- Previous Lecture:
  - User-defined functions
    - Examples with varying numbers of input and output parameters
    - Local memory space
- Today's Lecture:
  - Subfunctions
  - 1-d array—vector
  - More MATLAB graphics
  - Probability and random numbers
- Announcements:
  - Discussion section this week in the lab, UP B7
  - Please register your clicker online even if you've registered in class

What is the output?

```
x = 1;
x = f(x+1);
y = x+1;
disp(y)
```

```
function y = f(x)
x = x+1;
y = x+1;
```

A: 1 B: 2 C: 3 D: 4 E: 5

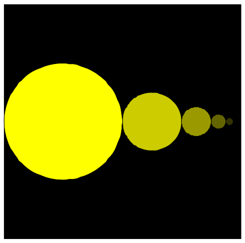
Execute the statement `y= foo(x)`

- Matlab looks for a function called foo (m-file called foo.m)
- Argument (value of x) is copied into function foo's local parameter
  - called "pass-by-value," one of several argument passing schemes used by programming languages
- Function code executes within its own workspace
- At the end, the function's output argument (value) is sent from the function to the place that calls the function. E.g., the value is assigned to y.
- Function's workspace is deleted
  - If foo is called again, it starts with a new, empty workspace

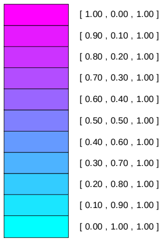
Subfunction

- There can be more than one function in an M-file
- top function is the main function and has the name of the file
- remaining functions are subfunctions, accessible only by the functions in the same m-file
- Each (sub)function in the file begins with a function header
- Keyword **end** is not necessary at the end of a (sub)function

Graphics and color interpolation



Used given `DrawDisk`

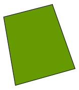


Learn about `fill`, `text` and practice working with *vectors*

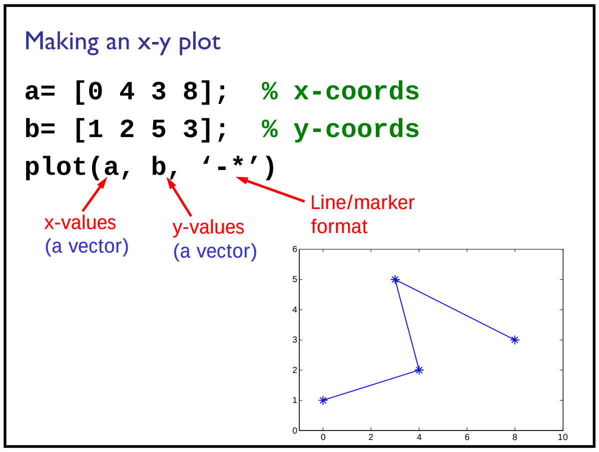
Color computation

- Color is a 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 black
- [1 1 1] is white



Making an x-y plot

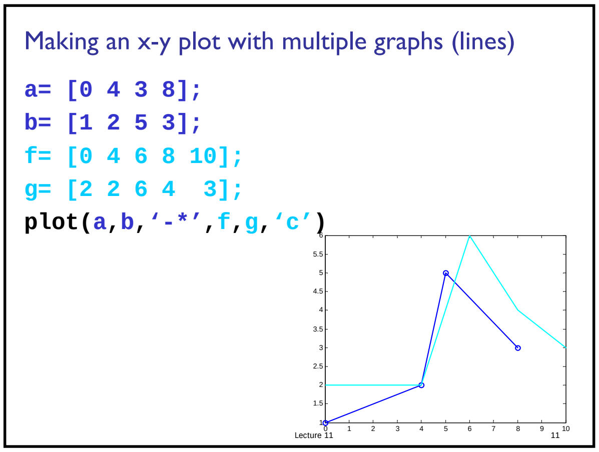
```
a= [0 4 3 8];   % x-coords
b= [1 2 5 3];   % y-coords
plot(a, b, ‘-*’)
```

x-values
(a vector)

y-values
(a vector)

