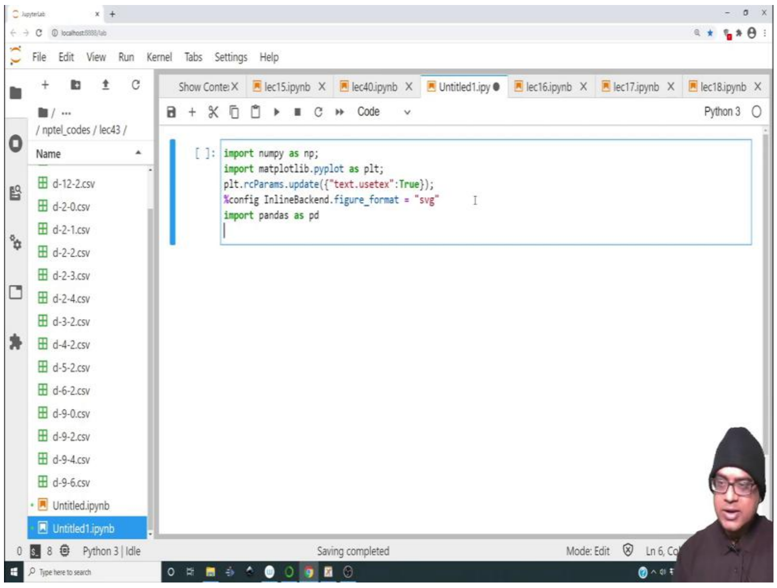
Python 3

(Refer Slide Time: 04:35)


Doppler effect

```
[1]: import numpy as np;
     import matplotlib.pyplot as plt;
     plt.rcParams.update({"text.usetex":True});
     %config InlineBackend.figure_format = "svg"
     import scipy.io.wavfile as sw

[2]: sam, d = sw.read("horn.wav");
     #tf = 9;
     #print(sam)
     #d1 = d[:tf*sam];
     t = np.arange(np.size(d))*1/sam;
     plt.plot(t, d);
     plt.title("audio of a moving sound source")
     plt.xlabel("time");
     plt.ylabel("waveform")

[2]: Text(0, 0.5, 'waveform')
```

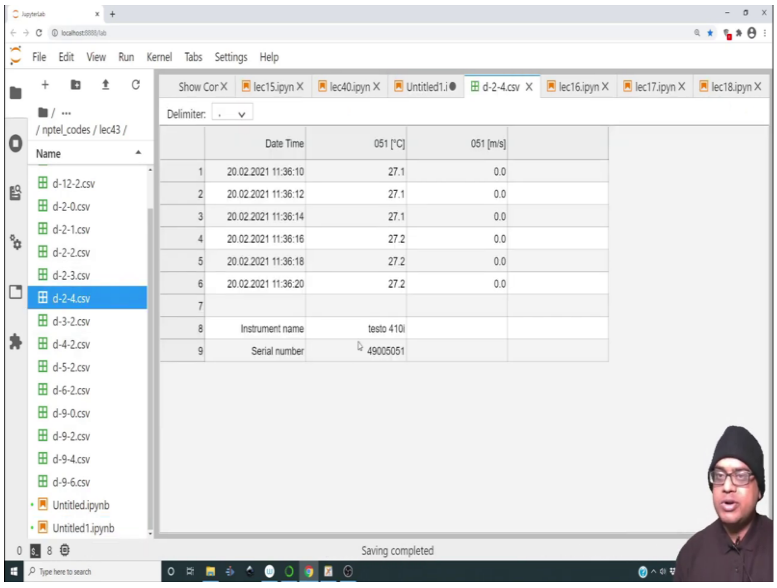
(Refer Slide Time: 04:39)



```
[ ]: import numpy as np;
     import matplotlib.pyplot as plt;
     plt.rcParams.update({"text.usetex":True});
     %config InlineBackend.figure_format = "svg"
     import pandas as pd
```

So, let us let us start processing the data let me create a new file let me select Python 3 as the interpreter. And we are going to need this standard thing we have been importing it, but in this particular problem additionally we going to require a data frame. So, we going to import pandas. So, pandas is a module with which you can do data analytics, but in this particular example we just want to use pandas for loading the data and because the data is actually csv file.

(Refer Slide Time: 05:11)



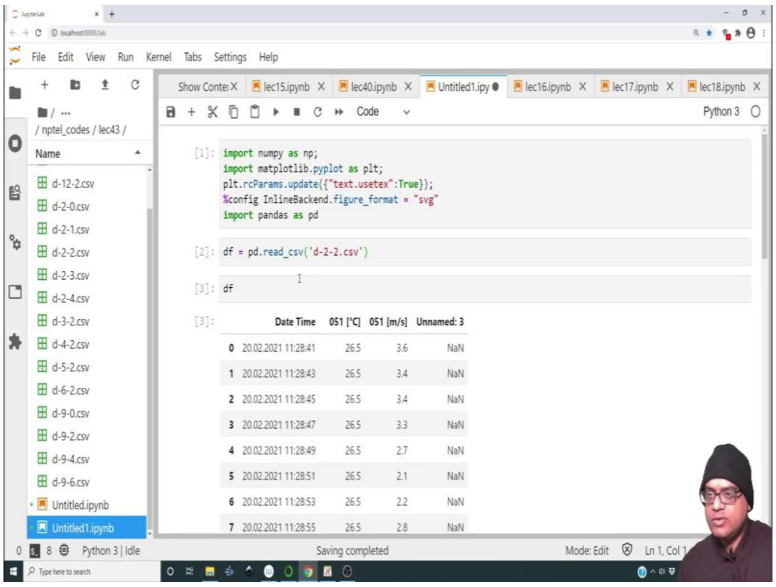
	Date Time	051 [°C]	051 [m/s]
1	20.02.2021 11:36:10	27.1	0.0
2	20.02.2021 11:36:12	27.1	0.0
3	20.02.2021 11:36:14	27.1	0.0
4	20.02.2021 11:36:16	27.2	0.0
5	20.02.2021 11:36:18	27.2	0.0
6	20.02.2021 11:36:20	27.2	0.0
7			
8	Instrument name	testo 410i	
9	Serial number	49005051	

So, let me open a data file and show you. So, this is how the data file looks like so, it stores the date and time it stores the temperature in degree Celsius, it stores the velocity in meters per second, and its it tells the name of the instrument its testo 410 i. So, the name of the brand is Testo and they manufacture such kinds of thing.

And the serial number of the device is 49005051 and the device is courtesy of the refrigeration lab Professor Sourab Mitra is the person whom I borrowed it from the Mechanical Engineering Department.

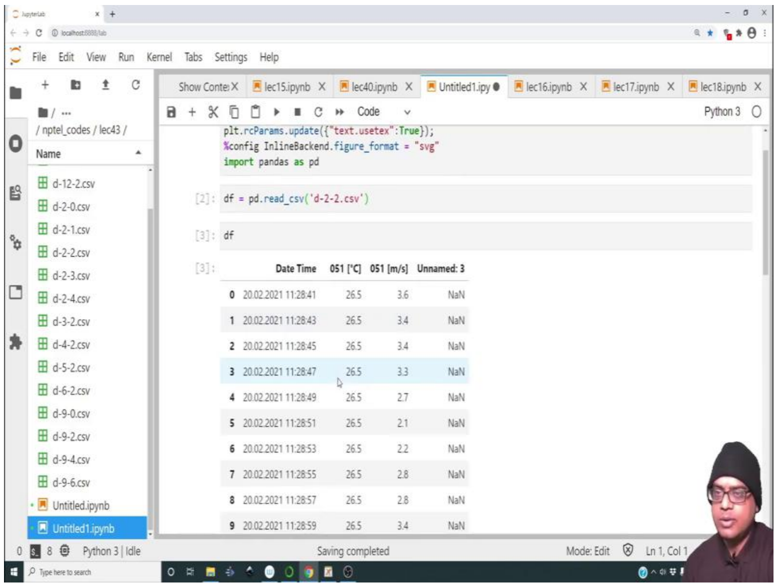
And we use this routinely in many of our experiments to measure wind speed. So, I am going to import pandas and the utility of pandas is primarily to just to help us import csv files right. So, let me do this. So, data frame equal to pd.csv ok pd.read csv and here we will give a file name

(Refer Slide Time: 06:13)



```
[1]: import numpy as np;
     import matplotlib.pyplot as plt;
     plt.rcParams.update({"text.usetex":True});
     %config InlineBackend.figure_format = "svg"
     import pandas as pd

[2]: df = pd.read_csv('d-2-2.csv')

[3]: df

[3]:
            Date Time  051 [°C]  051 [m/s]  Unnamed: 3
0  20.02.2021 11:28:41      26.5        3.6         NaN
1  20.02.2021 11:28:43      26.5        3.4         NaN
2  20.02.2021 11:28:45      26.5        3.4         NaN
3  20.02.2021 11:28:47      26.5        3.3         NaN
4  20.02.2021 11:28:49      26.5        2.7         NaN
5  20.02.2021 11:28:51      26.5        2.1         NaN
6  20.02.2021 11:28:53      26.5        2.2         NaN
7  20.02.2021 11:28:55      26.5        2.8         NaN
```

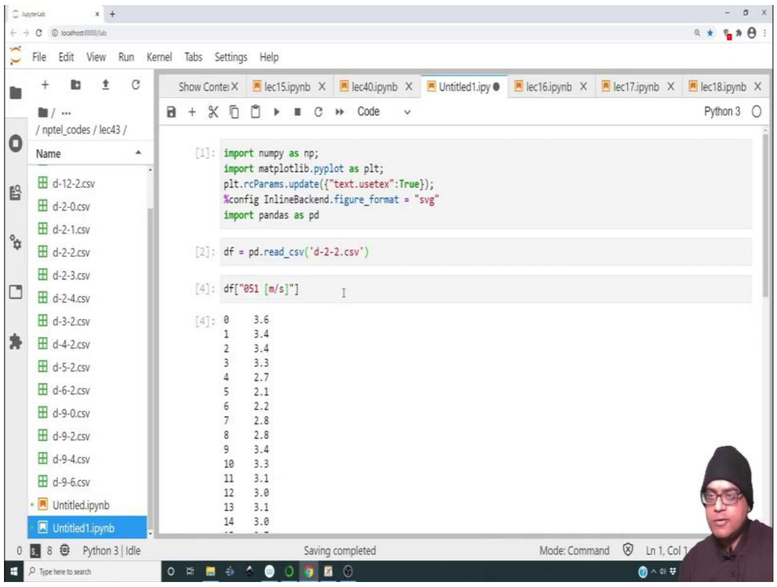
(Refer Slide Time: 06:41)



```
plt.rcParams.update({"text.usetex":True});
%config InlineBackend.figure_format = "svg"
import pandas as pd

[2]: df = pd.read_csv('d-2-2.csv')

[3]: df

[3]:
         Date Time  051 [°C]  051 [m/s]  Unnamed: 3
0  20.02.2021 11:28:41      26.5       3.6         NaN
1  20.02.2021 11:28:43      26.5       3.4         NaN
2  20.02.2021 11:28:45      26.5       3.4         NaN
3  20.02.2021 11:28:47      26.5       3.3         NaN
4  20.02.2021 11:28:49      26.5       2.7         NaN
5  20.02.2021 11:28:51      26.5       2.1         NaN
6  20.02.2021 11:28:53      26.5       2.2         NaN
7  20.02.2021 11:28:55      26.5       2.8         NaN
8  20.02.2021 11:28:57      26.5       2.8         NaN
9  20.02.2021 11:28:59      26.5       3.4         NaN
```

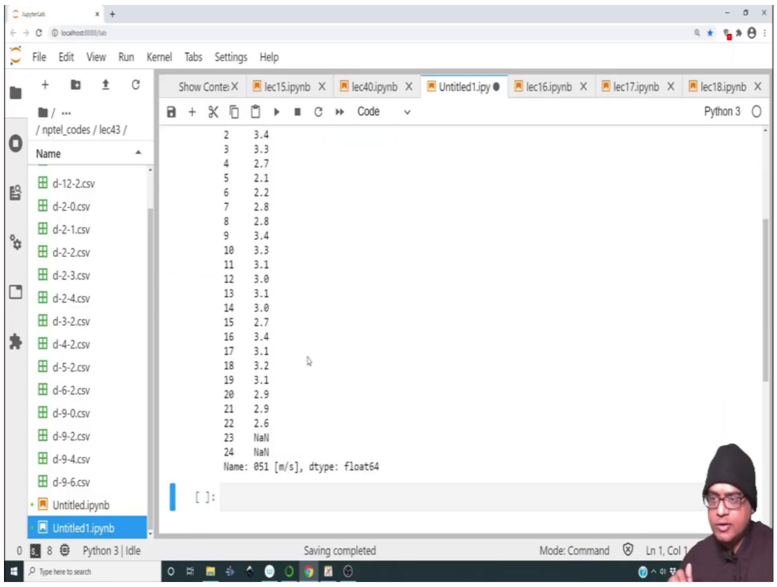
So, let us say d-2, 2. So, let us run this and let us see what df actually contains, So, df is actually this entire data is its this table ok. So, what can we do with this table? I mean we cannot really do anything with this table, but what we can do is sort of isolate the different columns from this ok.

(Refer Slide Time: 06:57)



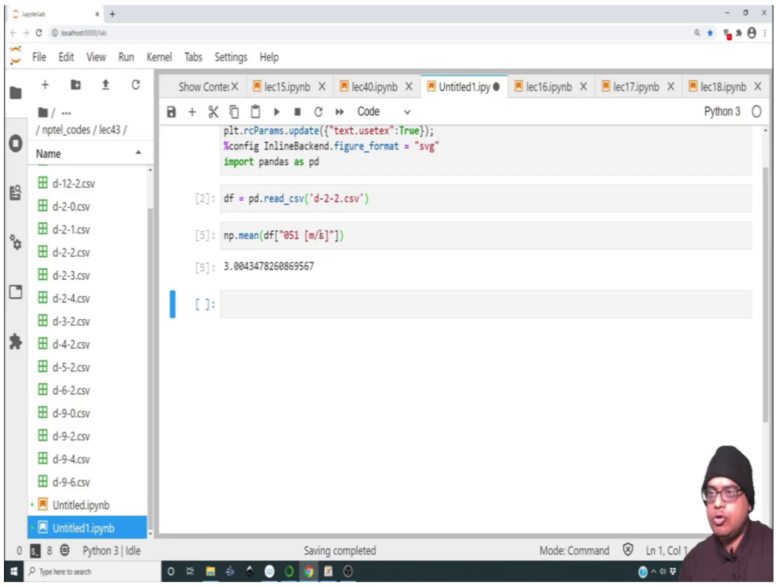
```
[1]: import numpy as np;
     import matplotlib.pyplot as plt;
     plt.rcParams.update({"text.usetex":True});
     %config InlineBackend.figure_format = "svg"
     import pandas as pd

[2]: df = pd.read_csv('d-2-2.csv')

[4]: df["051 [m/s]"]

[4]: 0     3.6
     1     3.4
     2     3.4
     3     3.3
     4     2.7
     5     2.1
     6     2.2
     7     2.8
     8     2.8
     9     3.4
     10    3.3
     11    3.1
     12    3.0
     13    3.1
     14    3.0
```

(Refer Slide Time: 07:03)



