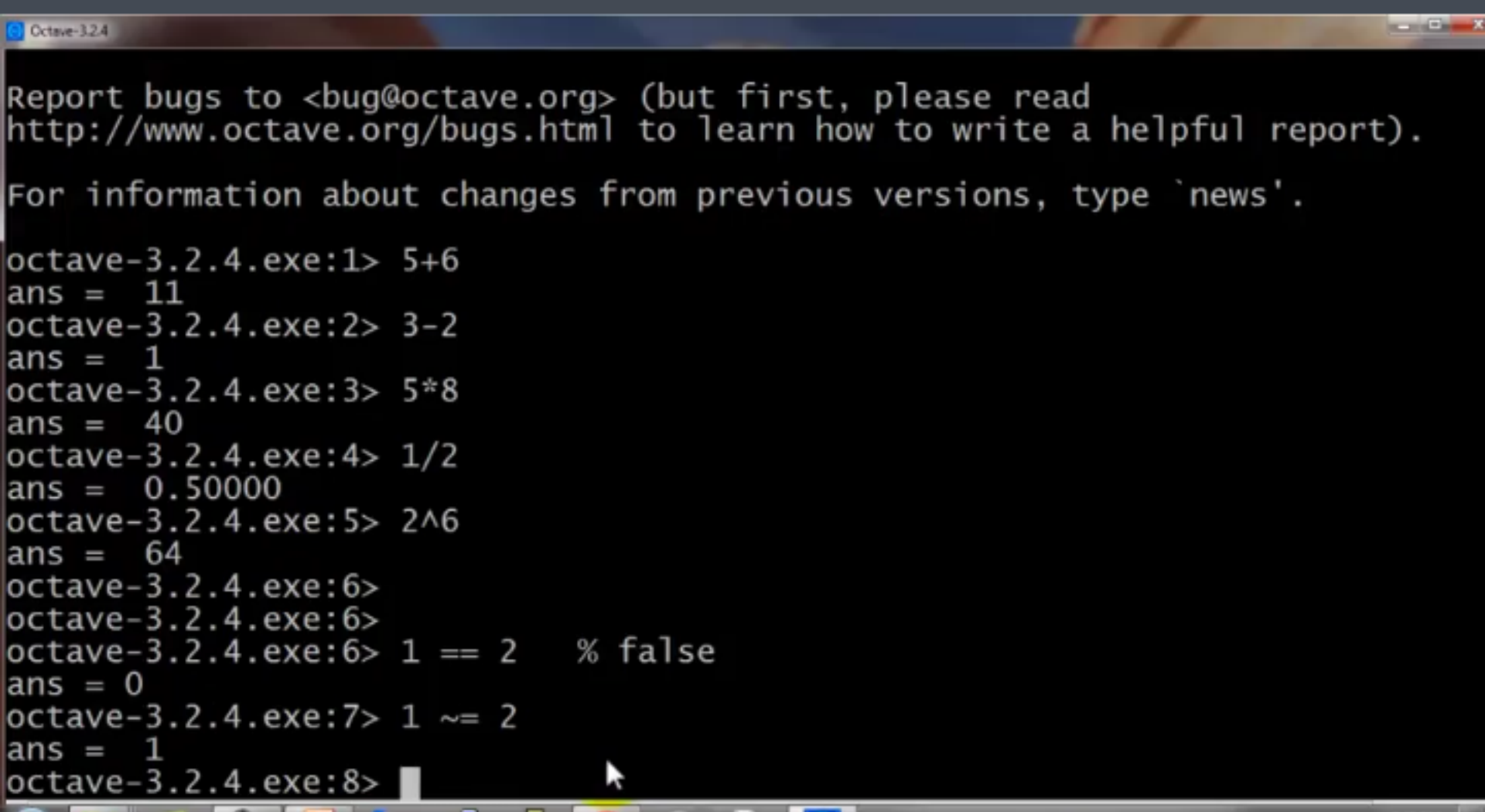
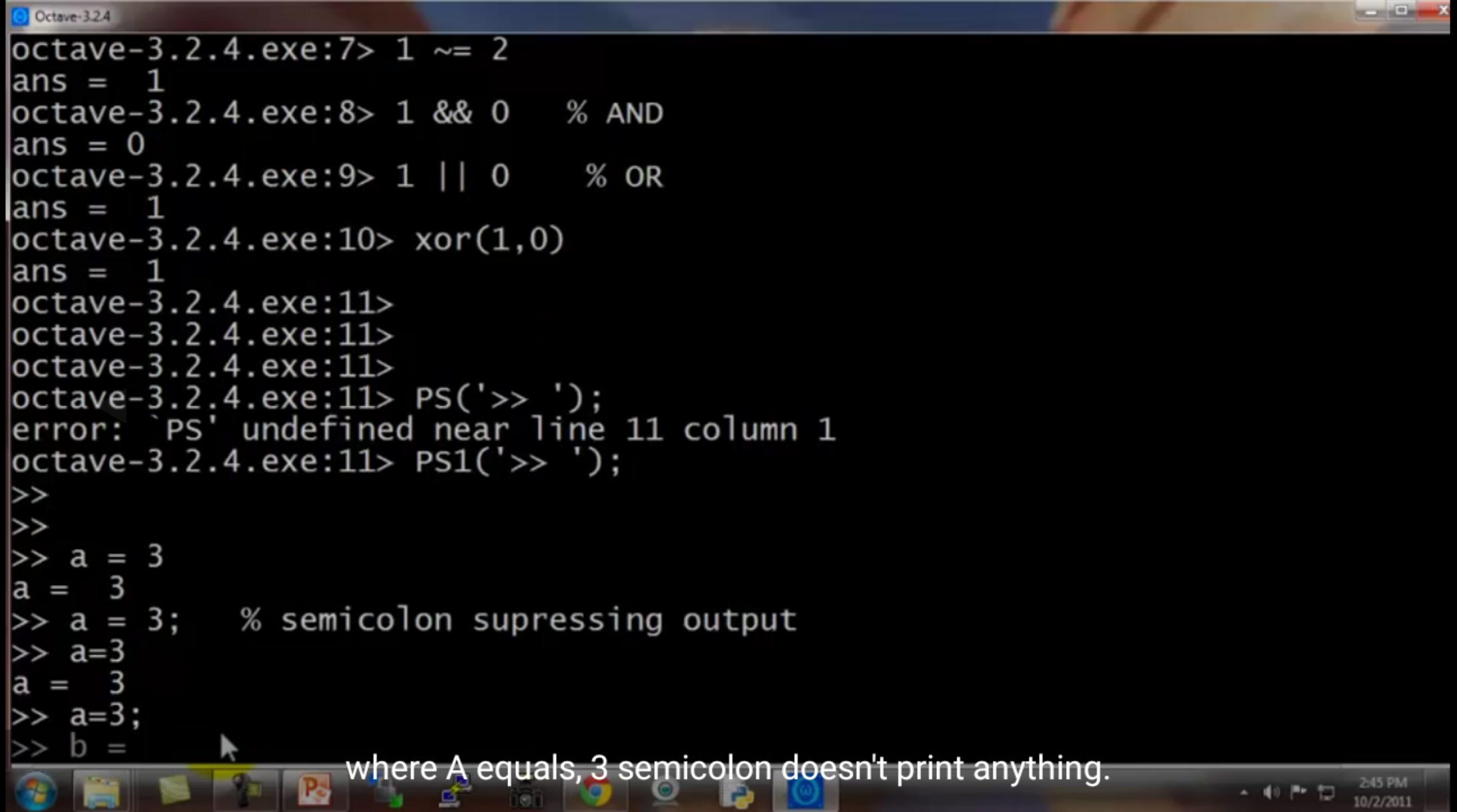
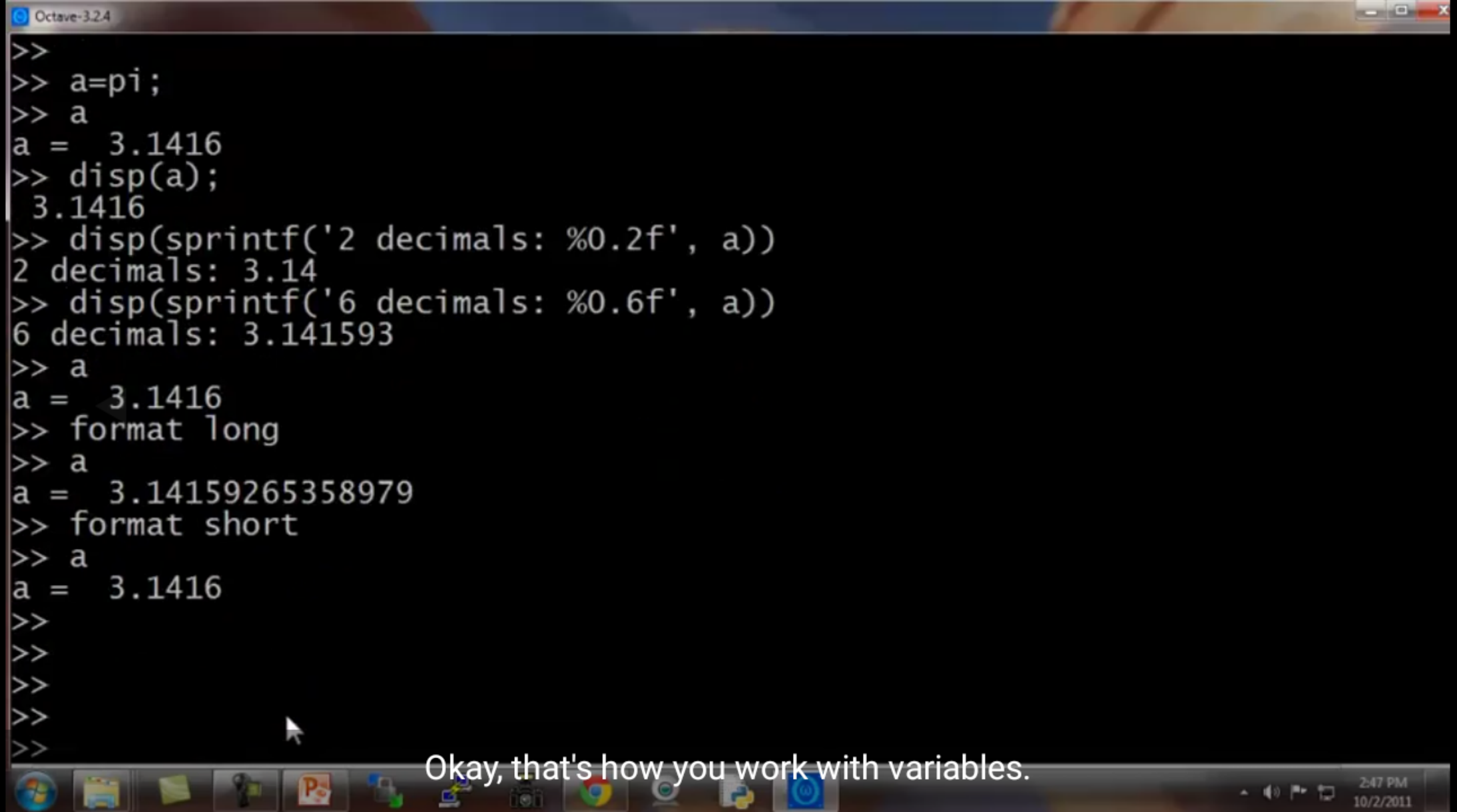
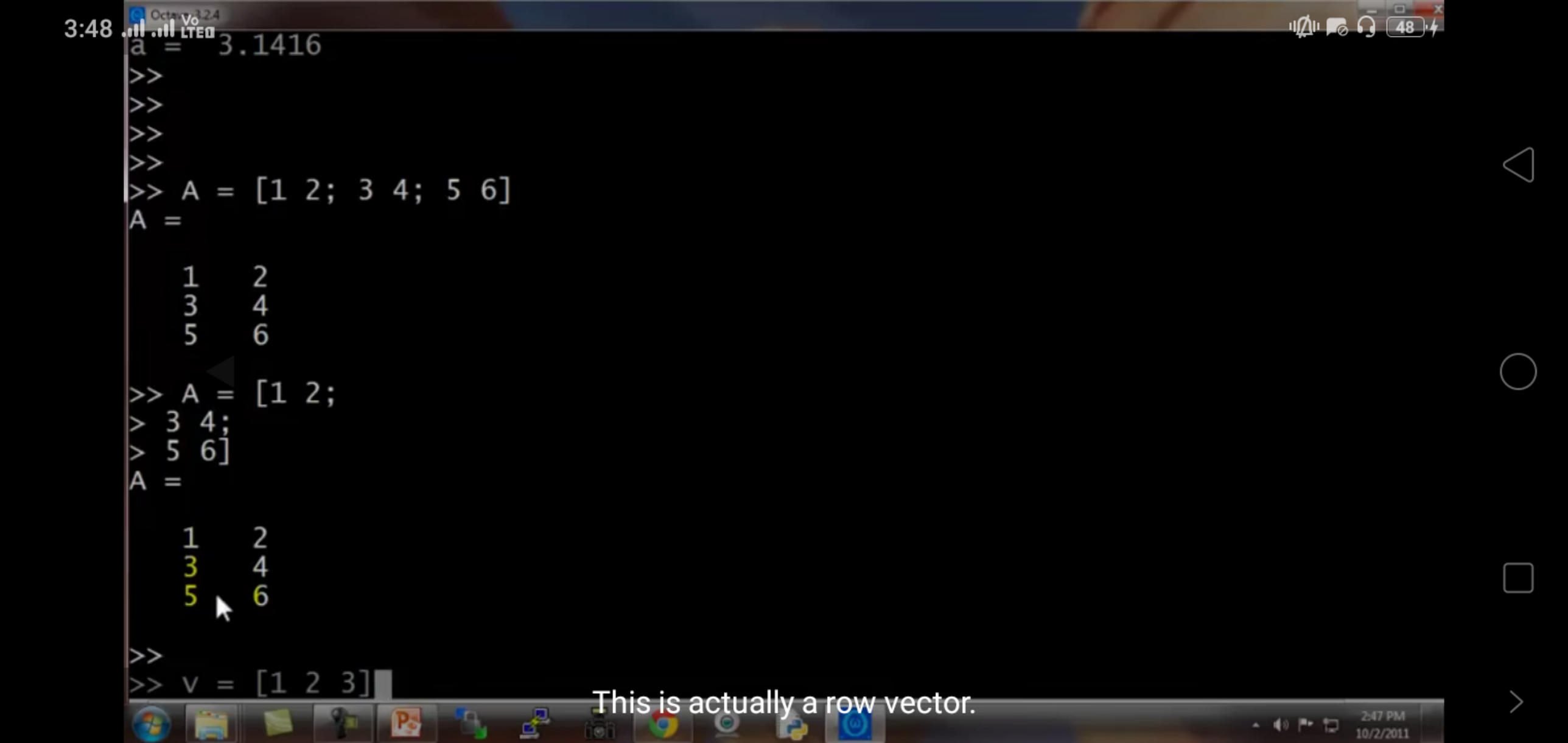