Line/marker
format

6
5
4
3
2
1
0
0
2
4
6
8
10

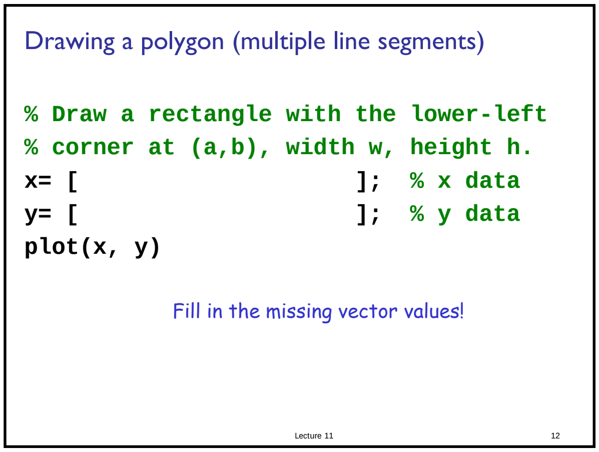


Making an x-y plot with multiple graphs (lines)

```
a= [0 4 3 8];
b= [1 2 5 3];
f= [0 4 6 8 10];
g= [2 2 6 4  3];
plot(a,b,‘-*’,f,g,‘c’)
```

6
5.5
5
4.5
4
3.5
3
2.5
2
1.5
1
0
1
2
3
4
5
6
7
8
9
10

Lecture 11 11

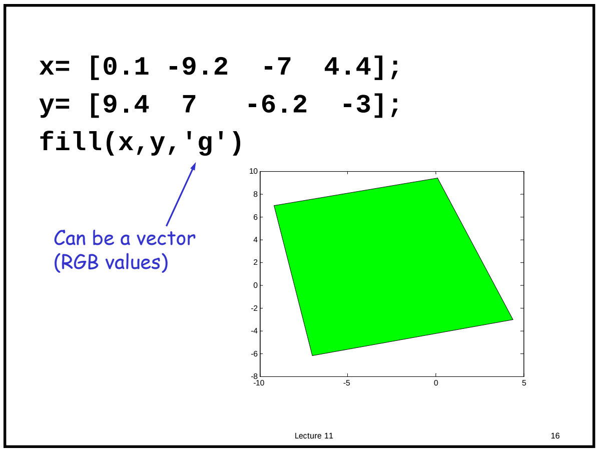


Drawing a polygon (multiple line segments)

```
% 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!

Lecture 11 12

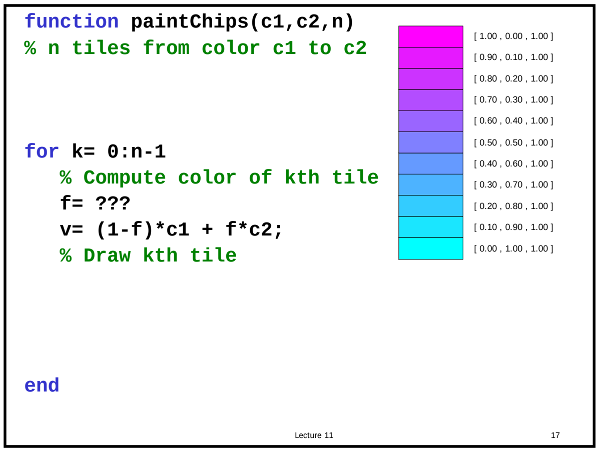


```
x= [0.1 -9.2  -7  4.4];
y= [9.4  7   -6.2  -3];
fill(x,y,'g')
```

Can be a vector
(RGB values)

10
8
6
4
2
0
-2
-4
-6
-8
-10
-5
0
5

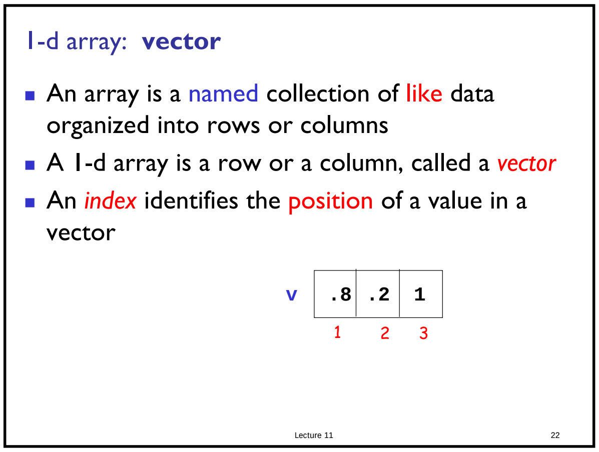
Lecture 11 16