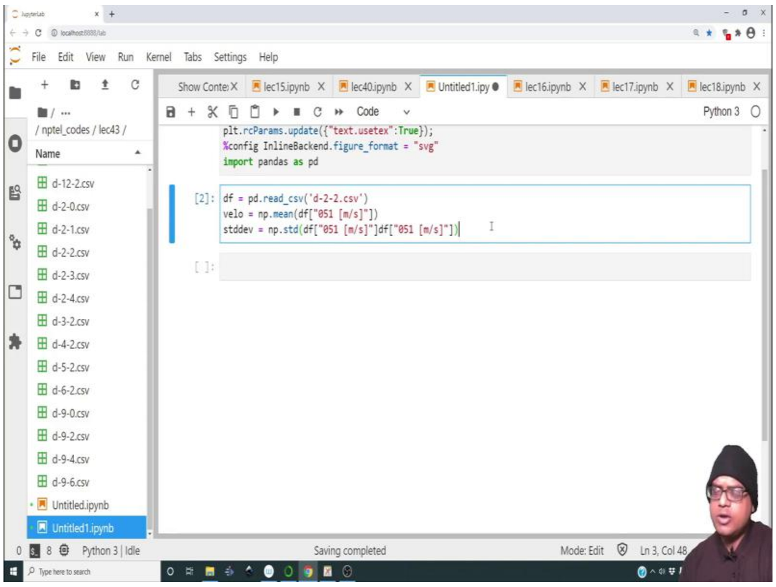
```
2     3.4
3     3.3
4     2.7
5     2.1
6     2.2
7     2.8
8     2.8
9     3.4
10    3.3
11    3.1
12    3.0
13    3.1
14    3.0
15    2.7
16    3.4
17    3.1
18    3.2
19    3.1
20    2.9
21    2.9
22    2.6
23    NaN
24    NaN
Name: 051 [m/s], dtype: float64
```

(Refer Slide Time: 07:07)



```
plt.rcParams.update({"text.usetex":True});
%config InlineBackend.figure_format = "svg"
import pandas as pd

[2]: df = pd.read_csv('d-2-2.csv')

[5]: np.mean(df["051 [m/s]"])

[5]: 3.0043478260869567

[ ]:
```

(Refer Slide Time: 07:20)



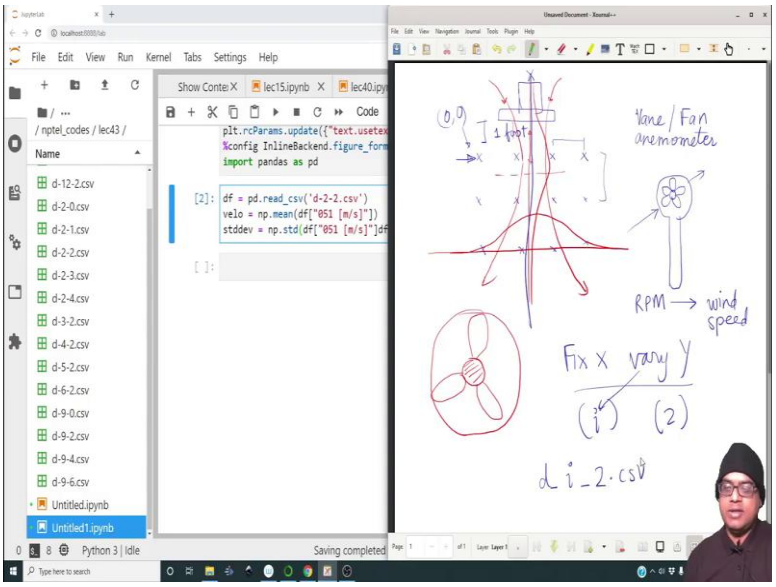
```
plt.rcParams.update({"text.usetex":True});
%config InlineBackend.figure_format = "svg"
import pandas as pd
```

```
[2]: df = pd.read_csv('d-2-2.csv')
     velo = np.mean(df["051 [m/s]"])
     stddev = np.std(df["051 [m/s]"]df["051 [m/s]"])
```

So, df this will give us the velocities and actually once we have this we can do np.mean to obtain the mean velocity. So, this gives us a clue on how to go about. So, let me sorry, let me do that, let me cut this, let me put it over here, let me delete this cell.

So now, we have read the file we found out the mean velocity alright. So, let me allot it to velo right. Now, let us also have the standard deviations stddev = np.std of this same thing so, it will give us the standard deviation as well alright.

(Refer Slide Time: 08:08)

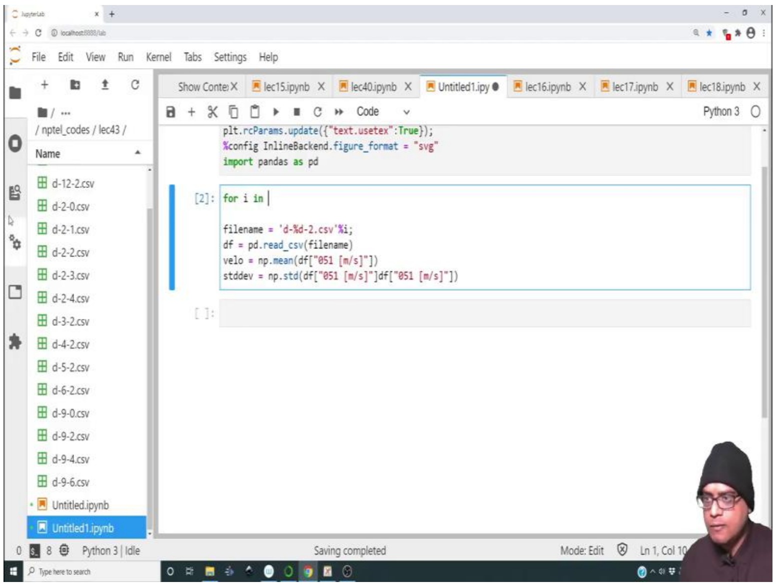


So, for this particular file ok let us do this.

So, for this particular file ok let us do this. Let us do this particular transact first this particular transact. So, let us fix up a value of x so we going to fix X and vary Y meaning, we are going to fix the 2nd index in this case we have a bunch of data files with the index 2 and we are going to vary the 1st index.

So, basically we are going to loop over that di,i_2.csv and stuff like that alright. So, instead of this we going to write file name and file name is going to be d-%d-2.csv;

(Refer Slide Time: 08:48)

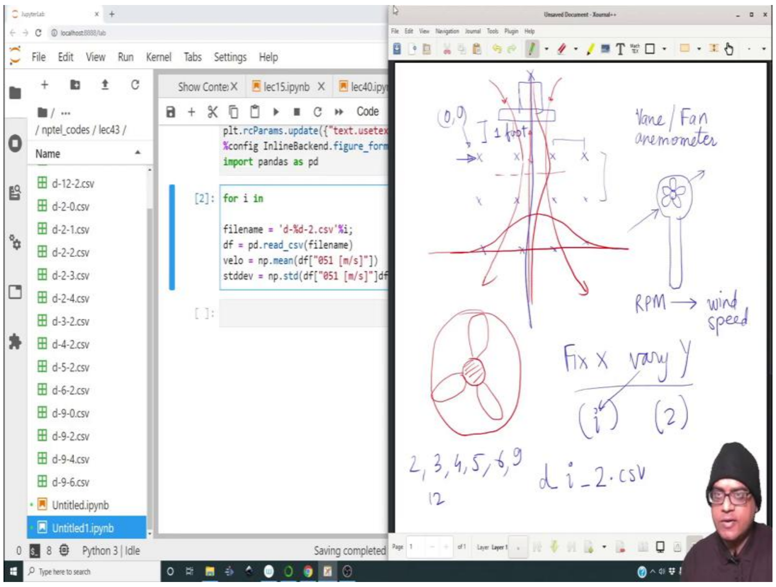


```
plt.rcParams.update({"text.usetex":True});
%config InlineBackend.figure_format = "svg"
import pandas as pd
```

```
[2]: for i in 

     filename = 'd-%d-2.csv'%i;
     df = pd.read_csv(filename)
     velo = np.mean(df["051 [m/s]"])
     stddev = np.std(df["051 [m/s]"]df["051 [m/s]"])
```

```
[ ]:
```

(Refer Slide Time: 09:43)

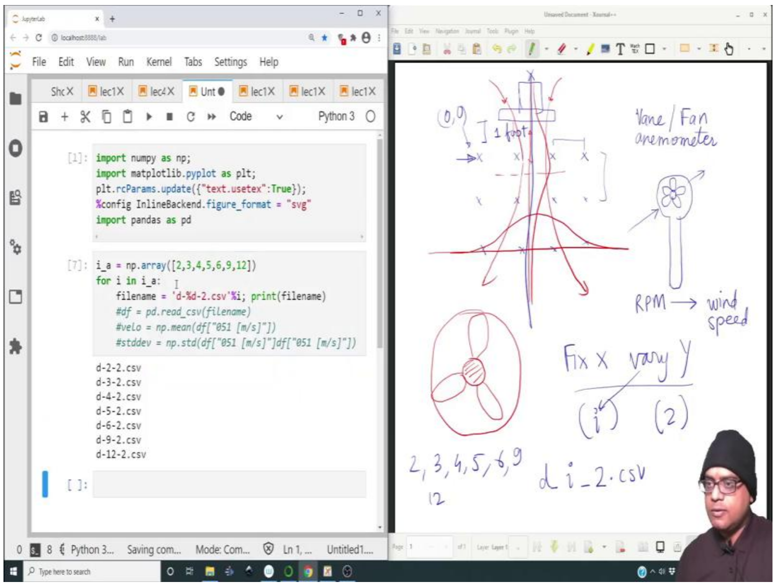


```
plt.rcParams.update({"text.usetex
%config InlineBackend.figure_form
import pandas as pd
```

```
[2]: for i in

     filename = 'd-%d-2.csv'%i;
     df = pd.read_csv(filename)
     velo = np.mean(df["051 [m/s]"])
     stddev = np.std(df["051 [m/s]"]df
```

And this is going to be the data file but we have to replace the %d over here with say let us let me substitute it with i ok. So, now we must perform a loop. So, for i in so, well let us see what different values we have. So, we have 12, 2 so we have 12 then 2, 3, 2 so, 2, 2, 3, 2, 4, 5, 6, 9 and 12.

(Refer Slide Time: 10:08)

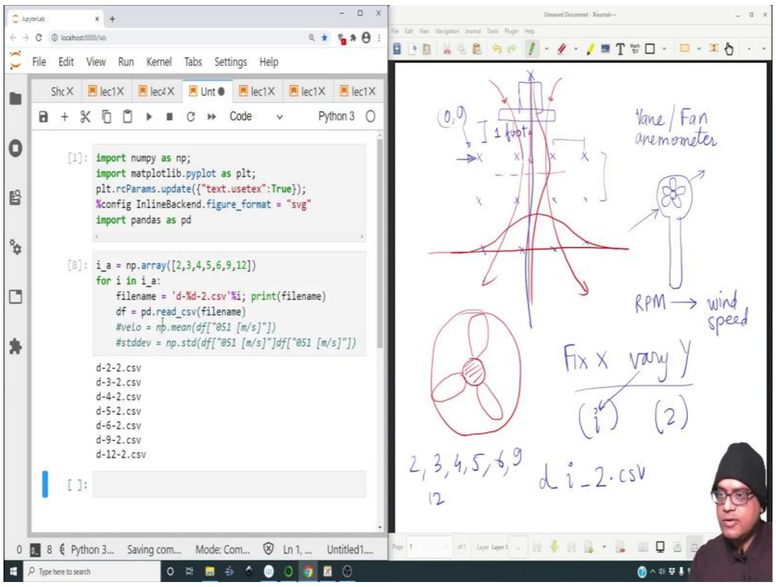


```
[1]: import numpy as np;
     import matplotlib.pyplot as plt;
     plt.rcParams.update({"text.usetex":True});
     %config InlineBackend.figure_format = "svg"
     import pandas as pd

[7]: i_a = np.array([2,3,4,5,6,9,12])
     for i in i_a:
         filename = 'd-%d-2.csv'%i; print(filename)
         #df = pd.read_csv(filename)
         #velo = np.mean(df["051 [m/s]"])
         #stddev = np.std(df["051 [m/s]"]df["051 [m/s]"])

     d-2-2.csv
     d-3-2.csv
     d-4-2.csv
     d-5-2.csv
     d-6-2.csv
     d-9-2.csv
     d-12-2.csv

[ ]:
```

So, let me create an array which contains these values. So, let me create an array so i = np.array(2, 3, 4, 5, 6, 9, 12). So, for in i array so then filename will be this. In fact, let me show you whether it is going to pick up the correct file names.

So, let me just print file name forgot this alright great. So, we are we are able to obtain all these file names. So now, we going to read all those so let us see if there is an error there is no error. So, we are able to load all the files into a data frame.

(Refer Slide Time: 10:47)

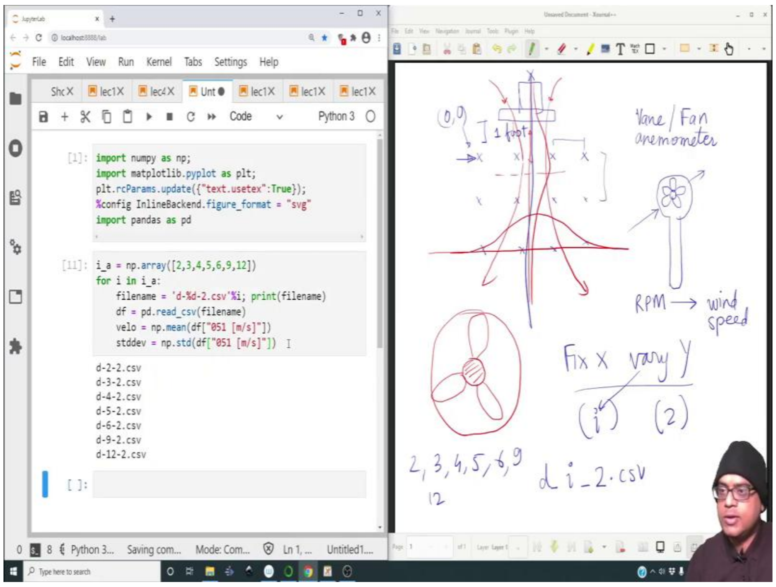


```
[1]: import numpy as np;
     import matplotlib.pyplot as plt;
     plt.rcParams.update({"text.usetex":True});
     %config InlineBackend.figure_format = "svg"
     import pandas as pd
```

```
[8]: i_a = np.array([2,3,4,5,6,9,12])
     for i in i_a:
         filename = 'd-%d-2.csv'%i; print(filename)
         df = pd.read_csv(filename)
         #velo = np.mean(df["051 [m/s]"])
         #stddev = np.std(df["051 [m/s]"]df["051 [m/s]"])
```

```
d-2-2.csv
d-3-2.csv
d-4-2.csv
d-5-2.csv
d-6-2.csv
d-9-2.csv
d-12-2.csv
```

(Refer Slide Time: 10:53)

