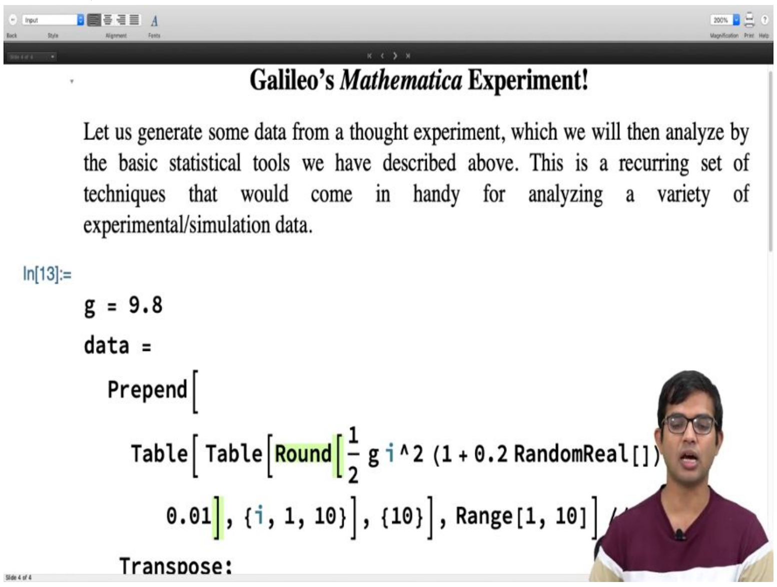
# Physics through Computational Thinking

**Professor Auditya Sharma**
**Professor Ambar Jain**
**Department of Physics**
**Indian Institute of Science Education and Research, Bhopal**

**Lecture 17**
**Introduction to Data Analysis 2**

(Refer Slide Time: 0:26)



## Galileo's *Mathematica* Experiment!

Let us generate some data from a thought experiment, which we will then analyze by the basic statistical tools we have described above. This is a recurring set of techniques that would come in handy for analyzing a variety of experimental/simulation data.

In[13]:=

```
g = 9.8
data =
  Prepend[
    Table[Table[Round[1/2 g i^2 (1 + 0.2 RandomReal[]),
      0.01], {i, 1, 10}], {10}], Range[1, 10]]
   Transpose;
```

Okay, so let us try and apply some of these ideas of data analysis and let us imagine ourselves to be in the shoes of Galileo. And let us say that Galileo somehow has access to Mathematica and then we design a thought experiment. So we, of course, know that if you drop a ball then the distance covered by it as a function of time is just ½ $gt^2$, right.

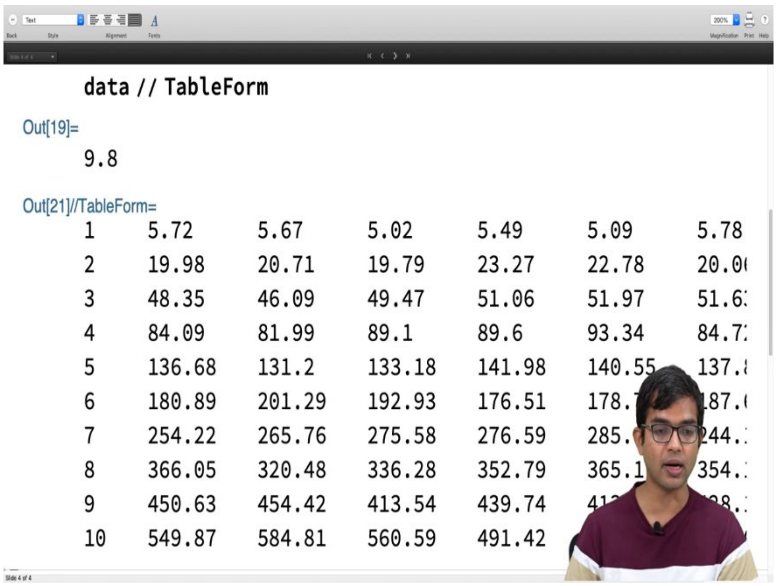
So, what I am going to do is use Mathematica to give me some data, right. I mean, I am imagining carrying out an experiment in which I drop a ball and then after a certain, let us say after 1 second, I find out how much distance has it covered, after 2 seconds how much distance it has covered, after 3 seconds how much distance it has covered, and so on, all the way up to 10 seconds right.

And then, so this is going to give me one set of data. And then I repeat this experiment a second time. Once again, I will get another set of numbers, it is not going to be identical to what I had the first time, but it is going to be roughly in the same ballpark. Then I do an experiment a third time, then again, I have a bunch of 10 numbers, which are all different but very close to the, you know, the first set and so on.

So, you imagine repeating this experiment a large number of times right. So, this is in the spirit of how one carries out experiments right. So, suppose Galileo had no idea that there is a quadratic relationship between time and distance, just by carrying out this experiment, we do not know we do not get into how he would actually measure these distances. Suppose it were possible. So, in principle it should be, one should be able to extract a neat quadratic curve right and with error bars right.

So, the focus of this current discussion is all about how to do statistical analysis of this. So, in order to do this, I am going to find a way to use Mathematica to give me some data, it is a fictitious data. So what I will do is, so first of all define the gravitational constant as g = 9.8. Of course, I already know the functional form to be ½ $gt^2$. So, what I will do is, I have this code, which I will just first run this whole code and then I will explain to you how I am generating this. So I have, so basically I am telling you that if I were to use this entire code.

(Refer Slide Time: 3:01)



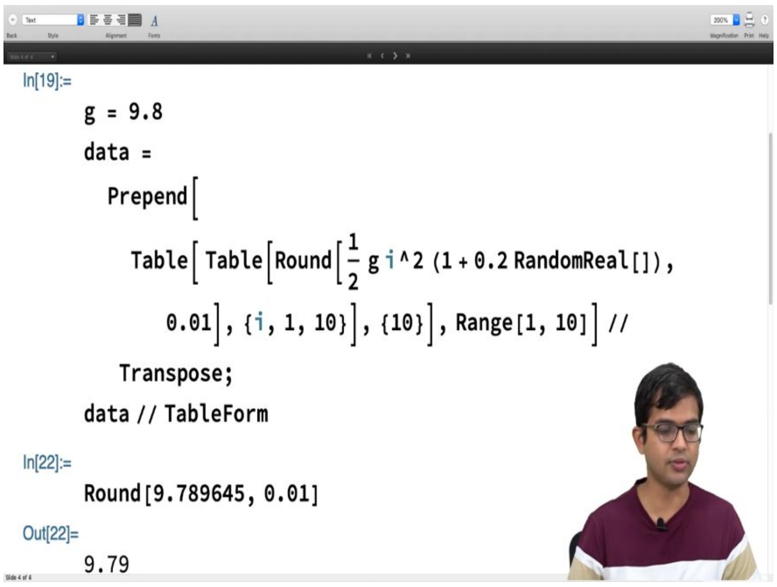
```
      data // TableForm
Out[19]=
      9.8
Out[21]//TableForm=
      1    5.72     5.67     5.02     5.49     5.09     5.78
      2    19.98    20.71    19.79    23.27    22.78    20.0
      3    48.35    46.09    49.47    51.06    51.97    51.6
      4    84.09    81.99    89.1     89.6     93.34    84.7
      5    136.68   131.2    133.18   141.98   140.55   137.
      6    180.89   201.29   192.93   176.51   178.     187.
      7    254.22   265.76   275.58   276.59   285.     244.
      8    366.05   320.48   336.28   352.79   365.1    354.
      9    450.63   454.42   413.54   439.74   41       [illegible]
      10   549.87   584.81   560.59   491.42   [illegible]
```

I can generate a table where after 1 second, maybe the ball has reached the distance 5.72 in some, in meters. So, then after 2 seconds, it is at 19.98. After 3 seconds, it is at this, and so on. After 10 seconds, it is 549.87.

But if I were to do the experiment the second time, you will see that the numbers are slightly different, but they are all in the same ballpark. There is a spread and this spread is something that I have used Mathematica to generate right. So, what am I doing here? So, whenever you see code of this kind in Mathematica, the correct approach is to sort of go inwards out. So, you start with the simplest function. So, I have this, I have already described what table does.

So I will, I am using something called round, I am using something called random real, I am using something called a prepend, you know. These are all functions which Mathematica has inbuilt in them. So, I will, let me let me try and open it up for you. So, I will start with let us say just round. So, if I were to go down here, so I will use input.

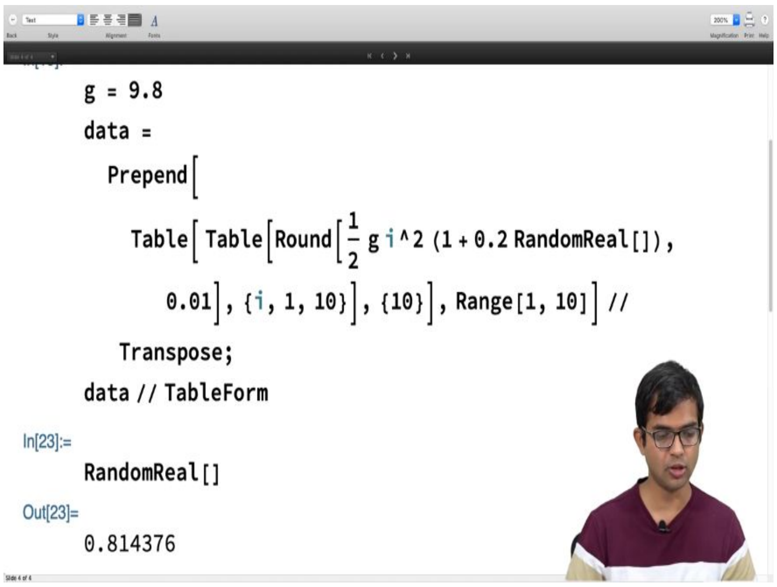
(Refer Slide Time: 04:09)



```
In[19]:=
      g = 9.8
      data =
        Prepend[
          Table[Table[Round[1/2 g i^2 (1 + 0.2 RandomReal[]),
             0.01], {i, 1, 10}], {10}], Range[1, 10]] //
         Transpose;
      data // TableForm
In[22]:=
      Round[9.789645, 0.01]
Out[22]=
      9.79
```

Okay, so if I were to do round of 9.789645, let us say. And if I want to round this to 0.01 and I go ahead and hit shift enter, so it gives me an answer of 9.79. So, which is what you expect up to two decimal place accuracy 9.789645 is 9.79. And that is what Mathematica tells me. So, here I want to generate this data half gt squared and I want to put this in a table, but I do not want only ½ $gt^2$, I want ½ $gt^2$ but with a spread around it with a random real. Random real is is a function which gives me random numbers in the range 0 to 1.