operations. So one equals two. This evaluates to false. The percent command here means a comment. So, one equals two, evaluates to false. Which is represents by zero. One not equals to two. This is true. So that returns one.

Note that a not equal sign is this tilde equals symbol. And not bang equals. Which is what some other programming languages use. Lets see logical operations one and zero use a double ampersand sign to the logical AND. And that evaluates false. One or zero is the OR operation.


Octave-3.2.4
octave-3.2.4.exe:7> 1 ~= 2
ans = 1
octave-3.2.4.exe:8> 1 && 0 % AND
ans = 0
octave-3.2.4.exe:9> 1 || 0 % OR
ans = 1
octave-3.2.4.exe:10> xor(1,0)
ans = 1
octave-3.2.4.exe:11>
octave-3.2.4.exe:11>
octave-3.2.4.exe:11>
octave-3.2.4.exe:11> PS('>> ');
error: `PS' undefined near line 11 column 1
octave-3.2.4.exe:11> PS1('>> ');
>>
>>
>> a = 3
a = 3
>> a = 3; % semicolon supressing output
>> a=3
a = 3
>> a=3;
>> b =
where A equals, 3 semicolon doesn't print anything.
2:45 PM
10/2/2011


Octave-3.2.4
>>
>> a=pi;
>> a
a = 3.1416
>> disp(a);
3.1416
>> disp(sprintf('2 decimals: %0.2f', a))
2 decimals: 3.14
>> disp(sprintf('6 decimals: %0.6f', a))
6 decimals: 3.141593
>> a
a = 3.1416
>> format long
>> a
a = 3.14159265358979
>> format short
>> a
a = 3.1416
>>
>>
>>
>>
>>
Okay, that's how you work with variables.
2:47 PM
10/2/2011


a = 3.1416
>>
>>
>>
>>
>> A = [1 2; 3 4; 5 6]
A =
1 2
3 4
5 6
>> A = [1 2;
> 3 4;
> 5 6]
A =
1 2
3 4
5 6
>>
>> v = [1 2 3]
This is actually a row vector.