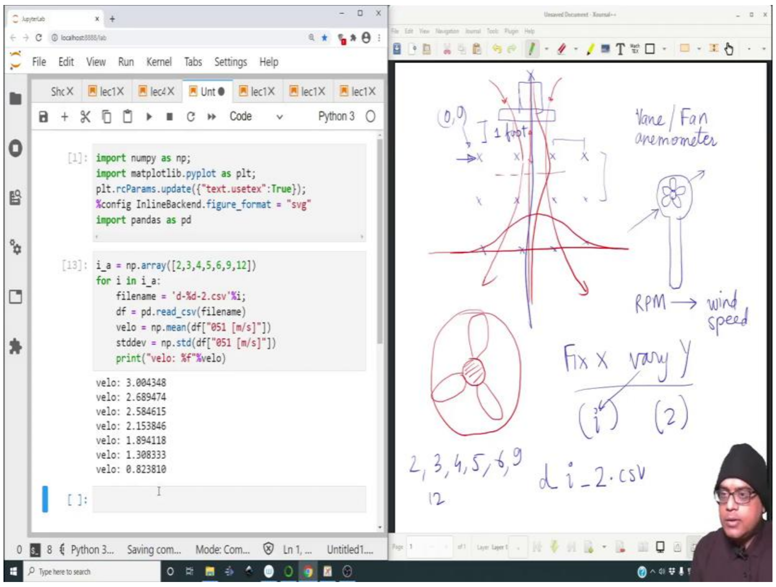


```
[1]: import numpy as np;
     import matplotlib.pyplot as plt;
     plt.rcParams.update({"text.usetex":True});
     %config InlineBackend.figure_format = "svg"
     import pandas as pd
```

```
[11]: i_a = np.array([2,3,4,5,6,9,12])
      for i in i_a:
          filename = 'd-%d-2.csv'%i; print(filename)
          df = pd.read_csv(filename)
          velo = np.mean(df["051 [m/s]"])
          stddev = np.std(df["051 [m/s]"])
```

```
d-2-2.csv
d-3-2.csv
d-4-2.csv
d-5-2.csv
d-6-2.csv
d-9-2.csv
d-12-2.csv
```

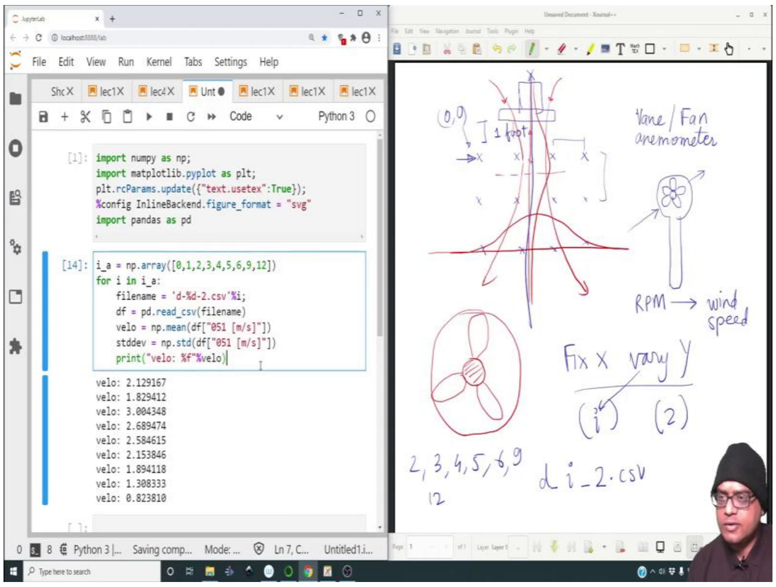
(Refer Slide Time: 11:11)



```
[1]: import numpy as np;
     import matplotlib.pyplot as plt;
     plt.rcParams.update({"text.usetex":True});
     %config InlineBackend.figure_format = "svg"
     import pandas as pd

[13]: i_a = np.array([2,3,4,5,6,9,12])
      for i in i_a:
          filename = 'd-%d-2.csv'%i;
          df = pd.read_csv(filename)
          velo = np.mean(df["051 [m/s]"])
          stddev = np.std(df["051 [m/s]"])
          print("velo: %f"%velo)

      velo: 3.004348
      velo: 2.689474
      velo: 2.584615
      velo: 2.153846
      velo: 1.894118
      velo: 1.308333
      velo: 0.823810

[ ]:
```

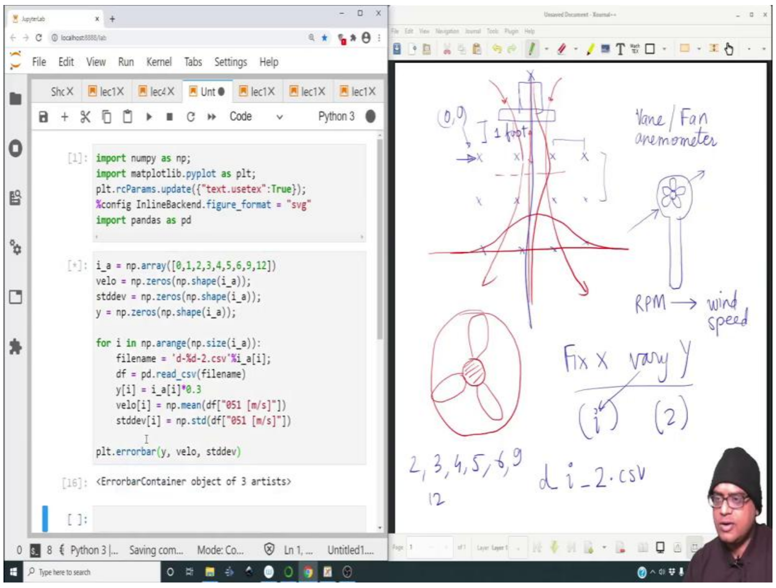
And yeah let us oops so, yeah something mismatched. So in fact, let us print all this print so, let me remove this print file name. So, instead let me print the velocity oops right. So, these are the velocities that we obtained ok. So, we did have another data set 0, 2 and 1, 2.

(Refer Slide Time: 12:07)



```
[1]: import numpy as np;
     import matplotlib.pyplot as plt;
     plt.rcParams.update({"text.usetex":True});
     %config InlineBackend.figure_format = "svg"
     import pandas as pd

[14]: i_a = np.array([0,1,2,3,4,5,6,9,12])
      for i in i_a:
          filename = 'd-%d-2.csv'%i;
          df = pd.read_csv(filename)
          velo = np.mean(df["051 [m/s]"])
          stddev = np.std(df["051 [m/s]"])
          print("velo: %f"%velo)

      velo: 2.129167
      velo: 1.829412
      velo: 3.004348
      velo: 2.689474
      velo: 2.584615
      velo: 2.153846
      velo: 1.894118
      velo: 1.308333
      velo: 0.823810
```

(Refer Slide Time: 12:34)



```
[1]: import numpy as np;
     import matplotlib.pyplot as plt;
     plt.rcParams.update({"text.usetex":True});
     %config InlineBackend.figure_format = "svg"
     import pandas as pd
```

```
[*]: i_a = np.array([0,1,2,3,4,5,6,9,12])
     velo = np.zeros(np.shape(i_a));
     stddev = np.zeros(np.shape(i_a));
     y = np.zeros(np.shape(i_a));

     for i in np.arange(np.size(i_a)):
         filename = 'd-%d-2.csv'%i_a[i];
         df = pd.read_csv(filename)
         y[i] = i_a[i]*0.3
         velo[i] = np.mean(df["051 [m/s]"])
         stddev[i] = np.std(df["051 [m/s]"])

     plt.errorbar(y, velo, stddev)
```

```
[16]: <ErrorbarContainer object of 3 artists>
```

So, let me add those indices over here alright. So, let me run this this is the velocity set that we have and let us add all of this to a variable ok. So, corresponding to this i_a. Let me declare some additional arrays. So, let me declare velo as an array so, np**.**zeros(np**.**shape(i_a)).

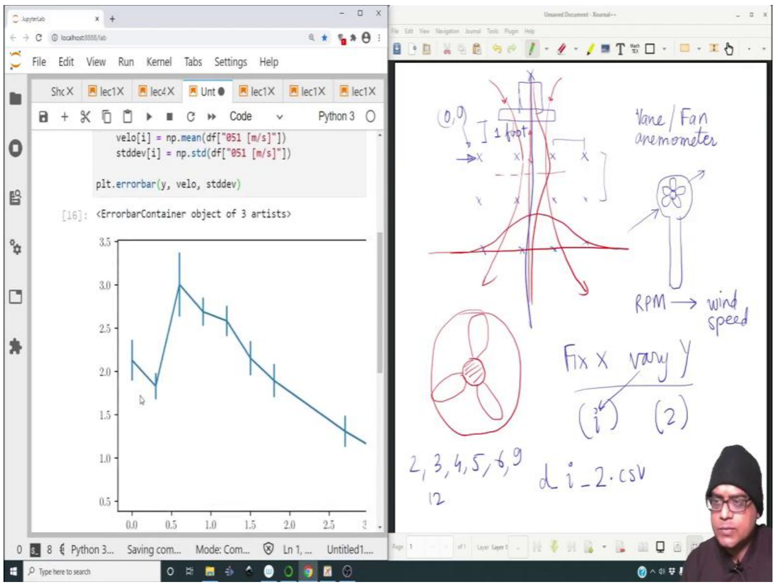
Let me copy this paste so this will be for the standard deviation, let me make another this will be for the y locations because, you also want to plot the y locations. So, what will be the y location? So, like I said the y locations are actually, the y locations are actually the index multiplied by 1 foot.

So, 1 foot is approximately 30 centimeters. So, let me do this so y[i] = [i]*****0.3. Actually I cannot do y[i] because I will actually have these indices. So, what I have to do is i **in** np**.**arange(np**.**size(i_a)) and this file name has to be then i_a[i].

So, this will make it loop normally and this has to be i_a[i]. So, just think about what I have done, I have not done something very complicated instead of using i directly from the array I am using i as a loop counter nothing else there are multiple ways of getting this done.

So, do not think this is the only way. Now, because I have defined velo as an array, I can define this and I can define this as well. So, let me suppress this printing let me run this great there is no error. So now, what do we have? We have a bunch of y's and we have the corresponding velocities and standard deviations.

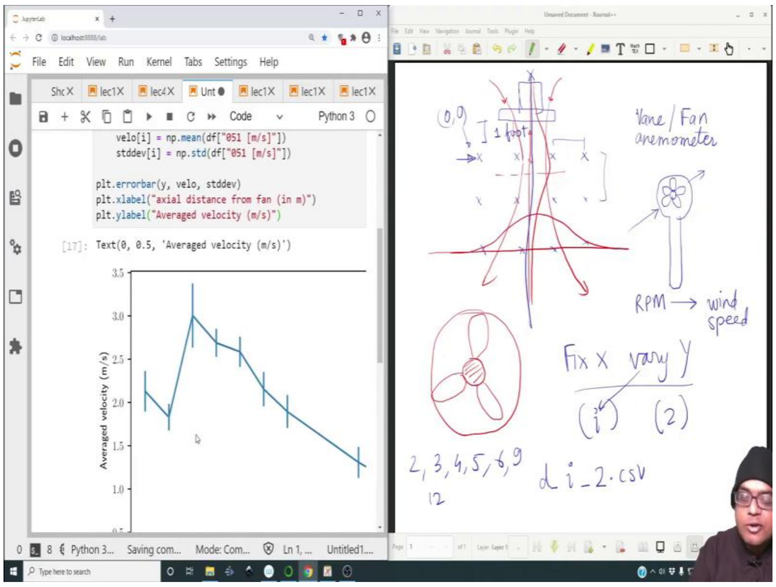
(Refer Slide Time: 15:00)



```
    velo[i] = np.mean(df["051 [m/s]"])
    stddev[i] = np.std(df["051 [m/s]"])

plt.errorbar(y, velo, stddev)
```

```
[16]: <ErrorbarContainer object of 3 artists>
```

(Refer Slide Time: 15:05)



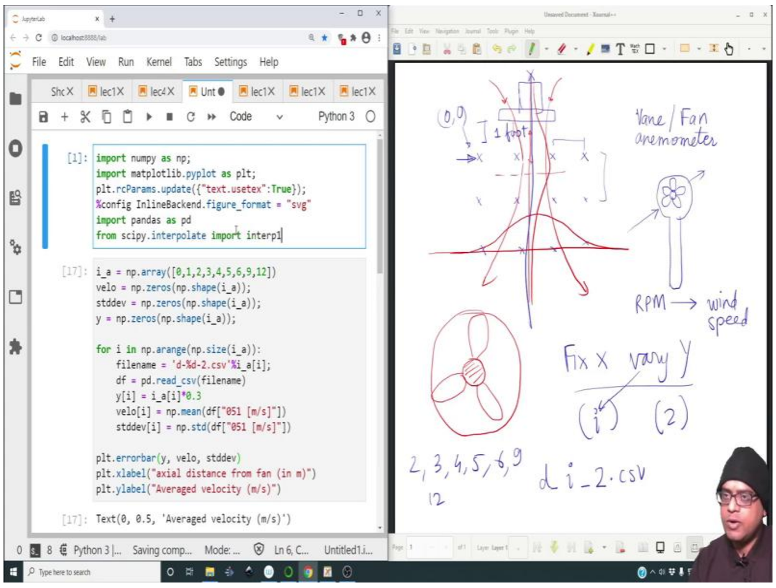
```
    velo[i] = np.mean(df["051 [m/s]"])
    stddev[i] = np.std(df["051 [m/s]"])

plt.errorbar(y, velo, stddev)
plt.xlabel("axial distance from fan (in m)")
plt.ylabel("Averaged velocity (m/s)")
```

```
[17]: Text(0, 0.5, 'Averaged velocity (m/s)')
```

So, now what we can do is we can plot the error. So, we are going to do plt.errorbar(y, velo, stddev). So, let us see great. So, this is what the plot we have. In fact, let me put plt.xlabel("axial distance from fan (in m)"), plt.ylabel("Averaged velocity (m/s)"), that is great.

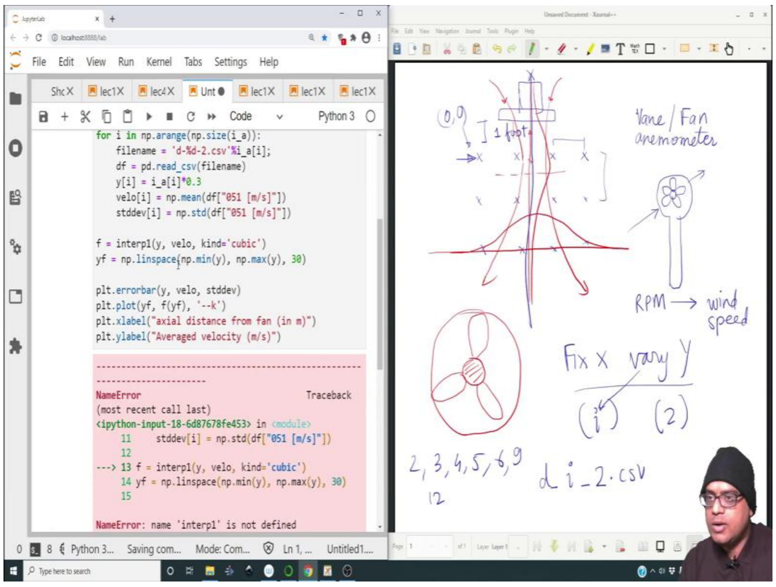
Hm, but because this is scarcity of data I have not taken data at a very large number of points, usually that is the case you do not have so much resource to do that.

