Introduction to Matlab

ECE 111 Introduction to ECE

Jake Glower - Week #1

Please visit Bison Academy for corresponding
lecture notes, homework sets, and solutions

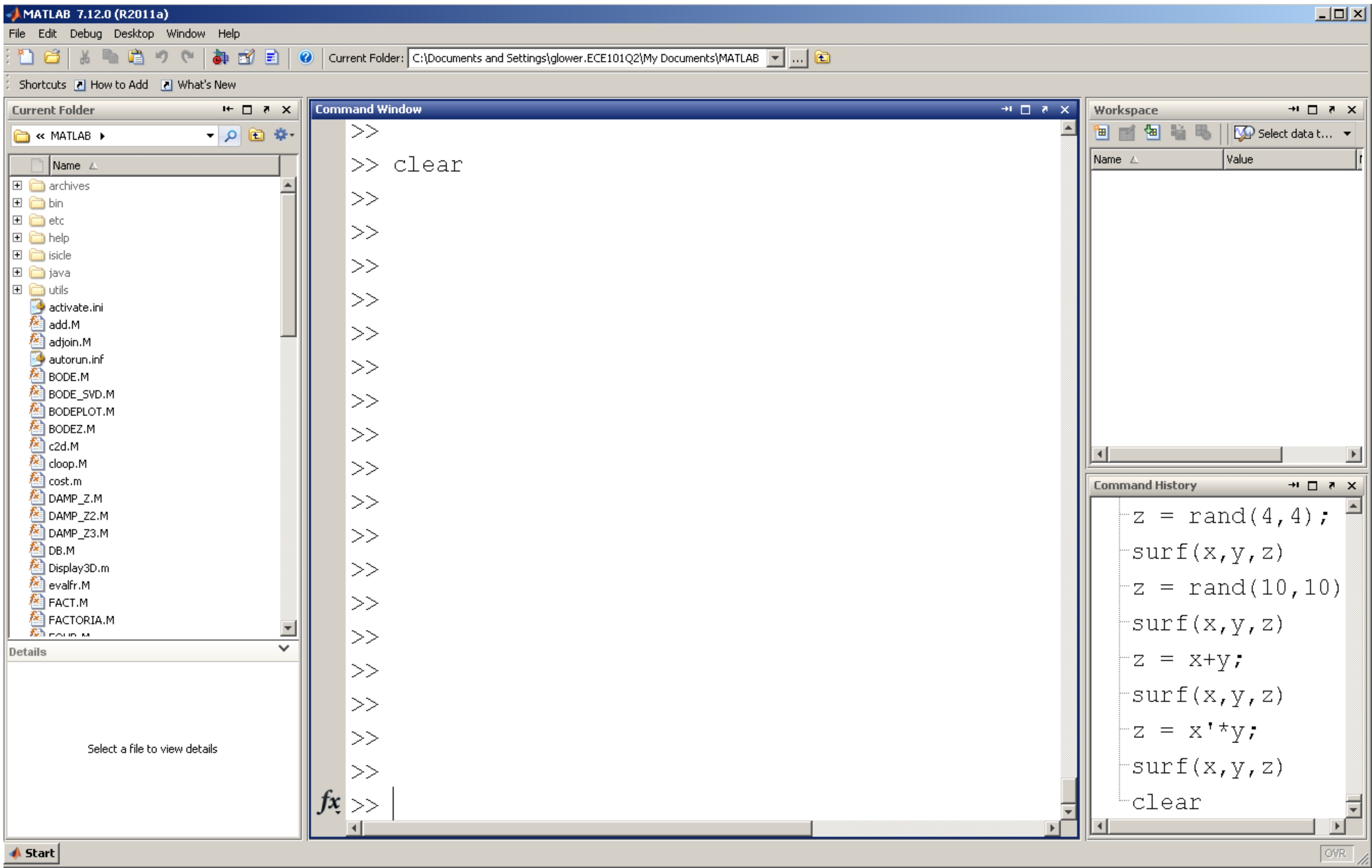
Becoming familiar with MATLAB

- Using the command window
- Using scripts
- Plotting with Matlab
- Random numbers in Matlab
- If-Statements
- For-Loops
- While-Loops
- Monte-Carlo Simulations

General environment and the console

Startup Screen:

- I usually close everything down except the command window



Command Window

Matlab works like a calculator

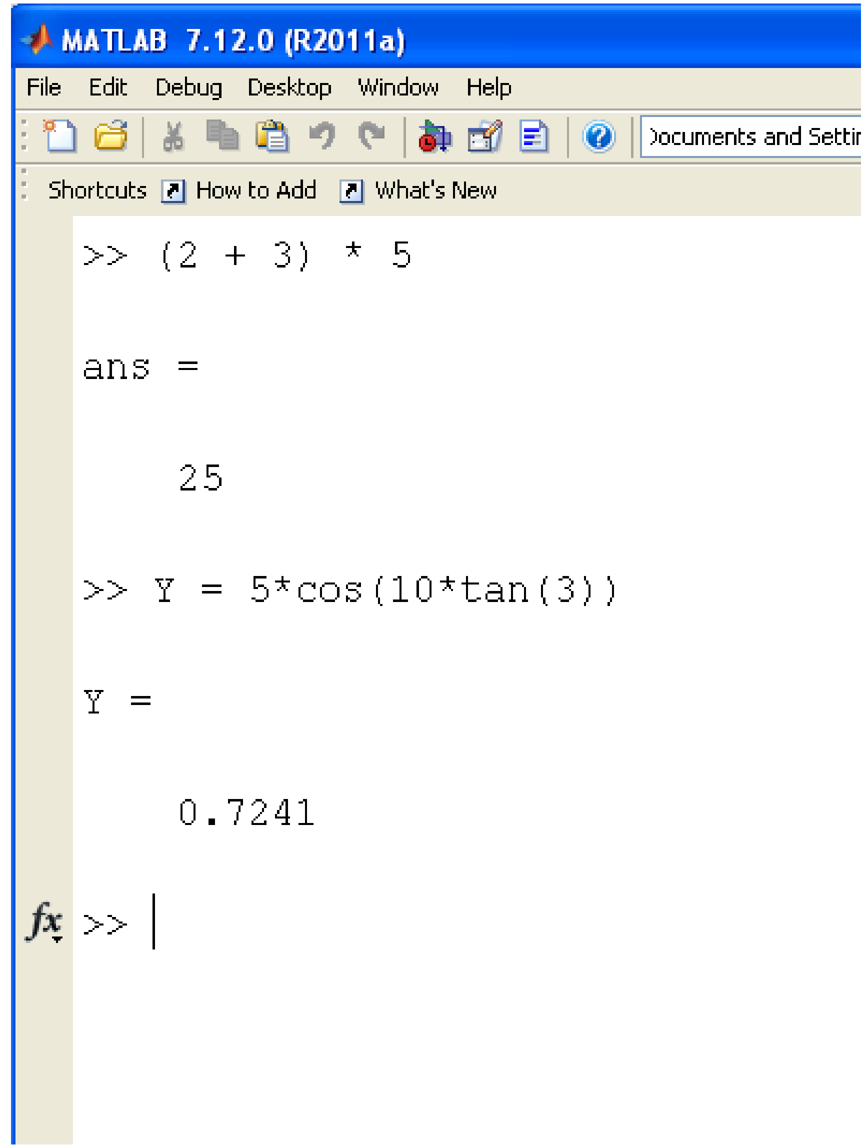
- Solve

```
( 2 + 3 ) * 5
```

- Solve

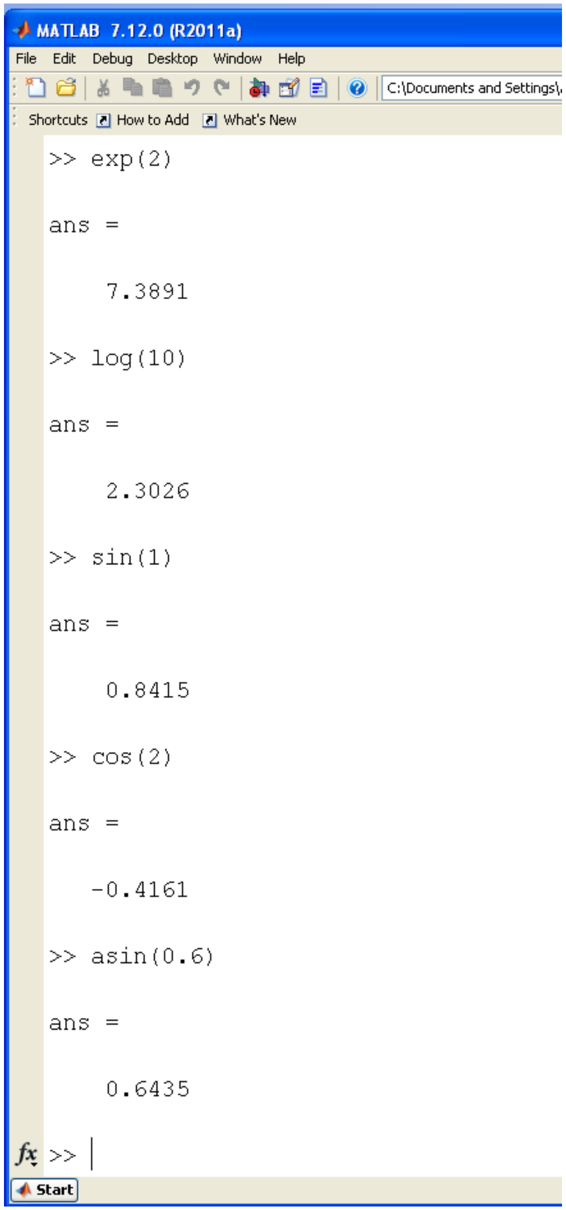
```
Y = 5*cos(10*tan(3))
```

You type it the way it looks:



Matlab Function Names

- pi π
- exp(x) e^x
- 10^x 10^x
- log(x) $\ln(x)$
- log10(x) $\log_{10}(x)$
- sin(x) $\sin(x)$ *units = radians*
- cos(x) $\cos(x)$ " "
- tan(x) $\tan(x)$ " "
- asin(x) $\arcsin(x)$
- acos(x) $\arccos(x)$
- atan(y,x) $\arctan(y/x)$
- + many more