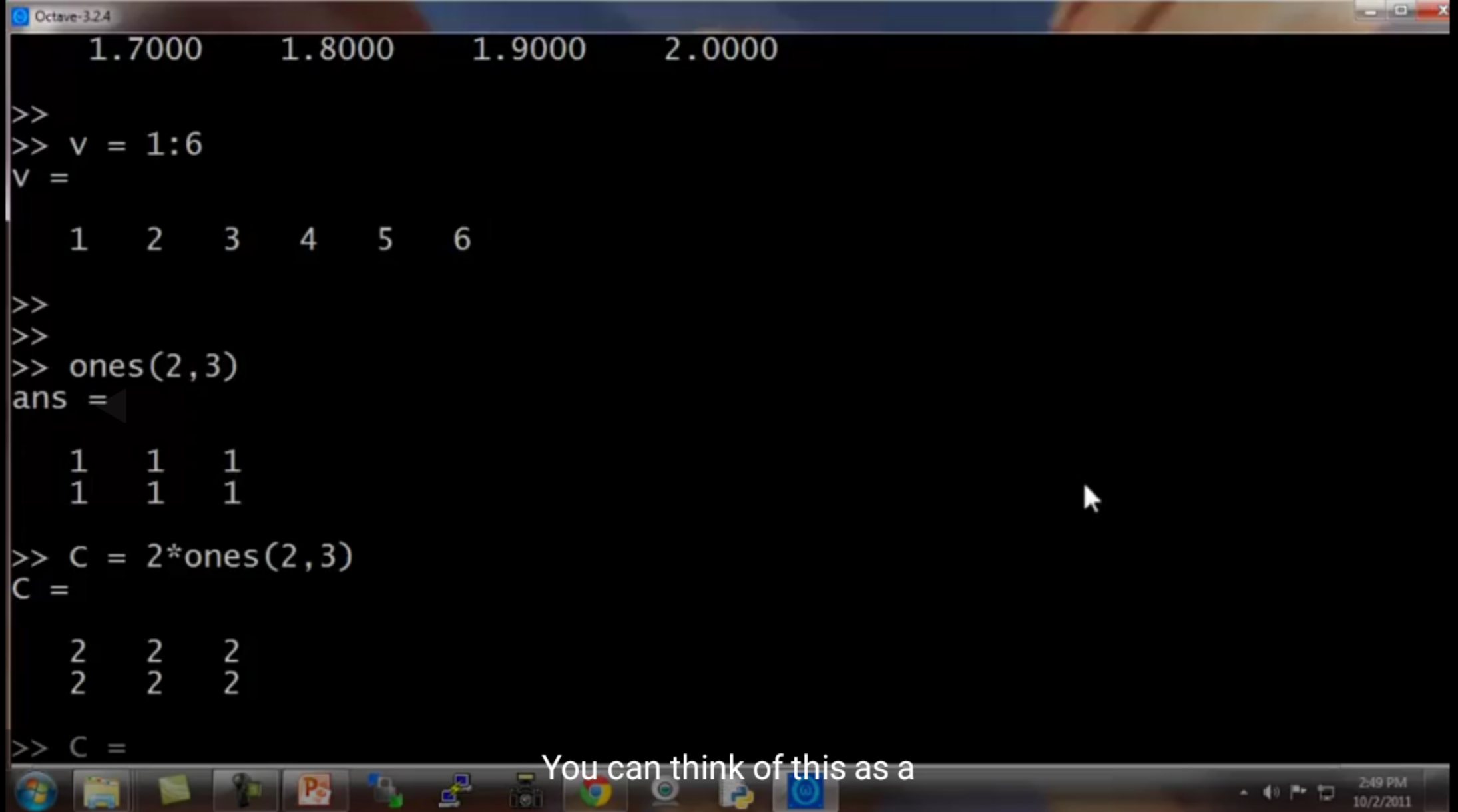
Basic Operations

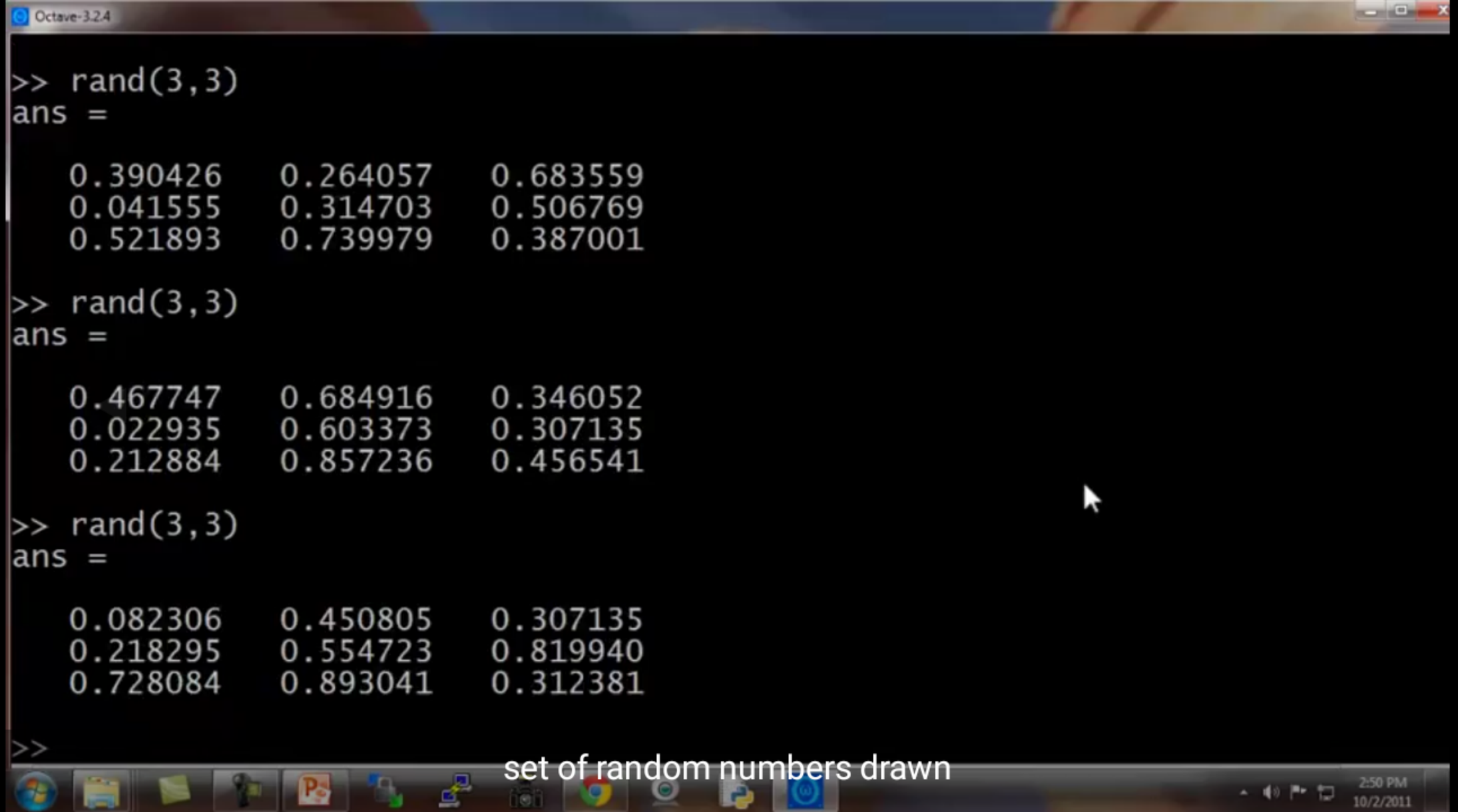

```
Octave-3.2.4
   1   2   3

>> v = [1; 2; 3]
v =

   1
   2
   3

>>
>> v = 1:0.1:2
v =

 Columns 1 through 7:

    1.0000    1.1000    1.2000    1.3000    1.4000    1.5000    1.6000

 Columns 8 through 11:

    1.7000    1.8000    1.9000    2.0000

>>
>>
```

V equals 1: 0.1: 2. What this does is it sets V to the bunch of elements that start from 1. And increments and steps of 0.1 until you get up to 2. So if I do this, V is going to be this, you know, row vector. This is what one by eleven matrix really.