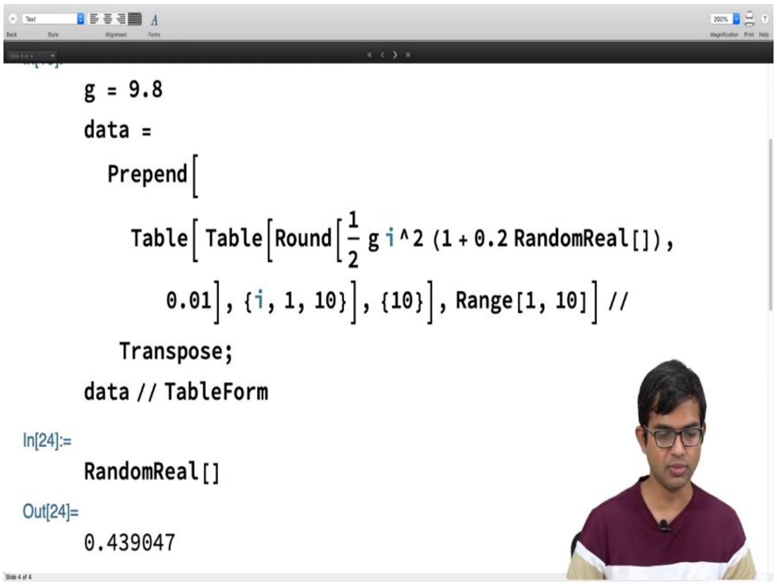
(Refer Slide Time: 05:20)



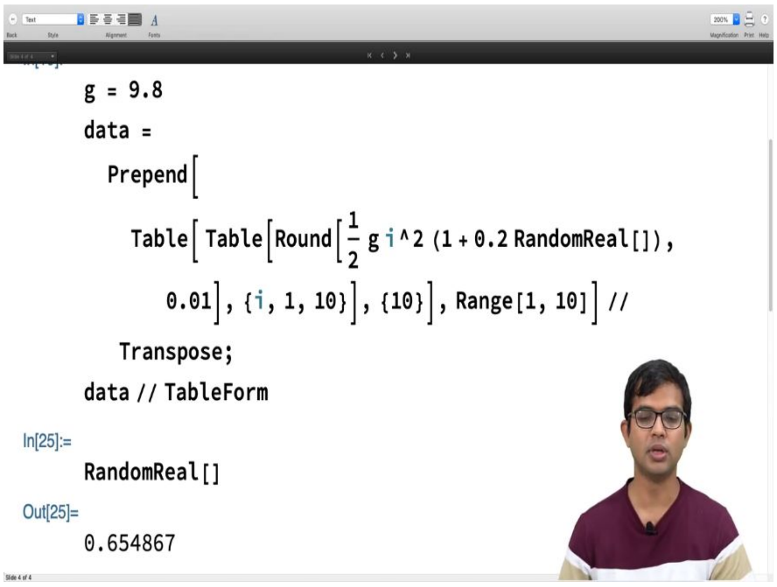
```
g = 9.8
data =
  Prepend[
    Table[Table[Round[1/2 g i^2 (1 + 0.2 RandomReal[]),
       0.01], {i, 1, 10}], {10}], Range[1, 10]] //
   Transpose;
data // TableForm
In[23]:=
RandomReal[]
Out[23]=
0.814376
```

So, if I do random real and it does not require any. So, if you do not put any arguments, it means that you are looking for a number between 0 and 1. So, there you go. So, it gave me 0.8. If I were to repeat this, it would give me some other number, some other number and so on.

(Refer Slide Time: 05:41)



```
g = 9.8
data =
  Prepend[
    Table[Table[Round[1/2 g i^2 (1 + 0.2 RandomReal[]),
       0.01], {i, 1, 10}], {10}], Range[1, 10]] //
   Transpose;
data // TableForm
In[24]:=
RandomReal[]
Out[24]=
0.439047
```



```
g = 9.8
data =
  Prepend[
    Table[Table[Round[1/2 g i^2 (1 + 0.2 RandomReal[]),
      0.01], {i, 1, 10}], {10}], Range[1, 10]] //
   Transpose;
data // TableForm
```

In[25]:=

```
RandomReal[]
```

Out[25]=

```
0.654867
```

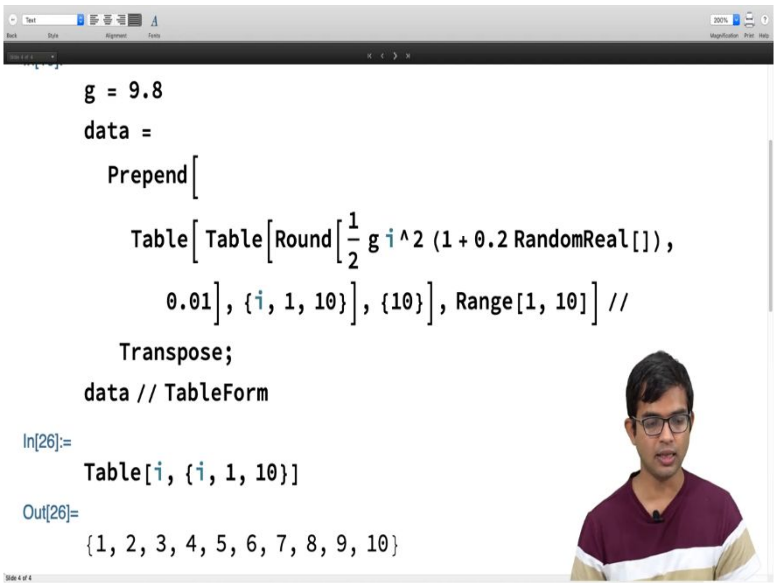
It gives me some bunch of numbers. So, what I am asking Mathematica to do in this code is to generate a number close to half gt squared. So, i, you see, i goes from 1 to 10. So, this table generates for me a bunch of numbers. And so I get, it is supposed to give me ½ $gi^2$, where i will go from 1 to 10. But I also ask it to give me some spread, which is going to be like 20 percent of this value.

I mean, I have just put some number, like some fraction, I have taken it to be 20 percent, you could try it with some other percentage. You could or you could make it a constant error or constant spread of some kind, but I am choosing to make it to be a percentage of the value itself, I, well, let us not go into whether this is exactly how a real experiment would give or not, I am just playing right.

So, when you are playing computer games, we are free to choose these parameters. And so here I am choosing it to be 0.2. So, later on of course, we can go back and change this and in general, one can make it a parameter with which you can play with and you can study it more systematically on its own.

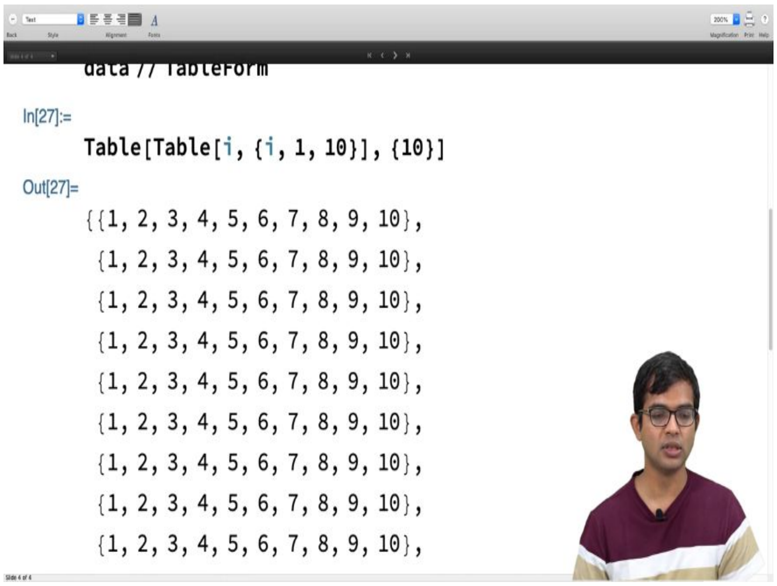
But for our purposes, we are doing something very simple right. So, we are generating a table of numbers, which are in the ballpark of half gt squared then I want to, I nest it around with another table. Why do I do this? Because I carry the same experiment 10 times. So, that is the second 10 that you are seeing. So let us, let me try and explain this step by step.

(Refer Slide Time: 07:23)



```
g = 9.8
data =
  Prepend[
    Table[Table[Round[1/2 g i^2 (1 + 0.2 RandomReal[]),
      0.01], {i, 1, 10}], {10}], Range[1, 10]] //
   Transpose;
data // TableForm
```

```
In[26]:=
  Table[i, {i, 1, 10}]
Out[26]=
  {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```



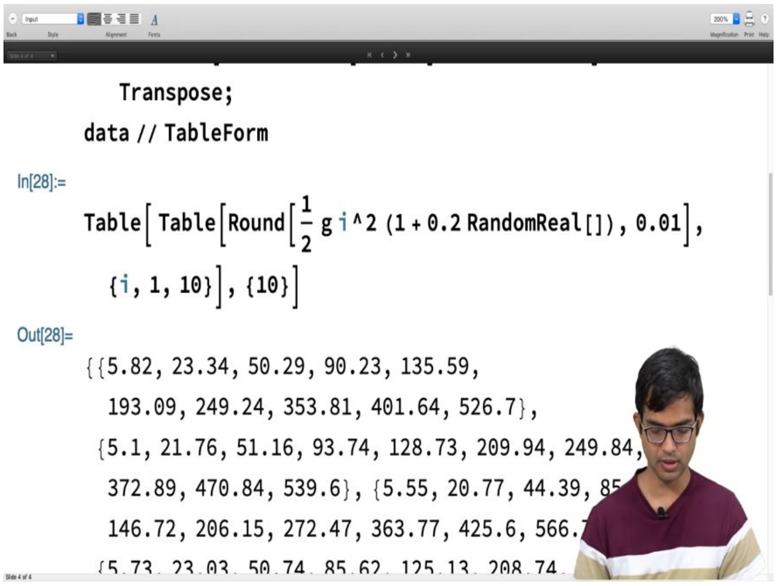
```
data // TableForm
In[27]:=
  Table[Table[i, {i, 1, 10}], {10}]
Out[27]=
  {{1, 2, 3, 4, 5, 6, 7, 8, 9, 10},
   {1, 2, 3, 4, 5, 6, 7, 8, 9, 10},
   {1, 2, 3, 4, 5, 6, 7, 8, 9, 10},
   {1, 2, 3, 4, 5, 6, 7, 8, 9, 10},
   {1, 2, 3, 4, 5, 6, 7, 8, 9, 10},
   {1, 2, 3, 4, 5, 6, 7, 8, 9, 10},
   {1, 2, 3, 4, 5, 6, 7, 8, 9, 10},
   {1, 2, 3, 4, 5, 6, 7, 8, 9, 10},
   {1, 2, 3, 4, 5, 6, 7, 8, 9, 10},
```

So, the first table. So, I have told you that if I were to generate a table of some numbers, let us say i, and i goes from 1 to 10. It is going to just give me numbers 1 to 10. And then, if I were to put another table around it. So, Mathematica is going to say I need to generate this entire table itself so many times. So if I were to do 10, it gives me an array of arrays.

So here, of course, I am doing something really dumb because why would you want these numbers from 1 to 10 repeated 10 times? But on the other hand, so in the experiment or that I am trying to simulate, what I am trying to do is get some spread. So, it is not going to be just the same numbers again and again, but because I have random real I am going to get so many different realizations of this.

And then round, make sure that these numbers are always cropped in a nice way and I have only two decimal place accuracy. And then finally, I have this. So, let me let me just try and explain this prepend.

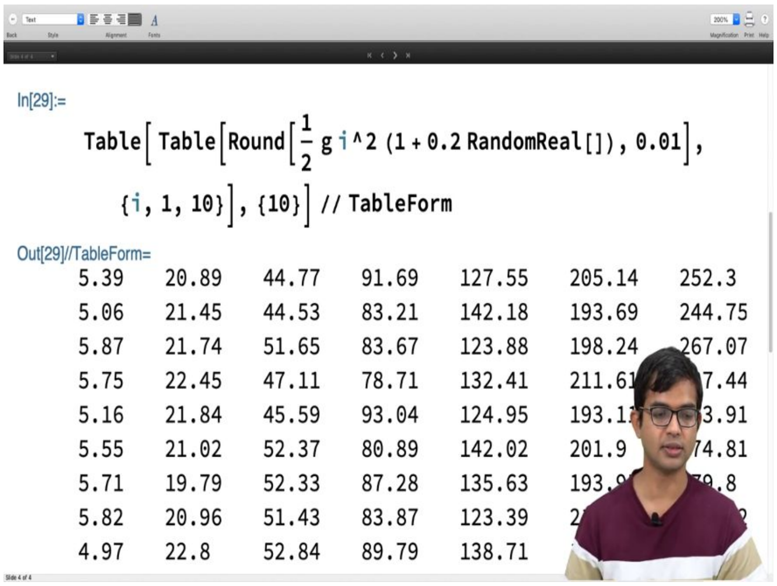
(Refer Slide Time: 08:51)



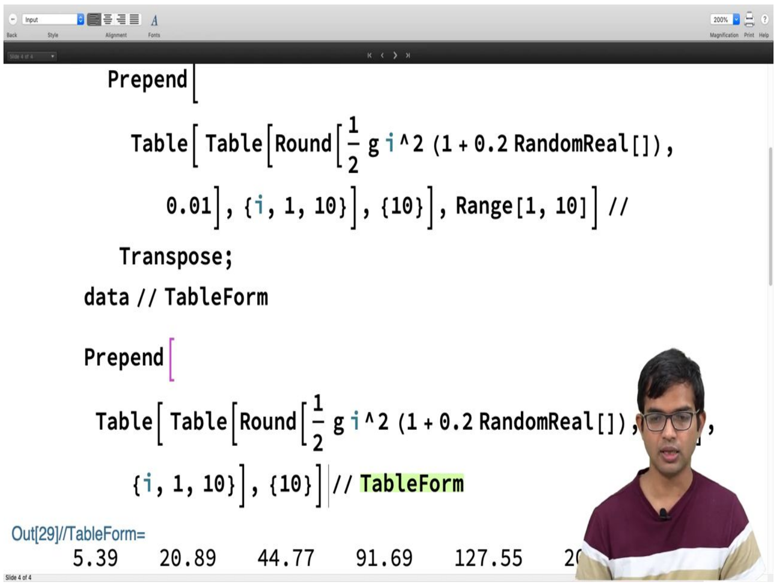
```
      Transpose;
    data // TableForm