(Refer Slide Time: 16:01)



```
[1]: import numpy as np;
     import matplotlib.pyplot as plt;
     plt.rcParams.update({"text.usetex":True});
     %config InlineBackend.figure_format = "svg"
     import pandas as pd
     from scipy.interpolate import interp1

[17]: i_a = np.array([0,1,2,3,4,5,6,9,12])
      velo = np.zeros(np.shape(i_a));
      stddev = np.zeros(np.shape(i_a));
      y = np.zeros(np.shape(i_a));

      for i in np.arange(np.size(i_a)):
          filename = 'd-%d-2.csv'%i_a[i];
          df = pd.read_csv(filename)
          y[i] = i_a[i]*0.3
          velo[i] = np.mean(df["051 [m/s]"])
          stddev[i] = np.std(df["051 [m/s]"])

      plt.errorbar(y, velo, stddev)
      plt.xlabel("axial distance from fan (in m)")
      plt.ylabel("Averaged velocity (m/s)")

[17]: Text(0, 0.5, 'Averaged velocity (m/s)')
```

(Refer Slide Time: 16:20)



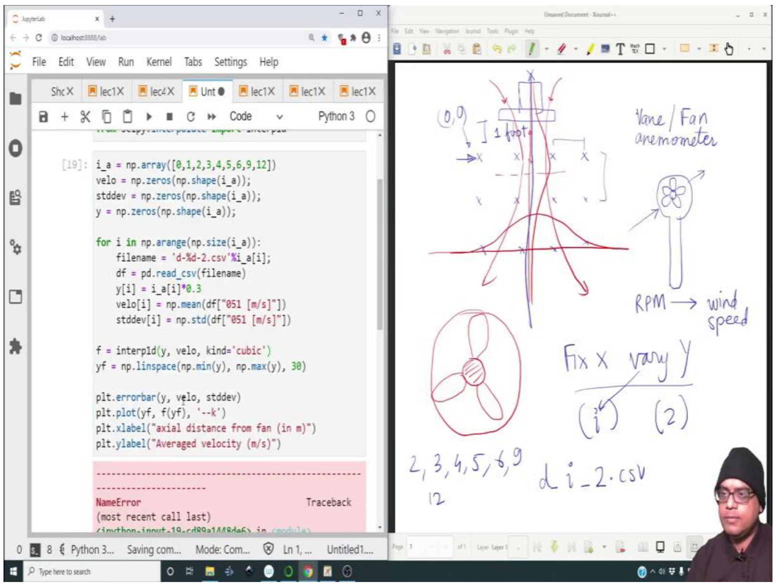
```
for i in np.arange(np.size(i_a)):
    filename = 'd-%d-2.csv'%i_a[i];
    df = pd.read_csv(filename)
    y[i] = i_a[i]*0.3
    velo[i] = np.mean(df["051 [m/s]"])
    stddev[i] = np.std(df["051 [m/s]"])

f = interp1(y, velo, kind='cubic')
yf = np.linspace(np.min(y), np.max(y), 30)

plt.errorbar(y, velo, stddev)
plt.plot(yf, f(yf), '--k')
plt.xlabel("axial distance from fan (in m)")
plt.ylabel("Averaged velocity (m/s)")
```

```
---------------------------------------------------------------------
---------------------
NameError                                 Traceback
(most recent call last)
<ipython-input-18-6d87678fe453> in <module>
     11     stddev[i] = np.std(df["051 [m/s]"])
     12
---> 13 f = interp1(y, velo, kind='cubic')
     14 yf = np.linspace(np.min(y), np.max(y), 30)
     15

NameError: name 'interp1' is not defined
```

So, let me interpolate the data so that we can make some kind of a cubic approximation of the data. So, we have already done this before. So, let me import scipy or rather from scipy.interpolate **import** interp1d that is 1 dimensional interpolation. Let me now define f = interp1d(y, velo, kind='cubic') and let me declare another yf = np**.**linspace(np**.**min(y), np**.**max(y), 30). Let me take 30 points.

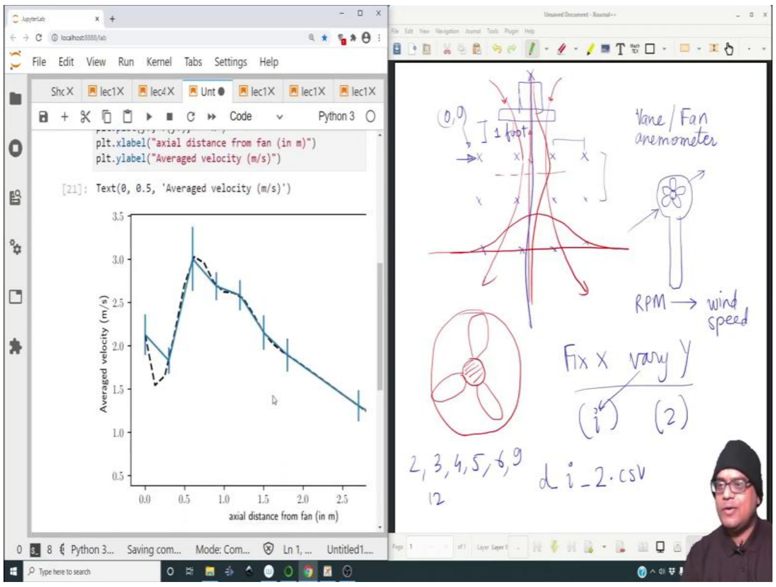
So, basically I am just taking these discrete points, whatever these points are 00.3 and so on. I am creating a new array which has more number of points in them. So, yf is this yf stands for y fine. So, on top of this I am going to plt.plot(yf, f(yf), '--k') and let me put it at a dotted line of black color. So, let me put this this should be interp1d ok.

(Refer Slide Time: 17:29)



```
[19]: i_a = np.array([0,1,2,3,4,5,6,9,12])
      velo = np.zeros(np.shape(i_a));
      stddev = np.zeros(np.shape(i_a));
      y = np.zeros(np.shape(i_a));

      for i in np.arange(np.size(i_a)):
          filename = 'd-%d-2.csv'%i_a[i];
          df = pd.read_csv(filename)
          y[i] = i_a[i]*0.3
          velo[i] = np.mean(df["051 [m/s]"])
          stddev[i] = np.std(df["051 [m/s]"])

      f = interp1d(y, velo, kind='cubic')
      yf = np.linspace(np.min(y), np.max(y), 30)

      plt.errorbar(y, velo, stddev)
      plt.plot(yf, f(yf), '--k')
      plt.xlabel("axial distance from fan (in m)")
      plt.ylabel("Averaged velocity (m/s)")

      ---------------------------------------------------------------------------
      NameError                                 Traceback
      (most recent call last)
```

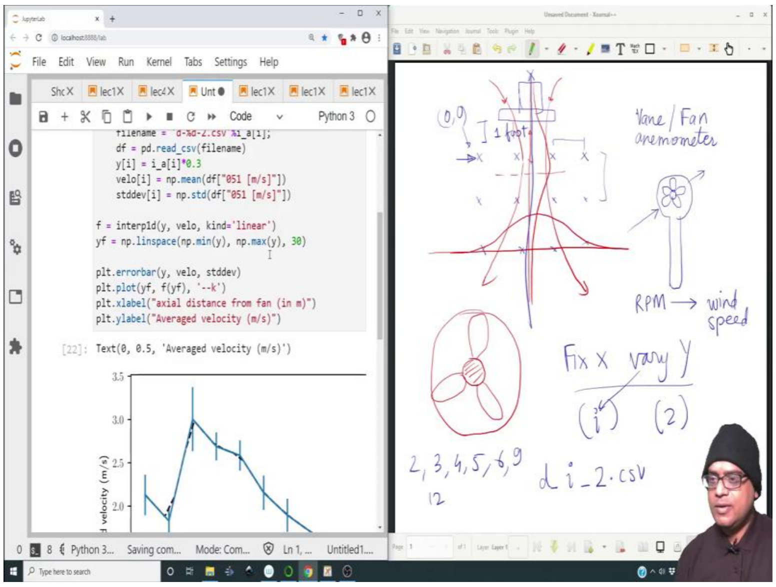
(Refer Slide Time: 17:38)



```
      plt.xlabel("axial distance from fan (in m)")
      plt.ylabel("Averaged velocity (m/s)")

[21]: Text(0, 0.5, 'Averaged velocity (m/s)')
```

There is still an error interp1d I have not yet run this cell it is a classic mistake alright.

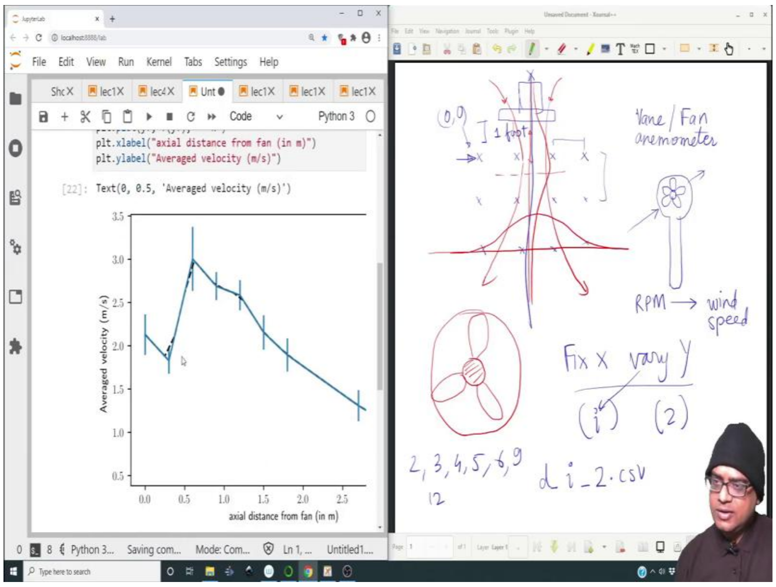
(Refer Slide Time: 17:47)



```
    filename = 'd-%d-2.csv'%i_a[i];
    df = pd.read_csv(filename)
    y[i] = i_a[i]*0.3
    velo[i] = np.mean(df["051 [m/s]"])
    stddev[i] = np.std(df["051 [m/s]"])

f = interp1d(y, velo, kind='linear')
yf = np.linspace(np.min(y), np.max(y), 30)

plt.errorbar(y, velo, stddev)
plt.plot(yf, f(yf), '--k')
plt.xlabel("axial distance from fan (in m)")
plt.ylabel("Averaged velocity (m/s)")
```

```
[22]: Text(0, 0.5, 'Averaged velocity (m/s)')
```

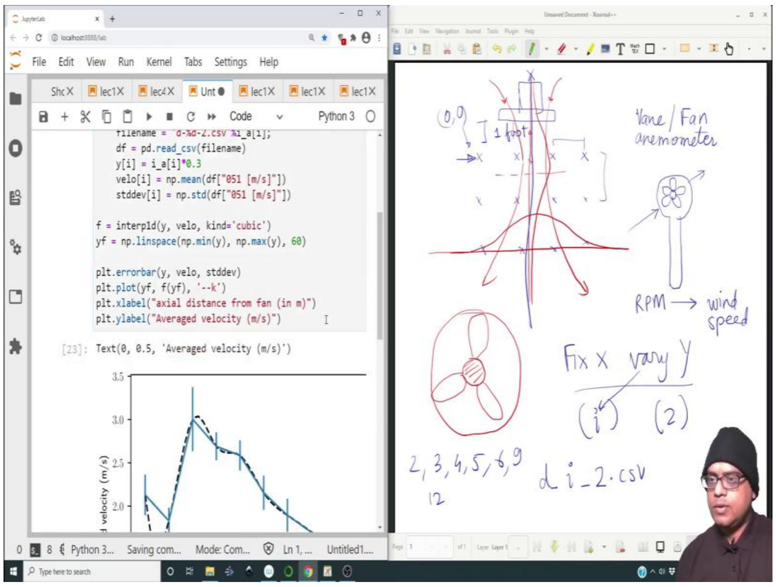
(Refer Slide Time: 17:50)



```
plt.xlabel("axial distance from fan (in m)")
plt.ylabel("Averaged velocity (m/s)")
```

```
[22]: Text(0, 0.5, 'Averaged velocity (m/s)')
```

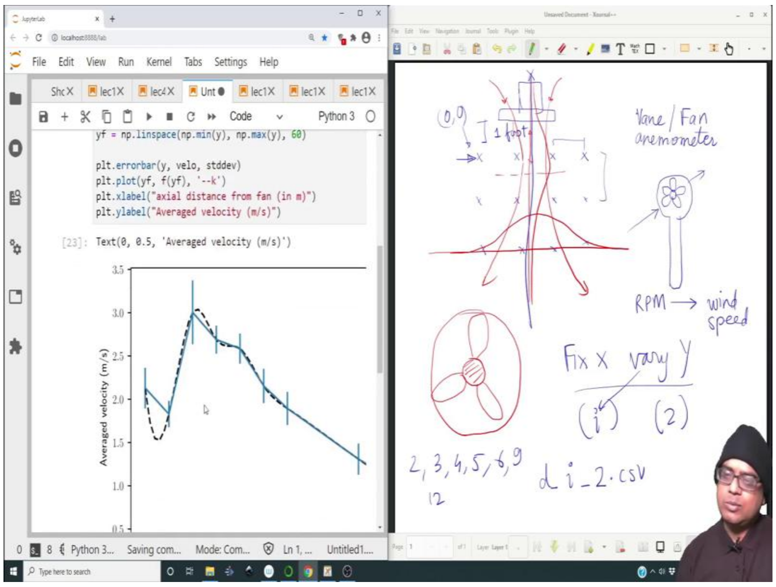
(Refer Slide Time: 17:54)



```
        filename = "d-%d-2.csv" %i_a[i];
        df = pd.read_csv(filename)
        y[i] = i_a[i]*0.3
        velo[i] = np.mean(df["051 [m/s]"])
        stddev[i] = np.std(df["051 [m/s]"])

    f = interp1d(y, velo, kind='cubic')
    yf = np.linspace(np.min(y), np.max(y), 60)

    plt.errorbar(y, velo, stddev)
    plt.plot(yf, f(yf), '--k')
    plt.xlabel("axial distance from fan (in m)")
    plt.ylabel("Averaged velocity (m/s)")
```

```
[23]: Text(0, 0.5, 'Averaged velocity (m/s)')
```

(Refer Slide Time: 17:57)

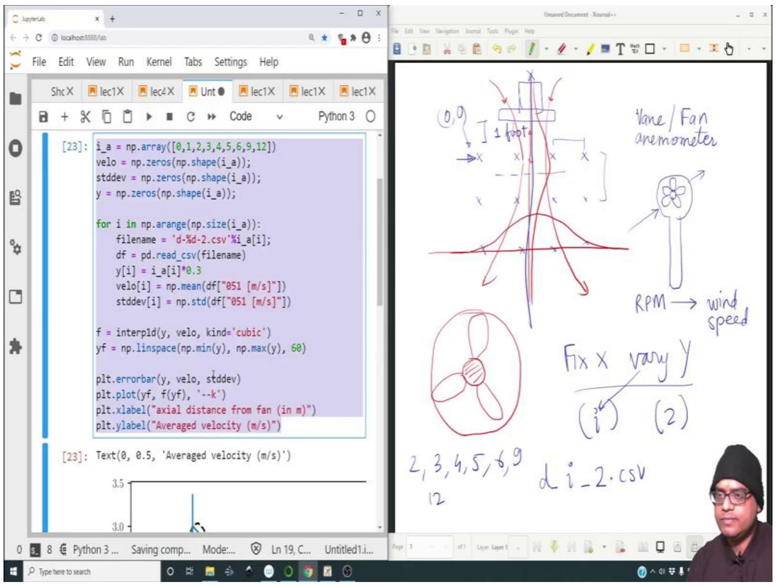


```
    yf = np.linspace(np.min(y), np.max(y), 60)

    plt.errorbar(y, velo, stddev)
    plt.plot(yf, f(yf), '--k')
    plt.xlabel("axial distance from fan (in m)")
    plt.ylabel("Averaged velocity (m/s)")
```

```
[23]: Text(0, 0.5, 'Averaged velocity (m/s)')
```

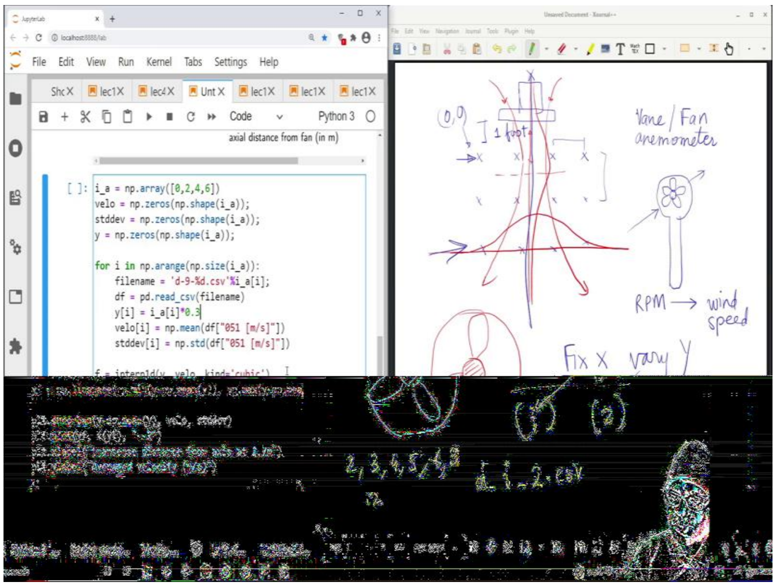
There is still an error interp1d I have not yet run this cell it is a classic mistake alright. So, this is how the velocity looks like once it is interpolated. You can do various kinds of interpolations linear this is a linear interpolation. And you can choose large number of points ok, it looks something like this.

