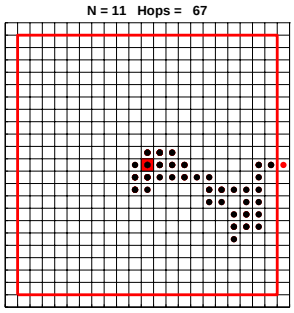
- Previous Lecture:
  - User-defined functions
    - Examples with varying numbers of input and output parameters
    - Local memory space
- Today's Lecture:
  - 1-d array—vector
  - Probability and random numbers
- Announcement:
  - Project 3 Part A posted on Tuesday; Part B to be posted. Both parts are due on Thursday 10/7.

2-dimensional random walk

Start in the middle tile, (0,0).

For each step, randomly choose between N,E,S,W and then walk one tile. Each tile is 1×1.

Walk until you reach the boundary.

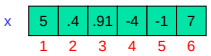



```
function [x, y] = RandomWalk2D(N)
% 2D random walk in 2N-1 by 2N-1 grid.
% Walk randomly from (0,0) to an edge.
% Vectors x,y represent the path.
```



```
function [x, y] = RandomWalk2D(N)

k=0;  xc=0;  yc=0;

while   not at an edge
   % Choose random dir, update xc,yc



   % Record new location in x, y

end
```


Array index starts at 1



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)


```
% Standing at (xc,yc)
% Randomly select a step
    r= rand(1);
    if r < .25
        yc= yc + 1;  % north
    elseif r < .5
        xc= xc + 1;  % east
    elseif r < .75
        yc= yc -1;   % south
    else
        xc= xc -1;   % west
    end
```

RandomWalk2D.m

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

Accessing values in a vector

score	93	99	87	80	85	82
	1	2	3	4	5	6

Given the vector **score** …

```
score(4)= 80;
score(5)= (score(4)+score(5))/2;
k= 1;
score(k+1)= 99;
```


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

Another representation for the random step

- Observe that each update has the form

  $xc = xc + \Delta x$

  $yc = yc + \Delta y$

  no matter which direction is taken.
- So let's get rid of the if statement!
- Need to create two "change vectors" deltaX and deltaY

deltaX

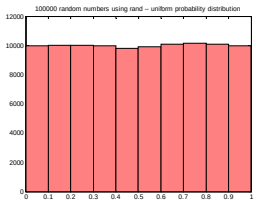
deltaY

RandomWalk2D_v2.m

Random numbers

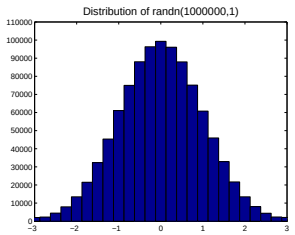
- *Pseudorandom* numbers in programming
- Function **`rand(…)`** generates random real numbers in the interval (0,1). All numbers in the interval (0,1) are equally likely to occur—uniform probability distribution.
- Examples:

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



Uniform probability distribution in (0,1)
`rand`

Normal distribution with zero mean and unit standard deviation
`randn`

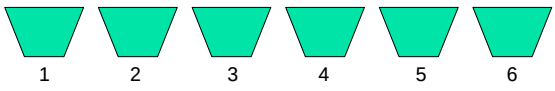


Simulate a fair 6-sided die

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*

Possible outcomes from rolling a fair 6-sided die



Algorithm

Repeat the following:

```
% roll the die

% increment correct "bin"
```



```
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
```