Order of Operations

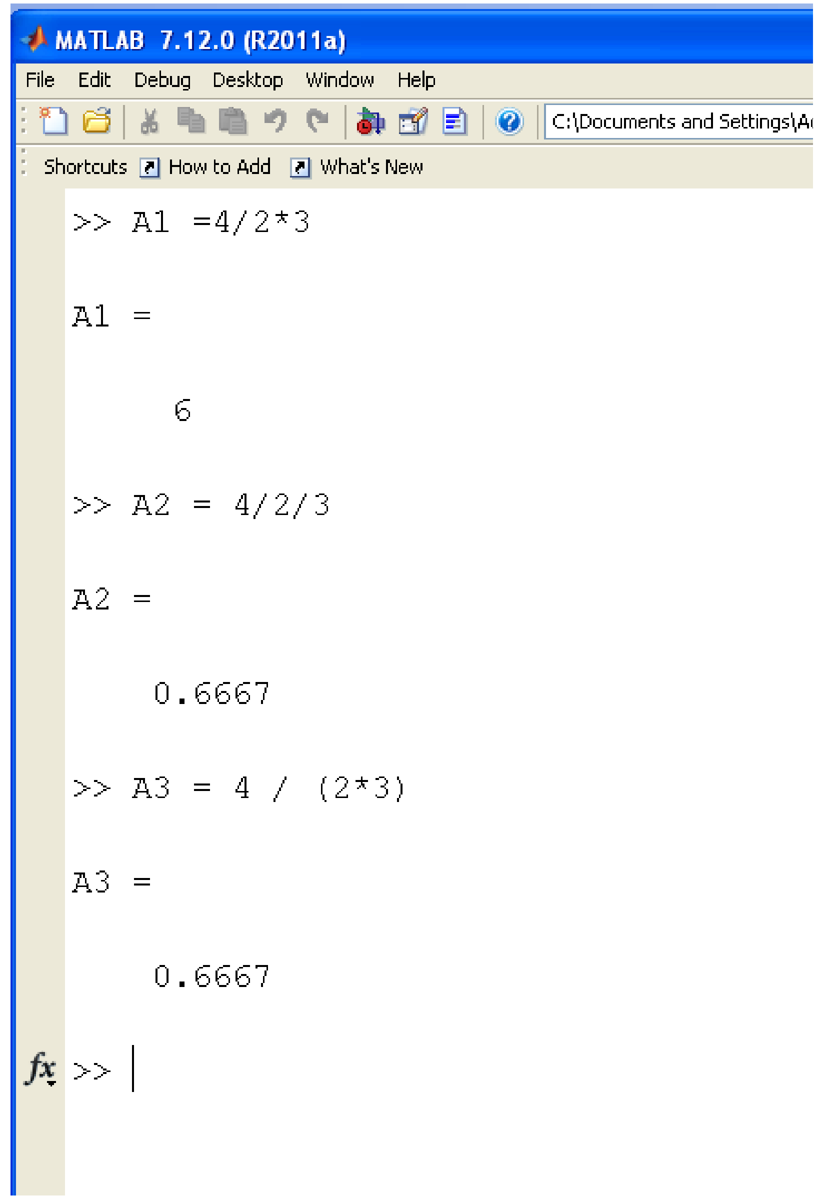
- 1st: ^
- 2nd: *, /
- 3rd: +, -

Equations are executed

- By order of operations, then
- Left to right

Paranthesis never hurt,

- They can aviod confusion



Matlab as a Calculator

You can define variables as you go

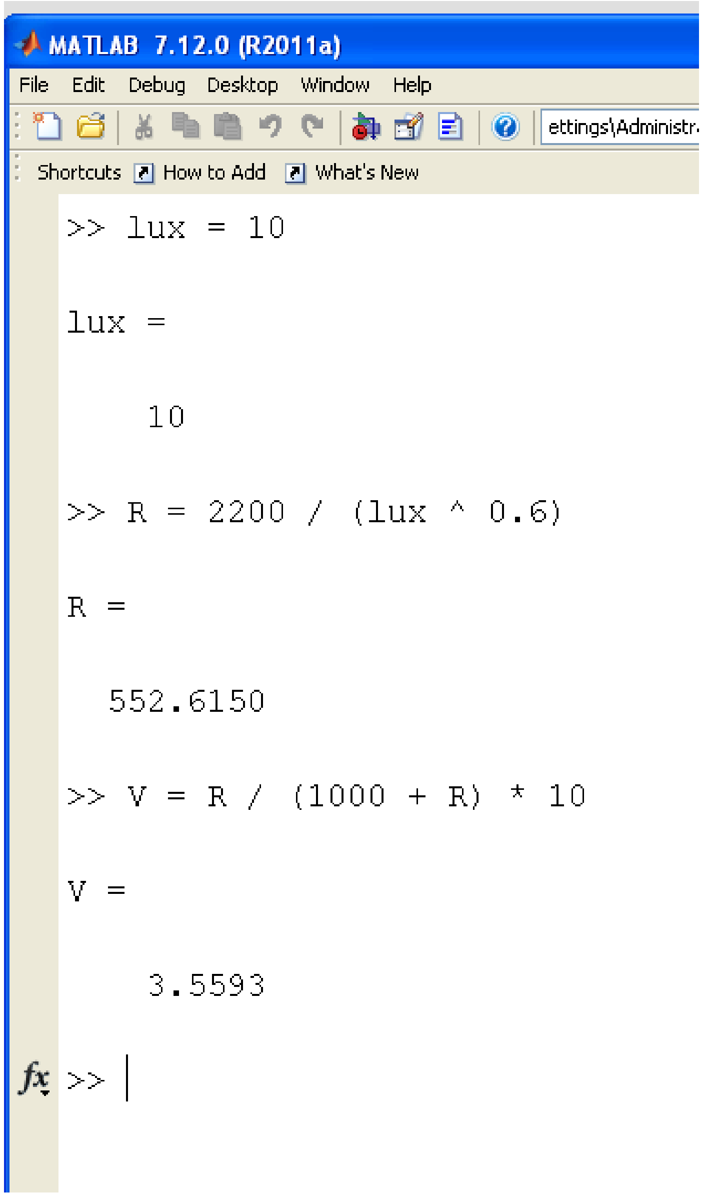
- 1st character must be a letter
- 2nd onward can be letters or numbers
- Case sensitive

Example: Light Sensor

$$R = \left(\frac{2200}{(lux)^{0.6}} \right) \Omega$$

$$V = \left(\frac{R}{1000+R} \right) 10V$$

Find R and V @ 10 Lux



Doing Several Operations at Once

Matlab is a matrix language

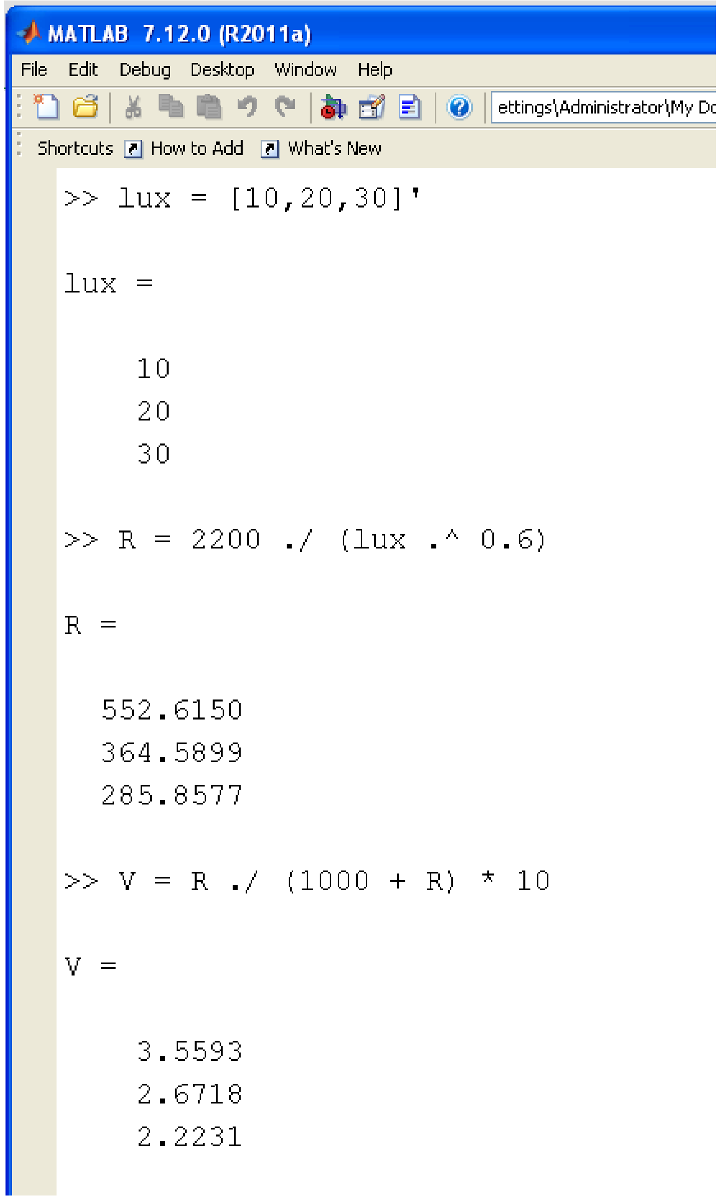
```
[        start of a matrix
,        next column (a space also works)
;        next row
]        end of matrix.
'        transpose
```

Multiplication, Power

- *, ^
- Matrix operations (coming soon)

Dot-Notation

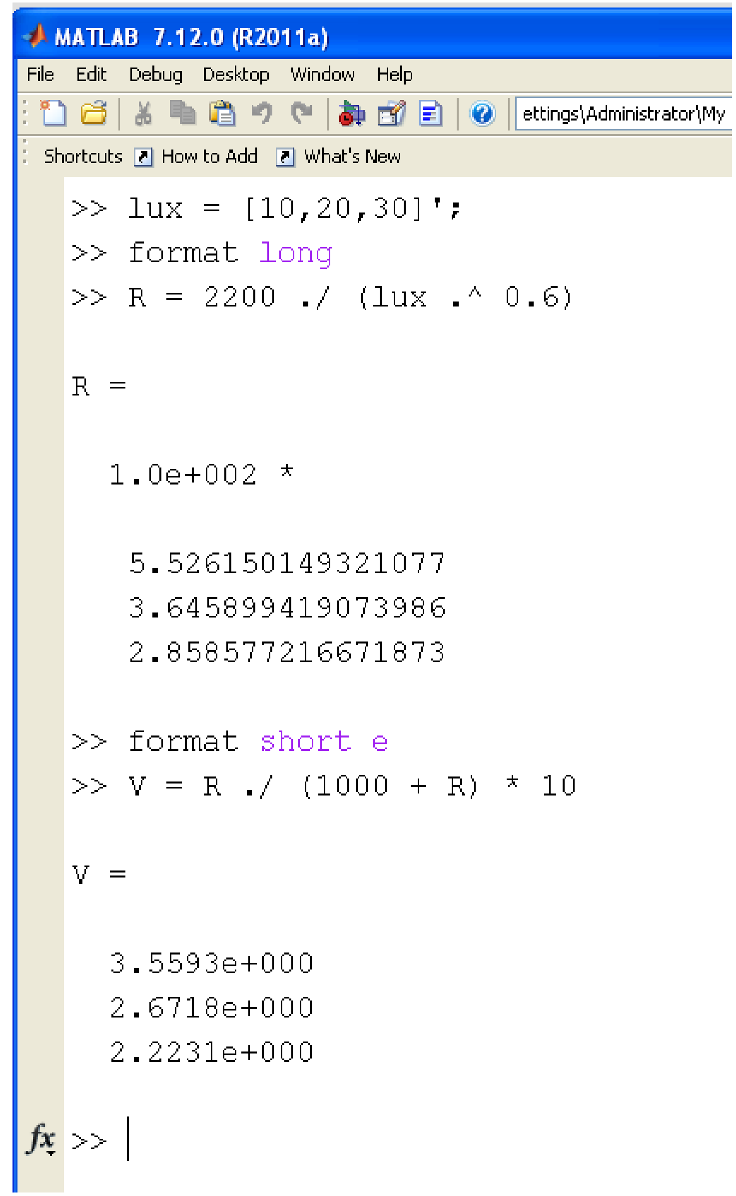
- .*   ./   .^
- Element-by-element operations
- Allows you to do several operations at once



Formatting Output

- Terminate a line with a semi-colon if you don't what the result displayed
- Leave off the semi-colon if you *do* want to see the result

```
format short
pi

   3.1416

format long
pi

  3.141592653589793


format shorteng
pi^30

  821.2893e+012
```



Matlab as a Graphing Calculator

Matlab has pretty good graphics

This is useful if you want to know what happens over a range of values.

Example: Find R for 1 < lux < 100

$$R = \left(\frac{2200}{(lux)^{0.6}}\right)\Omega$$

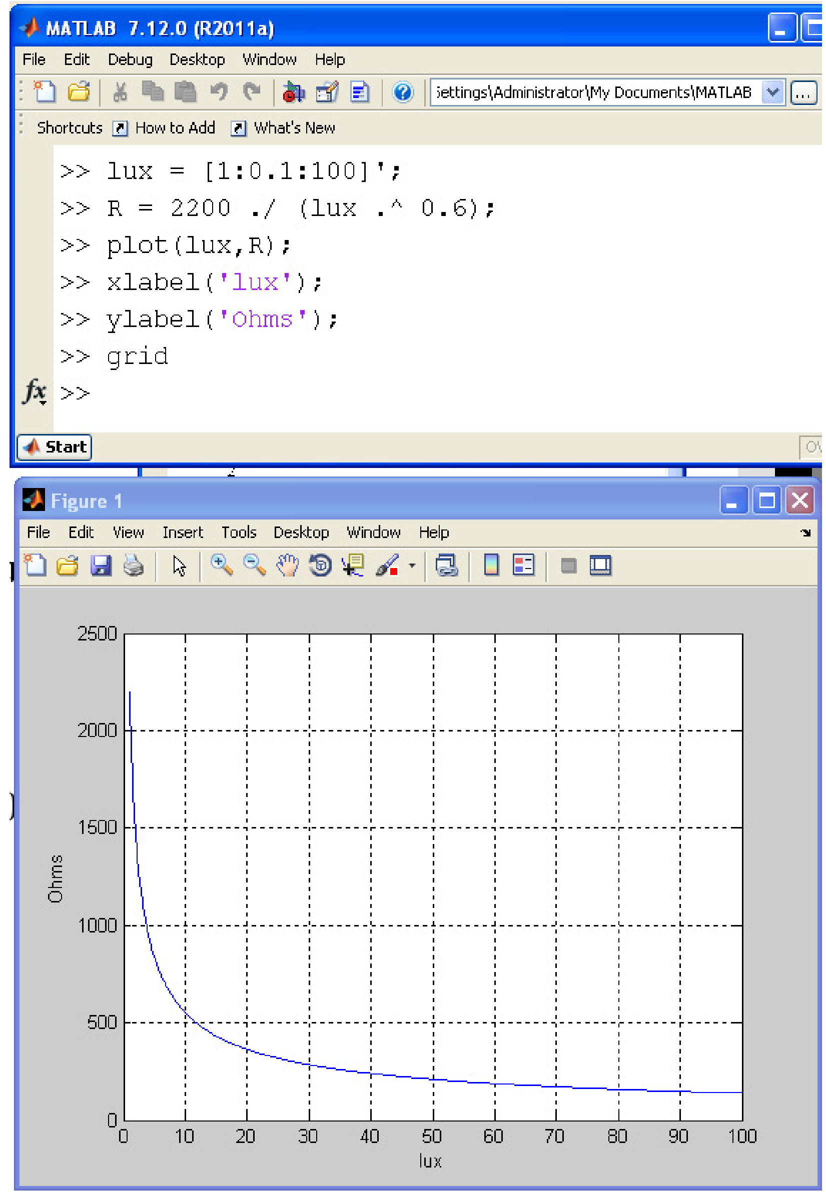
Linear spacing

- 1 to 100 lux, step size = 0.1

```
lux = [1 : 0.1 : 100]';
```

Log spacing from 10^{-2} to 10^{3} with 100 points

```
lux = logspace(-2, 3, 100)';
```