(Refer Slide Time: 18:15)



```
[23]: i_a = np.array([0,1,2,3,4,5,6,9,12])
      velo = np.zeros(np.shape(i_a));
      stddev = np.zeros(np.shape(i_a));
      y = np.zeros(np.shape(i_a));

      for i in np.arange(np.size(i_a)):
          filename = 'd-%d-2.csv'%i_a[i];
          df = pd.read_csv(filename)
          y[i] = i_a[i]*0.3
          velo[i] = np.mean(df["051 [m/s]"])
          stddev[i] = np.std(df["051 [m/s]"])

      f = interp1d(y, velo, kind='cubic')
      yf = np.linspace(np.min(y), np.max(y), 60)

      plt.errorbar(y, velo, stddev)
      plt.plot(yf, f(yf), '--k')
      plt.xlabel("axial distance from fan (in m)")
      plt.ylabel("Averaged velocity (m/s)")

[23]: Text(0, 0.5, 'Averaged velocity (m/s)')
```

(Refer Slide Time: 18:18)



```
[ ]: i_a = np.array([0,2,4,6])
     velo = np.zeros(np.shape(i_a));
     stddev = np.zeros(np.shape(i_a));
     y = np.zeros(np.shape(i_a));

     for i in np.arange(np.size(i_a)):
         filename = 'd-9-%d.csv'%i_a[i];
         df = pd.read_csv(filename)
         y[i] = i_a[i]*0.3
         velo[i] = np.mean(df["051 [m/s]"])
         stddev[i] = np.std(df["051 [m/s]"])

     f = interp1d(y, velo, kind='cubic')
```

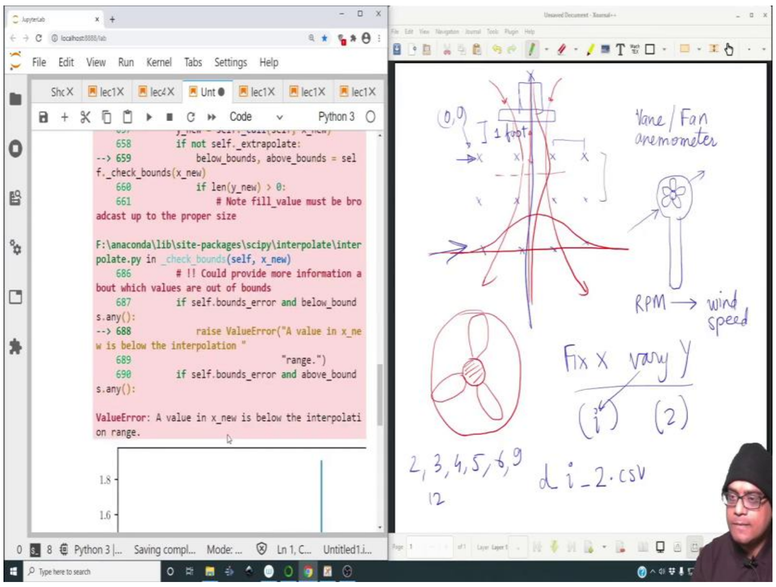
So, great so, so far we have done this transit and similarly you can do this transect as well. So, let us see how that can be done. So, what are the points that we have? If I remember correctly for the for transect I did take some points this 90 92 94 96 alright.

So, the 1st index is fixed meaning y is fixed and x is varying ok. So, the 1st index is fixed meaning this y is fixed the row number is fixed. So, what I am going to do is modify this so that it contains 0, 2, 4 and 6 and modify this.

So, that it is 9-%d and in fact, it is very easy to reuse that above code. Instead of the level being actual distance we will say. So, we are going to say transverse distance from axis and in fact, when we do the transverse distance from the axis we are going to take y minus mean of y. So, that we can show that the y = 0 has the maximum value. And similarly we are going to do – np.mean(yf) alright.

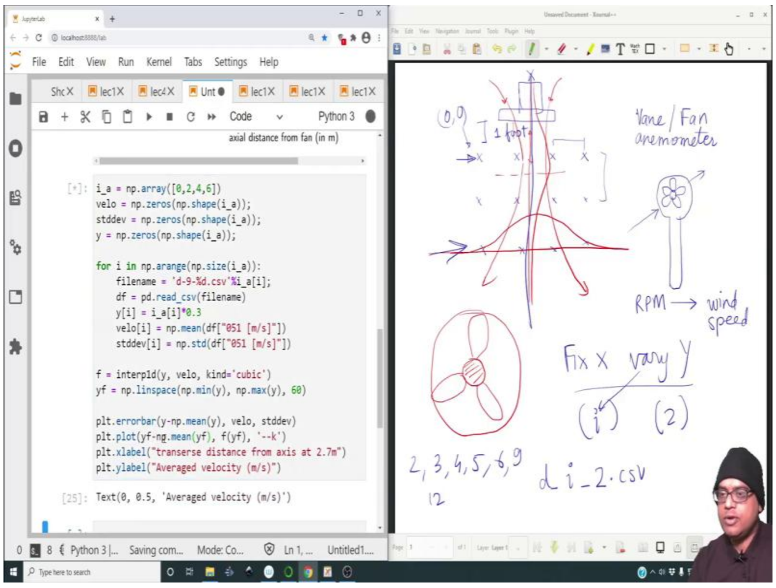
So, once I do this actually I have to feed so, yf can remain as it is we can simply modify this ok, at 2.7 meter because it is all done at a y location of 9 other the row number 9 so, 9 corresponds to a distance of 9 times 0.3 which is 2.7. So, let me run this there is an error let us see ok.

(Refer Slide Time: 20:48)



```
    658         if not self._extrapolate:
--> 659             below_bounds, above_bounds = sel
f._check_bounds(x_new)
    660             if len(y_new) > 0:
    661                 # Note fill_value must be bro
adcast up to the proper size

F:\anaconda\lib\site-packages\scipy\interpolate\inter
polate.py in _check_bounds(self, x_new)
    686         # !! Could provide more information a
bout which values are out of bounds
    687         if self.bounds_error and below_bound
s.any():
--> 688             raise ValueError("A value in x_ne
w is below the interpolation "
    689                              "range.")
    690         if self.bounds_error and above_bound
s.any():

ValueError: A value in x_new is below the interpolati
on range.
```

(Refer Slide Time: 21:16)

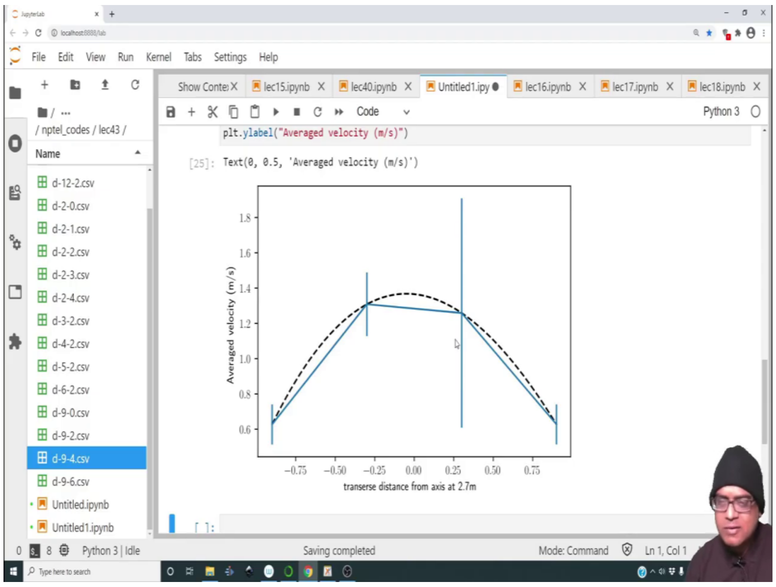


axial distance from fan (in m)

```
[*]: i_a = np.array([0,2,4,6])
     velo = np.zeros(np.shape(i_a));
     stddev = np.zeros(np.shape(i_a));
     y = np.zeros(np.shape(i_a));

     for i in np.arange(np.size(i_a)):
         filename = 'd-9-%d.csv'%i_a[i];
         df = pd.read_csv(filename)
         y[i] = i_a[i]*0.3
         velo[i] = np.mean(df["051 [m/s]"])
         stddev[i] = np.std(df["051 [m/s]"])

     f = interp1d(y, velo, kind='cubic')
     yf = np.linspace(np.min(y), np.max(y), 60)

     plt.errorbar(y-np.mean(y), velo, stddev)
     plt.plot(yf-np.mean(yf), f(yf), '--k')
     plt.xlabel("transerse distance from axis at 2.7m")
     plt.ylabel("Averaged velocity (m/s)")

[25]: Text(0, 0.5, 'Averaged velocity (m/s)')
```

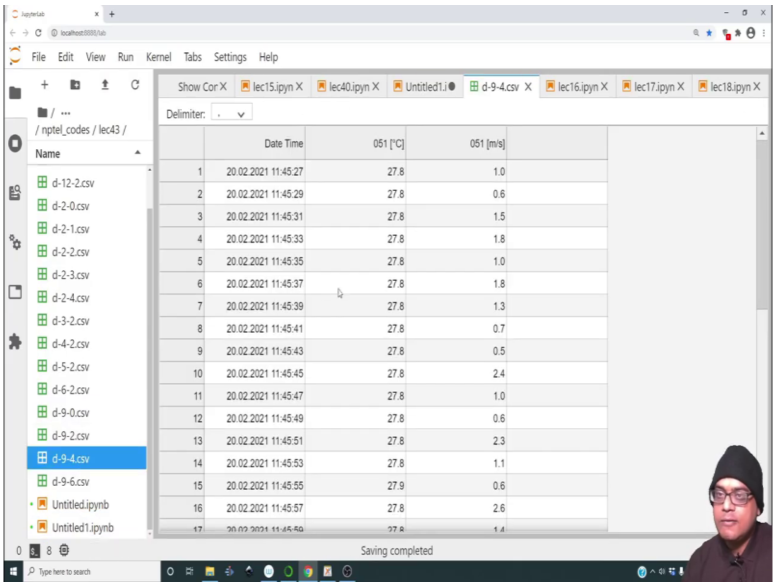
So, the interpolation goes out of bound because, we did an interpolation over here so, in fact let me keep things as they are. And I mean it is not that difficult to figure out at all, but yeah I mean you can try to do you know what I intended to do so, you can try to do it. We can plot the shifted plots plotting it should not be an issue let us see great.

(Refer Slide Time: 21:41)



```
plt.ylabel("Averaged velocity (m/s)")

[25]: Text(0, 0.5, 'Averaged velocity (m/s)')
```

(Refer Slide Time: 21:56)



/ nptel_codes / lec43 /

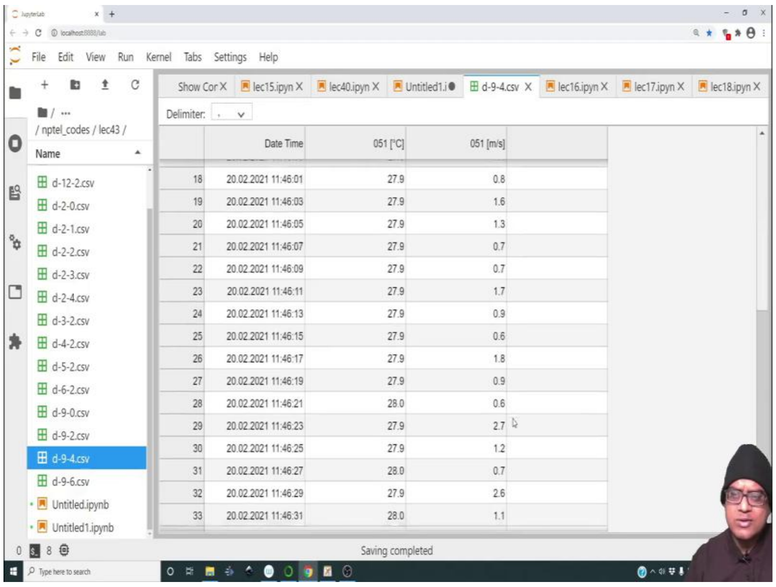
d-12-2.csv
d-2-0.csv
d-2-1.csv
d-2-2.csv
d-2-3.csv
d-2-4.csv
d-3-2.csv
d-4-2.csv
d-5-2.csv
d-6-2.csv
d-9-0.csv
d-9-2.csv
d-9-4.csv
d-9-6.csv
Untitled.ipynb
Untitled1.ipynb

Delimiter: ,

	Date Time	051 [°C]	051 [m/s]
1	20.02.2021 11:45:27	27.8	1.0
2	20.02.2021 11:45:29	27.8	0.6
3	20.02.2021 11:45:31	27.8	1.5
4	20.02.2021 11:45:33	27.8	1.8
5	20.02.2021 11:45:35	27.8	1.0
6	20.02.2021 11:45:37	27.8	1.8
7	20.02.2021 11:45:39	27.8	1.3
8	20.02.2021 11:45:41	27.8	0.7
9	20.02.2021 11:45:43	27.8	0.5
10	20.02.2021 11:45:45	27.8	2.4
11	20.02.2021 11:45:47	27.8	1.0
12	20.02.2021 11:45:49	27.8	0.6
13	20.02.2021 11:45:51	27.8	2.3
14	20.02.2021 11:45:53	27.8	1.1
15	20.02.2021 11:45:55	27.9	0.6
16	20.02.2021 11:45:57	27.8	2.6
17	20.02.2021 11:45:59	27.8	1.4

Saving completed

(Refer Slide Time: 21:57)