In[28]:=
    Table[Table[Round[1/2 g i^2 (1 + 0.2 RandomReal[]), 0.01],
      {i, 1, 10}], {10}]

Out[28]=
    {{5.82, 23.34, 50.29, 90.23, 135.59,
      193.09, 249.24, 353.81, 401.64, 526.7},
     {5.1, 21.76, 51.16, 93.74, 128.73, 209.94, 249.84,
      372.89, 470.84, 539.6}, {5.55, 20.77, 44.39, 85
      146.72, 206.15, 272.47, 363.77, 425.6, 566.7
     {5.73, 23.03, 50.74, 85.62, 125.13, 208.74,
```

So, before I suppose I did it without prepend, it is going to give me something like this. Let me copy this stuff and paste it here. It is going to give me data, which is going to look like this. Let me put it in table form is just a nice form it gives you, gives your data.

(Refer Slide Time: 09:03)



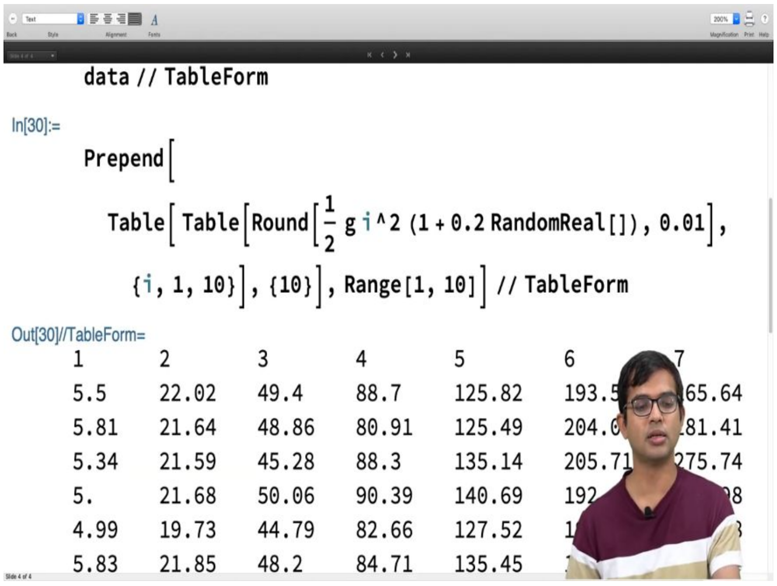
```
In[29]:=
    Table[Table[Round[1/2 g i^2 (1 + 0.2 RandomReal[]), 0.01],
      {i, 1, 10}], {10}] // TableForm

Out[29]//TableForm=
    5.39    20.89   44.77   91.69   127.55   205.14   252.3
    5.06    21.45   44.53   83.21   142.18   193.69   244.75
    5.87    21.74   51.65   83.67   123.88   198.24   267.07
    5.75    22.45   47.11   78.71   132.41   211.61   7.44
    5.16    21.84   45.59   93.04   124.95   193.1    3.91
    5.55    21.02   52.37   80.89   142.02   201.9    74.81
    5.71    19.79   52.33   87.28   135.63   193.    9.8
    5.82    20.96   51.43   83.87   123.39   2
    4.97    22.8    52.84   89.79   138.71
```



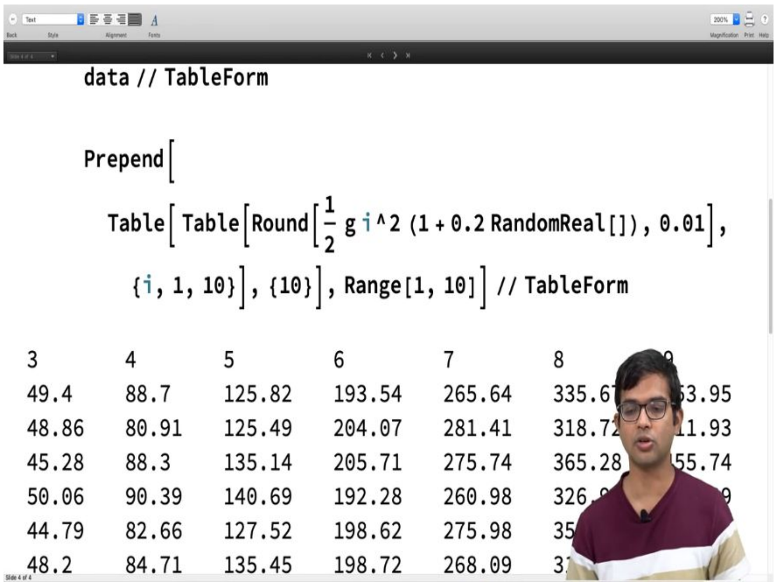
```
Prepend[
  Table[Table[Round[1/2 g i^2 (1 + 0.2 RandomReal[]),
     0.01], {i, 1, 10}], {10}], Range[1, 10]] //
   Transpose;
data // TableForm

Prepend[
 Table[Table[Round[1/2 g i^2 (1 + 0.2 RandomReal[]),
    {i, 1, 10}], {10}] // TableForm
Out[29]//TableForm=
 5.39    20.89    44.77    91.69    127.55    2
```

So, if I did not do this, then I have these numbers, after one second, two seconds, three seconds. So, it is along the horizontal axis. It is increasing. So, but I will also like the time to appear right at the beginning. So, what I do is use this thing called prepend. If I were to do prepend, if I do prepend, p r e p e n d and then put this entire thing around it, prepend.

(Refer Slide Time: 09:38)



```
data // TableForm
In[30]:=
 Prepend[
  Table[Table[Round[1/2 g i^2 (1 + 0.2 RandomReal[]), 0.01],
    {i, 1, 10}], {10}], Range[1, 10]] // TableForm
Out[30]//TableForm=
 1       2        3        4        5         6        7
 5.5     22.02    49.4     88.7     125.82    193.5    65.64
 5.81    21.64    48.86    80.91    125.49    204.0    81.41
 5.34    21.59    45.28    88.3     135.14    205.71   275.74
 5.      21.68    50.06    90.39    140.69    192      8
 4.99    19.73    44.79    82.66    127.52    1        3
 5.83    21.85    48.2     84.71    135.45
```

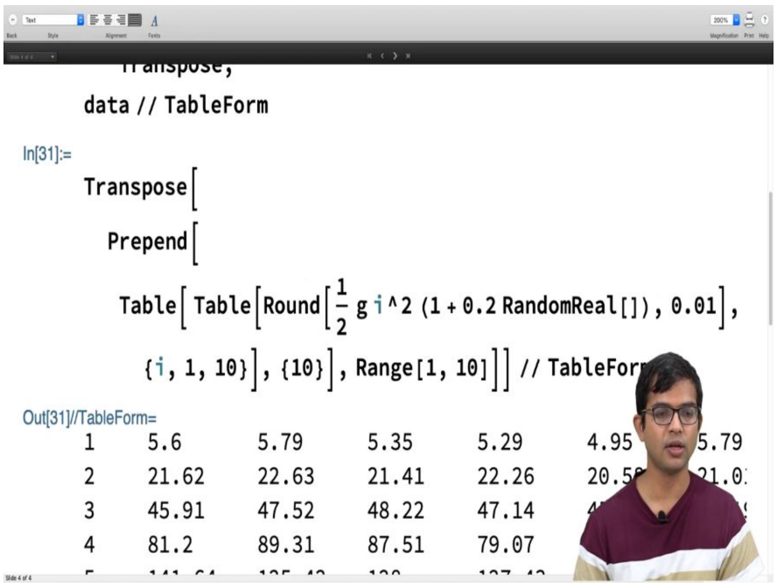


```
data // TableForm

Prepend[
  Table[Table[Round[1/2 g i^2 (1 + 0.2 RandomReal[]), 0.01],
    {i, 1, 10}], {10}], Range[1, 10]] // TableForm
```

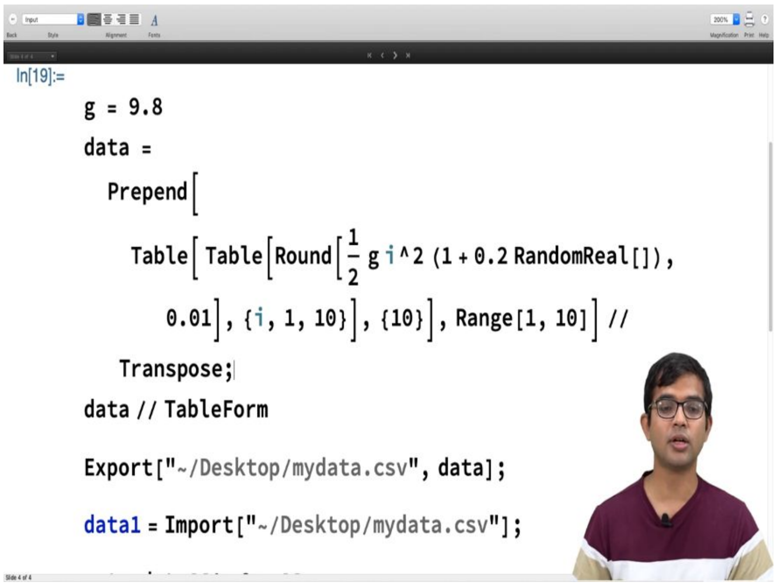
```
3        4        5        6        7        8        9
49.4     88.7     125.82   193.54   265.64   335.6    3.95
48.86    80.91    125.49   204.07   281.41   318.72   1.93
45.28    88.3     135.14   205.71   275.74   365.28   55.74
50.06    90.39    140.69   192.28   260.98   326.     9
44.79    82.66    127.52   198.62   275.98   35
48.2     84.71    135.45   198.72   268.09   3
```

Then I have to provide this range. Its numbers ranging from 1 to 1 to 1 to 10. If I do this and then again table form can stay. So, now you see that I have these numbers 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 right at the top. See at the moment, I have it arranged in this sort of column-wise, but I want this to become row-wise. And that is why I also do transpose, I can, I do, I use this thing called, it is a post processing operation. So if I do slash and then do transpose then that is my final answer or I can also, of course, go ahead and do transpose of this entire thing. That is going to be the same, transpose of this whole thing. And then look at it in table form.

(Refer Slide Time: 10:21)