```
function paintChips(c1,c2,n)
% n tiles from color c1 to c2



for k= 0:n-1
   % Compute color of kth tile
   f= ???
   v= (1-f)*c1 + f*c2;
   % Draw kth tile




end
```

[1.00 , 0.00 , 1.00]
[0.90 , 0.10 , 1.00]
[0.80 , 0.20 , 1.00]
[0.70 , 0.30 , 1.00]
[0.60 , 0.40 , 1.00]
[0.50 , 0.50 , 1.00]
[0.40 , 0.60 , 1.00]
[0.30 , 0.70 , 1.00]
[0.20 , 0.80 , 1.00]
[0.10 , 0.90 , 1.00]
[0.00 , 1.00 , 1.00]

Lecture 11 17

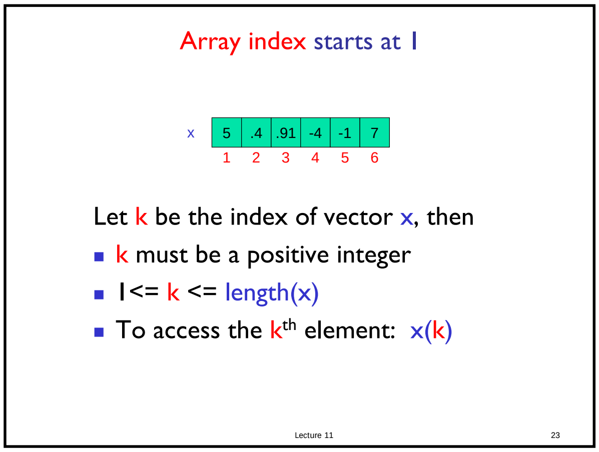


1-d array: **vector**

- 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

v	.8	.2	1
	1	2	3

Lecture 11 22

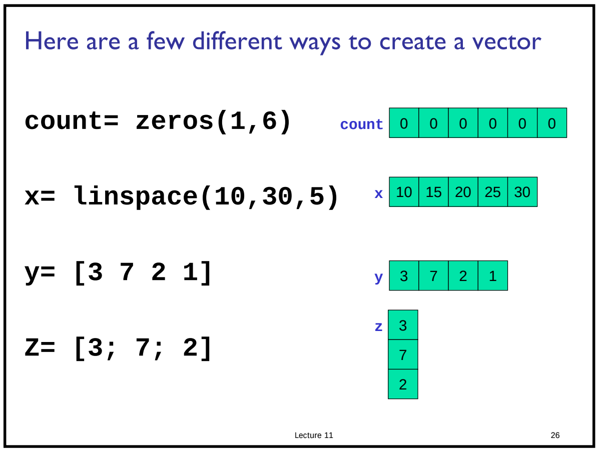


Array index starts at 1

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 kth element: x(k)



Here are a few different ways to create a vector

```
count= zeros(1,6)
```

count	0	0	0	0	0	0

```
x= linspace(10,30,5)
```

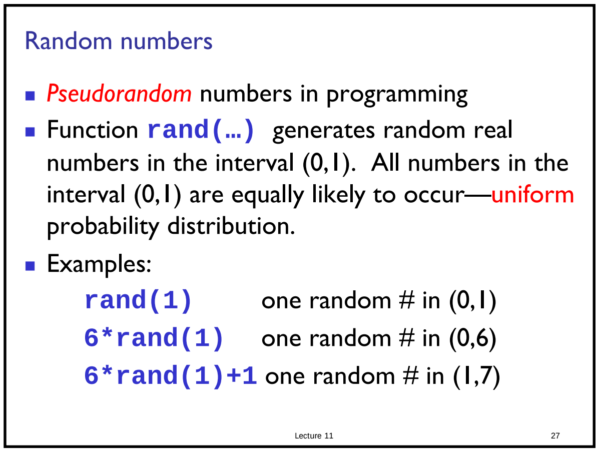
x	10	15	20	25	30