/ nptel_codes / lec43 /

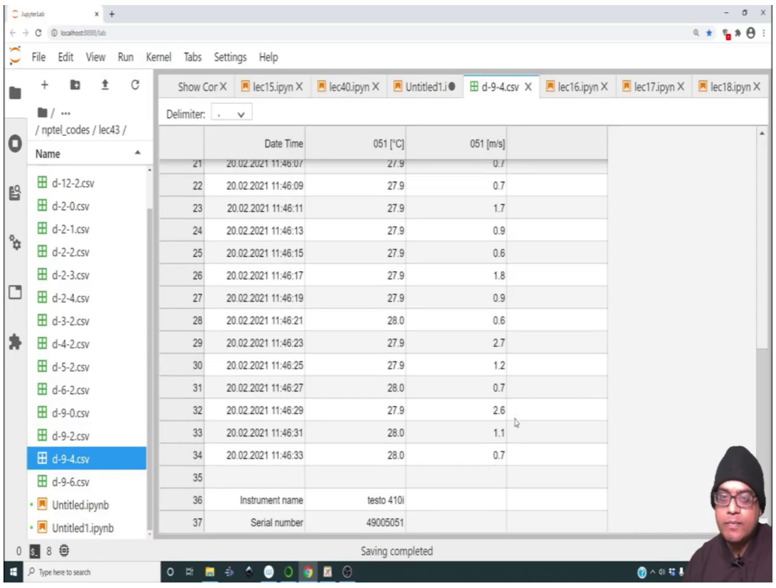
d-12-2.csv
d-2-0.csv
d-2-1.csv
d-2-2.csv
d-2-3.csv
d-2-4.csv
d-3-2.csv
d-4-2.csv
d-5-2.csv
d-6-2.csv
d-9-0.csv
d-9-2.csv
d-9-4.csv
d-9-6.csv
Untitled.ipynb
Untitled1.ipynb

Delimiter: ,

	Date Time	051 [°C]	051 [m/s]
18	20.02.2021 11:46:01	27.9	0.8
19	20.02.2021 11:46:03	27.9	1.6
20	20.02.2021 11:46:05	27.9	1.3
21	20.02.2021 11:46:07	27.9	0.7
22	20.02.2021 11:46:09	27.9	0.7
23	20.02.2021 11:46:11	27.9	1.7
24	20.02.2021 11:46:13	27.9	0.9
25	20.02.2021 11:46:15	27.9	0.6
26	20.02.2021 11:46:17	27.9	1.8
27	20.02.2021 11:46:19	27.9	0.9
28	20.02.2021 11:46:21	28.0	0.6
29	20.02.2021 11:46:23	27.9	2.7
30	20.02.2021 11:46:25	27.9	1.2
31	20.02.2021 11:46:27	28.0	0.7
32	20.02.2021 11:46:29	27.9	2.6
33	20.02.2021 11:46:31	28.0	1.1

Saving completed

(Refer Slide Time: 21:58)



Delimiter: ,

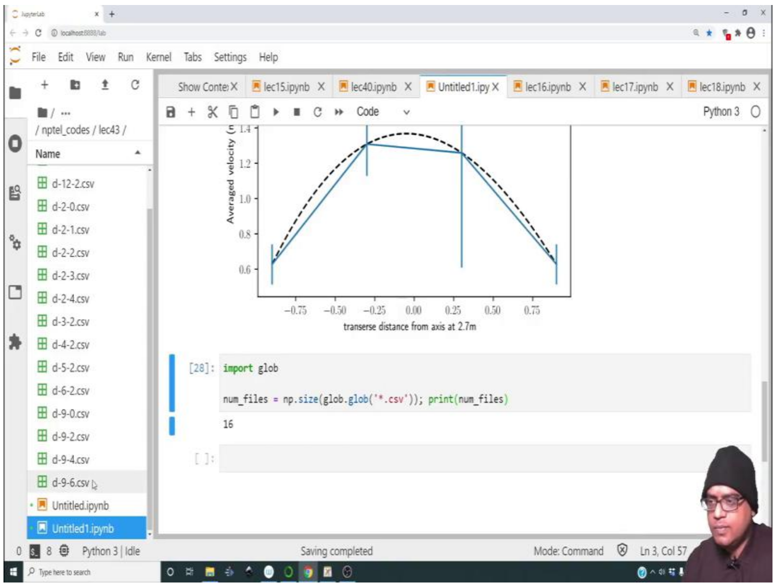
	Date Time	051 [°C]	051 [m/s]
21	20.02.2021 11:46:07	27.9	0.7
22	20.02.2021 11:46:09	27.9	0.7
23	20.02.2021 11:46:11	27.9	1.7
24	20.02.2021 11:46:13	27.9	0.9
25	20.02.2021 11:46:15	27.9	0.6
26	20.02.2021 11:46:17	27.9	1.8
27	20.02.2021 11:46:19	27.9	0.9
28	20.02.2021 11:46:21	28.0	0.6
29	20.02.2021 11:46:23	27.9	2.7
30	20.02.2021 11:46:25	27.9	1.2
31	20.02.2021 11:46:27	28.0	0.7
32	20.02.2021 11:46:29	27.9	2.6
33	20.02.2021 11:46:31	28.0	1.1
34	20.02.2021 11:46:33	28.0	0.7
35			
36	Instrument name	testo 410i	
37	Serial number	49005051	

So, what do we see? So, we need to see a maxima somewhere near the origin and there is a big error bar over here, that would be to less number of points. So, this would be 9, 3 let us open up 9 4 ok.

So, this this data variation from 0.6 all the way to 2.4 so, that does seem to be some error ideally I would have taken a lot of points and then done the averaging. So, that is would that would have led to smaller error bars that is ok. Ok, so far we have seen how to do all this but.

But we have a bunch of files and the indexing of the files does indicate the x and y of that particular data. So, there has to be some way of easily extracting the x and y from the file name it is self, looping over all the csv files and then making a 2 dimensional plot of the velocity field. Given that we have only a velocity field at a few scattered points, but still it should be possible. So, let us see how we can get that done. So, first of all in order to list all the files in this folder.

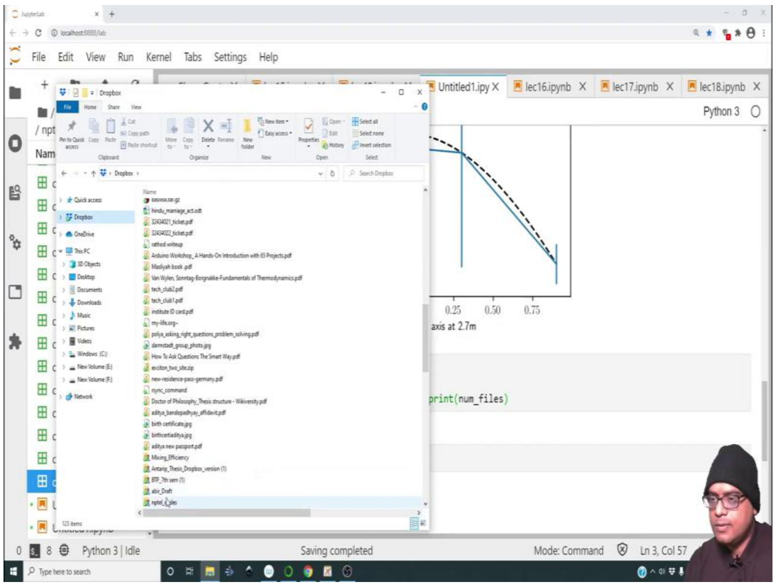
(Refer Slide Time: 22:57)



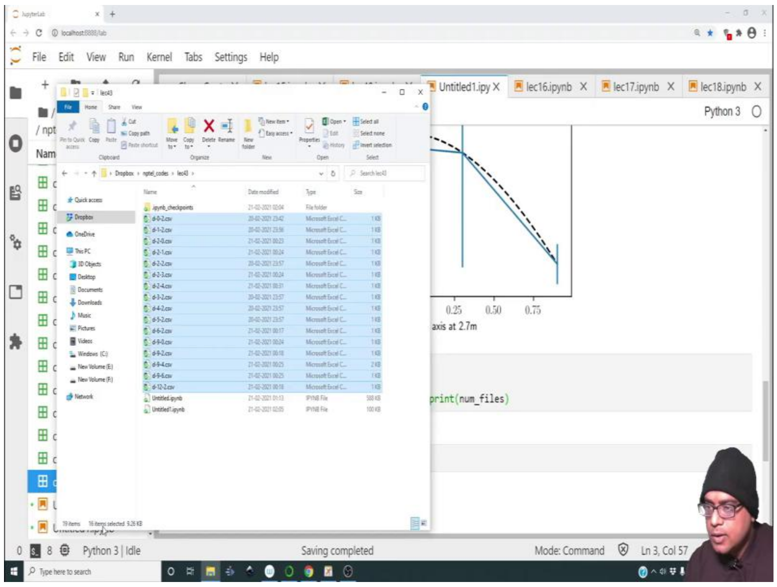
```
[28]: import glob

      num_files = np.size(glob.glob('*.csv')); print(num_files)

      16
```

We must import glob ok. So, let me run this. So, let us check the number of files so, num_files = np.size(glob.glob('*.csv')). Let me print the number of files. So, it says 16. So, it is 1. So, let me see whether they are indeed 16 they are 16 ok.

(Refer Slide Time: 23:31)

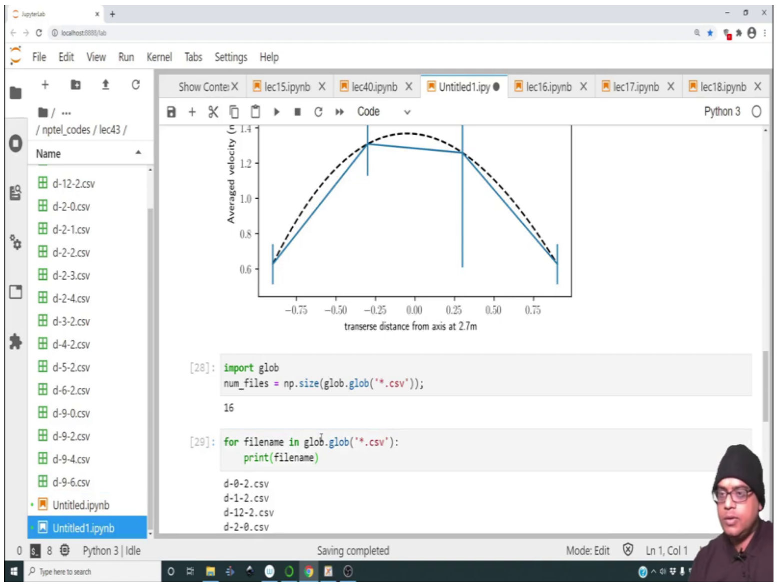


(Refer Slide Time: 23:40)



```
print(num_files)
```

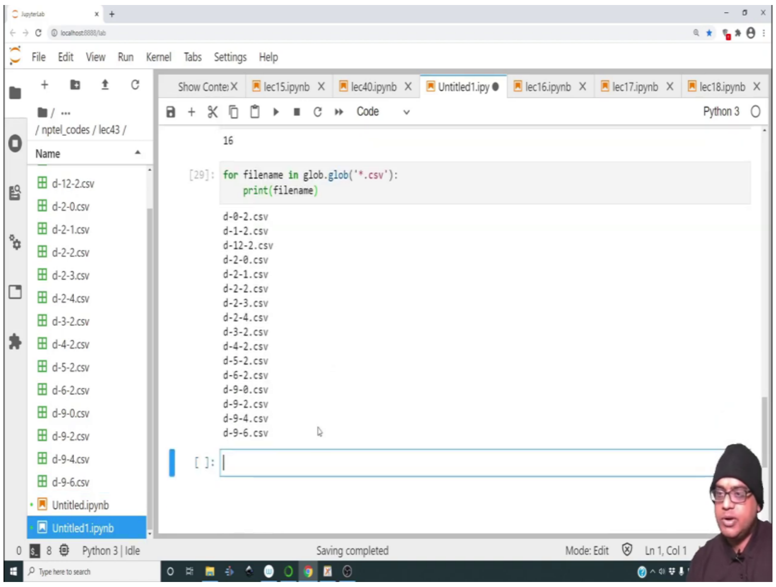
(Refer Slide Time: 23:49)



```
[28]: import glob
      num_files = np.size(glob.glob('*.csv'));

      16

[29]: for filename in glob.glob('*.csv'):
          print(filename)

      d-0-2.csv
      d-1-2.csv
      d-12-2.csv
      d-2-0.csv
```

So, we have 16 files of the current csv this is how you can get all the files. So, let me remove this ok. So, what do we do with this ok so, **for** filename **in** glob.glob('*.csv') print file name. So, it prints out all the csv files ok. So this is also good, we can move this to the main cell let me delete this alright.

(Refer Slide Time: 24:20)

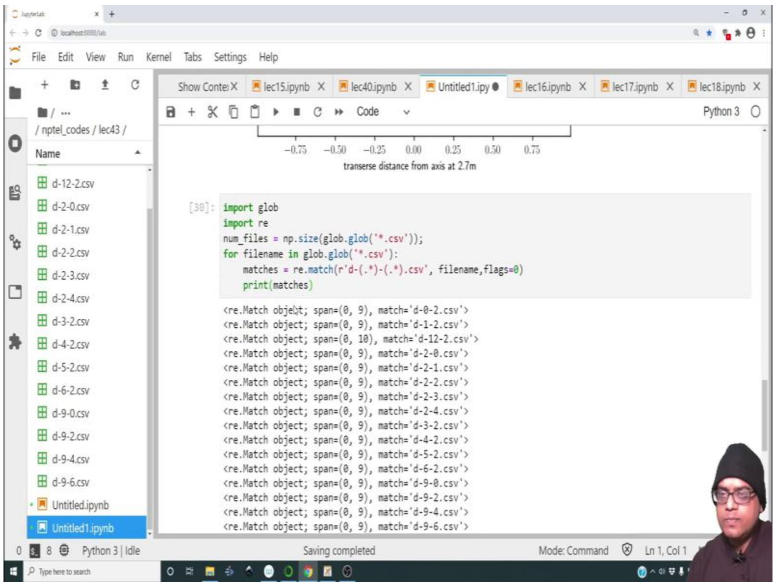


```
16
```

```
[29]: for filename in glob.glob('*.csv'):
          print(filename)
```

```
d-0-2.csv
d-1-2.csv
d-12-2.csv
d-2-0.csv
d-2-1.csv
d-2-2.csv
d-2-3.csv
d-2-4.csv
d-3-2.csv
d-4-2.csv
d-5-2.csv
d-6-2.csv
d-9-0.csv
d-9-2.csv
d-9-4.csv
d-9-6.csv
```

(Refer Slide Time: 24:25)



```
[30]: import glob
      import re
      num_files = np.size(glob.glob('*.csv'));
      for filename in glob.glob('*.csv'):
          matches = re.match(r'd-(.*)-(.*).csv', filename,flags=0)
          print(matches)
```

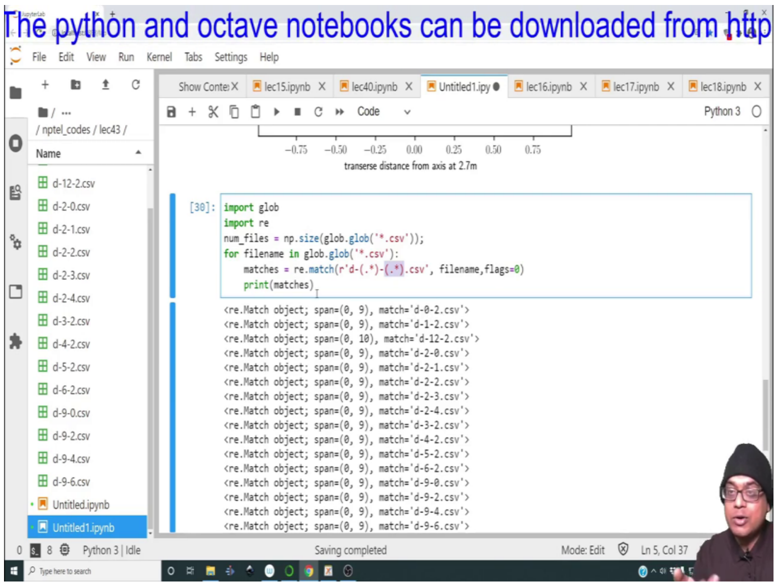
```
<re.Match object; span=(0, 9), match='d-0-2.csv'>
<re.Match object; span=(0, 9), match='d-1-2.csv'>
<re.Match object; span=(0, 10), match='d-12-2.csv'>
<re.Match object; span=(0, 9), match='d-2-0.csv'>
<re.Match object; span=(0, 9), match='d-2-1.csv'>
<re.Match object; span=(0, 9), match='d-2-2.csv'>
<re.Match object; span=(0, 9), match='d-2-3.csv'>
<re.Match object; span=(0, 9), match='d-2-4.csv'>
<re.Match object; span=(0, 9), match='d-3-2.csv'>
<re.Match object; span=(0, 9), match='d-4-2.csv'>
<re.Match object; span=(0, 9), match='d-5-2.csv'>
<re.Match object; span=(0, 9), match='d-6-2.csv'>
<re.Match object; span=(0, 9), match='d-9-0.csv'>
<re.Match object; span=(0, 9), match='d-9-2.csv'>
<re.Match object; span=(0, 9), match='d-9-4.csv'>
<re.Match object; span=(0, 9), match='d-9-6.csv'>
```

So, now we have a way of sort of printing out the file name we need to extract those 2digits from the file. So, for this we will utilize what is called as a regular expression or regex functionality of python actually it is a functionality of Unix.