That's 1, 1.1, 1.2, 1.3 and so on until we get up to two. Now, and I can also set V equals one colon six, and that sets V to be these numbers. 1 through 6, okay. Now here are some other ways to generate matrices.

Ones 2.3 is a command that generates a matrix that is a two by three matrix that is the matrix of all


Octave-3.2.4
1.7000 1.8000 1.9000 2.0000
>>
>> v = 1:6
v =
1 2 3 4 5 6
>>
>>
>> ones(2,3)
ans =
1 1 1
1 1 1
>> C = 2*ones(2,3)
C =
2 2 2
2 2 2
>> C =
You can think of this as a
2:49 PM
10/2/2011

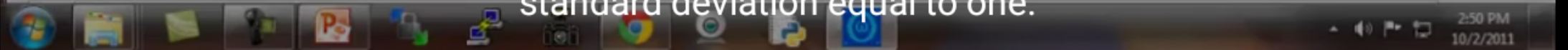


set of random numbers drawn

Octave-3.2.4

```
   0.467747   0.684916   0.346052
   0.022935   0.603373   0.307135
   0.212884   0.857236   0.456541

>> rand(3,3)
ans =

   0.082306   0.450805   0.307135
   0.218295   0.554723   0.819940
   0.728084   0.893041   0.312381

>> w = randn(1,3)
w =

  -1.44264  -1.27860  -0.69640

>> w = randn(1,3)
w =

  -0.33517   1.26847  -0.28211

>>
```

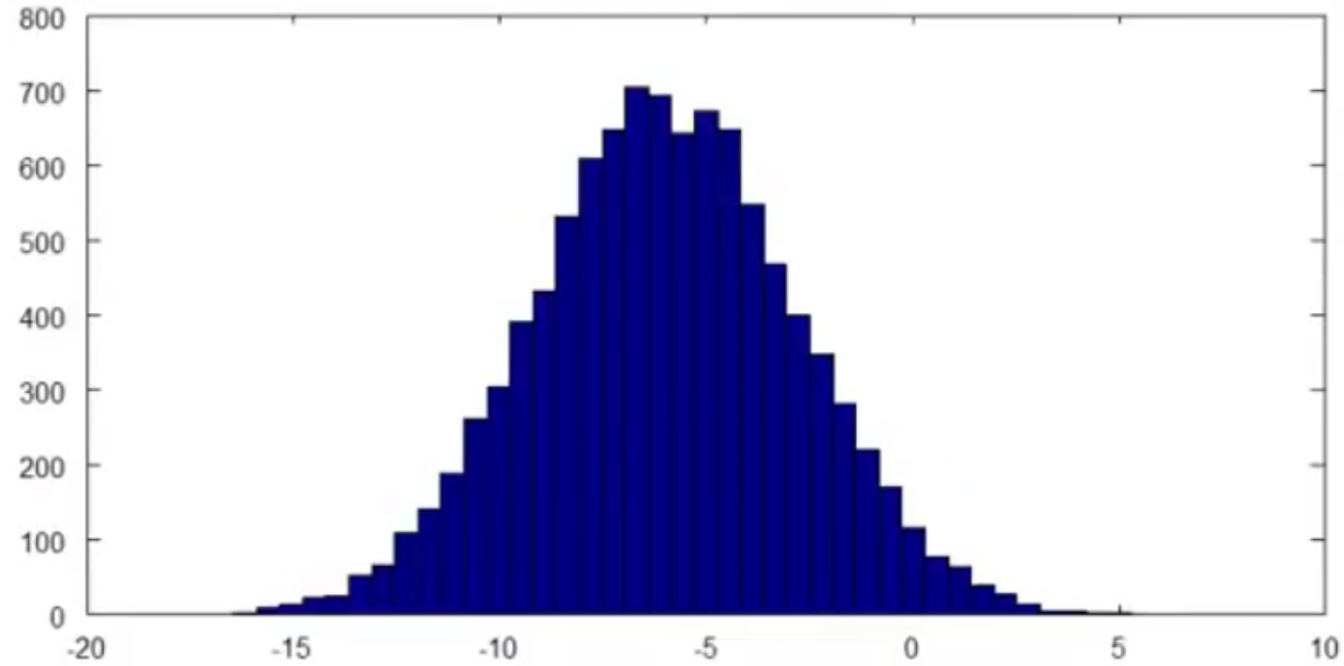
standard deviation equal to one.



Octave-3.2.4

```
ans =

    0.082306
    0.218295
    0.728084

>> w = randn(1
w =

   -1.44264   -1

>> w = randn(1
w =

   -0.33517    1

>>
>>
>>
>> w = -6 + sqrt(10)*(randn(1,10000))
>> hist(w)
>> hist(w,50)
>>
```