```
  Transpose;
data // TableForm

In[31]:=
Transpose[
  Prepend[
    Table[Table[Round[1/2 g i^2 (1 + 0.2 RandomReal[]), 0.01],
      {i, 1, 10}], {10}], Range[1, 10]]] // TableFor

Out[31]//TableForm=
1    5.6      5.79     5.35     5.29     4.95     5.79
2    21.62    22.63    21.41    22.26    20.5     21.0
3    45.91    47.52    48.22    47.14
4    81.2     89.31    87.51    79.07
```



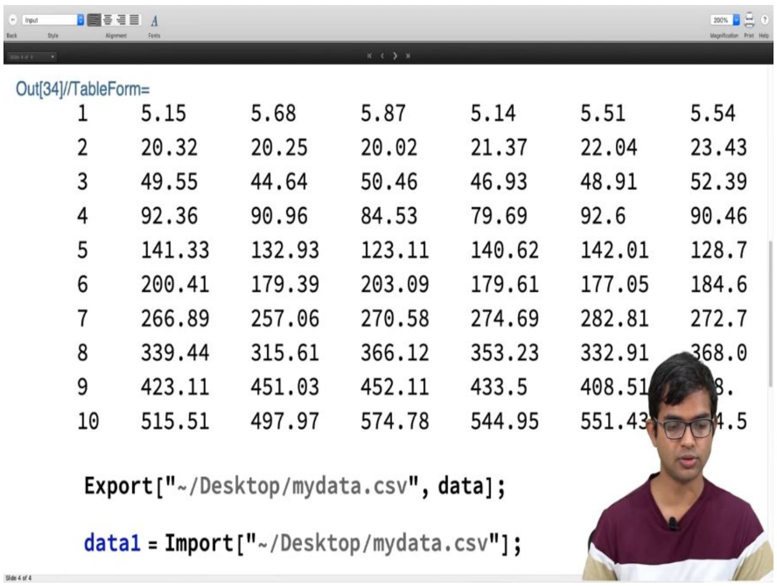
```
In[19]:=
g = 9.8
data =
  Prepend[
    Table[Table[Round[1/2 g i^2 (1 + 0.2 RandomReal[]),
        0.01], {i, 1, 10}], {10}], Range[1, 10]] //
   Transpose;
data // TableForm

Export["~/Desktop/mydata.csv", data];

data1 = Import["~/Desktop/mydata.csv"];
```

That is this. So, this is exactly what I have, but I have used the other approach to do transpose. And then finally, I want to store all of this data in a variable, which I am calling data so that I can use this variable itself later on. To do my analysis, I am going to be working on data. Okay, so I am going to erase all this. So, this was just some explanation of all the steps involved in working out this code.

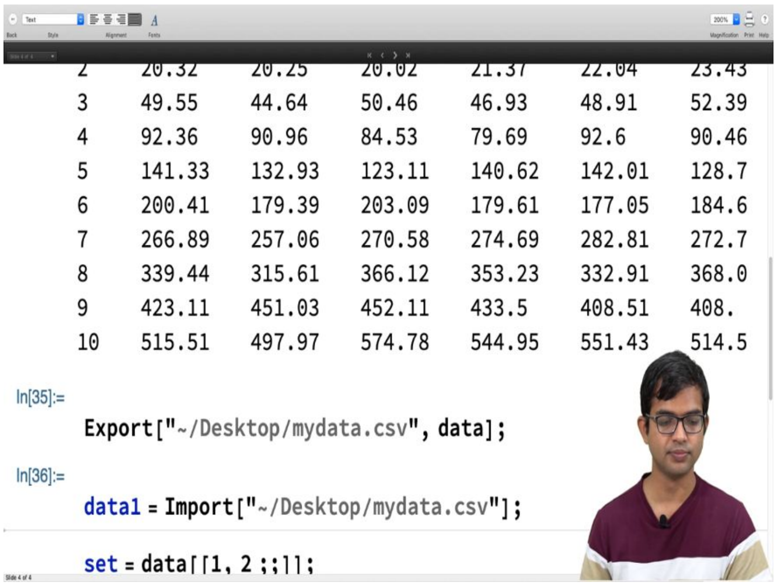
So, let me remove this, we would be bit cluttering it otherwise, but what I just did is also illustrative of how one looks at Mathematica code or code in general is to be analysed as you know bits and pieces. You take them apart, you pull them apart, you ask what is each individual bit doing and then you try to integrate them and when you are integrating them, you must figure out how to make it compact.

On the one hand, the code should be as clean and as minimalist as possible, but it must also be efficient, right. So, these are things that you would pick up as you go along with the more you code, the better you become. Alright, so data once we have, we want to look at it with this table form. So, let me just implement this whole thing again, I have this data, and then I will also do table form. So, it is already there, table form is also taken care of.

(Refer Slide Time: 11: 58)



```
Out[34]//TableForm=
      1    5.15     5.68     5.87     5.14     5.51     5.54
      2    20.32    20.25    20.02    21.37    22.04    23.43
      3    49.55    44.64    50.46    46.93    48.91    52.39
      4    92.36    90.96    84.53    79.69    92.6     90.46
      5    141.33   132.93   123.11   140.62   142.01   128.7
      6    200.41   179.39   203.09   179.61   177.05   184.6
      7    266.89   257.06   270.58   274.69   282.81   272.7
      8    339.44   315.61   366.12   353.23   332.91   368.0
      9    423.11   451.03   452.11   433.5    408.51   8.
      10   515.51   497.97   574.78   544.95   551.43   4.5

       Export["~/Desktop/mydata.csv", data];

       data1 = Import["~/Desktop/mydata.csv"];
```



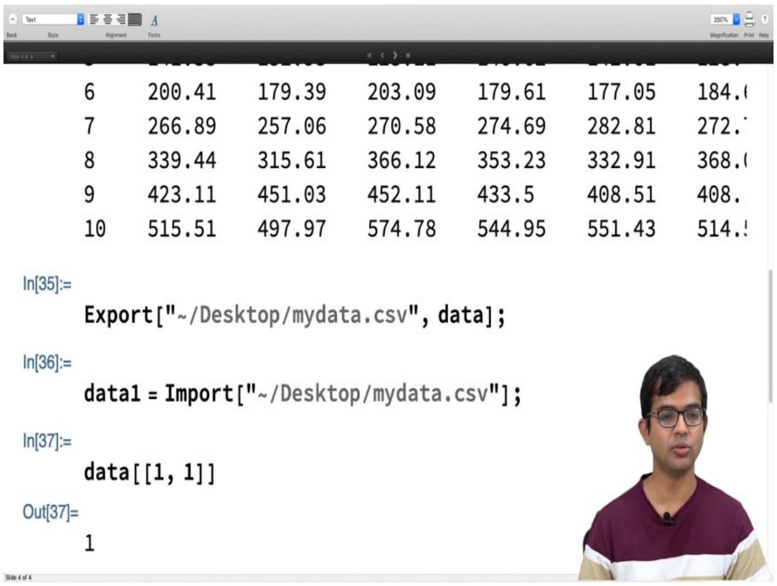
```
      2    20.32    20.25    20.02    21.37    22.04    23.43
      3    49.55    44.64    50.46    46.93    48.91    52.39
      4    92.36    90.96    84.53    79.69    92.6     90.46
      5    141.33   132.93   123.11   140.62   142.01   128.7
      6    200.41   179.39   203.09   179.61   177.05   184.6
      7    266.89   257.06   270.58   274.69   282.81   272.7
      8    339.44   315.61   366.12   353.23   332.91   368.0
      9    423.11   451.03   452.11   433.5    408.51   408.
      10   515.51   497.97   574.78   544.95   551.43   514.5

In[35]:=
       Export["~/Desktop/mydata.csv", data];

In[36]:=
       data1 = Import["~/Desktop/mydata.csv"];

       set = data[[1, 2 ;;]];
```

Finally, when I have all of this data, I might want to export this entire data to my desktop, for example, or somewhere else. And so this is the syntax, we have already seen this. And then I could do this and then again, import it back right. You could pretend for a moment that all this information was obtained, was got from some experiment. And so then you would just start from this step alone. You do not go, do this whole business of getting the data.

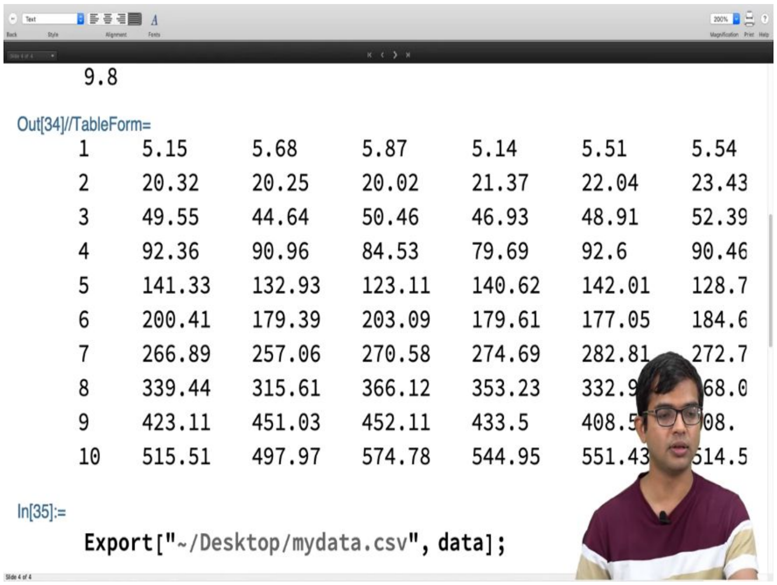
We created the data using Mathematica and next we are going to analyse the data using Mathematica. So, in some sense, all that we did so far is an artificial way of generating data. So, so far, we are not really applied the concepts that we discussed earlier. So, in some sense, only what follows is the critical aspects. So far, it was some tricks to generate data, you can say or it is a method to do simulation right. So, we might get into a systematic discussion of simulation techniques at a later time or maybe in a more advanced course. But for now, all we want to do is actually to be able to interpret data right.

So, we have, you know, all the data is stored in this variable called data 1, but let me just do my analysis on my data. So, now there is more syntax. So, suppose I want to just look at, let us say, I am interested in looking at some particular number. So Mathematica tells us that your variable, you have access to any column and any row and in order to do that, this is, suppose I want to get the first row first column data. This is how I would do it, I would take data of.

(Refer Slide Time: 13:57)



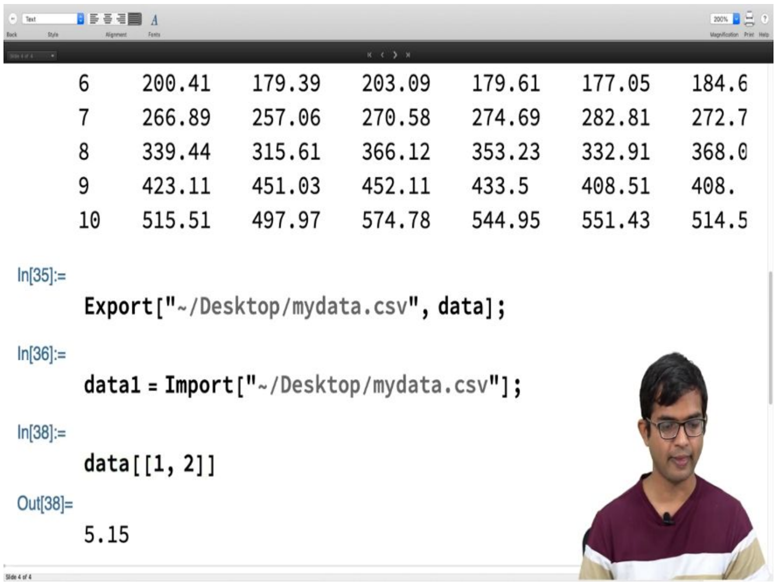
```
    6    200.41   179.39   203.09   179.61   177.05   184.
    7    266.89   257.06   270.58   274.69   282.81   272.
    8    339.44   315.61   366.12   353.23   332.91   368.
    9    423.11   451.03   452.11   433.5    408.51   408.
    10   515.51   497.97   574.78   544.95   551.43   514.