So, we have we will import regex regular expression; and what this will help us it will help us to identify strings inside file. So, once you see what it is you will understand what it really means matches is equal to ra dot match now.

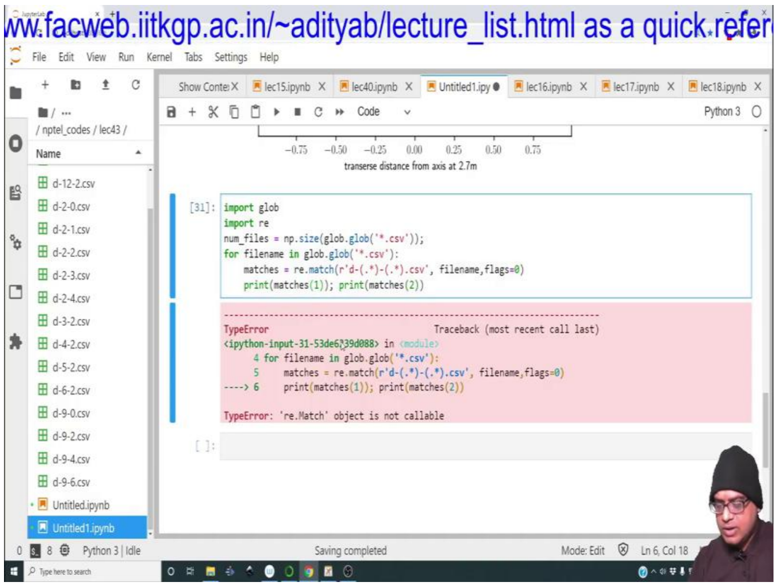
We are going to pass the raw string that we want to check and the raw string that we want to check is d then we want to ok let me just do this .csv(.*)-(.*); and we will check for this kind of string and file name and we will have no flags ok. So, let me print matches. So, what I am doing is I am taking this raw string I am taking the file name.

(Refer Slide Time: 25:49)



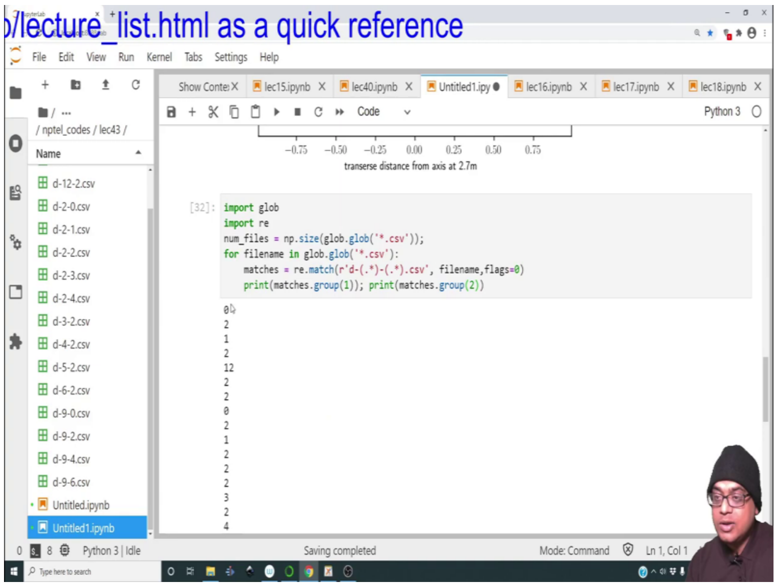
```
[30]: import glob
      import re
      num_files = np.size(glob.glob('*.csv'));
      for filename in glob.glob('*.csv'):
          matches = re.match(r'd-(.*)-(.*).csv', filename,flags=0)
          print(matches)

<re.Match object; span=(0, 9), match='d-0-2.csv'>
<re.Match object; span=(0, 9), match='d-1-2.csv'>
<re.Match object; span=(0, 10), match='d-12-2.csv'>
<re.Match object; span=(0, 9), match='d-2-0.csv'>
<re.Match object; span=(0, 9), match='d-2-1.csv'>
<re.Match object; span=(0, 9), match='d-2-2.csv'>
<re.Match object; span=(0, 9), match='d-2-3.csv'>
<re.Match object; span=(0, 9), match='d-2-4.csv'>
<re.Match object; span=(0, 9), match='d-3-2.csv'>
<re.Match object; span=(0, 9), match='d-4-2.csv'>
<re.Match object; span=(0, 9), match='d-5-2.csv'>
<re.Match object; span=(0, 9), match='d-6-2.csv'>
<re.Match object; span=(0, 9), match='d-9-0.csv'>
<re.Match object; span=(0, 9), match='d-9-2.csv'>
<re.Match object; span=(0, 9), match='d-9-4.csv'>
<re.Match object; span=(0, 9), match='d-9-6.csv'>
```

(Refer Slide Time: 26:08)



```
[31]: import glob
      import re
      num_files = np.size(glob.glob('*.csv'));
      for filename in glob.glob('*.csv'):
          matches = re.match(r'd-(.*)-(.*).csv', filename,flags=0)
          print(matches(1)); print(matches(2))

---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-31-53de6?39d088> in <module>
      4 for filename in glob.glob('*.csv'):
      5     matches = re.match(r'd-(.*)-(.*).csv', filename,flags=0)
----> 6     print(matches(1)); print(matches(2))

TypeError: 're.Match' object is not callable

[ ]:
```

(Refer Slide Time: 26:10)


/lecture_list.html as a quick reference

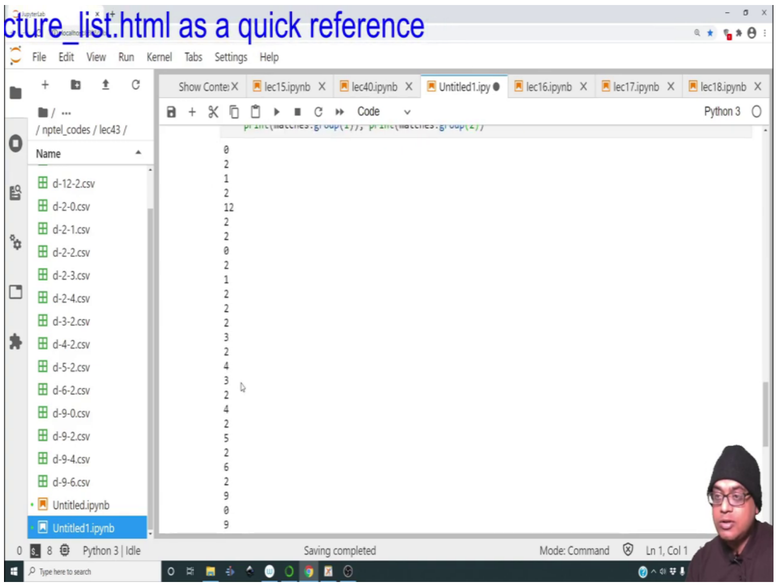
```
[32]: import glob
      import re
      num_files = np.size(glob.glob('*.csv'));
      for filename in glob.glob('*.csv'):
          matches = re.match(r'd-(.*)-(.*).csv', filename,flags=0)
          print(matches.group(1)); print(matches.group(2))
```

```
0
2
1
2
12
2
2
0
2
1
2
2
2
3
2
4
```

And comparing the file name to this raw string. So, dot star means whatever comes in between da dash something and dash then, whatever appears in in this placeholder ok. So, in this placeholder you will have 0 and 2.

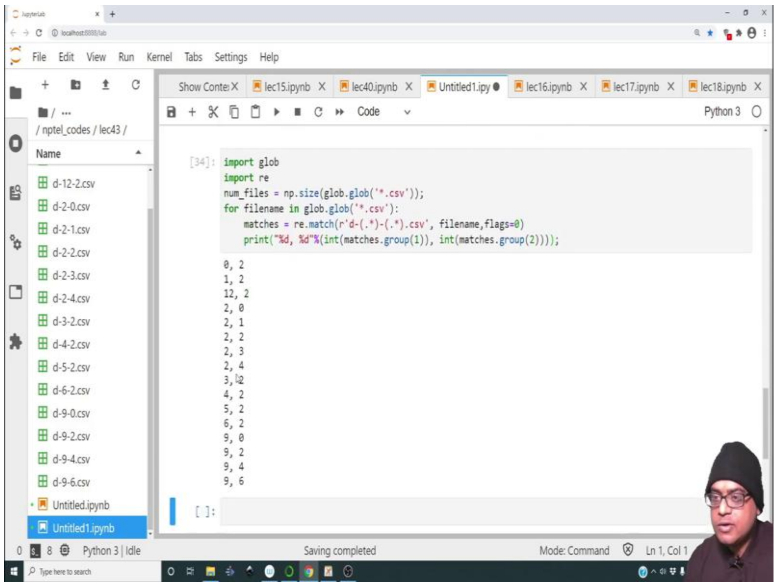
So, let me do this print matches 0 print matches 1 not 0, but this will be so, 0 will give me all the string. So, 1, 2 so, its matches dot group 1 group 2. So, this actually gives me all the different strings and if you are not convinced whether this is working or not ah.

(Refer Slide Time: 26:38)


cture_list.html as a quick reference

```
0
2
1
2
12
2
2
0
2
1
2
2
2
3
2
4
3
2
4
2
5
2
6
2
9
0
9
```

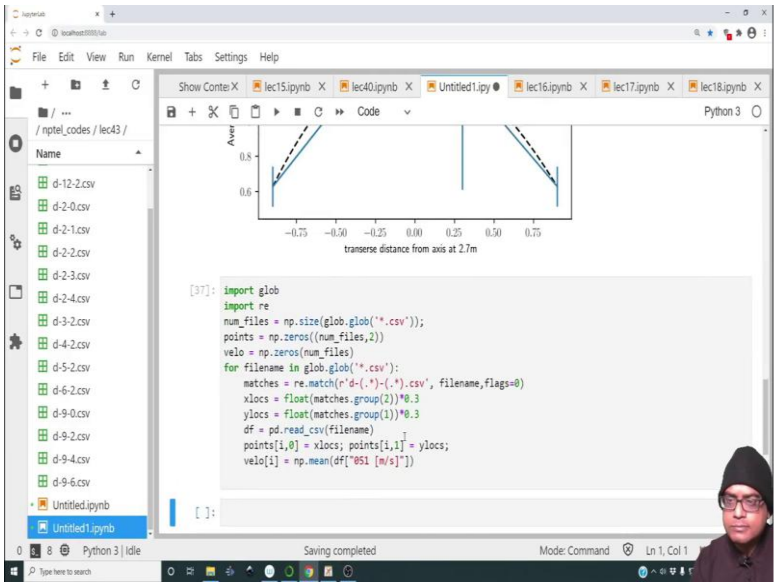
(Refer Slide Time: 26:48)



```
[34]: import glob
      import re
      num_files = np.size(glob.glob('*.csv'));
      for filename in glob.glob('*.csv'):
          matches = re.match(r'd-(.*)-(.*).csv', filename,flags=0)
          print("%d, %d"%(int(matches.group(1)), int(matches.group(2))));

      0, 2
      1, 2
      12, 2
      2, 0
      2, 1
      2, 2
      2, 3
      2, 4
      3, 2
      4, 2
      5, 2
      6, 2
      9, 0
      9, 2
      9, 4
      9, 6
```

Let me do this now what is required alright. So, look its giving us all the strings we want and I have type casted the string to an integer that is standard I mean because, it is giving you a string output the matches is giving you a string output you must type cast it to integer, if you want to use it in further calculation.

(Refer Slide Time: 27:47)



```
[37]: import glob
      import re
      num_files = np.size(glob.glob('*.csv'));
      points = np.zeros((num_files,2))
      velo = np.zeros(num_files)
      for filename in glob.glob('*.csv'):
          matches = re.match(r'd-(.*)-(.*).csv', filename,flags=0)
          xlocs = float(matches.group(2))*0.3
          ylocs = float(matches.group(1))*0.3
          df = pd.read_csv(filename)
          points[i,0] = xlocs; points[i,1] = ylocs;
          velo[i] = np.mean(df["051 [m/s]"])
```

So, good so group 1 is going to be the y coordinate and group 2 is going to be the x coordinate as simple as that. So, xlocs = float(matches**.**group(2))*****0.3 actually I have to type caste it to float so, I will talk type caste it to float multiplied by this xlocs =

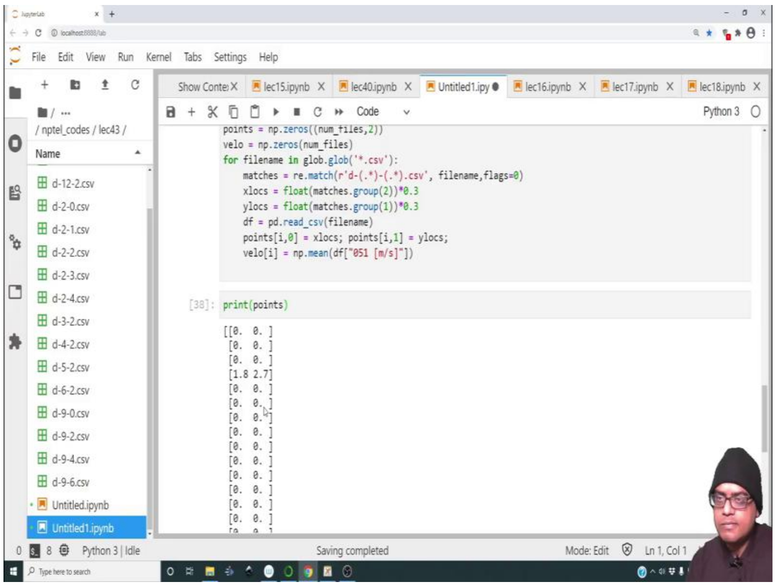
float(matches.group(2))*0.3 match or matches ok. So, everything runs fine and now that we have the file name we can simply reuse all this so, we are going to copy this bit of code.

We are going to import it paste it over here. So, file name velo why you do not need y anymore? Actually we do need y because, y[i] will be ylocs and actually let me write it as points[i,0] = xlocs and points[i,1] = ylocs ok. So, once we are going in this loop it will take xlocs it will put in points 0 comma.

So, we need to define what points is so because number of files is 16 so, points will be np dot zeros we are going to initialize it and it will be np dot so it will be actually num files comma 2.