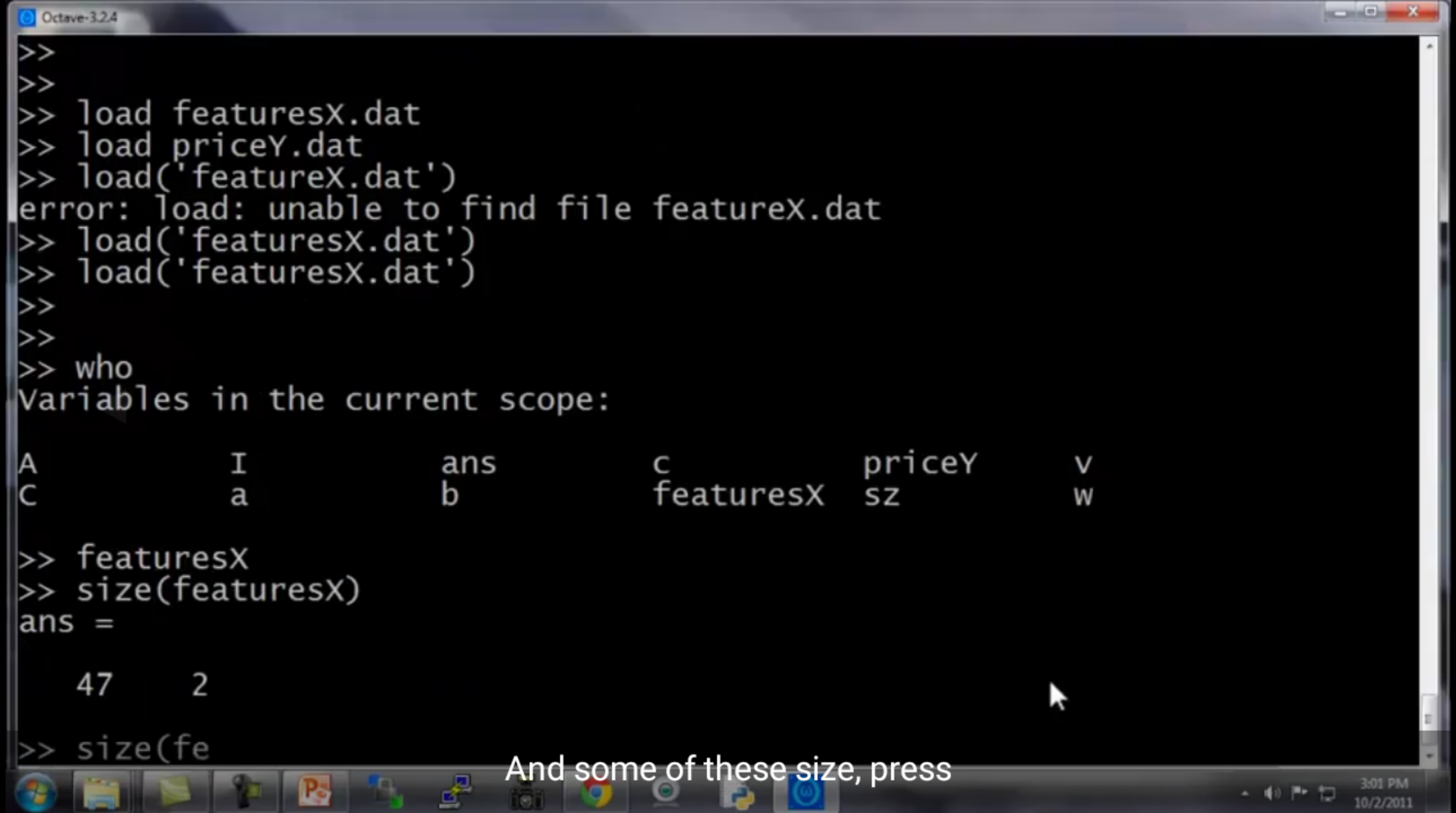
Figure 1

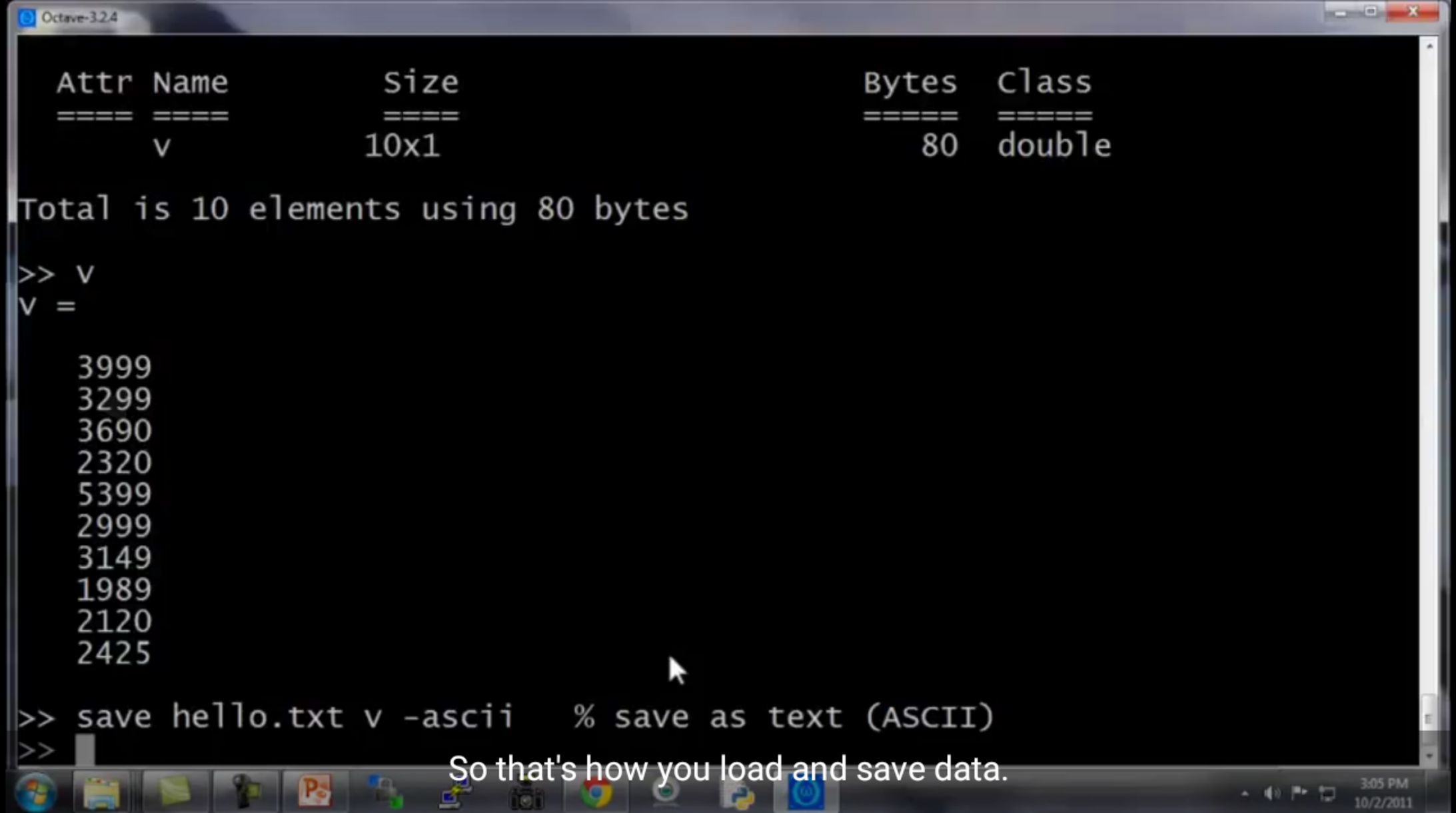


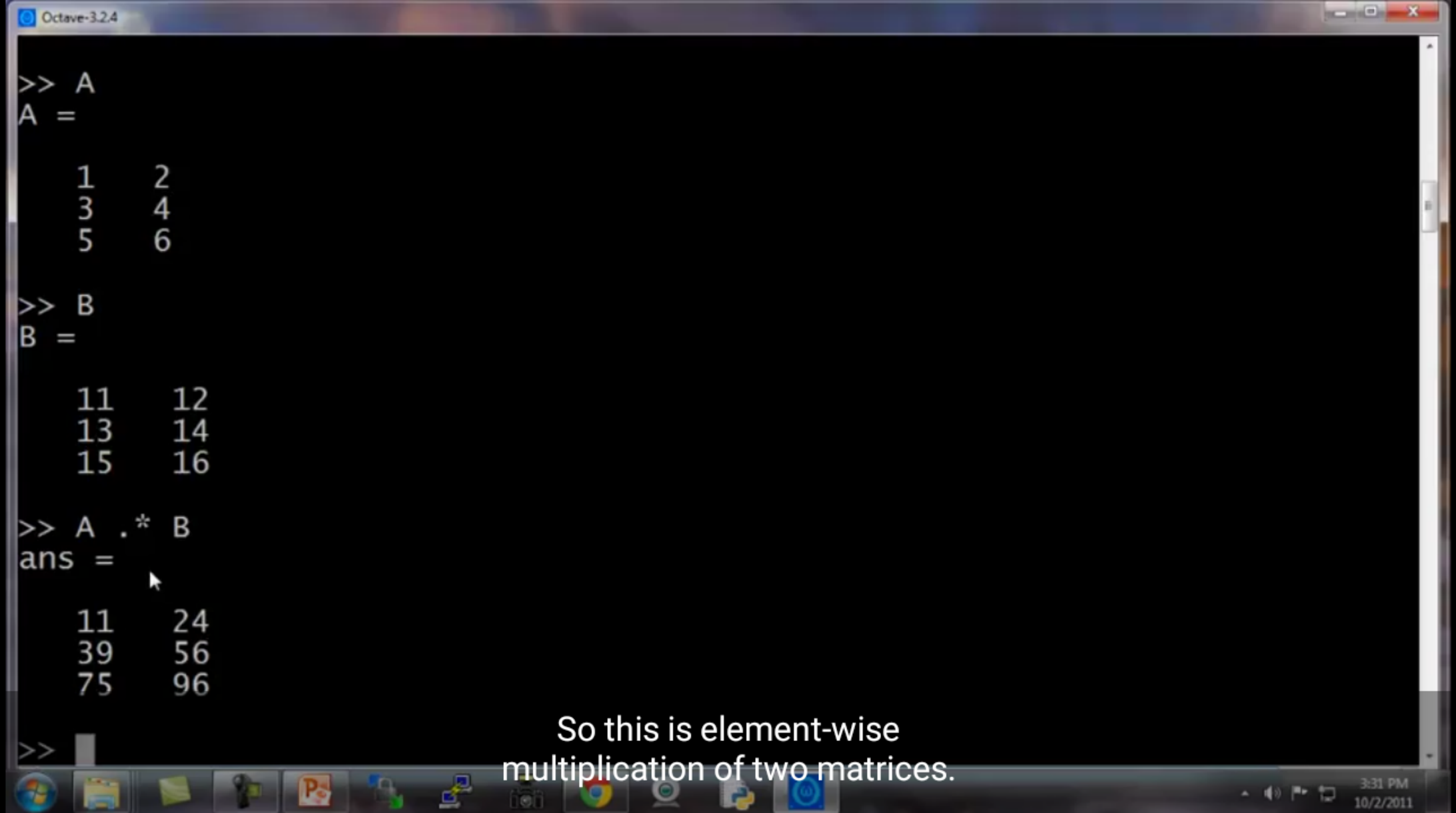
-2.15065, -126.034

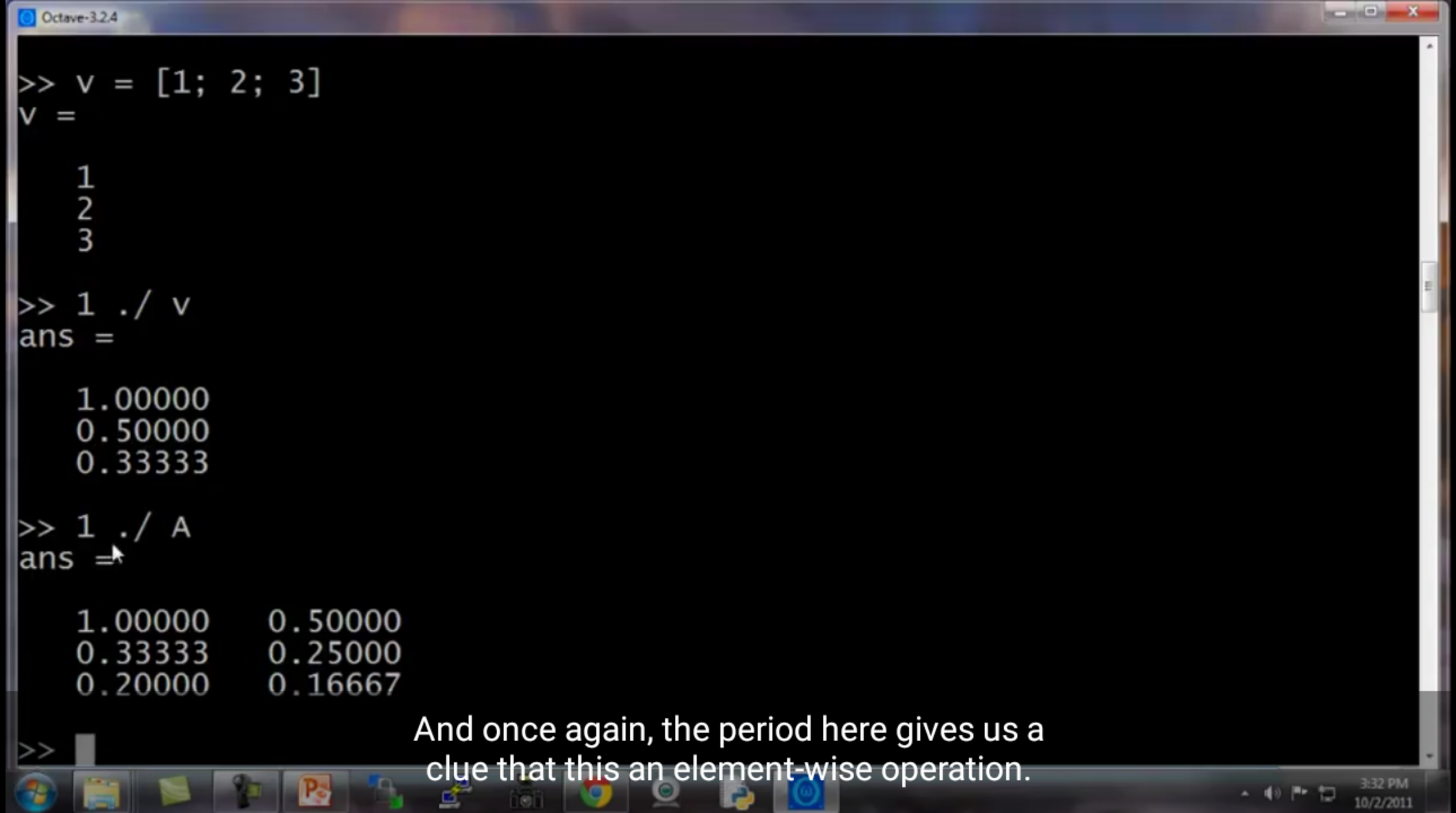
more buckets, with more bins, with say, 50 bins.

2:51 PM
10/2/2011


Octave-3.2.4
>>
>>
>> load featuresX.dat
>> load priceY.dat
>> load('featureX.dat')
error: load: unable to find file featureX.dat
>> load('featuresX.dat')
>> load('featuresX.dat')
>>
>>
>> who
Variables in the current scope:
A I ans c priceY v
C a b featuresX sz w
>> featuresX
>> size(featuresX)
ans =
47 2
>> size(fe