In[35]:=
    Export["~/Desktop/mydata.csv", data];

In[36]:=
    data1 = Import["~/Desktop/mydata.csv"];

In[37]:=
    data[[1, 1]]

Out[37]=
    1
```



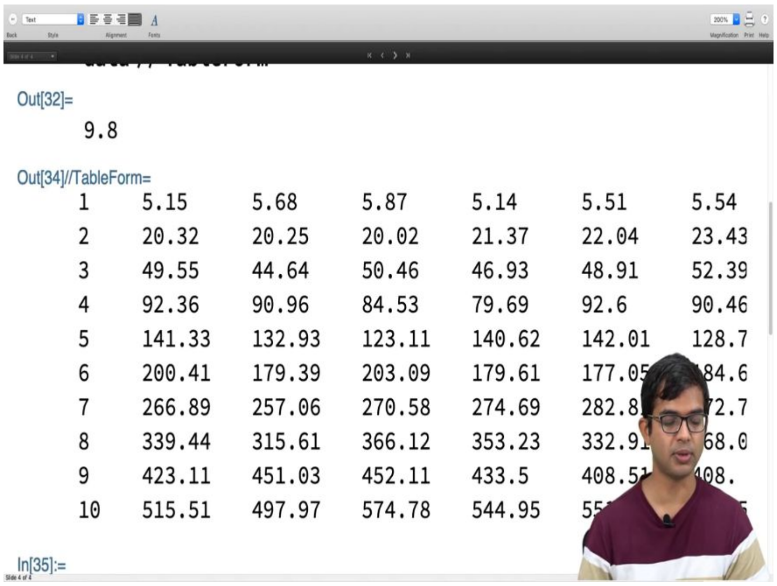
```
    9.8

Out[34]//TableForm=
    1    5.15     5.68     5.87     5.14     5.51     5.54
    2    20.32    20.25    20.02    21.37    22.04    23.43
    3    49.55    44.64    50.46    46.93    48.91    52.39
    4    92.36    90.96    84.53    79.69    92.6     90.46
    5    141.33   132.93   123.11   140.62   142.01   128.7
    6    200.41   179.39   203.09   179.61   177.05   184.6
    7    266.89   257.06   270.58   274.69   282.81   272.7
    8    339.44   315.61   366.12   353.23   332.9    68.0
    9    423.11   451.03   452.11   433.5    408.5    08.
    10   515.51   497.97   574.78   544.95   551.43   514.5

In[35]:=
    Export["~/Desktop/mydata.csv", data];
```

So, I would have to put this inside, enclose them in two brackets of this kind. And then, if I want to go to 1,1 and then if I do shift enter, then there you go, you have the number 1 appears. So, if I had done 1,2 then I have 5.15.

(Refer Slide Time: 14:21)



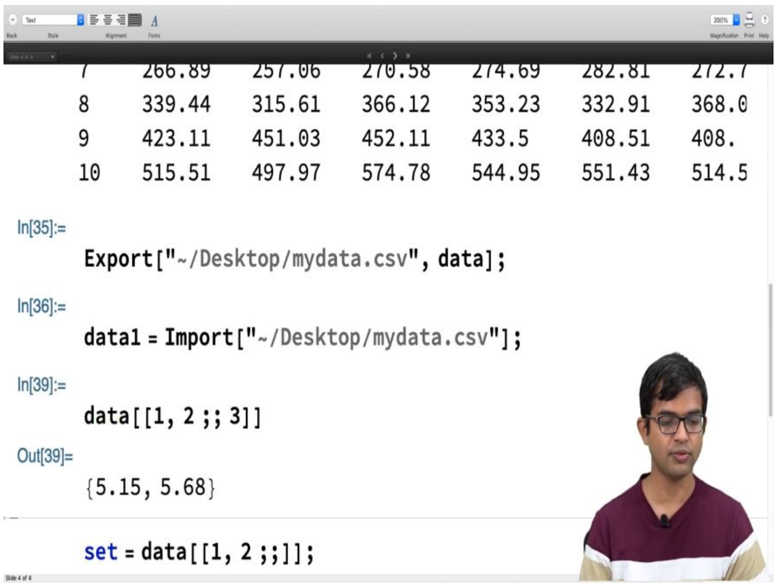
```
      6    200.41   179.39   203.09   179.61   177.05   184.6
      7    266.89   257.06   270.58   274.69   282.81   272.7
      8    339.44   315.61   366.12   353.23   332.91   368.0
      9    423.11   451.03   452.11   433.5    408.51   408.
      10   515.51   497.97   574.78   544.95   551.43   514.5

In[35]:=
      Export["~/Desktop/mydata.csv", data];

In[36]:=
      data1 = Import["~/Desktop/mydata.csv"];

In[38]:=
      data[[1, 2]]

Out[38]=
      5.15
```



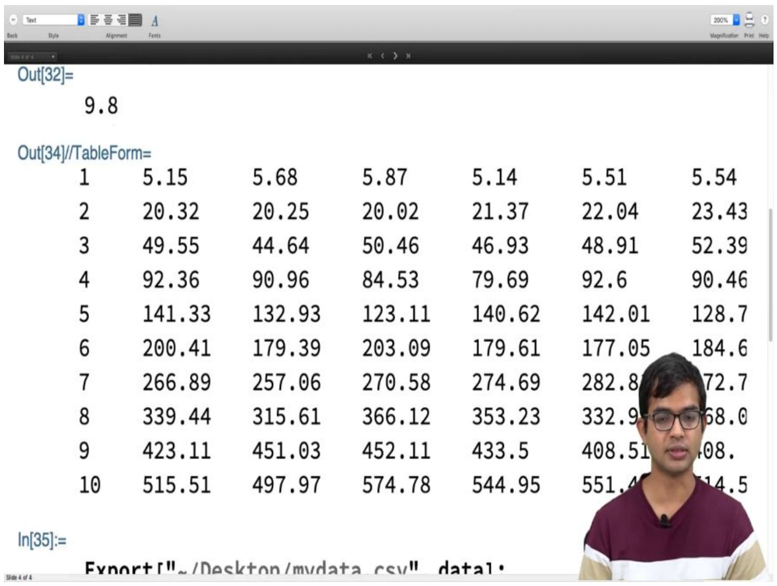
```
Out[32]=
      9.8

Out[34]//TableForm=
      1    5.15     5.68     5.87     5.14     5.51     5.54
      2    20.32    20.25    20.02    21.37    22.04    23.43
      3    49.55    44.64    50.46    46.93    48.91    52.39
      4    92.36    90.96    84.53    79.69    92.6     90.46
      5    141.33   132.93   123.11   140.62   142.01   128.7
      6    200.41   179.39   203.09   179.61   177.05
      7    266.89   257.06   270.58   274.69   282.8
      8    339.44   315.61   366.12   353.23   332.9
      9    423.11   451.03   452.11   433.5    408.5
      10   515.51   497.97   574.78   544.95

In[35]:=
```

Again, not a surprise. So it is the first row, second column. Now, but suppose, I want all the columns, but in the first row, then the syntax for this is to do this.

(Refer Slide Time: 14:38)



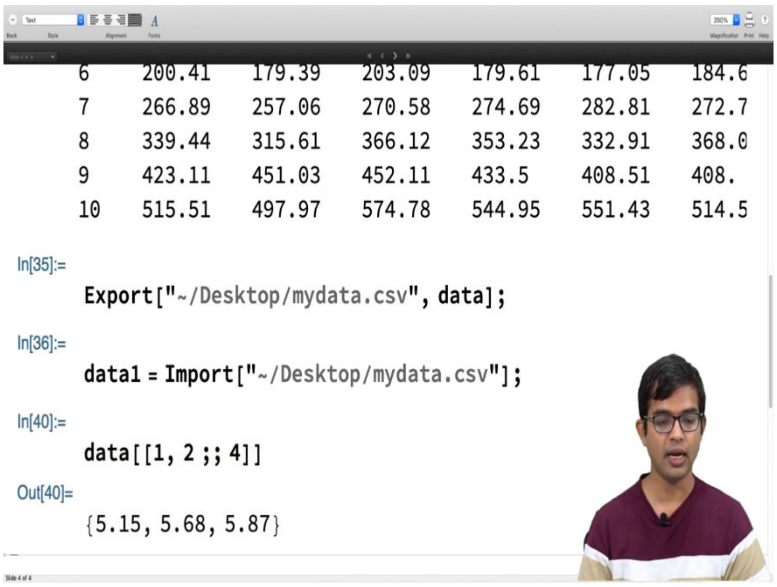
```
 7    266.89   257.06   270.58   274.69   282.81   272.7
 8    339.44   315.61   366.12   353.23   332.91   368.0
 9    423.11   451.03   452.11   433.5    408.51   408.
 10   515.51   497.97   574.78   544.95   551.43   514.5

In[35]:=
      Export["~/Desktop/mydata.csv", data];

In[36]:=
      data1 = Import["~/Desktop/mydata.csv"];

In[39]:=
      data[[1, 2 ;; 3]]
Out[39]=
      {5.15, 5.68}

      set = data[[1, 2 ;;]];
```



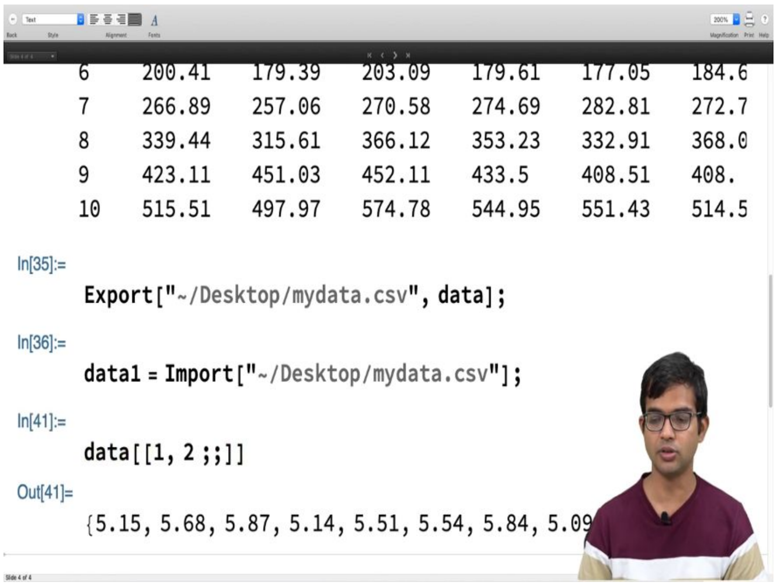
```
Out[32]=
      9.8