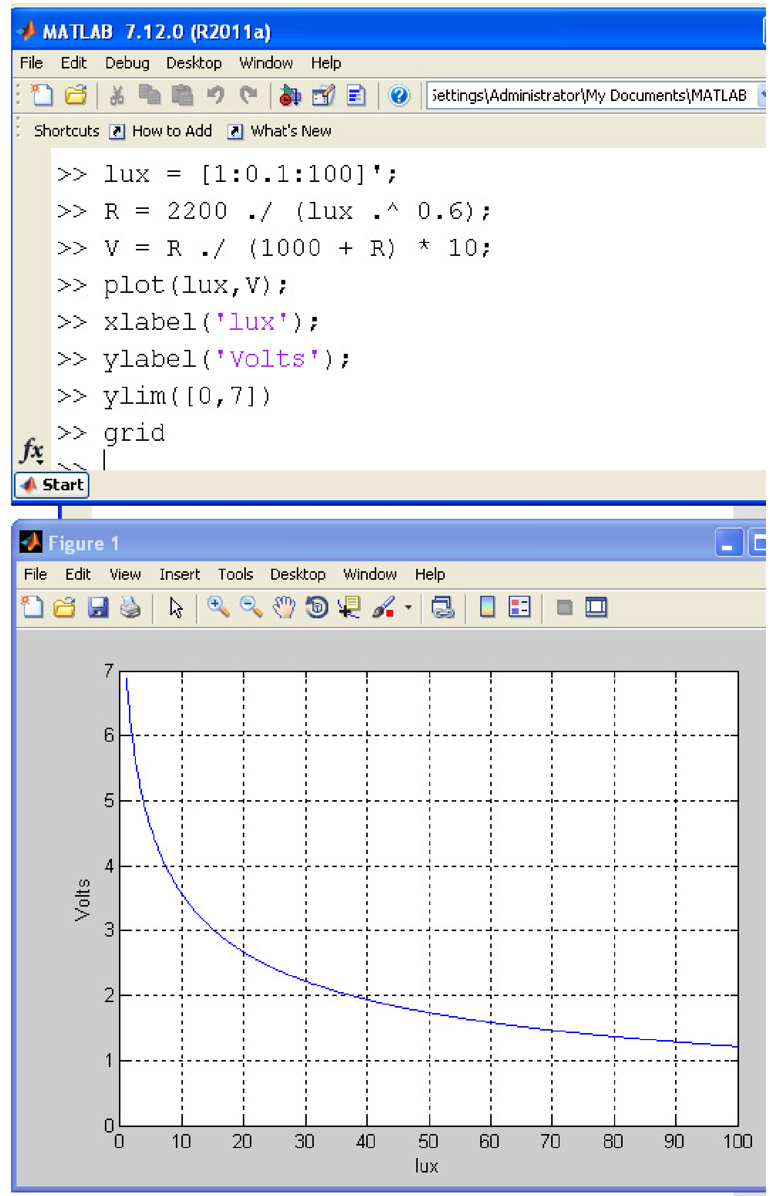
Matlab as a Graphing Calculator

Graphs are useful

- They show you how the two variables are related
- They allow you to determine V over a range of lux
- They allow you to determine lux if you know V

Example: If you read 2.00 Volts, what is the light level?

- Read it off the graph
- light = 36 lux (approx)

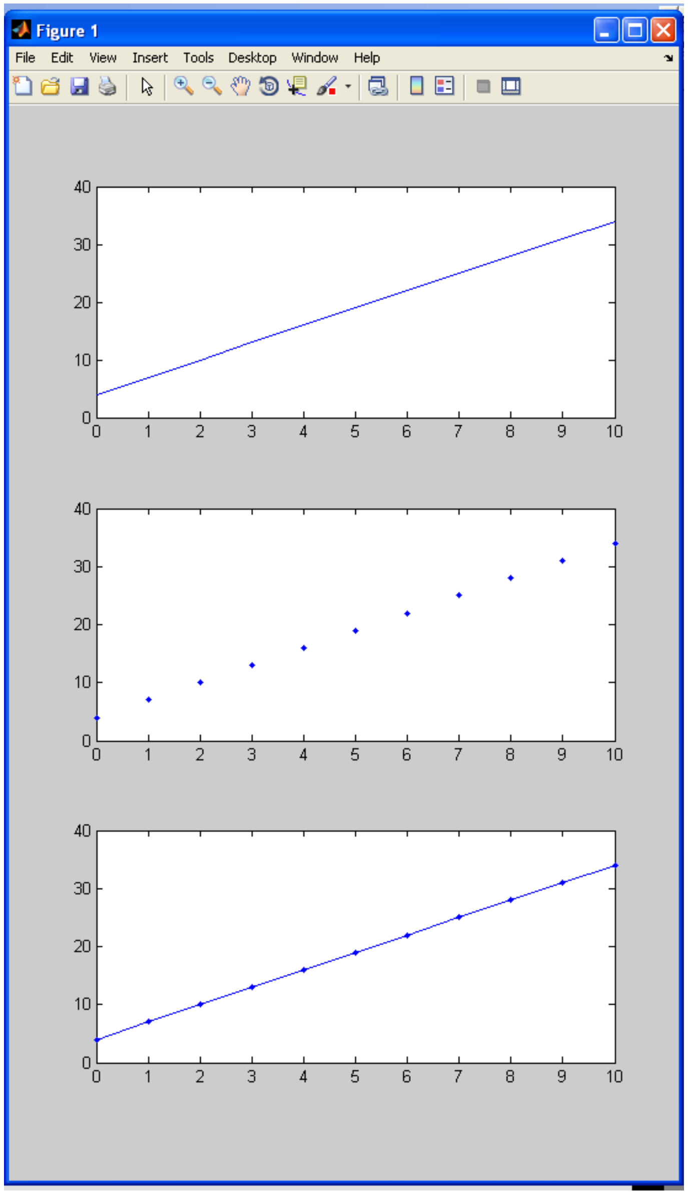


Plotting Functions in Matlab:

Matlab has some pretty good graphics capabilities.

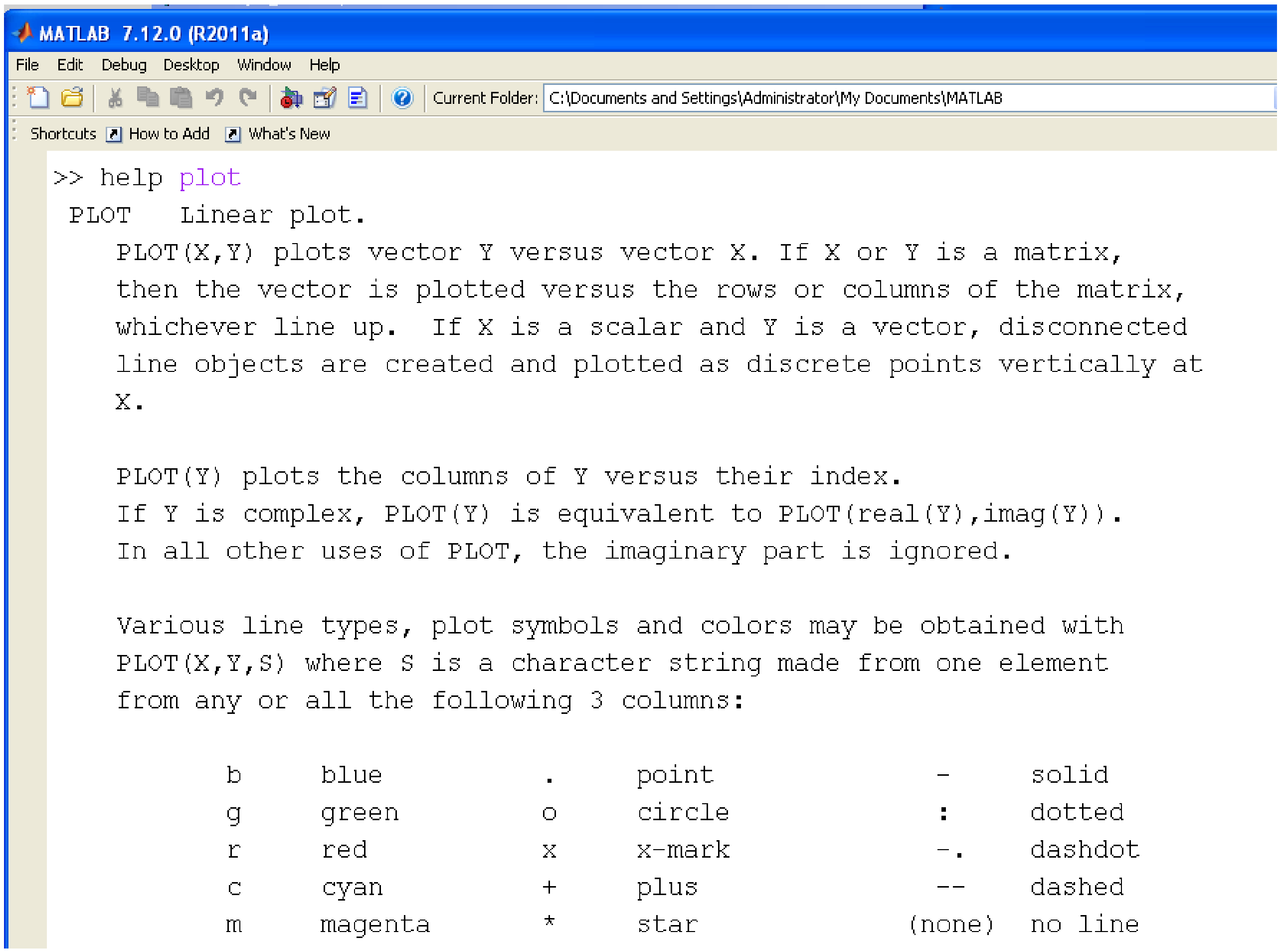
Matlab Plot Command	x axis	y axis	type of function
plot(x,y)	linear	linear	$y = ax + b$
semilogx(x,y)	log()	linear	$y = a\log(bx)$
semilogy(x,y)	linear	log()	$y = a\,e^{bx}$
loglog(x,y)	log()	log()	$y = a \cdot b^{x}$
subplot(abc)	Create 'a' rows, 'b' columns of graphs. Starting at #c		

```
x = [0:1:10]';
y = 3*x + 4;

subplot(311)
plot(x,y);
subplot(312)
plot(x,y,'.');
subplot(313);
plot(x,y,'.-');
```



Matlab Help

If you forget how to use a function, type help




```
>> help plot
 PLOT   Linear plot.
    PLOT(X,Y) plots vector Y versus vector X. If X or Y is a matrix,
    then the vector is plotted versus the rows or columns of the matrix,
    whichever line up.  If X is a scalar and Y is a vector, disconnected
    line objects are created and plotted as discrete points vertically at
    X.

    PLOT(Y) plots the columns of Y versus their index.
    If Y is complex, PLOT(Y) is equivalent to PLOT(real(Y),imag(Y)).
    In all other uses of PLOT, the imaginary part is ignored.

    Various line types, plot symbols and colors may be obtained with
    PLOT(X,Y,S) where S is a character string made from one element
    from any or all the following 3 columns:

           b     blue          .     point              -     solid
           g     green         o     circle             :     dotted
           r     red           x     x-mark             -.    dashdot
           c     cyan          +     plus               --    dashed
           m     magenta       *     star             (none)  no line
```