And some of these size, press
3:01 PM
10/2/2011


Octave-3.2.4
Attr Name Size Bytes Class
v 10x1 80 double
Total is 10 elements using 80 bytes
>> v
v =
3999
3299
3690
2320
5399
2999
3149
1989
2120
2425
>> save hello.txt v -ascii % save as text (ASCII)
>>
So that's how you load and save data.
3:05 PM
10/2/2011


Octave-3.2.4
>> A
A =
1 2
3 4
5 6
>> B
B =
11 12
13 14
15 16
>> A .* B
ans =
11 24
39 56
75 96
>>
So this is element-wise
multiplication of two matrices.
3:31 PM
10/2/2011



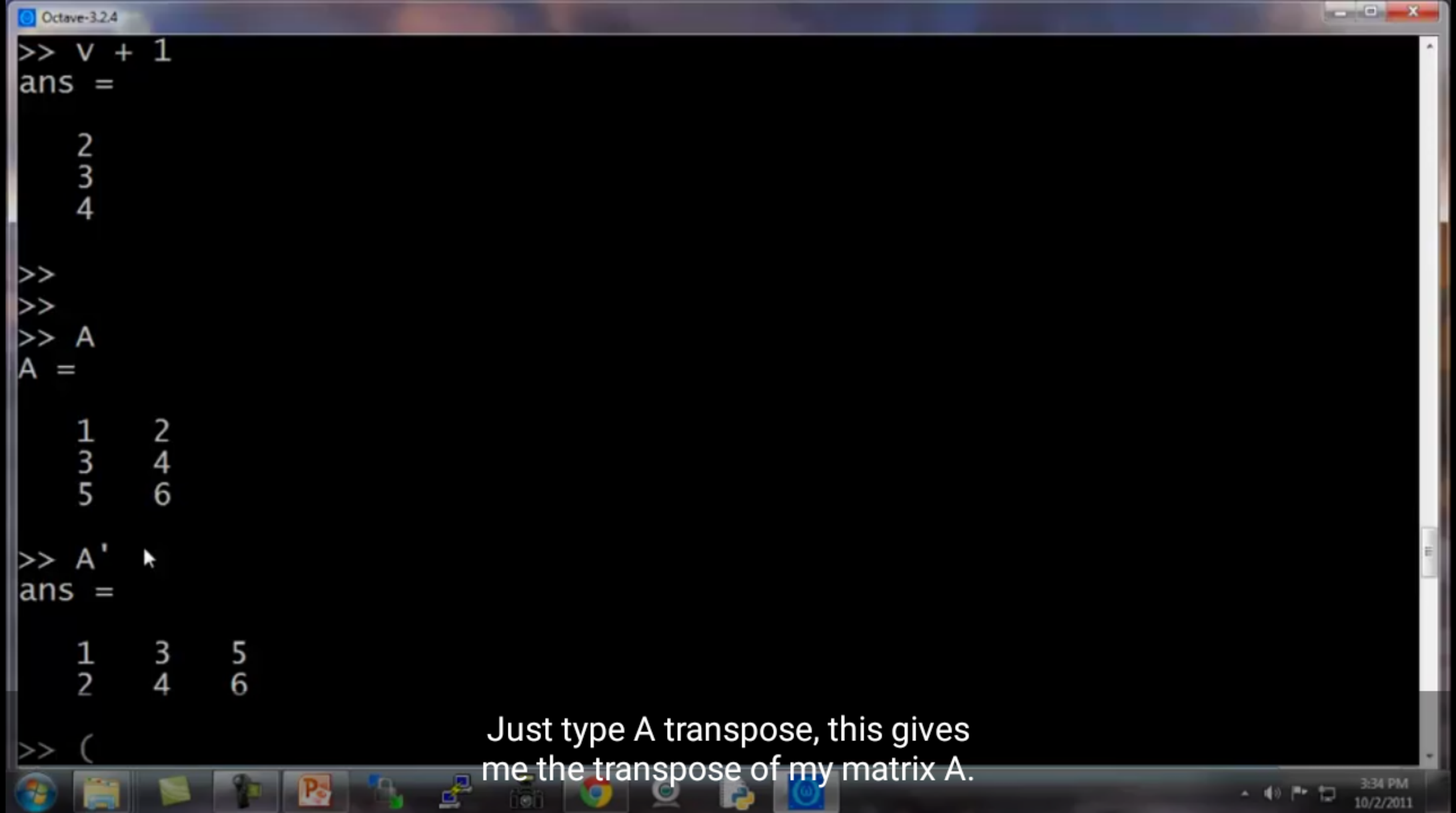
And once again, the period here gives us a clue that this an element-wise operation.