Out[34]//TableForm=
 1    5.15     5.68     5.87     5.14     5.51     5.54
 2    20.32    20.25    20.02    21.37    22.04    23.43
 3    49.55    44.64    50.46    46.93    48.91    52.39
 4    92.36    90.96    84.53    79.69    92.6     90.46
 5    141.33   132.93   123.11   140.62   142.01   128.7
 6    200.41   179.39   203.09   179.61   177.05   184.6
 7    266.89   257.06   270.58   274.69   282.8    72.7
 8    339.44   315.61   366.12   353.23   332.9    68.0
 9    423.11   451.03   452.11   433.5    408.51   08.
 10   515.51   497.97   574.78   544.95   551.4    14.5

In[35]:=
      Export["~/Desktop/mydata.csv", data];
```

So, it can go from 2. So if I were to go from the second column to third column that is then then that syntax is this. So, the starting column and the ending column should be separated by these semicolon and semicolon. So if I do this I have 5.15 and 5.68. So, if I wanted information from the second column all the way up to the fourth column then I would do this and I have these three numbers.

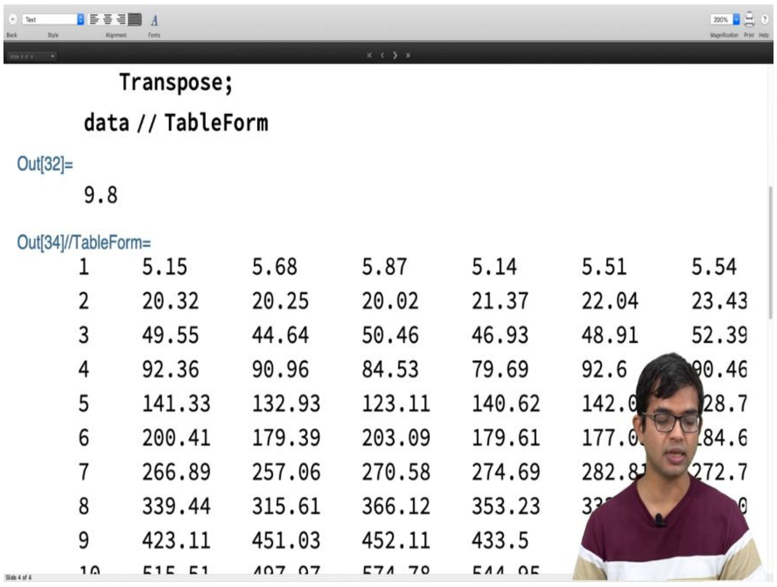
(Refer Slide Time: 15:03)



```
6     200.41   179.39   203.09   179.61   177.05   184.6
7     266.89   257.06   270.58   274.69   282.81   272.7
8     339.44   315.61   366.12   353.23   332.91   368.0
9     423.11   451.03   452.11   433.5    408.51   408.
10    515.51   497.97   574.78   544.95   551.43   514.5

In[35]:=
      Export["~/Desktop/mydata.csv", data];

In[36]:=
      data1 = Import["~/Desktop/mydata.csv"];

In[40]:=
      data[[1, 2 ;; 4]]

Out[40]=
      {5.15, 5.68, 5.87}
```



```
6     200.41   179.39   203.09   179.61   177.05   184.6
7     266.89   257.06   270.58   274.69   282.81   272.7
8     339.44   315.61   366.12   353.23   332.91   368.0
9     423.11   451.03   452.11   433.5    408.51   408.
10    515.51   497.97   574.78   544.95   551.43   514.5

In[35]:=
      Export["~/Desktop/mydata.csv", data];

In[36]:=
      data1 = Import["~/Desktop/mydata.csv"];

In[41]:=
      data[[1, 2 ;;]]

Out[41]=
      {5.15, 5.68, 5.87, 5.14, 5.51, 5.54, 5.84, 5.09
```

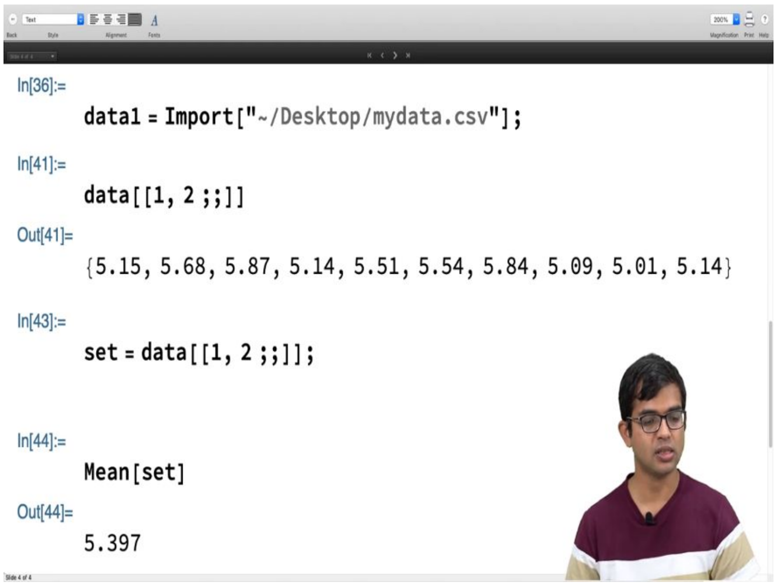


```
   Transpose;
 data // TableForm
Out[32]=
 9.8

Out[34]//TableForm=
 1    5.15     5.68     5.87     5.14     5.51     5.54
 2    20.32    20.25    20.02    21.37    22.04    23.43
 3    49.55    44.64    50.46    46.93    48.91    52.39
 4    92.36    90.96    84.53    79.69    92.6     90.46
 5    141.33   132.93   123.11   140.62   142.0    28.7
 6    200.41   179.39   203.09   179.61   177.0    84.6
 7    266.89   257.06   270.58   274.69   282.8    272.7
 8    339.44   315.61   366.12   353.23   33
 9    423.11   451.03   452.11   433.5
 10   515.51   497.97   574.78   544.95
```

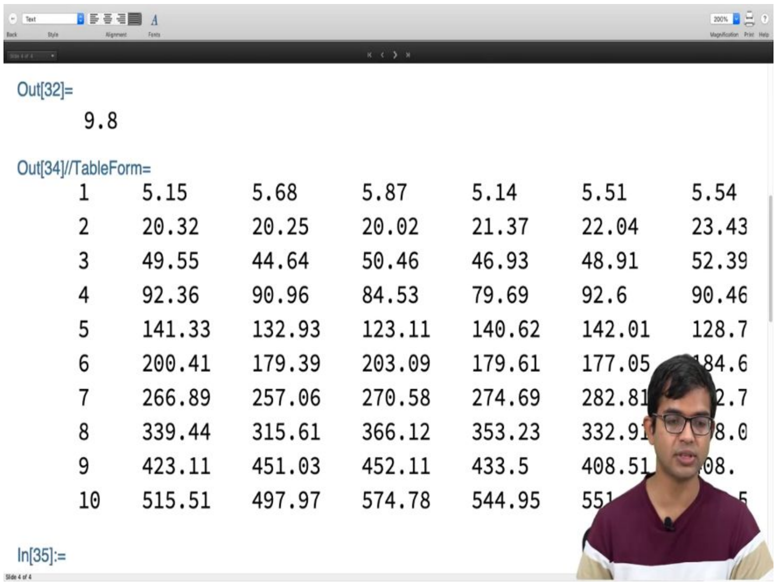
And if I want to go all the way till the last column, I could, of course, just specify column number 11, or I could just leave it blank. And it is going to give me data for all the columns right. So, I am interested in this particular number. So, this is just to illustrate how I can go ahead and compute the average of this.

So, after all, what are we interested in? We are interested in finally doing data analysis row by row. After one second, I want access to the mean and the error bar. An estimate of the mean and an estimate of the error in my knowledge of the mean. If I have this information, then I can go ahead and do further data analysis on this. Now, so let me call this whole quantity as set right. If I put a semicolon, it does not show all the numbers, but it is stored there, I have to still implement it.

(Refer Slide Time: 15:56)



```
In[36]:=
        data1 = Import["~/Desktop/mydata.csv"];

In[41]:=
        data[[1, 2 ;;]]
Out[41]=
        {5.15, 5.68, 5.87, 5.14, 5.51, 5.54, 5.84, 5.09, 5.01, 5.14}

In[43]:=
        set = data[[1, 2 ;;]];

In[44]:=
        Mean[set]
Out[44]=
        5.397
```

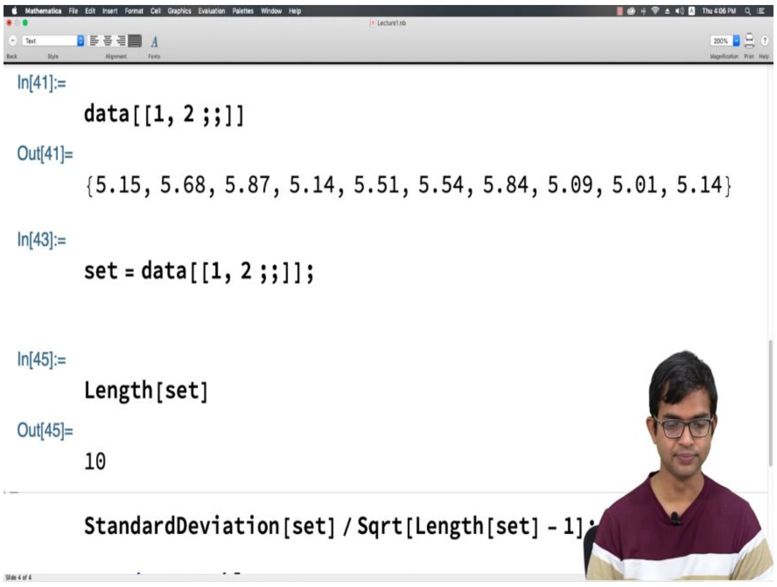


```
Out[32]=
        9.8

Out[34]//TableForm=
       1     5.15      5.68      5.87      5.14      5.51      5.54
       2     20.32     20.25     20.02     21.37     22.04     23.43
       3     49.55     44.64     50.46     46.93     48.91     52.39
       4     92.36     90.96     84.53     79.69     92.6      90.46
       5     141.33    132.93    123.11    140.62    142.01    128.7
       6     200.41    179.39    203.09    179.61    177.05    84.6
       7     266.89    257.06    270.58    274.69    282.81    2.7
       8     339.44    315.61    366.12    353.23    332.91    8.0
       9     423.11    451.03    452.11    433.5     408.51    08.
       10    515.51    497.97    574.78    544.95    551       5