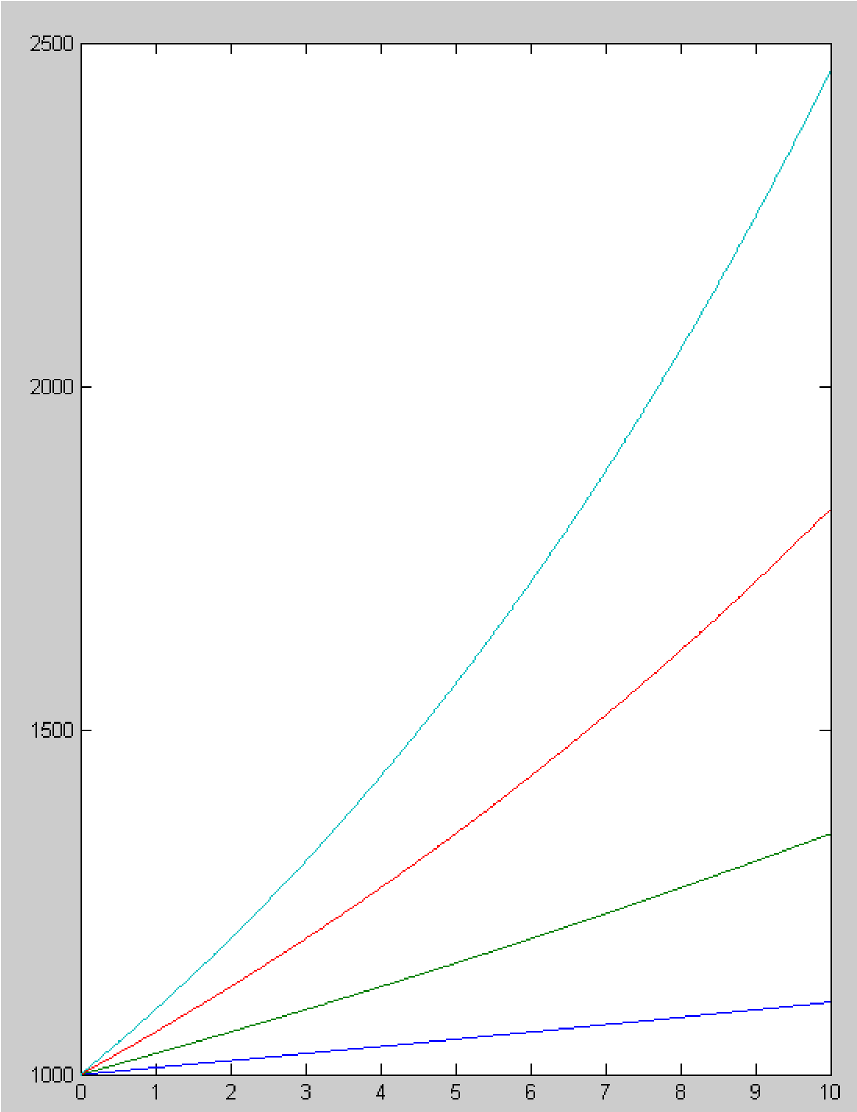
Multiple Plots on the same graph:

```
plot(x1,y1, x2,y2, x3,y3)
plot(x, [y1,y2,y3])
```

Invest $1000 for 10 years at...

- 1% interest
- 3% interest
- 6% interest
- 9% interest

```
t = [0:0.01:10]';
y1 = 1000 * exp(0.01*t);
y3 = 1000 * exp(0.03*t);
y6 = 1000 * exp(0.06*t);
y9 = 1000 * exp(0.09*t);

% Method #1
plot(t,y1,t,y3,t,y6,t,y9)

% Method #2
plot(t,[y1,y3,y6,y9])
```



Polynomials

poly([a,b,c])

- Give a polynomial with roots at (a, b, c)

roots([a,b,c,d])

- Find the roots of the polynomial

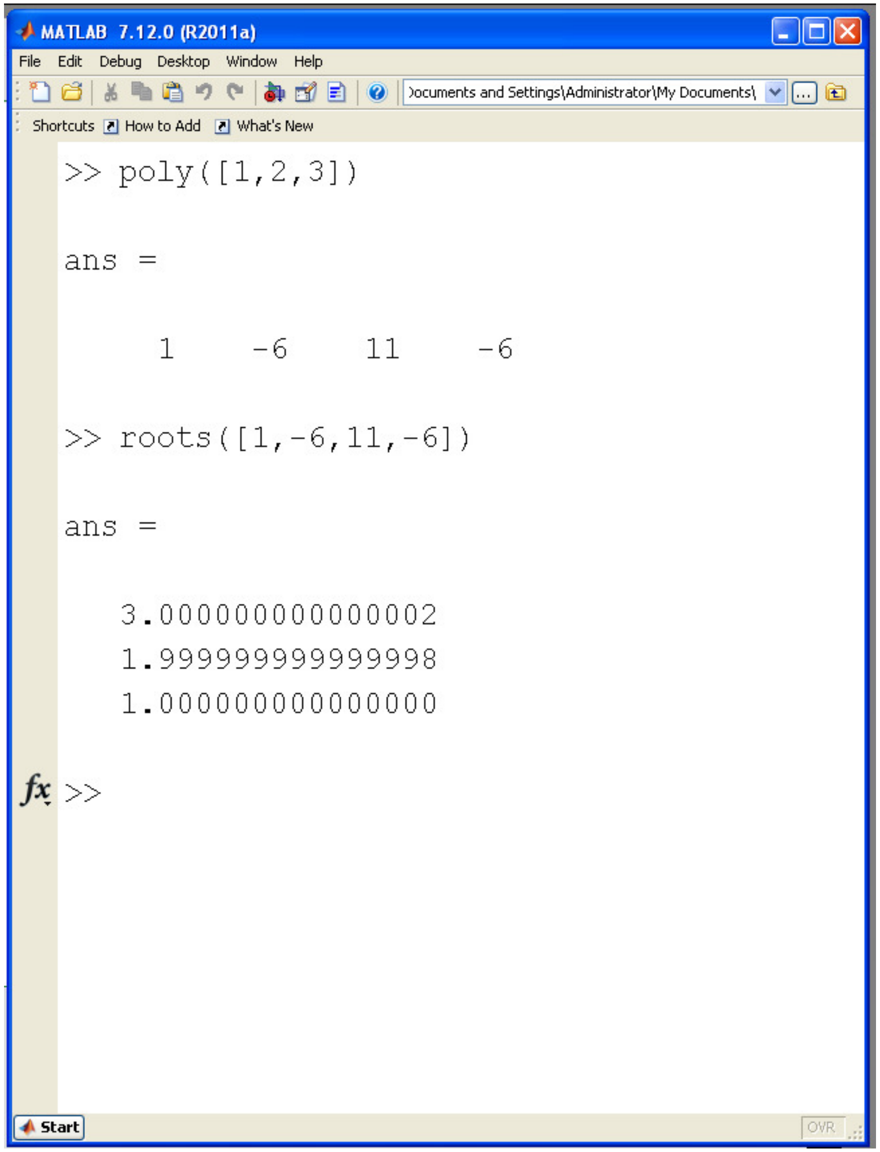
$$ax^3 + bx^2 + cx + d = 0$$

```
poly([1,2,3])

    1    -6    11    -6
```

$$y = x^3 - 6x^2 + 11x - 6$$

$$= (x-1)(x-2)(x-3)$$

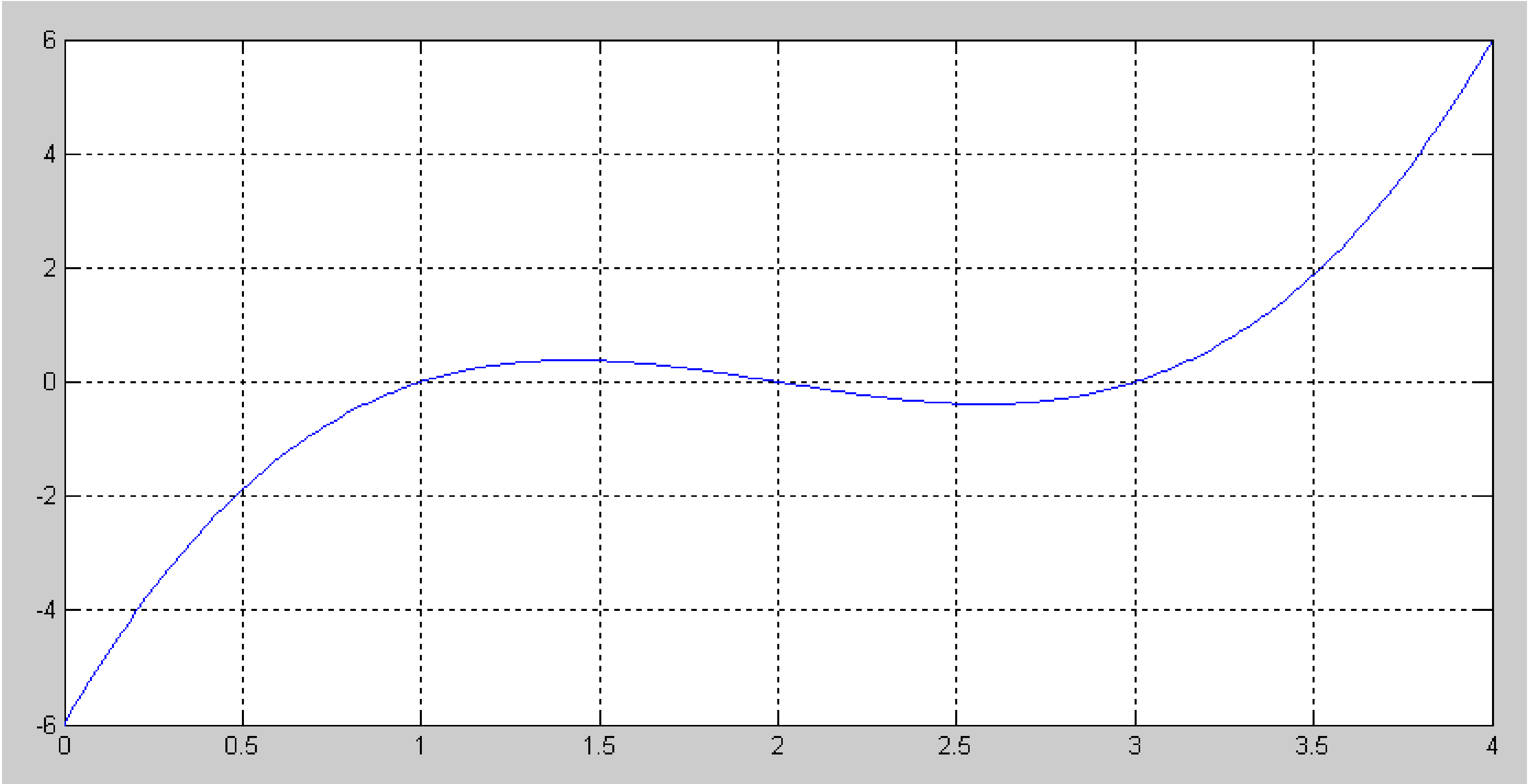
```
roots([1,-6,11,-6])

   3.0000
   2.0000
   1.0000
```



Note: The roots are the zero crossings

- Roots = { 1, 2, 3 }

```
x = [0:0.01:4]';
y = x.^3 - 6*(x.^2) + 11*x - 6;
plot(x,y);
grid on
```