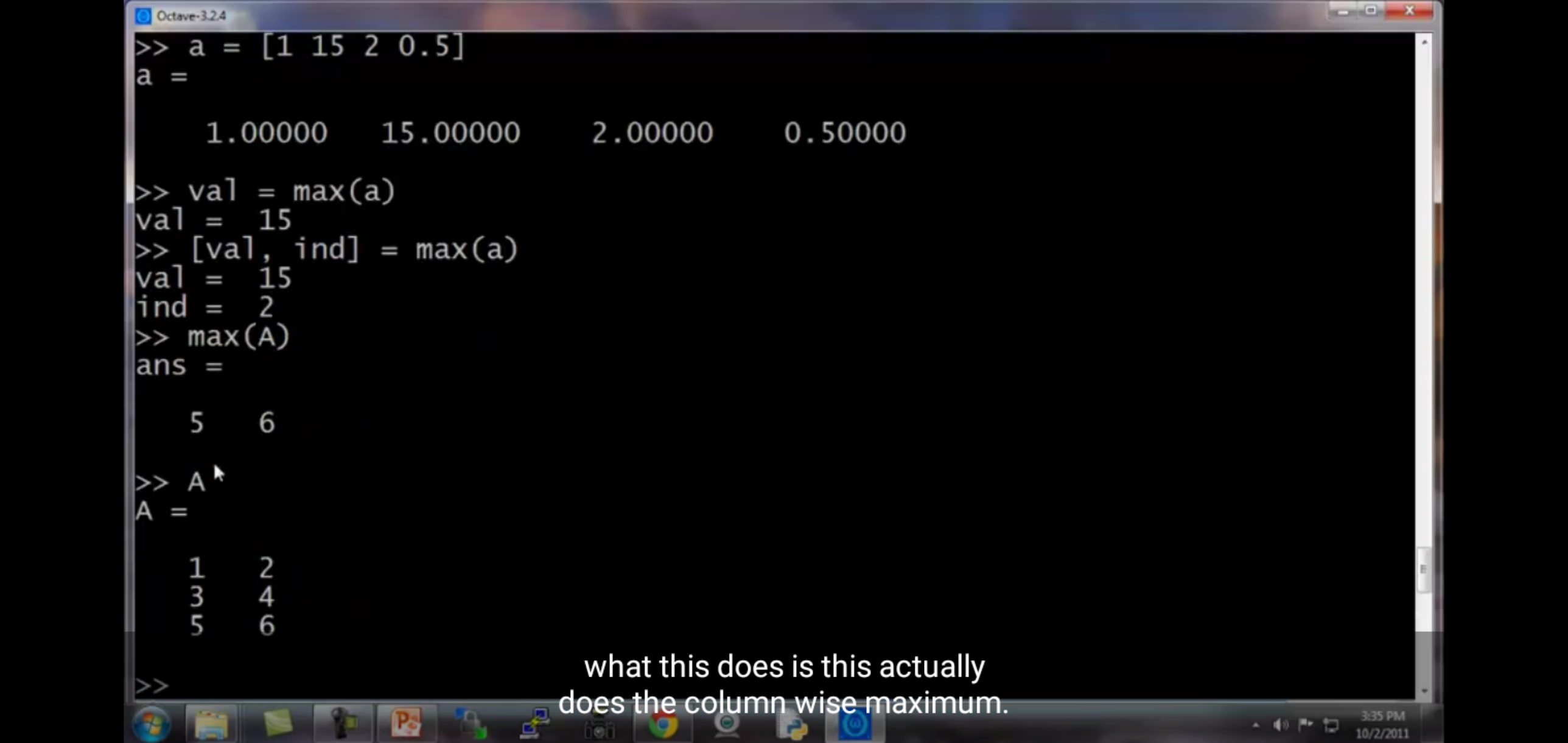
Octave-3.2.4

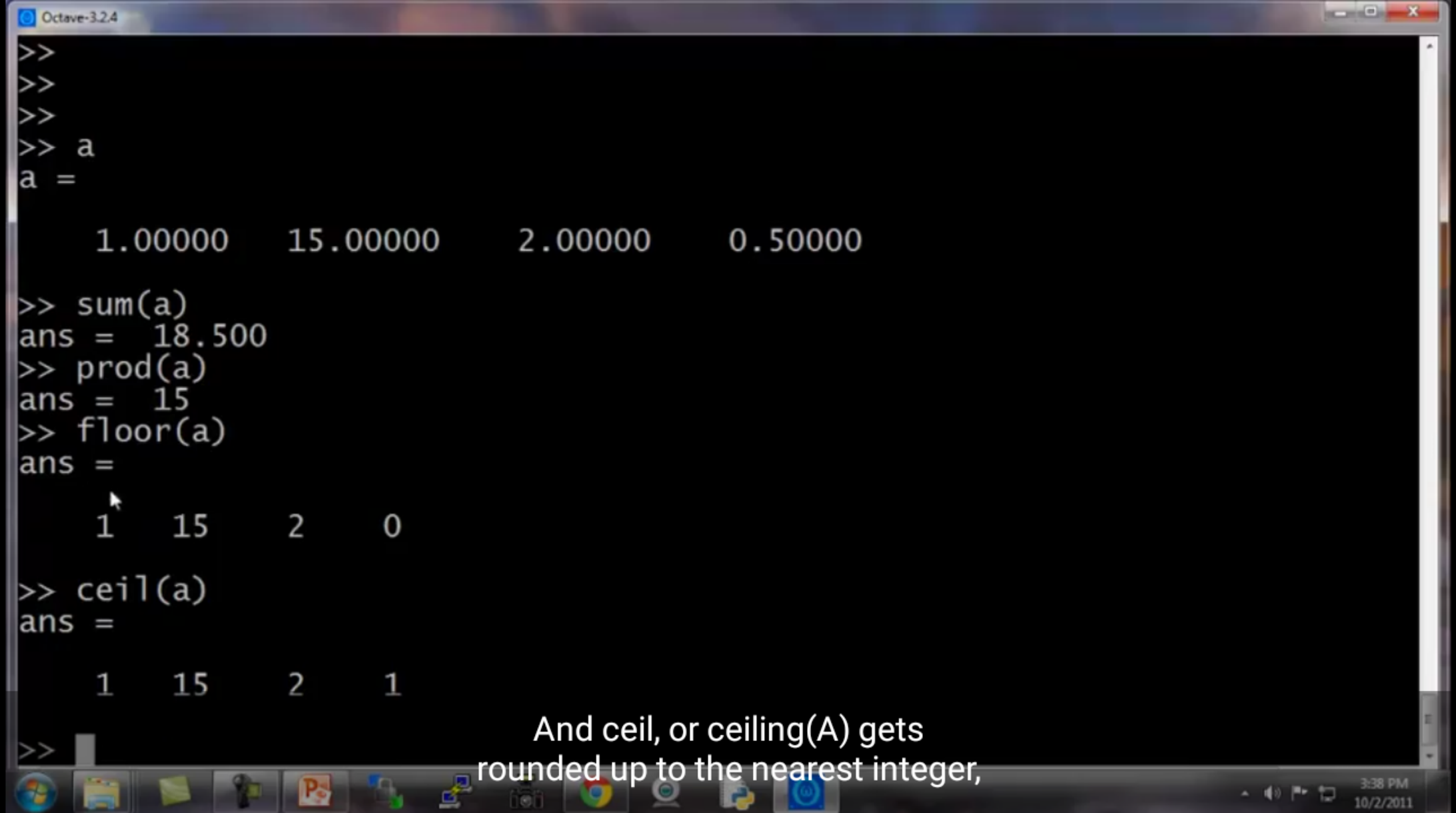
```
>> log(v)
ans =

   0.00000
   0.69315
   1.09861

>> exp(v)
ans =

    2.7183
    7.3891
   20.0855

>> v
v =

   1
   2
   3

>>
```