In[35]:=
```

So, I can go ahead and compute the mean of this. So, mean of this is simply the average of these 10 numbers right. So, I mean, so you see that I go from 2 all the way up to the last column. The first column is suppressed, because the first column is data of the time, it is only from second column onwards that we have information about the distance. So, the mean is 5.397. So, after one second, the mean distance covered by this fictitious particle is 5.397.

So, then we also have this length of set, is a useful, length is a useful useful function. So, it just simply tells you how many data points do you have. So, of course, we know that there is 10 of them. So, let me compute this. So, if I were to do mean I get this, I could have also done length of this.

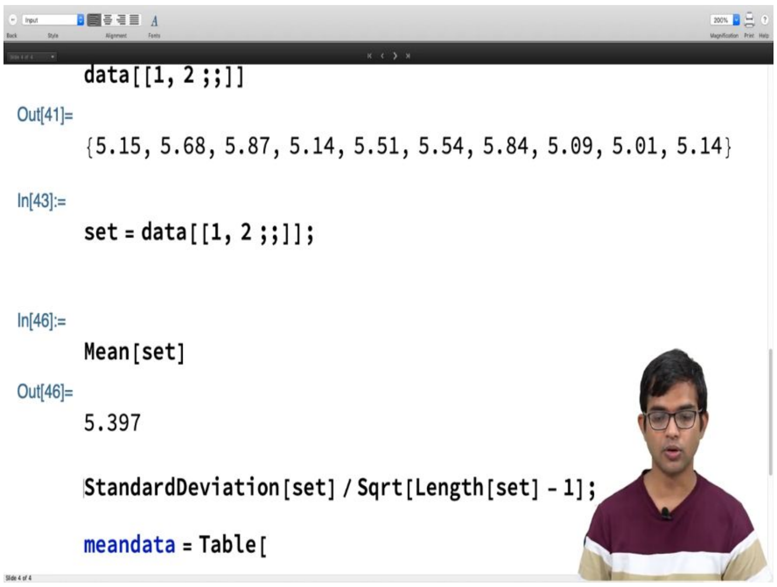
(Refer Slide Time: 16:55)



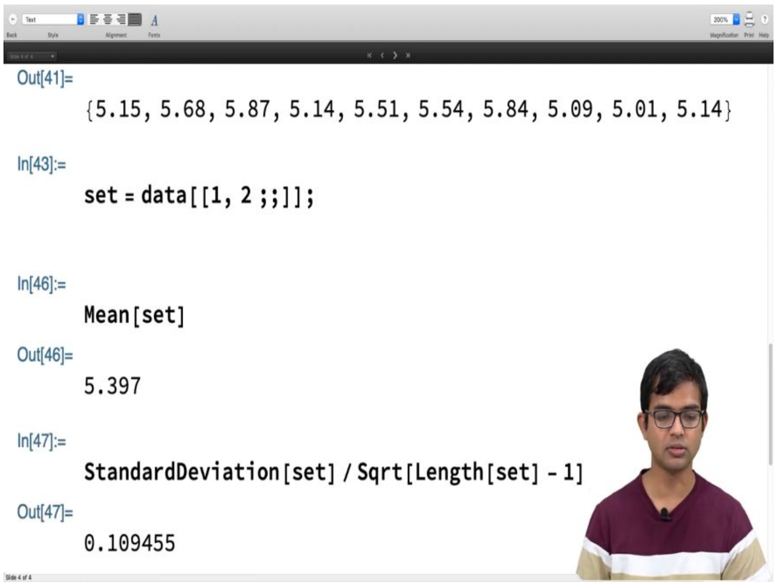
```
In[41]:=
        data[[1, 2 ;;]]
Out[41]=
        {5.15, 5.68, 5.87, 5.14, 5.51, 5.54, 5.84, 5.09, 5.01, 5.14}

In[43]:=
        set = data[[1, 2 ;;]];

In[45]:=
        Length[set]
Out[45]=
        10

        StandardDeviation[set] / Sqrt[Length[set] - 1];
```

Not a surprise that it is 10 right. So, I want to use, so you remember that you must find the sample mean is the mean of the distribution. And the sample standard deviation divided by √N-1 is a measure of the error bar. So, that is what I have here, I have square root of this, √N is nothing but √Length[set]-1. It is a square root of this whole thing. That is what I have down here. So, let me change this back to mean of set.

(Refer Slide Time: 17:38)



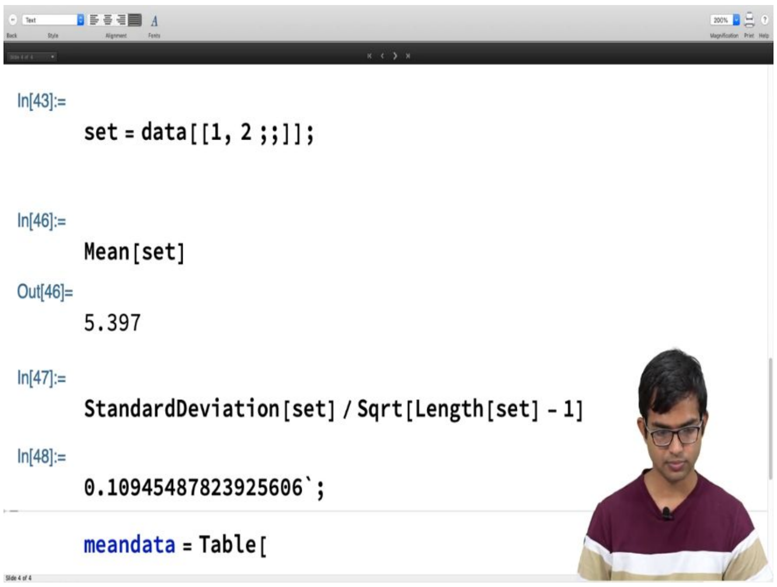
```
data[[1, 2 ;;]]
Out[41]=
{5.15, 5.68, 5.87, 5.14, 5.51, 5.54, 5.84, 5.09, 5.01, 5.14}
In[43]:=
set = data[[1, 2 ;;]];

In[46]:=
Mean[set]
Out[46]=
5.397

StandardDeviation[set] / Sqrt[Length[set] - 1];

meandata = Table[
```



```
Out[41]=
{5.15, 5.68, 5.87, 5.14, 5.51, 5.54, 5.84, 5.09, 5.01, 5.14}
In[43]:=
set = data[[1, 2 ;;]];

In[46]:=
Mean[set]
Out[46]=
5.397
In[47]:=
StandardDeviation[set] / Sqrt[Length[set] - 1]
Out[47]=
0.109455
```

Then I have standard deviation of set divided by √Length[set]-1. I can go ahead and find out this number, this is what the number is. I can also suppress it.

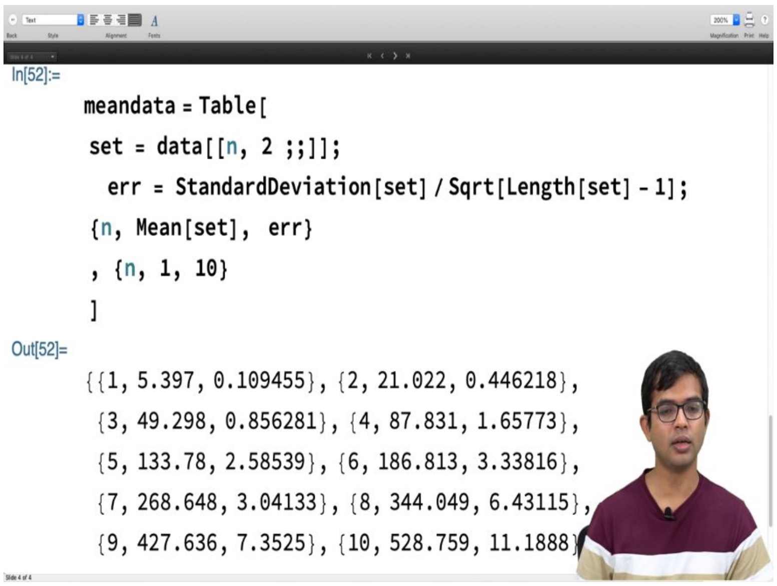
(Refer Slide Time: 17:53)



```
In[43]:=
        set = data[[1, 2 ;;]];

In[46]:=
        Mean[set]
Out[46]=
        5.397

In[47]:=
        StandardDeviation[set] / Sqrt[Length[set] - 1]

In[48]:=
        0.10945487823925606`;

        meandata = Table[
```

So, let me get rid of this cell, I can redo this just to check that it is all good. There you go and then I will suppress this.

(Refer Slide Time: 18:27)



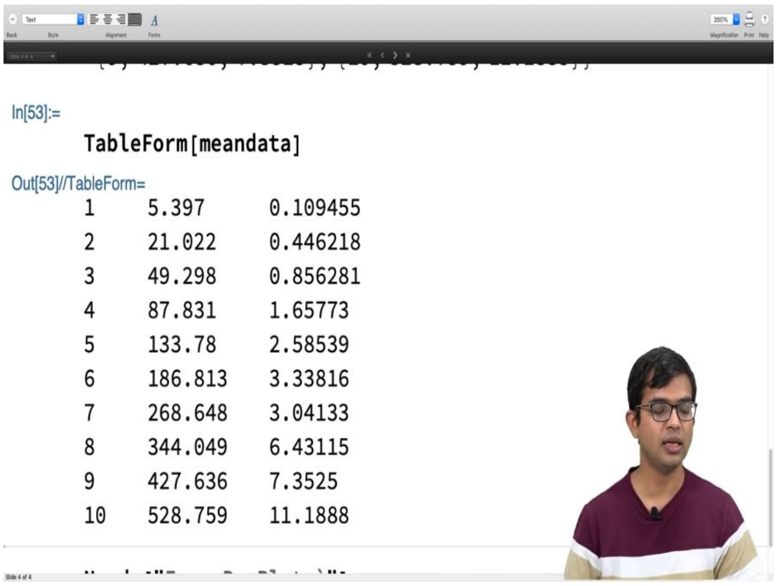
```
In[52]:=
        meandata = Table[
         set = data[[n, 2 ;;]];
          err = StandardDeviation[set] / Sqrt[Length[set] - 1];
         {n, Mean[set], err}
         , {n, 1, 10}
         ]
Out[52]=
        {{1, 5.397, 0.109455}, {2, 21.022, 0.446218},
         {3, 49.298, 0.856281}, {4, 87.831, 1.65773},
         {5, 133.78, 2.58539}, {6, 186.813, 3.33816},
         {7, 268.648, 3.04133}, {8, 344.049, 6.43115},
         {9, 427.636, 7.3525}, {10, 528.759, 11.1888}
```

Okay so then what I want to do next is actually carry out this kind of an exercise for each row right. Ultimately, what do I want? I want to have, so I know how to do the data analysis, but I want a compact, quick way to do this data analysis row by row for each of the times. So, there is again recourse to this table command helps. So, we will do set not just for the first row so, let us say that we are doing this set is equal to data of the nth row and again, columns going from 2 all the way up to the last column, and I will call it set.

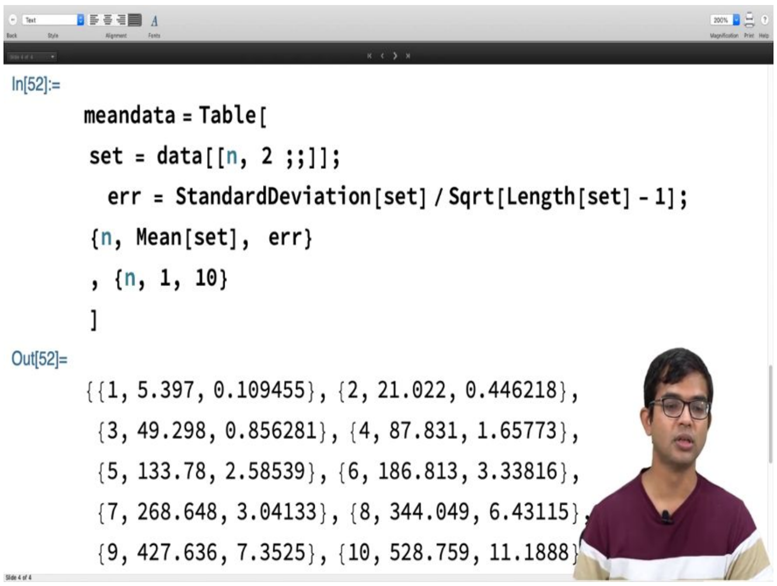
So, table is a new array which I am defining as mean data right. It is going to have the time n, will go from 1 to 10. And it is going to have the mean of the set, and it is going to have the error bar. So, error bar I know how to compute, which is just simply. So, in fact if I want to clean this up a little more, let me copy this or cut this and call this.

Alright, so I have defined something called err or error. And so at each step, it is going to compute the error. And then, I have to store this information. I want to store the time at which I am doing this, carrying out this, finding out the mean, and then the mean and the error at that particular time. So, if I were to just run this. Yeah, so there you go.

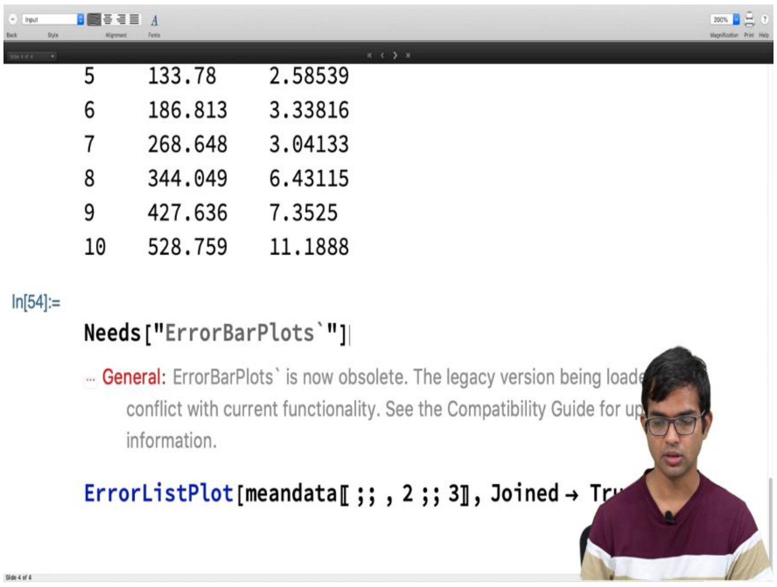
(Refer Slide Time: 20:22)