Change the Problem to Fit the Solution

roots() finds the zero crossings of a polynomial

$$0 = x^3 + 5x^2 + 7x + 2$$

If you want to find a different answer, change the problem

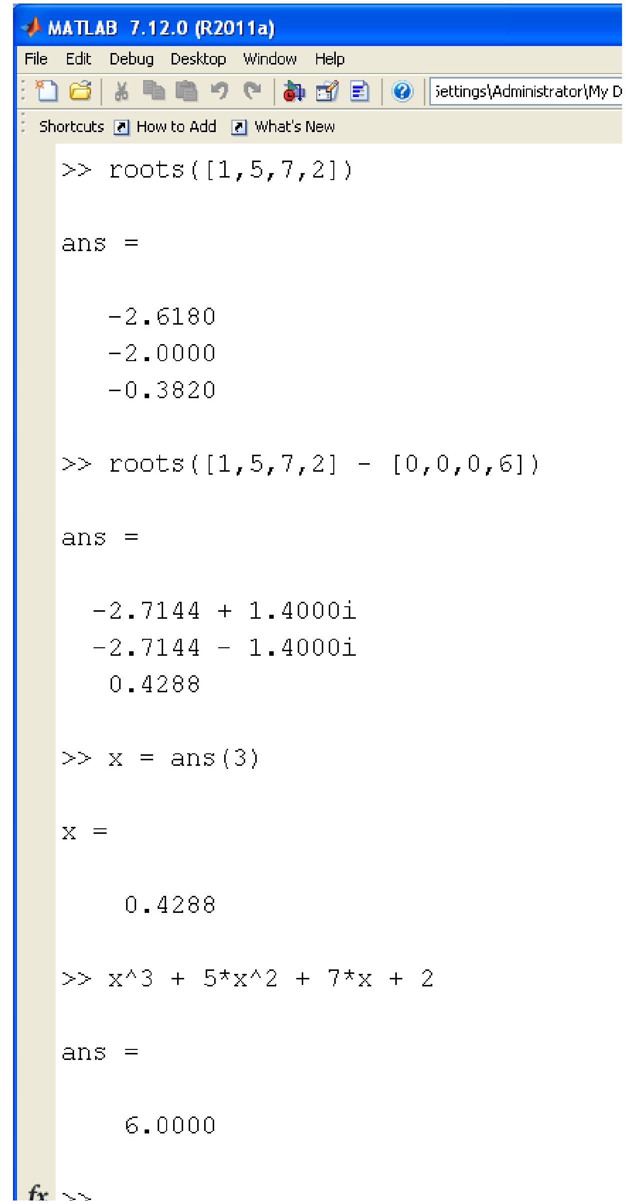
$$6 = x^3 + 5x^2 + 7x + 2$$

becomes

$$0 = (x^3 + 5x^2 + 7x + 2) - (6)$$

Note that

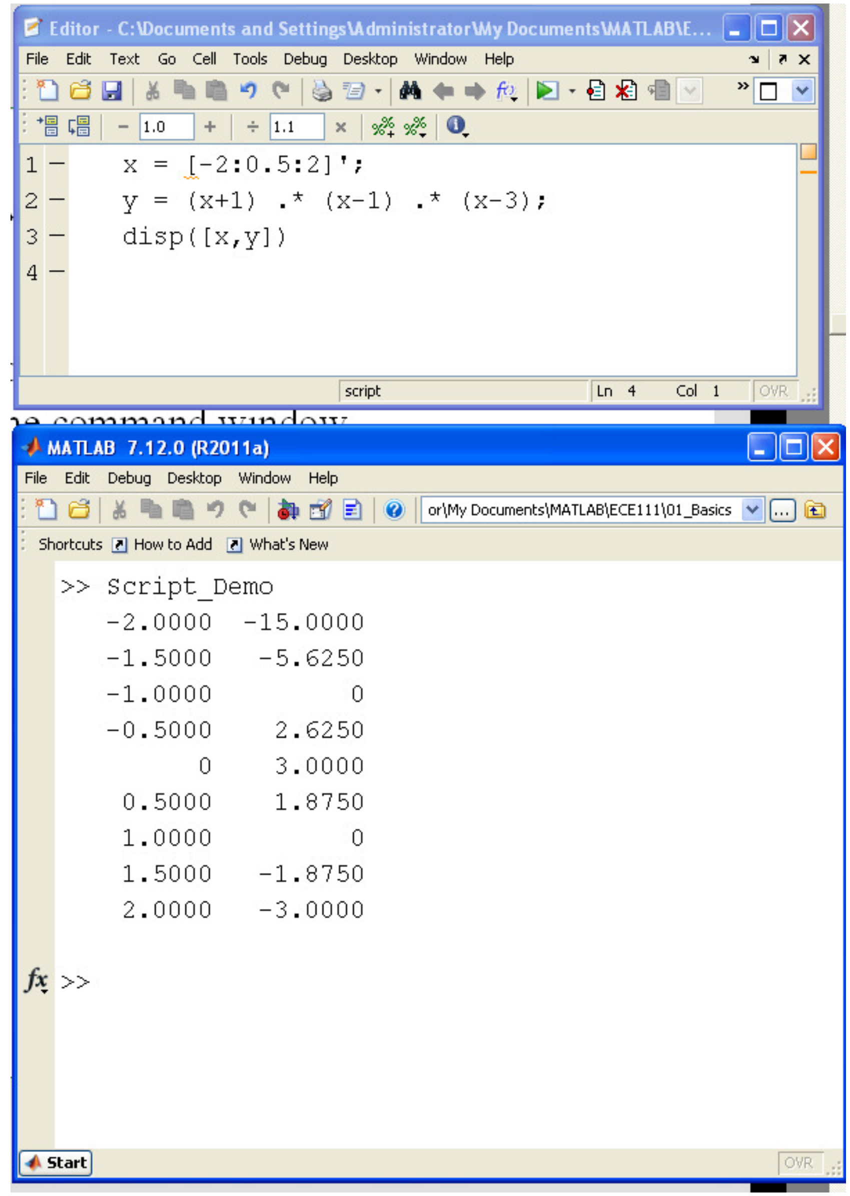
- $6 = 0x^3 + 0x^2 + 0x + 6$
- You have to use matricies with similar dimensions



Matlab Scripts

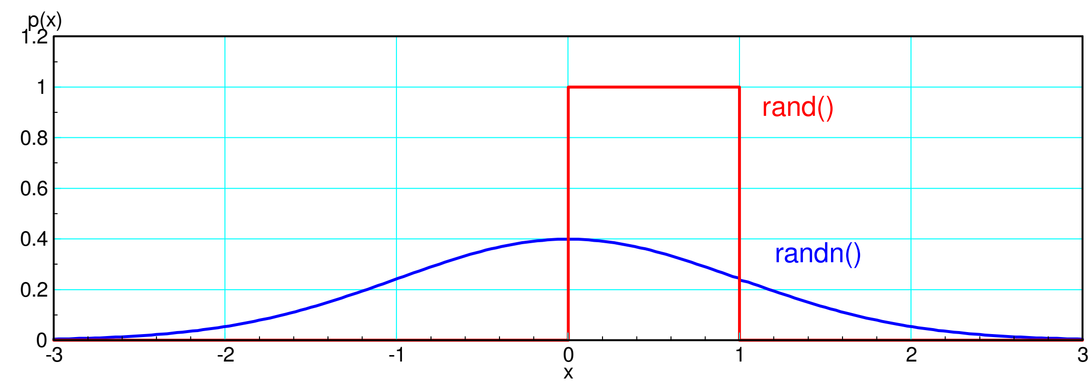
Instead of typing the same set of commands over-and-over again, you can place these Matlab commands in a file (a Matlab script)

- The file must have a .m extension
- You can execute this script using the green arrow
- You can execute this script by calling it from the command window

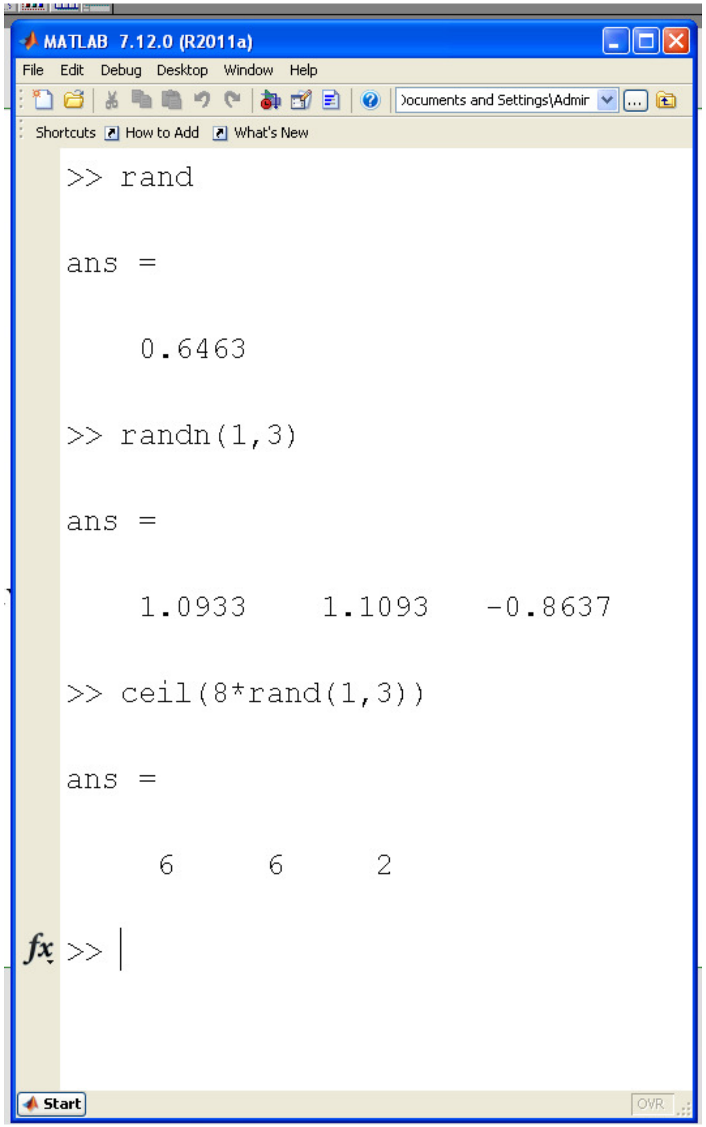


Random Numbers: Rolling Dice

```
rand            random number: (0,1)
randn           standard normal random #
```




```
rand(1,5)               1x5 matrix of random #

ceil( 6*rand )          6-sided die

ceil( 8*rand(1,3) )     3d8

sum( 6*rand(5,1) )      sum of 5d6
                        (level 5 fireball)
```



If-Statement

```
if - end
if - else - end
if - elseif - end
```

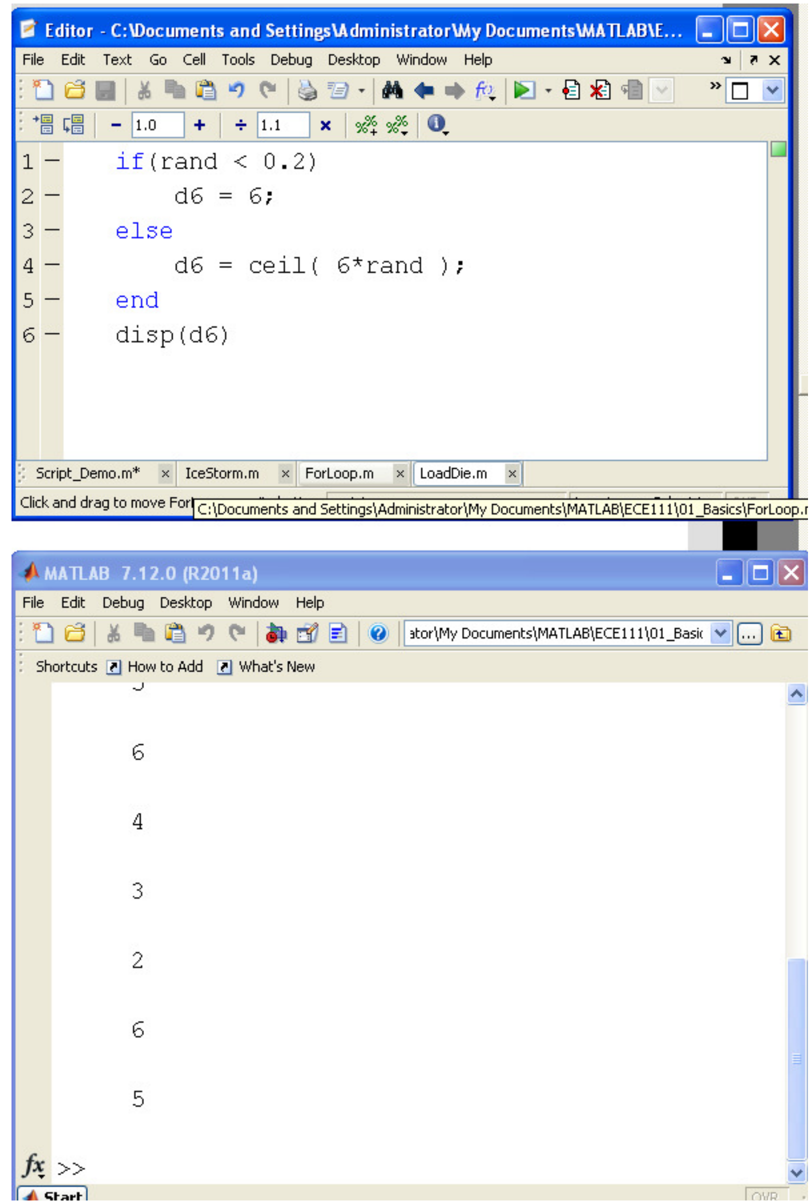
Do a set of operations if a statement is true

Valid boolean statements:

`(N == 3)`	$N = 3$
`(N > 3)`	$N > 3$
`(N >= 3)`	$N \geq 3$
`(N != 3)`	$N \neq 3$
`(N >= 3)*(N <= 7)`	$3 \leq N \leq 7$

Example: Roll a loaded die

- 20% of the time you always roll a 6
- The rest of the time it's a fair die
- Each time you run the script, you get a new die roll