so this is E, this is E squared EQ,
because this was V, and

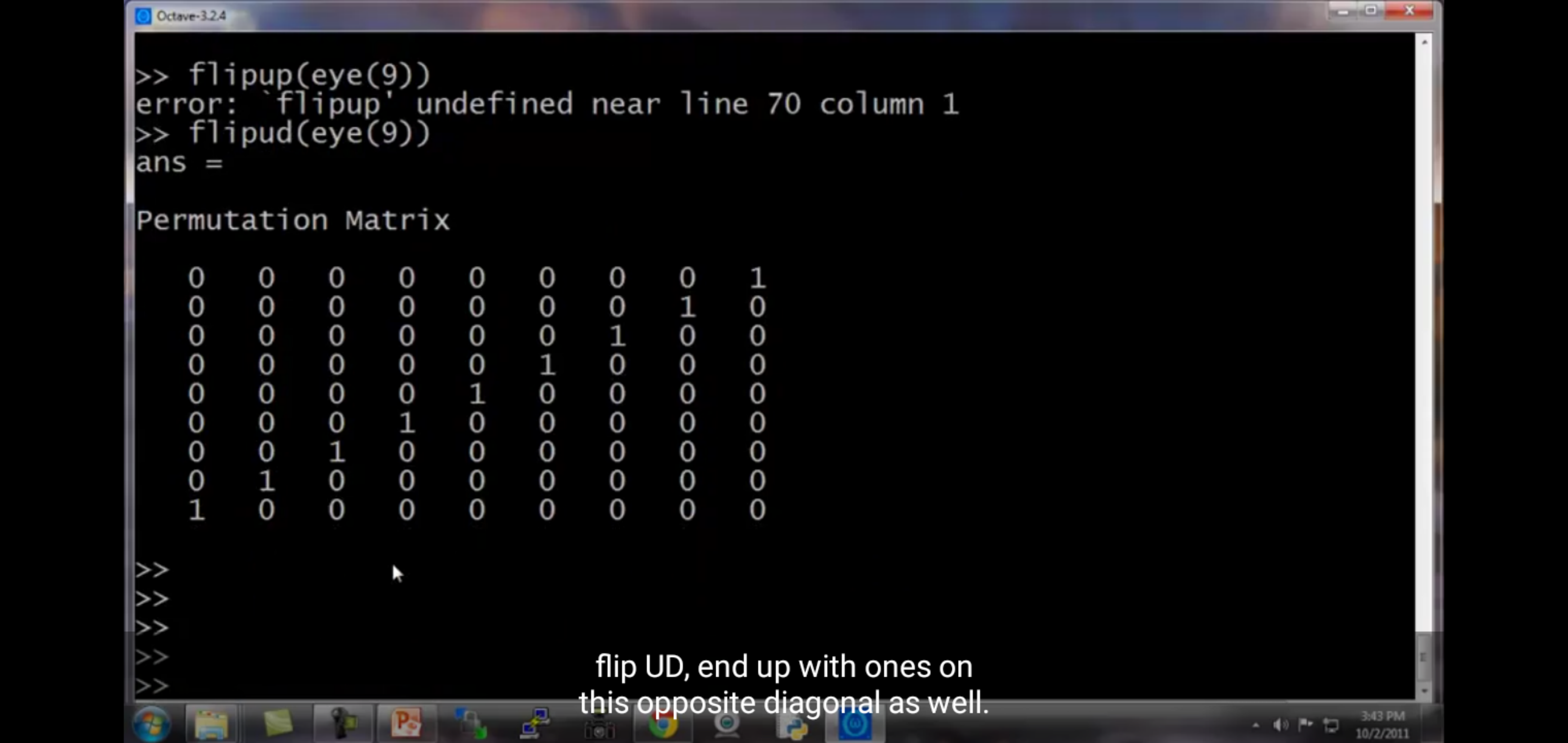
3:32 PM
10/2/2011


Octave-3.2.4
>> v + 1
ans =
2
3
4
>>
>>
>> A
A =
1 2
3 4
5 6
>> A'
ans =
1 3 5
2 4 6
>> (
Just type A transpose, this gives
me the transpose of my matrix A.
3:34 PM
10/2/2011


Octave-3.2.4
>> a = [1 15 2 0.5]
a =
1.00000 15.00000 2.00000 0.50000
>> val = max(a)
val = 15
>> [val, ind] = max(a)
val = 15
ind = 2
>> max(A)
ans =
5 6
>> A
A =
1 2
3 4
5 6
>>
what this does is this actually
does the column wise maximum.
3:35 PM
10/2/2011



And ceil, or ceiling(A) gets
rounded up to the nearest integer,


Octave-3.2.4
>> flipup(eye(9))
error: `flipup' undefined near line 70 column 1
>> flipud(eye(9))
ans =
Permutation Matrix
0 0 0 0 0 0 0 0 1
0 0 0 0 0 0 0 1 0
0 0 0 0 0 0 1 0 0
0 0 0 0 0 1 0 0 0
0 0 0 0 1 0 0 0 0
0 0 0 1 0 0 0 0 0
0 0 1 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0
1 0 0 0 0 0 0 0 0
>>
>>
>>
>>
>>
flip UD, end up with ones on
this opposite diagonal as well.
3:43 PM
10/2/2011

```
Octave-3.2.4
ans = C:\Octave\3.2.4_gcc-4.4.0\bin
>> cd 'C:\User\ang\Desktop'
error: C:\User\ang\Desktop: No such file or directory
>> cd 'C:\Users\ang\Desktop'
>> squareThisNumber(5)
ans =  25
>>
>>
>> % Octave search path (advanced/optional)
>> addpath('C:\Users\ang\Desktop')
>> cd 'C:\'
>> squareThisNumber(5)
ans =  25
>> pwd
ans = C:\
>>
>>
>> [a,b] = squareAndCubeThisNumber(5);
>> a
a =  25
>> b
b =  125
>>
```

So, some of you depending on what programming language you use, if you're familiar with, you know, CC++ your offer. Often, we think of the function as return in just one value. But just so the syntax in Octave that should return multiple values. Now back in the Octave window.

If I type, you know, a, b equals square and cube this number 5 then a is now equal to 25 and b is equal to the cube of 5 equal to 125. So, this is often convenient if you needed to define a function that returns multiple values.

Finally, I'm going to show you just one more sophisticated example of

Vectorization

Vectorization example.

$$h_\theta(x) = \sum_{j=0}^{n} \theta_j x_j = \theta^T x$$

$$\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \end{bmatrix} \quad x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix}$$

theta(1) theta(2) theta(3)

Unvectorized implementation

```
prediction = 0.0;
for j = 1:n+1,
  prediction = prediction +
               theta(j) * x(j)
end;
```

Vectorized implementation

```
prediction = theta' * x;
```

And what this line of code on the right will do is, it will use Octaves highly optimized numerical linear algebra routines to compute this inner product between the two vectors, theta and X, and not only is the vectorized implementation simpler, it will also run much more efficiently.

So that was octave, but the issue of vectorization applies to other programming language as well. Lets look on the example in C++. Here's what an unvectorized implementation might look like. We again initialize prediction to 0.0 and then we now how

Vectorization

Vectorization example.

$$h_\theta(x) = \sum_{j=0}^{n} \theta_j x_j$$

$$= \theta^T x$$

Unvectorized implementation

```
double prediction = 0.0;
for (int j = 0; j <= n; j++)
  prediction += theta[j] * x[j];
```

Vectorized implementation

```
double prediction
  = theta.transpose() * x;
```

So depending on the details of your numerical linear algebra library, you might be able to have an object, this is a C++ object, which is vector theta, and a C++ object which is vector x, and you just take theta.transpose * x, where this times becomes a C++ sort of overload operator so you can just multiply these two vectors in C++.

And depending on the details of your numerical linear algebra library, you might end up using a slightly different syntax, but by relying on the library to do this inner product, you can get a much simpler piece of code and a