So, number of rows will be the number of files and 2 columns for x location and my location alright. Corresponding to these points we will have the velo = np.zeros(num_files). So, we are soft coding everything we can put np 0.016, but we are soft coding everything.

(Refer Slide Time: 30:28)



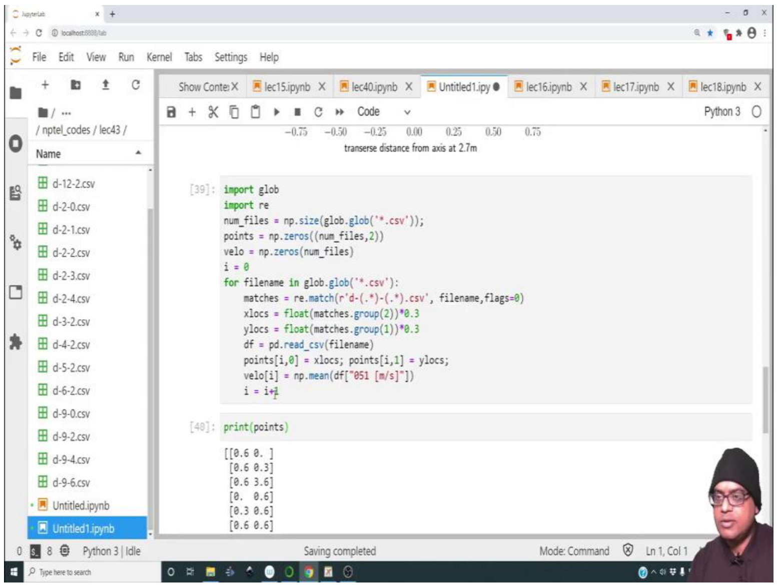
```
points = np.zeros((num_files,2))
velo = np.zeros(num_files)
for filename in glob.glob('*.csv'):
    matches = re.match(r'd-(.*)-(.*).csv', filename,flags=0)
    xlocs = float(matches.group(2))*0.3
    ylocs = float(matches.group(1))*0.3
    df = pd.read_csv(filename)
    points[i,0] = xlocs; points[i,1] = ylocs;
    velo[i] = np.mean(df["051 [m/s]"])
```

```
[38]: print(points)
```

```
[[0.  0. ]
 [0.  0. ]
 [0.  0. ]
 [1.8 2.7]
 [0.  0. ]
 [0.  0. ]
 [0.  0. ]
 [0.  0. ]
 [0.  0. ]
 [0.  0. ]
 [0.  0. ]
 [0.  0. ]
 [0.  0. ]
 [0.  0. ]
```

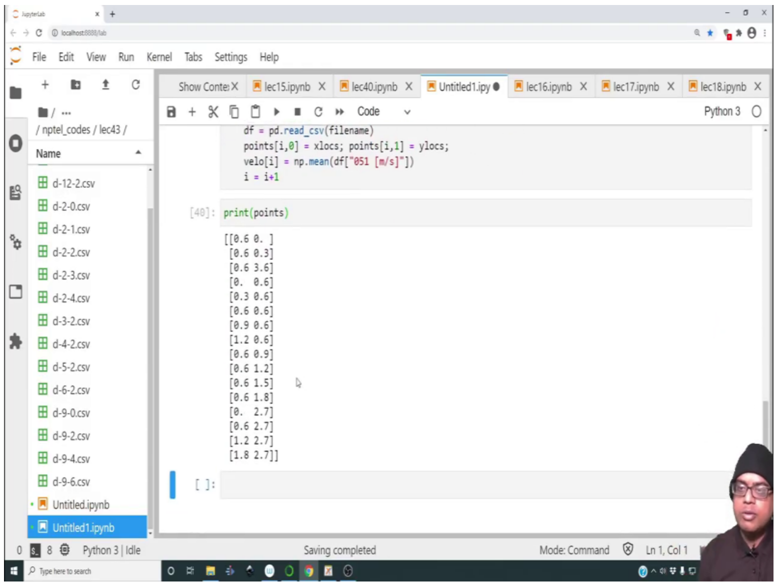
Hm we do not need the standard deviation in this one that is fine. So, let me run this to see if the scenario there is no error great, hm. Let me print what points is ok, does so I have to set i = 0 over here. At the end of it I have to do i = i+1 let me run this let me run this great.

(Refer Slide Time: 30:40)



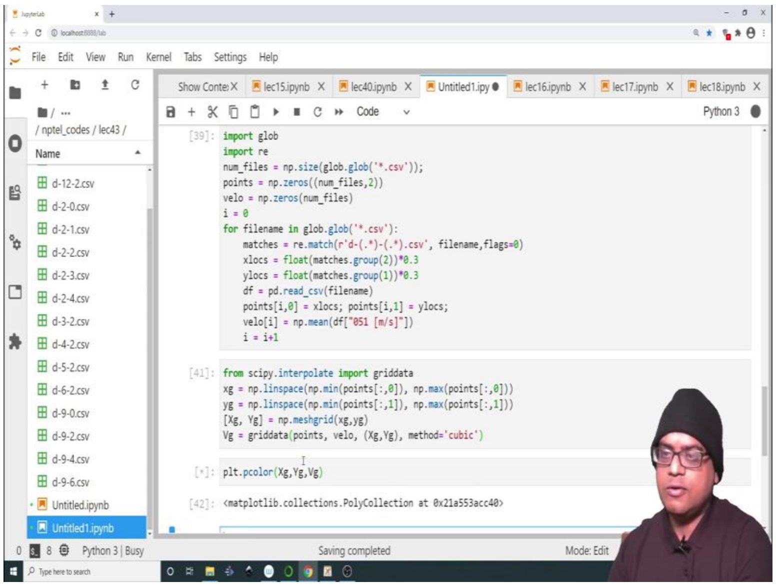
```
[39]: import glob
      import re
      num_files = np.size(glob.glob('*.csv'));
      points = np.zeros((num_files,2))
      velo = np.zeros(num_files)
      i = 0
      for filename in glob.glob('*.csv'):
          matches = re.match(r'd-(.*)-(.*).csv', filename,flags=0)
          xlocs = float(matches.group(2))*0.3
          ylocs = float(matches.group(1))*0.3
          df = pd.read_csv(filename)
          points[i,0] = xlocs; points[i,1] = ylocs;
          velo[i] = np.mean(df["051 [m/s]"])
          i = i+1

[40]: print(points)

      [[0.6 0. ]
       [0.6 0.3]
       [0.6 3.6]
       [0.  0.6]
       [0.3 0.6]
       [0.6 0.6]
```

(Refer Slide Time: 30:49)



```
          df = pd.read_csv(filename)
          points[i,0] = xlocs; points[i,1] = ylocs;
          velo[i] = np.mean(df["051 [m/s]"])
          i = i+1

[40]: print(points)

      [[0.6 0. ]
       [0.6 0.3]
       [0.6 3.6]
       [0.  0.6]
       [0.3 0.6]
       [0.6 0.6]
       [0.9 0.6]
       [1.2 0.6]
       [0.6 0.9]
       [0.6 1.2]
       [0.6 1.5]
       [0.6 1.8]
       [0.  2.7]
       [0.6 2.7]
       [1.2 2.7]
       [1.8 2.7]]

[ ]:
```

(Refer Slide Time: 31:03)



```
[39]: import glob
      import re
      num_files = np.size(glob.glob('*.csv'));
      points = np.zeros((num_files,2))
      velo = np.zeros(num_files)
      i = 0
      for filename in glob.glob('*.csv'):
          matches = re.match(r'd-(.*)-(.*).csv', filename,flags=0)
          xlocs = float(matches.group(2))*0.3
          ylocs = float(matches.group(1))*0.3
          df = pd.read_csv(filename)
          points[i,0] = xlocs; points[i,1] = ylocs;
          velo[i] = np.mean(df["051 [m/s]"])
          i = i+1

[41]: from scipy.interpolate import griddata
      xg = np.linspace(np.min(points[:,0]), np.max(points[:,0]))
      yg = np.linspace(np.min(points[:,1]), np.max(points[:,1]))
      [Xg, Yg] = np.meshgrid(xg,yg)
      Vg = griddata(points, velo, (Xg,Yg), method='cubic')

[*]: plt.pcolor(Xg,Yg,Vg)

[42]: <matplotlib.collections.PolyCollection at 0x21a553acc40>
```

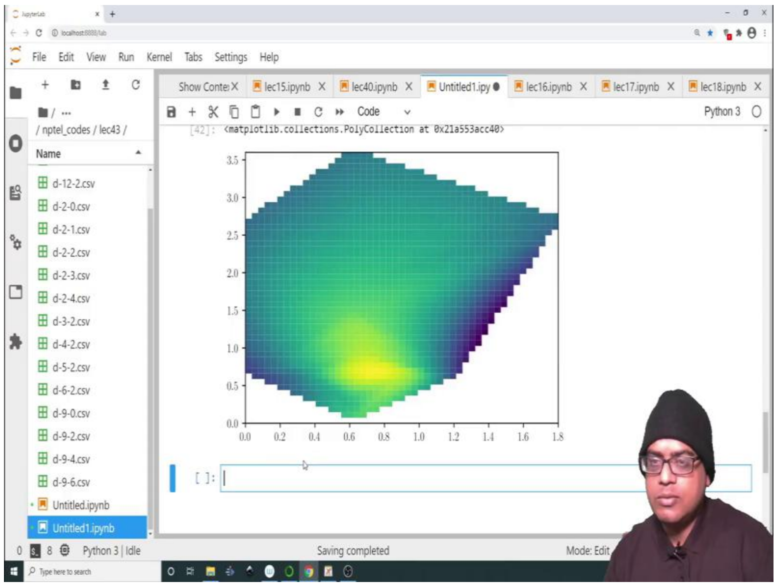
We have all the pairs of point's hm. So, now that we have all the pairs of points, we must do a 2 dimensional interpolation ok. So, let us get to that so first of all we must import grid data. So, **from** scipy.interpolate **import** griddata. So, what grid data does is, if you have x y value triplets all over domain if its scattered you can take those scattered points and you can sort of interpolate them over a grid data using this particular function, it will be clear what it means.

So, let me write xg = np**.**linspace(np**.**min(points[:,0]), np**.**max(points[:,0])) similarly, I will define yg = np**.**linspace(np**.**min(points[:,1]), np**.**max(points[:,1])). So, 0th column are the all the x values the minimum of all the x values and the maximum of all the x values will be sort of used to create the finer grid by default it is 50. So, this should be np.min and this should be np.max similarly this should be np.max and this should be np.min ok.

Now, with the help of these two axis we are going to create the mesh. So, [Xg, Yg] = np**.**meshgrid(xg,yg) as simple as that now, once we have this values on this fine grid will be grid data.

So, the points pair the values. So, the values are velo then the Xg and Yg on which you want to obtain the interpolated values and finally the method. So, the method has to be there is a bunch of methods we can use linear, nearest, or cubic let us keep it cubic for now.

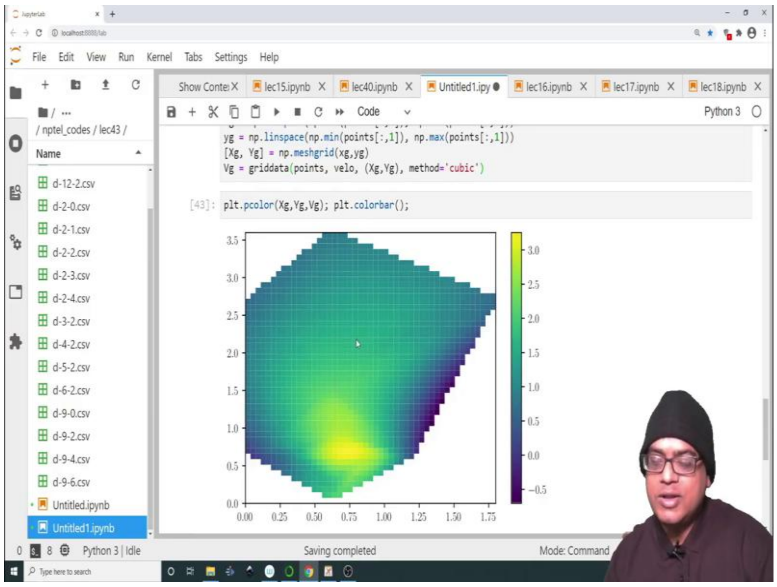
(Refer Slide Time: 33:50)



```
[42]: <matplotlib.collections.PolyCollection at 0x21a553acc40>

[ ]:
```

So, let me run this and see if there is an error there is no error. So, lastly what we can do is plt.pcolor(Xg,Yg,Vg). So, let us see what we obtain we obtain the interpolated data, wherever its white meaning it cannot interpolate because we do not have enough points from which we can create that interpolation.

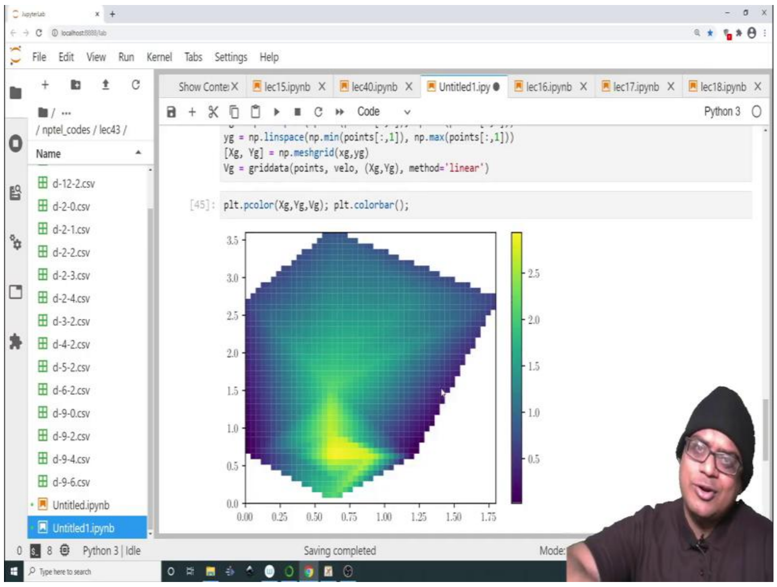
(Refer Slide Time: 34:07)



```
yg = np.linspace(np.min(points[:,1]), np.max(points[:,1]))
[Xg, Yg] = np.meshgrid(xg,yg)
Vg = griddata(points, velo, (Xg,Yg), method='cubic')

[43]: plt.pcolor(Xg,Yg,Vg); plt.colorbar();
```

So, this is great let me add a color bar plt.colorbar() alright. So, the velocity is higher near the fan so, this is the location of the fan and it sort of decays. So, this is the region if you sit near this region you get the maximum wind, if you sit near the edges you get nothing ok.

(Refer Slide Time: 34:31)



```
yg = np.linspace(np.min(points[:,1]), np.max(points[:,1]))
[Xg, Yg] = np.meshgrid(xg,yg)
Vg = griddata(points, velo, (Xg,Yg), method='linear')
```

```
[45]: plt.pcolor(Xg,Yg,Vg); plt.colorbar();
```

In fact, the interpolation predicts a negative velocity in fact; let me make this linear let us see yeah. The linear gets rid of those negative values because the cubic is going to undershoot, the cubic usually does this undershooting business that is ok because, the number of grid points that we had were quite less right.

So, just like that we were able to make a file through which we were able to take data analyze it make the 2d map of the data and I hope you will find this very useful in your endeavors.

So, with this we come to the conclusion of this course I hope you have learned a lot of new things and hopefully it will benefit you in your research or in your assignments or in fact, your learning as well. I have enjoyed this journey and I really hope you have enjoyed this as well with this I sign out and I will see you again someday with a new course bye.