```
In[53]:=
        TableForm[meandata]
Out[53]//TableForm=
        1     5.397      0.109455
        2     21.022     0.446218
        3     49.298     0.856281
        4     87.831     1.65773
        5     133.78     2.58539
        6     186.813    3.33816
        7     268.648    3.04133
        8     344.049    6.43115
        9     427.636    7.3525
        10    528.759    11.1888
```



```
In[52]:=
        meandata = Table[
         set = data[[n, 2 ;;]];
          err = StandardDeviation[set] / Sqrt[Length[set] - 1];
         {n, Mean[set], err}
         , {n, 1, 10}
         ]
Out[52]=
        {{1, 5.397, 0.109455}, {2, 21.022, 0.446218},
         {3, 49.298, 0.856281}, {4, 87.831, 1.65773},
         {5, 133.78, 2.58539}, {6, 186.813, 3.33816},
         {7, 268.648, 3.04133}, {8, 344.049, 6.43115},
         {9, 427.636, 7.3525}, {10, 528.759, 11.1888}
```



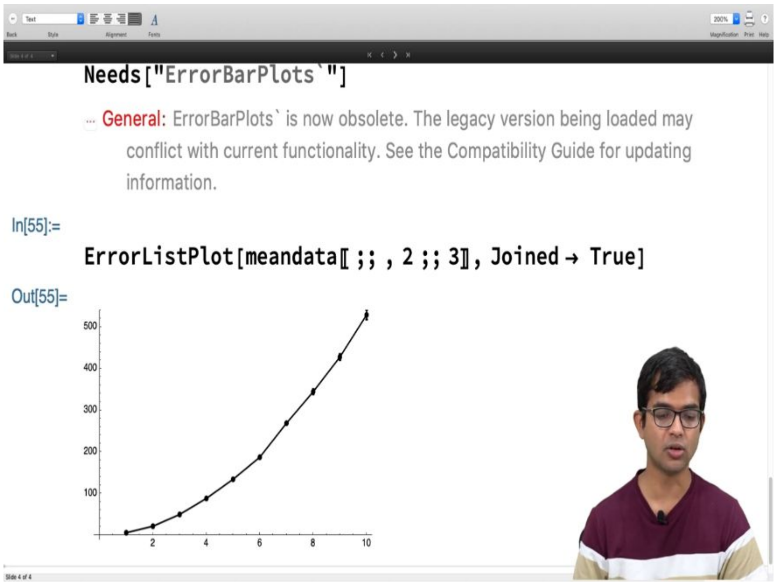
```
5     133.78     2.58539
6     186.813    3.33816
7     268.648    3.04133
8     344.049    6.43115
9     427.636    7.3525
10    528.759    11.1888
```

```
In[54]:=
Needs["ErrorBarPlots`"]
General: ErrorBarPlots` is now obsolete. The legacy version being loade
    conflict with current functionality. See the Compatibility Guide for up
    information.

ErrorListPlot[meandata[[;; , 2 ;; 3]], Joined → Tr
```

So, I have 1, 5.37397 and 0.1 and so on. So finally, I want to put this in a nice form, of course. It does not look so nice if it is just the way it is. So, you go to table form of mean data. So, then you see that after one second, it is roughly around 5.4. And then it has with errors of 0.1 and 21 bigger errors. And so the errors also grow because that is the way I set it up. So, you could set up a different model where the errors are roughly the same but so that is that is a whole different discussion altogether. But suppose these are the numbers, this is the way to do the data analysis.

So, now the final thing to do is, of course to make a plot of all of this right. So, this last piece of code you should study a little more carefully and then you will see that you know this again using the nested argument. You start from the table command, you start with each of these different functions, and then and then use them to, in a judicious way you can get this mean data. Final thing is you have to load something called error bar plots if you want to plot this with error bars. And then, ok so it is not clear whether this is going to show. Yeah, it does show up.

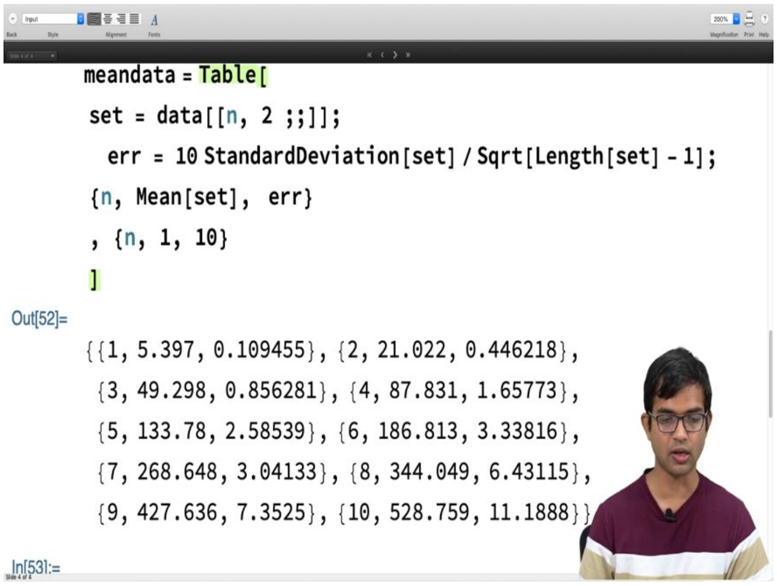
(Refer Slide Time: 12:50)



```
Needs["ErrorBarPlots`"]

General: ErrorBarPlots` is now obsolete. The legacy version being loaded may
  conflict with current functionality. See the Compatibility Guide for updating
  information.

In[55]:=
ErrorListPlot[meandata[[;; , 2 ;; 3]], Joined → True]

Out[55]=
```

So, is now obsolete, there is a new version of this available so we can work this out, we can figure this out. So, but anyway, it seems to work. So, there is some comment that Mathematica gives us. But even so, this is working out alright. As you can see the error bars are there, but they are not so visible because they are too tiny. So, let me in order to just exaggerate how these error bars are, I am going to just put in a factor of 10.

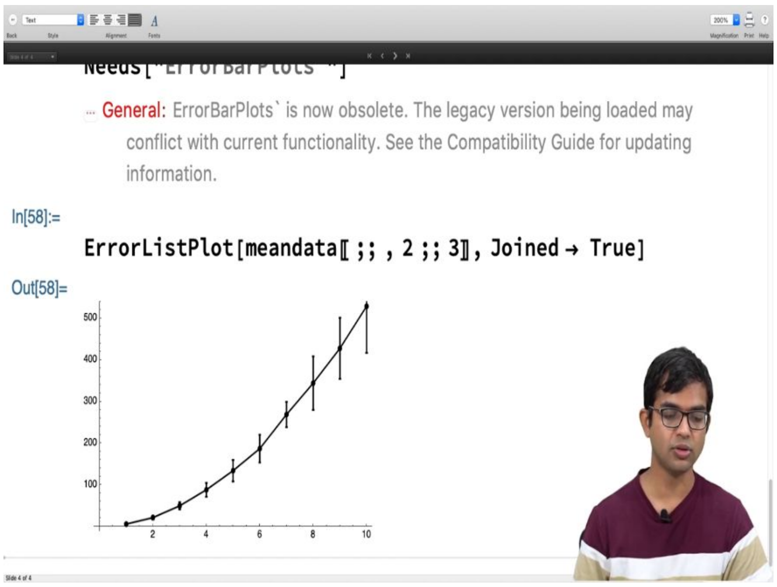
(Refer Slide Time: 22:15)



```
meandata = Table[
 set = data[[n, 2 ;;]];
   err = 10 StandardDeviation[set] / Sqrt[Length[set] - 1];
 {n, Mean[set], err}
 , {n, 1, 10}
 ]

Out[52]=
{{1, 5.397, 0.109455}, {2, 21.022, 0.446218},
 {3, 49.298, 0.856281}, {4, 87.831, 1.65773},
 {5, 133.78, 2.58539}, {6, 186.813, 3.33816},
 {7, 268.648, 3.04133}, {8, 344.049, 6.43115},
 {9, 427.636, 7.3525}, {10, 528.759, 11.1888}}

In[53]:=
```

It is a common factor of 10 for all and to show you how this is going to explicitly look like errors. Yeah, so there you go.

(Refer Slide Time: 22:26)



```
Needs["ErrorBarPlots`"]
```

... General: ErrorBarPlots` is now obsolete. The legacy version being loaded may conflict with current functionality. See the Compatibility Guide for updating information.

```
In[58]:=
ErrorListPlot[meandata[[ ;; , 2 ;; 3]], Joined → True]
Out[58]=
```

So, this is what one might see in a real experiment. But, of course, these numbers are exaggerated here. And also you see that the errors are changing as a function of time, which may or may not be the way a real world system works. So, anyway, so the whole point of this discussion is not really about whether the model we have for generating these these errors is the is the right one or not, but it is simply about doing data analysis and to illustrate how with Mathematica there is a quick and powerful way to take up all this data and to to analyse this using our method, and then to make a plot.

So, the key main message to take from this this lecture is that if you have a bunch of repeated experimental data then the average of all of these numbers is a good estimate of the mean of the distribution and the standard deviation of the sample standard deviation divided by √N-1, which if you have a lot of points, you can just take it to be √N.

So, it is $\sigma/\sqrt{N}$ is a good estimate of the error bars. And with Mathematica, we are able to extract all of this information and get an error plot with error bars in a nice compact way. That is this lecture. Thank you.