```
y= [3 7 2 1]
```

y	3	7	2	1

```
z= [3; 7; 2]
```

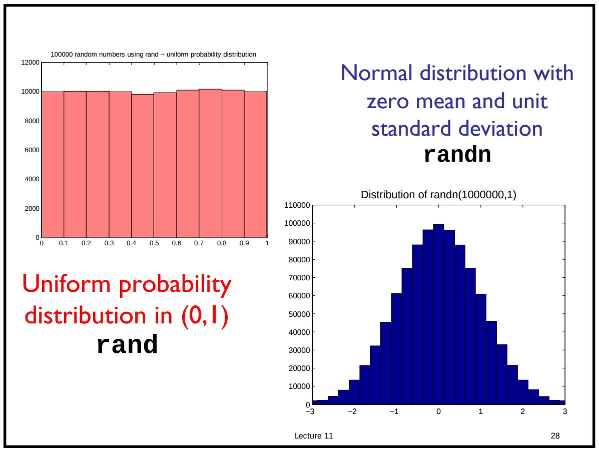
z
3
7
2



Random numbers

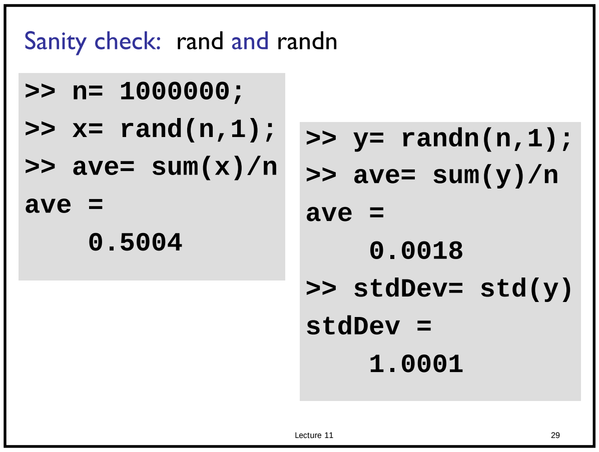
- *Pseudorandom* numbers in programming
- Function `rand(…)` generates random real numbers in the interval (0,1). All numbers in the interval (0,1) are equally likely to occur—uniform probability distribution.
- Examples:

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



Normal distribution with zero mean and unit standard deviation
randn

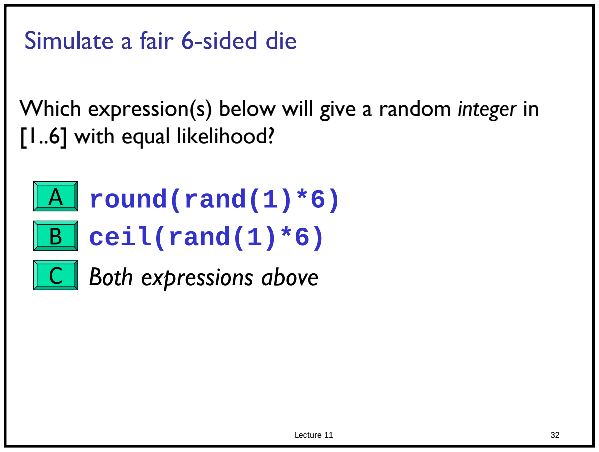
Uniform probability distribution in (0,1)
rand



Sanity check: rand and randn