For-Loops

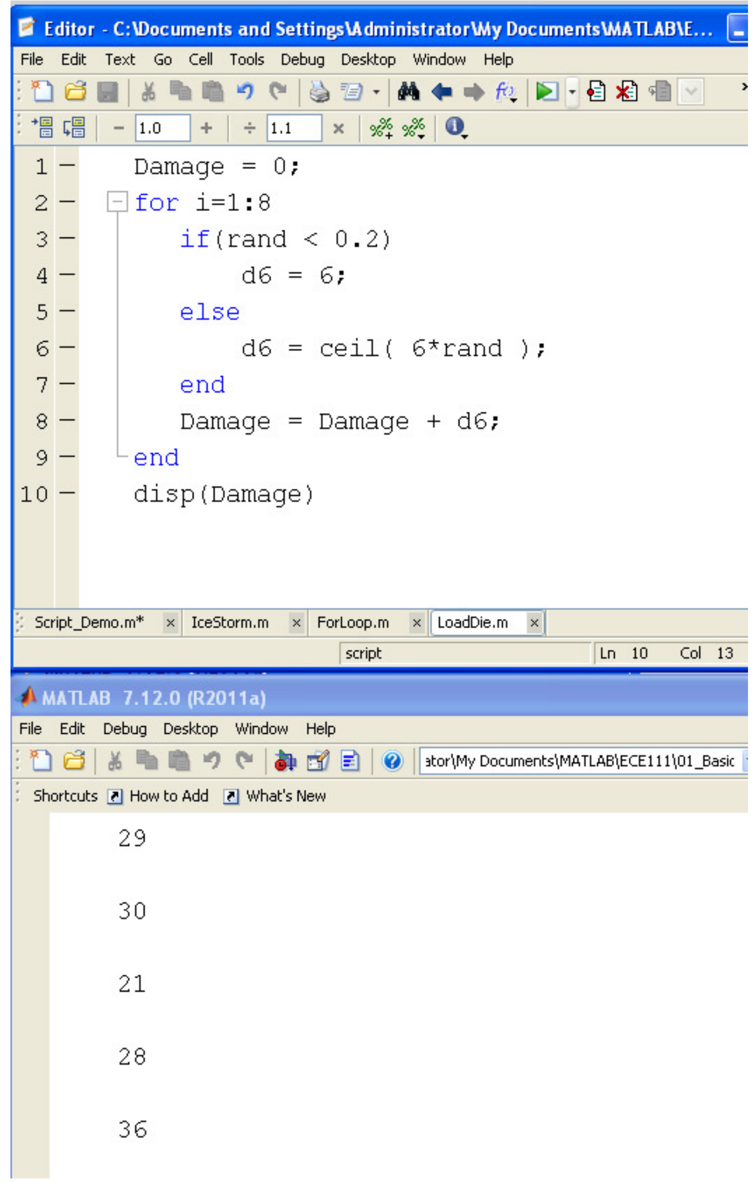
```
for i=1:100
    Matlab commands
    end
```

Repeat a set of commands a fixed number of times

Terminate with an *end* statement

Example: Cast a level-8 Fireball

- y = 8d6
- Use loaded dice (20% chance of a 6)
- Each time you run the script, you get a different result



Monte-Carlo Simulations

- One extermely useful capability of Matlab is to run Monte Carlo simulations
- To find the probability of an event, repeat an experiment 100,000 times
- The probability is then roughly the percentage of the time the outcome happened

Procedure:

- Write a script to run an experiment one time
- Once that works, repeat 100,000 times
- (place the code inside a for-loop)

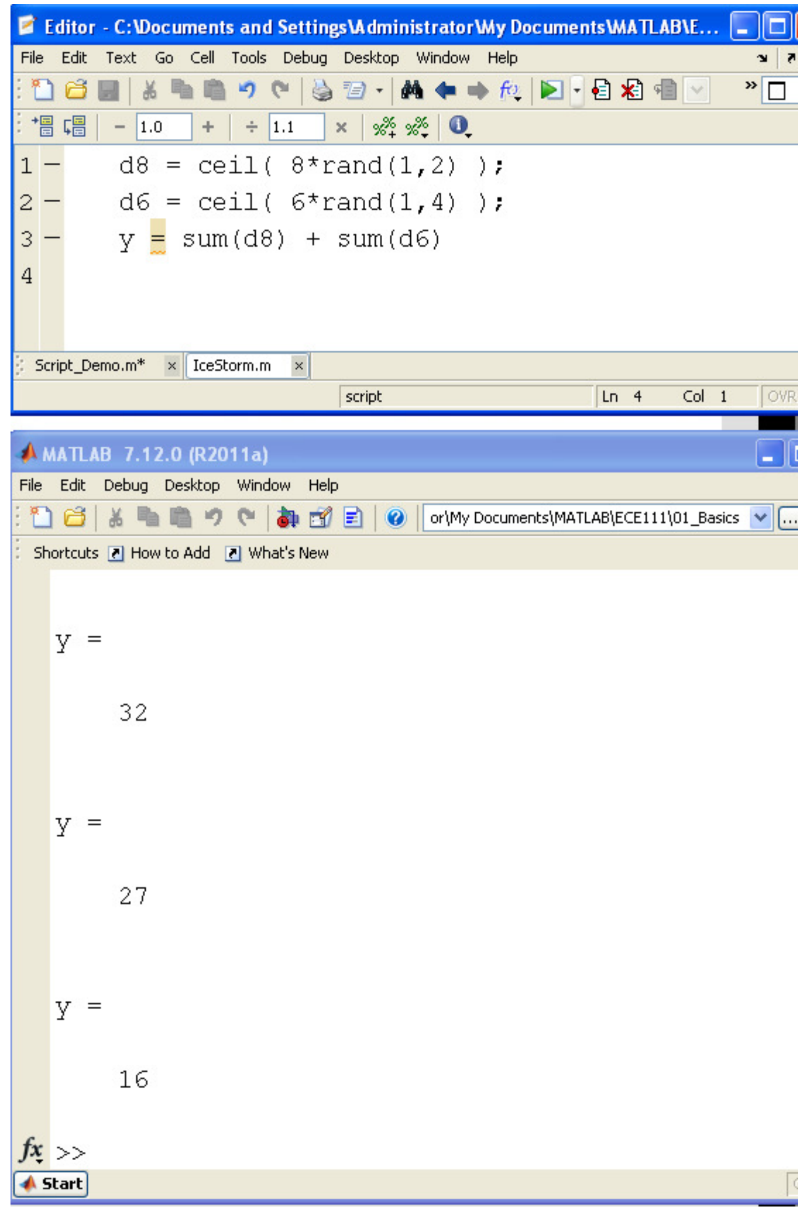
Example: Ice Storm

The Dungeons and Dragon's spell *Ice Storm* does 6-40 damage

- The sum of two 8-sided dice and four 6-sided dice
- y = 2d8 + 4d6

Determine...

- The probability of doing N damage
- The probability that N > 30

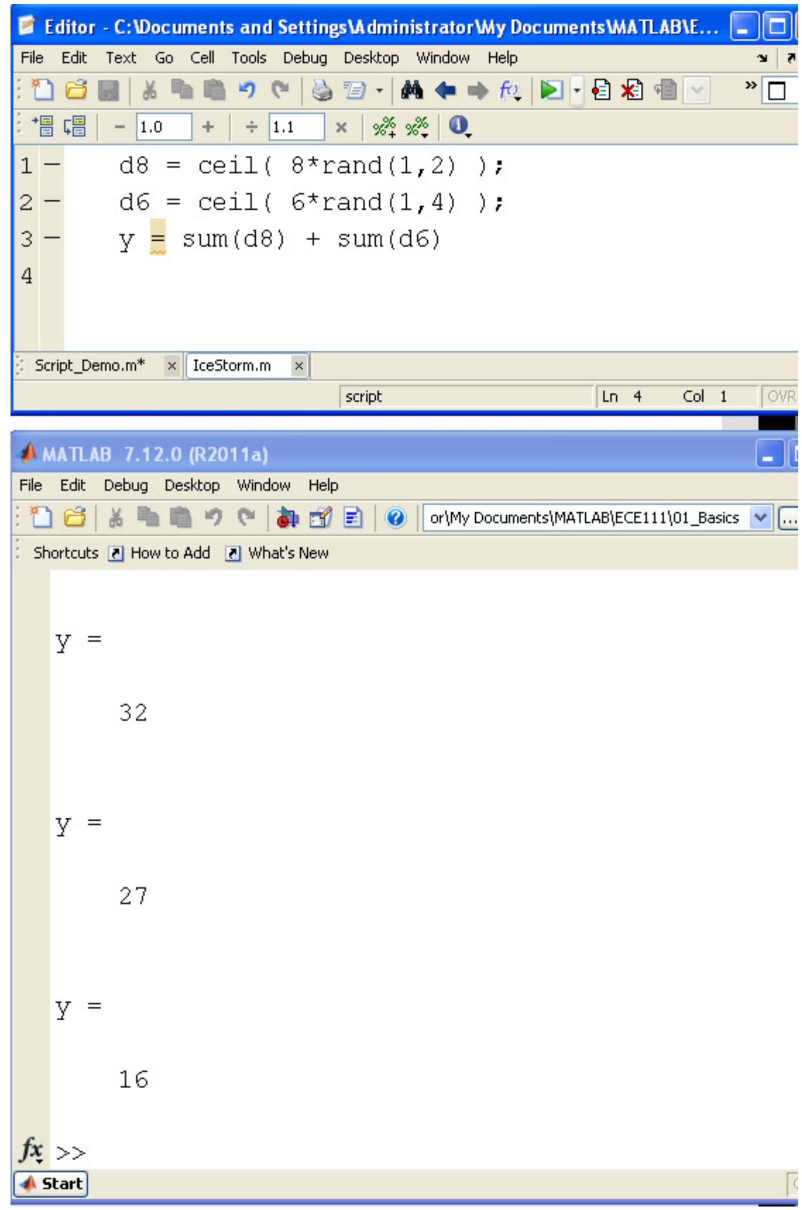


Solution: Step 1

- Create a file *IceStorm.m*
- Find y = 2d8 + 4d6

Note that every time you run this script, you get a different answer

- it's random



Solution: Step 2

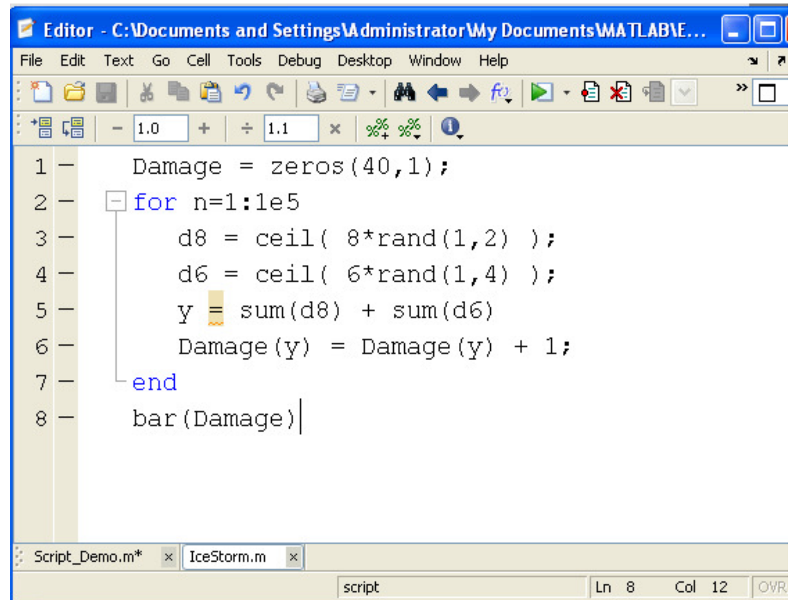
- Repeat 100,000 times
- Keep track of how many times you did y damage

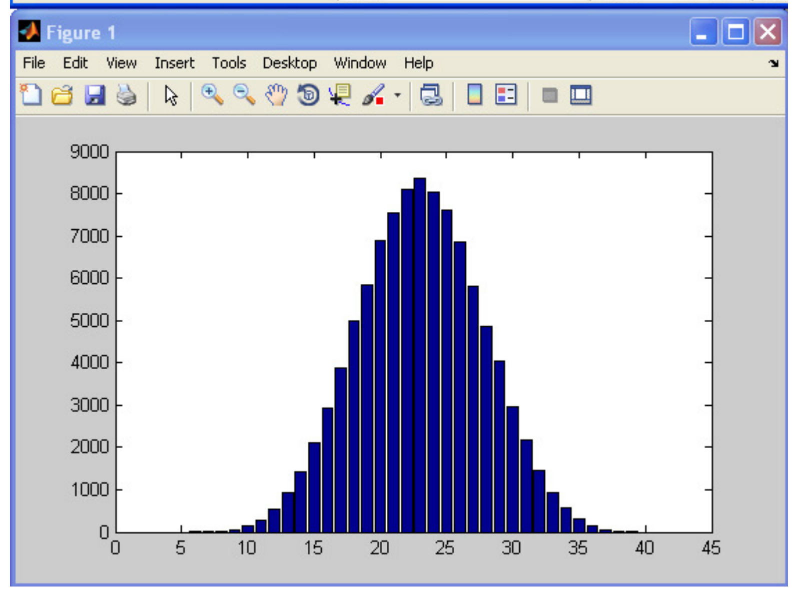
The odds of doing 30 damage

```
>> Damage(30) / 100000
ans =     0.0297
```

The odds of doing 30 or more damage

```
>> sum(Damage(30:40) / 100000)
ans =     0.0866
```





