- Previous Lecture:
  - User-defined functions
    - Examples with varying numbers of input and output parameters
    - Local memory space
- Today's Lecture:
  - Probability and random numbers
  - 1-d array—vector
  - More MATLAB graphics
- Announcement:
  - Section this week in the computer labs
  - P3 posted, due 3/6 at 6pm

Random numbers

- *Pseudorandom* numbers in programming
- Function **`rand(…)`** generates random real numbers in the interval (0,1). All numbers in the interval (0,1) are equally likely to occur—uniform probability distribution.
- Examples:

`rand(1)`	one random # in (0,1)
`6*rand(1)`	one random # in (0,6)
`6*rand(1)+1`	one random # in (1,7)

Simulate a fair 6-sided die

Which expression(s) below will give a random *integer* in [1..6] with equal likelihood?

A `round(rand(1)*6)`
B `ceil(rand(1)*6)`
C *Both expressions above*

% Simulate the rolling of 2 fair dice totalOutcome= ???

A `ceil(rand(1)*12)`
B `ceil(rand(1)*11)+1`
C `floor(rand(1)*11)+2`
D *2 of the above*
E *None of the above*

Discover the answer in section this week!

1-d array: **vector**

- An array is a named collection of like data organized into rows or columns
- A 1-d array is a row or a column, called a *vector*
- An *index* identifies the position of a value in a vector

score	93	92	87	0	90	82
	1	2	3	4	5	6

Array index starts at 1

x	5	.4	.91	-4	-1	7
	1	2	3	4	5	6

Let k be the index of vector x, then

- k must be a positive integer
- 1<= k <= length(x)
- To access the k^{th} element: x(k)

Drawing a single line segment

```
a= 0;  % x-coord of pt 1
b= 1;  % y-coord of pt 1
c= 5;  % x-coord of pt 2
d= 3;  % y-coord of pt 2
plot([a c], [b d], '-*')
```

Line/marker format

x-values
A vector!

y-values
A vector!

Drawing a polygon (multiple line segments)

```
% Draw a rectangle with the lower-left
% corner at (a,b), width w, height h.
x= [                    ];  % x data
y= [                    ];  % y data
plot(x, y)
```

Fill in the missing vector values!

Color is 3-vector, sometimes called the RGB values

- Any color is a mix of red, green, and blue
- Example:

```
c= [0.4  0.6  0]
```

- Each component is a real value in [0,1]
- [0 0 0] is white
- [1 1 1] is black

Let's *compute* colors!

Show "all combinations" of red and blue

- Assume some kind of granularity—discretize the color value range for red and blue
- Assume no contribution from green (set to 0)

Program development:

- Compute the color first; worry about drawing later
- Decide on granularity, say, Δ=.25


```
% All combinations of R and B

gran= 0.25;  %granularity
```


Things to consider/try on the color computation problem

- The granularity was the programmer's choice
- Choosing how to display the colors was a design problem!
- What if you want compute "all combinations" of the R, G, and B values? How would the program change?
- Another design problem: how to show all color combinations of the ***3***-vector on a ***2***-dimensional plot?