```
>> n= 1000000;
>> x= rand(n,1);
>> ave= sum(x)/n
ave =
    0.5004
```

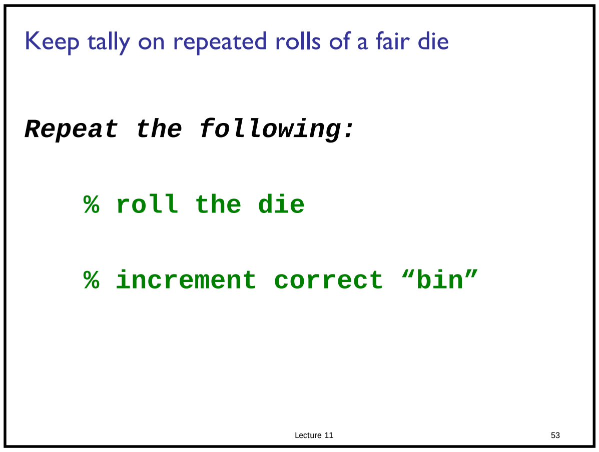
```
>> y= randn(n,1);
>> ave= sum(y)/n
ave =
    0.0018
>> stdDev= std(y)
stdDev =
    1.0001
```



Simulate a fair 6-sided die

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

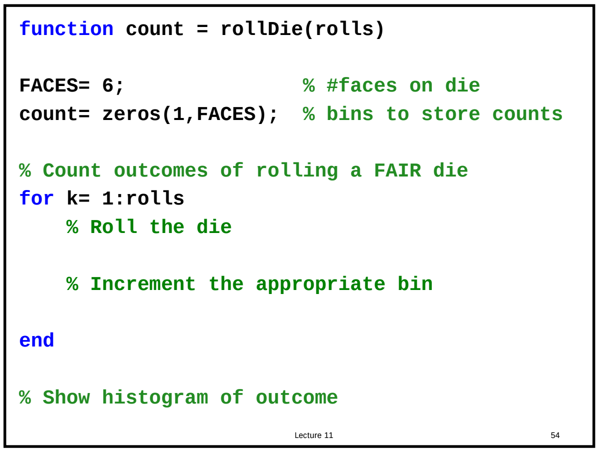
A `round(rand(1)*6)`
B `ceil(rand(1)*6)`
C *Both expressions above*



Keep tally on repeated rolls of a fair die

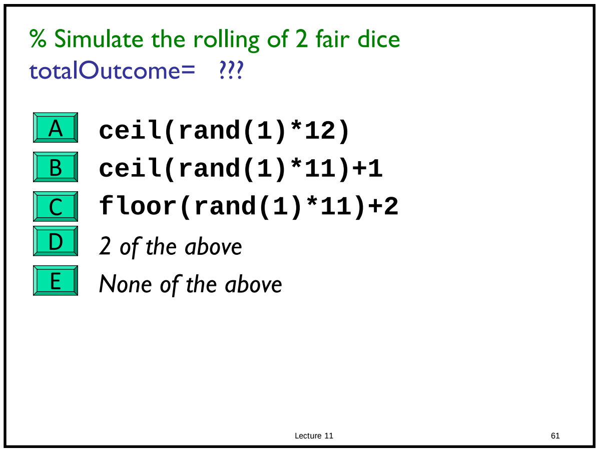
```
Repeat the following:

    % roll the die

    % increment correct “bin”
```

Lecture 11 53



```
function count = rollDie(rolls)

FACES= 6;               % #faces on die
count= zeros(1,FACES);  % bins to store counts

% Count outcomes of rolling a FAIR die
for k= 1:rolls
    % Roll the die

    % Increment the appropriate bin

end

% Show histogram of outcome
```

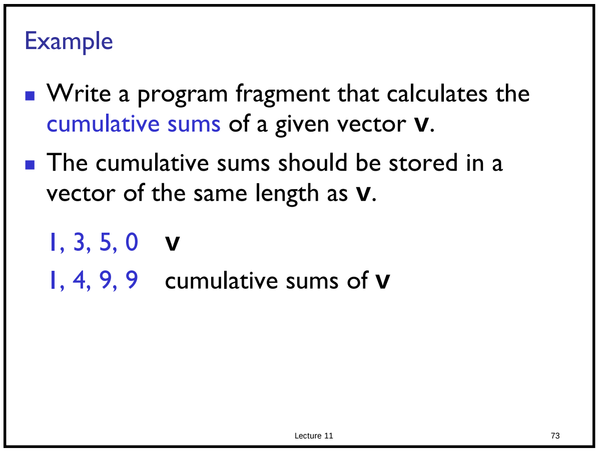
Lecture 11 54



% 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*

Lecture 11 61



Example

- Write a program fragment that calculates the cumulative sums of a given vector **v**.
- The cumulative sums should be stored in a vector of the same length as **v**.

1, 3, 5, 0	**v**
1, 4, 9, 9	cumulative sums of **v**

Lecture 11 73