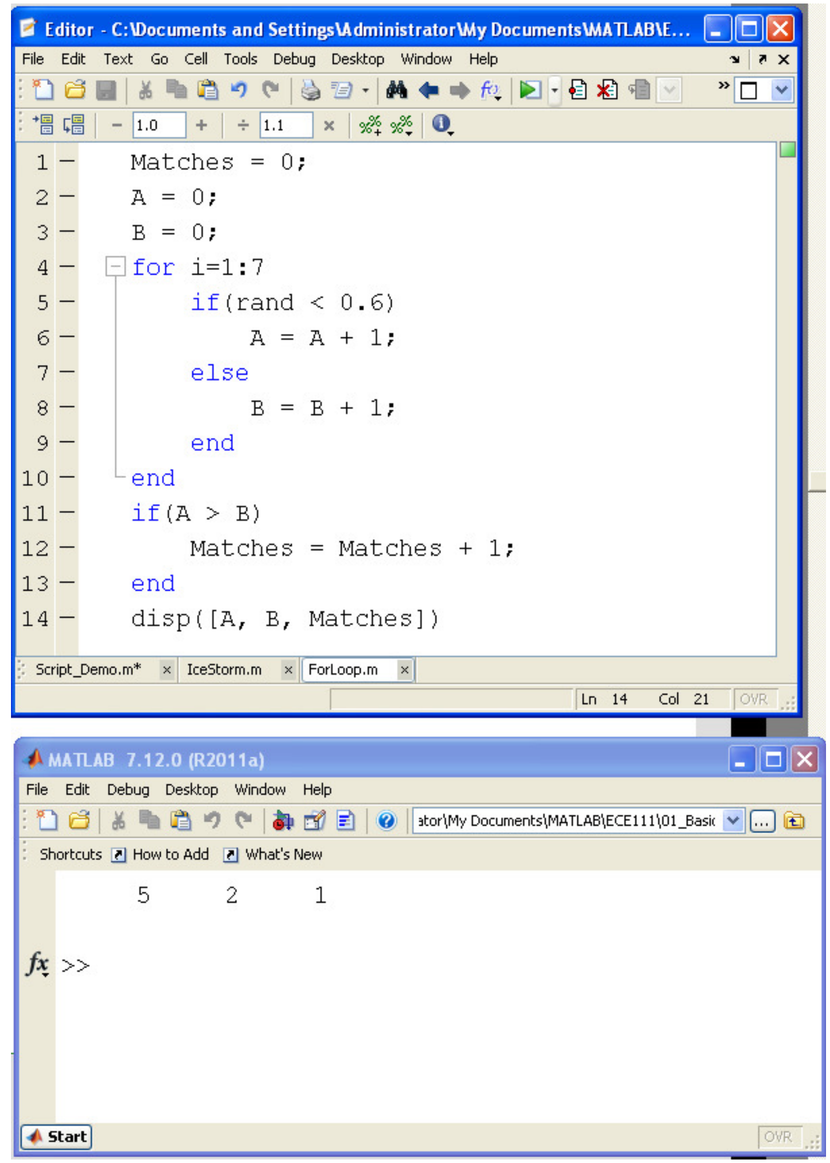
Monte-Carlo Example #2

A and B are playing a match

- A has a 60% chance of winning any given game.
- What is the probability that A will win the match?

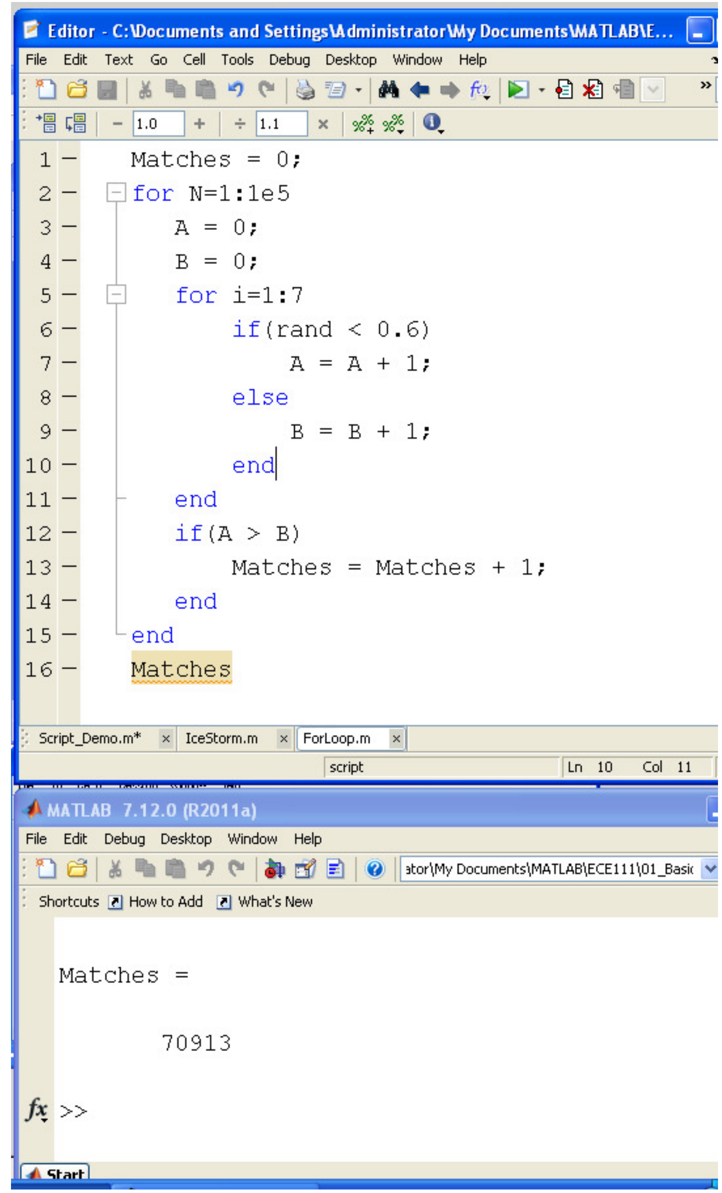
Start by playing a single match

- A won the match 5-2
- Different result each time you run the script



Now repeat for 100,000 matches

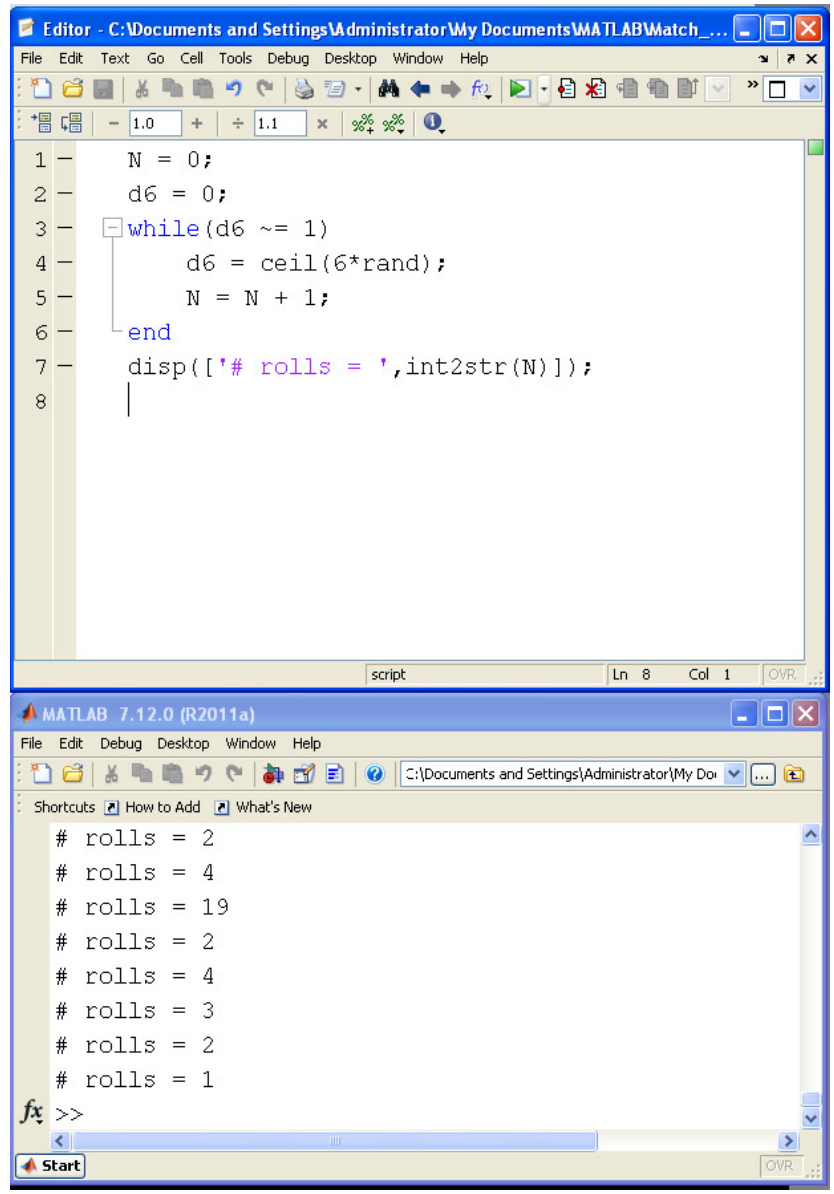
- A wins 70,913 times in 100,000 matches
- A has roughly a 70.9% chance of winning any given match



While-Loop

```
while(statement is true)
  do the following
  end
```

Example: Count how many times you roll a die until you get a 1



While-Loop (cont'd)

What is the chance that it will take 7 or more rolls to get a 1?

Repeat 100,000 times

- Monte-Carlo Simulation

In 100,000 trials

- It took 7 or more rolls 33,343 times
- There is about a 33.34% chance it will take 7 or more rolls to get a 1

MATLAB 7.12.0 (R2011a)

```
>> X = 0;

for n=1:1e5
   N = 0;
   d6 = 0;

   while( d6 ~= 1 )
      d6 = ceil(6*rand);
      N = N + 1;
      end
   if(N >= 7)
      X = X + 1;
      end
   end
>> disp(X)
       33343

>> disp(X/1e5)
    0.3334

>>
```

While-Loop (cont'd)

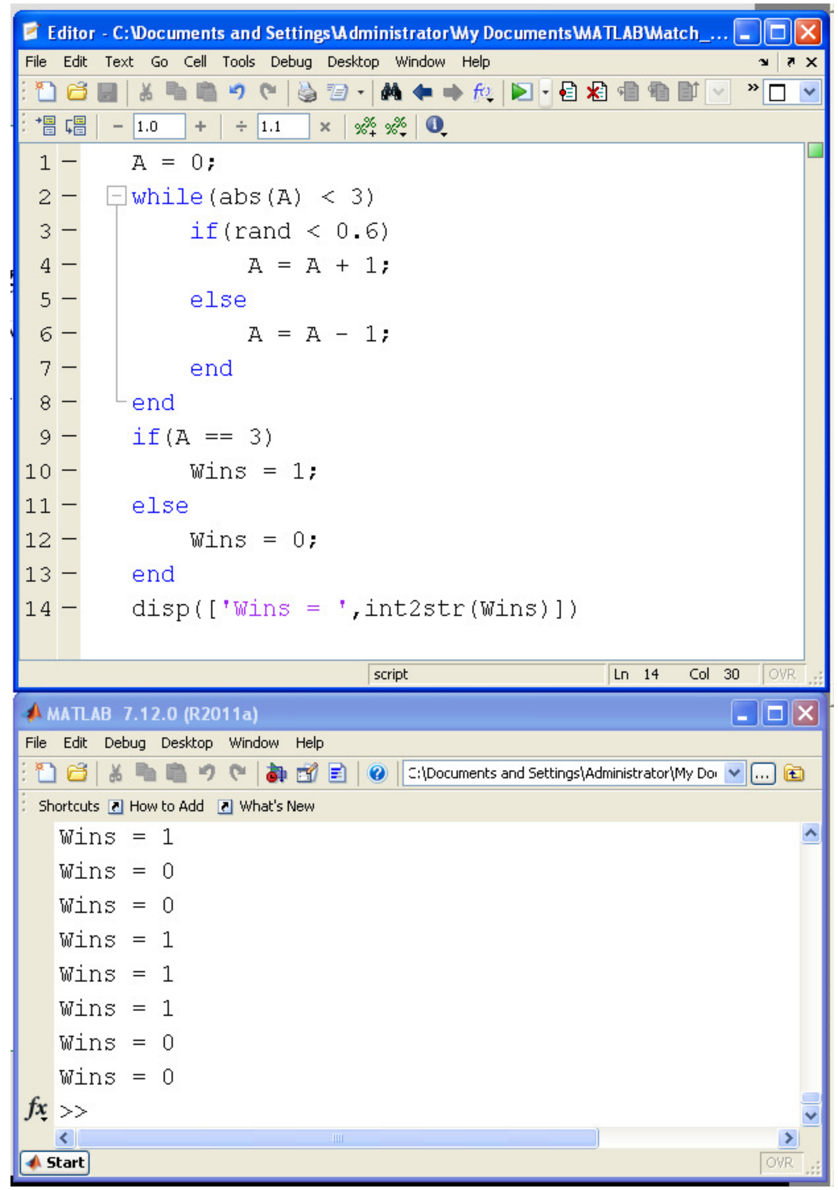
Player A and B are playing a match

- Player A has a 60% chance of winning any given game
- When a player is up 3 games, the match is over

What is the chance player A wins the match?

Solution: Play a single match

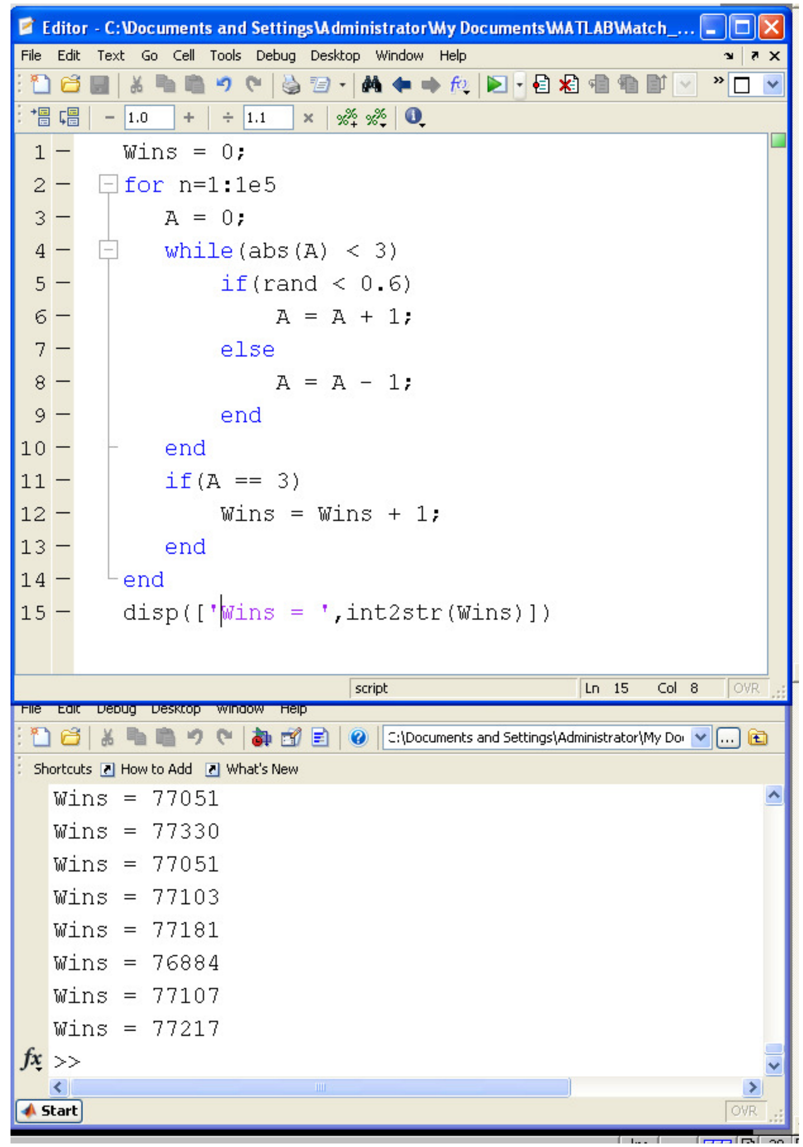
- If A wins, A gains 1 point.
- If A loses, A loses 1 point.
- Keep playing until A is up 3 or down 3



Win-by-3 (cont'd)

Now play 100,000 matches

- A wins about 77% of the time with this format
- TV hates this format since a match can take a very long time



While-Loop: Tennis

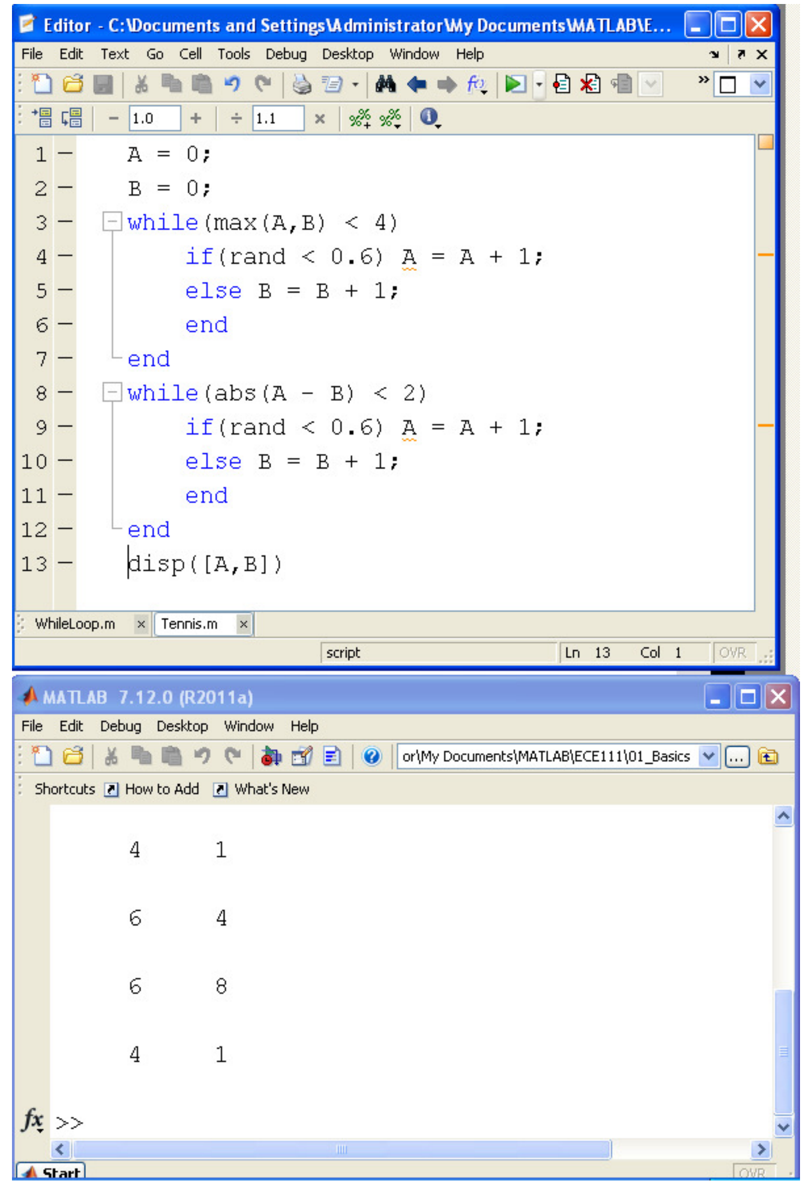
Assume A and B are playing a match

- A has a 60% chance of winning any given game
- If a player wins 4 games and is up by 2 games, the match is over.
- Otherwise, the match continues until a player is up two games.

Find the probability that A wins the match

Solution: Start with playing a single match

- A wins 4 - 1
- A wins 6 - 4
- A loses 8 - 6
- A wins 4 - 1



Now repeat for 100,000 matches

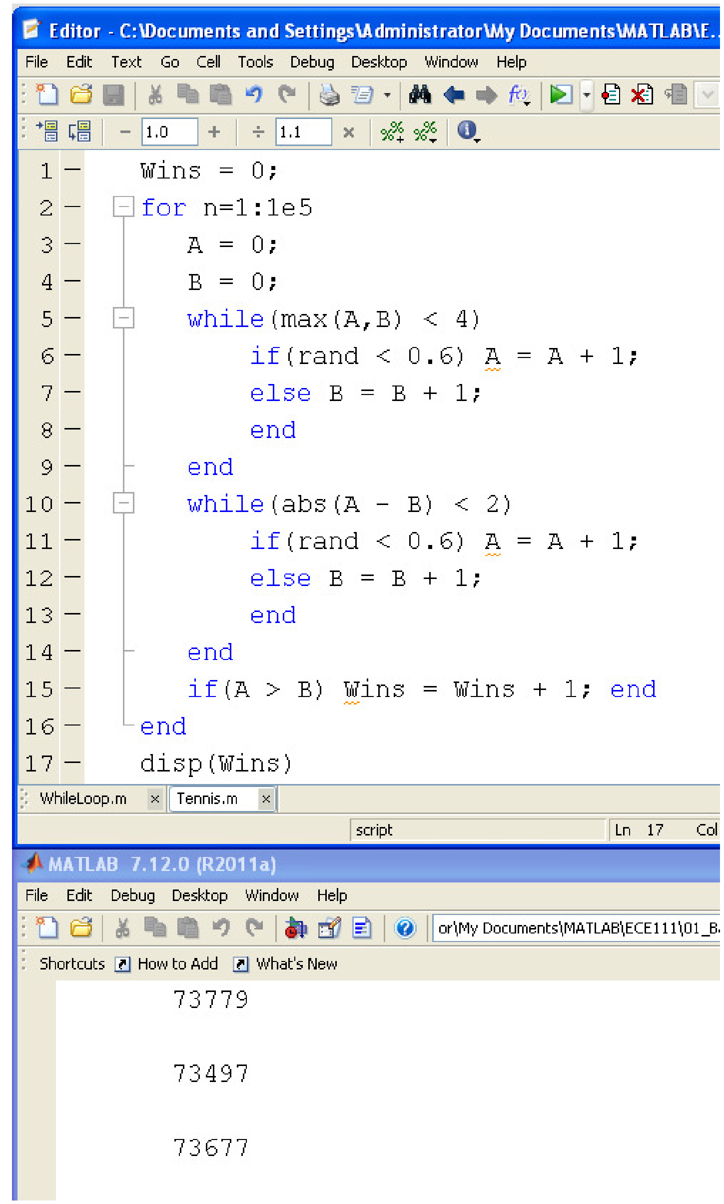
Result is A wins about 73,500 times

- A has about a 73.5% chance of winning the match

Results will vary each time you run this script

- it's random

To find the actual odds, you need to use a student-t test (week #15 of ECE 111)



Summary

Matlab is a fairly friendly computer language

You can use the command window as a calculator

- Adds, subtracts, multiplies, divides

Scripts allow you to try & modify code as you write it

For-loops let you run code multiple times

- Monte-Carlo simulations...

If-statements allow you to check for conditions

- If the sum is 25 or more...

While-loops let you run code until an event happens

- repeat until you roll a 1

Matlab Commands

Display

- format short display results to 4 decimal places
- format long display results to 13 decimal places
- format short e display using scientific notation
- format long e display using scientific notation

Polynomials

- poly(x)
- roots(x)
- conv(x,y)

Analysis

- sqrt(x) square root of x
- log(x) log base e
- log10(x) log base 10
- exp(x) e^x
- exp10(x) 10^x
- abs(x) |x|
- round(x) round to the nearest integer
- floor(x) round down (integer value of x)
- ceil(x) round up to the next integer
- real(x) real part of a complex number
- imag(x) imaginary part of a complex number
- abs(x) absolute value of x, magnitude of a complex number
- angle(x) angle of a complex number (answer in radians)
- unwrap(x) remove the discontinuity at pi (180 degrees) for a vector of angles
- sum(x) sum the columns of x
- prod(x) multiply the columns of x

Trig Functions

- sin(x) sin(x) where x is in radians
- cos(x) cos()
- tan(x) tan()
- asin(x) arcsin(x)
- acos(x) arccos(x)
- atan(x) arctan(x)
- atan2(y,x) angle to a point (x,y)

Probability and Statistics

- factorial(x) x!
- gamma(x) x!
- rand(n,m) create an nxm matrix of random numbers between 0 and 1
- randn(n,m) create an nxm matrix of random numbers with a normal distribution
- length(x) return the dimensions of x
- mean(x) mean (average) of the columns of x
- std() standard deviation of the columns of x

Display Functions

• plot(x)	plot x vs sample number
• plot(x,y)	plot x vs. y
• semilogx(x,y)	log(x) vs y
• semilogy(x,y)	x vs log(y)
• loglog(x,y)	log(x) vs log(y)
• mesh(x)	3d plot where the height is the value at x(a,b)
• contour(x)	contour plot
• bar(x,y)	draw a bar graph
• xlabel('time')	label the x axis with the word 'time'
• ylabel()	label the y axis
• title()	put a title on the plot
• grid()	draw the grid lines

Useful Commands

• hold on	don't erase the current graph
• hold off	do erase the current graph

- diary create a text file to save whatever goes to the screen
- linepace(a, b, n) create a 1xn array starting at a, increment by b
- logspace(a,b,n) create a 1xn array starting at 10^a going to 10^b, spaced logarithmically
- subplot() create several plots on the same screen
- disp('hello') display the message *hello*

Utilities

- format set the display format
- zeros(n,m) create an nxm matrix of zeros
- eye(n,m) create an nxm matrix with ones on the diagonal
- ones(n,m) create an nxm matrix of ones
- help help using different functions
- pause(x) pause x seconds (can be a fraction). Show the graph as well
- clock the present time
- etime the difference between two times
- tic start a stopwatch
- toc the number of seconds since tic