<WICC>

# CIS Partner Finding Social

**Tuesday, February 4**
**Gates 310 and 3rd Floor Lounge**

5-6 pm for 1000-2000 level classes
6-7pm for 3000+ level classes

- Previous Lecture:
  - Nesting `if`-statements
  - Logical operators, short-circuiting
  - Top-down design
- Today's Lecture:
  - Iteration using `for`
  - (at home) Watch MatTV episode "Troubleshooting for-loops"
- Announcements:
  - Discussion this week in the classrooms as listed in Student Center (Hollister 401)
  - Project 1 due tonight at 11pm; late submission accepted until tomorrow 11pm with 10% penalty
  - Read *Insight* §2.2 (or MatTV episode on `while`-loop) and *Insight* §3.2 before next lecture
  - Partner-finding social tonight at 5pm, Gates 3rd floor lounge

# Question

A 1 meter-long stick is split into two pieces. The breakpoint is randomly selected. On average, how long is the shorter piece?

Thought experiment? → analysis

Physical experiment?

Computational experiment! → simulation

} Need to <u>repeat</u> many trials!

# Question

A 1 meter-long stick is split into two pieces. The breakpoint is randomly selected (equally likely anywhere along the stick). On average, how long is the *shorter* piece?

A: ¼ m

B: 1/3 m

C: ½ m

D: other

Simulation:

use code to imitate the physical experiment

```
% one trial of the experiment
breakPt= rand();
if breakPt < 0.5
    shortPiece= breakPt;
else
    shortPiece= 1 - breakPt;
end
```

# More shortcuts: `min()`

```
% one trial of the experiment
breakPt= rand();
shortPiece= min(breakPt, 1-breakPt);
```

Want to do many trials, add up the lengths of the short pieces, and then divide by the number of trials to get the average length.

## Algorithm (bottom-up development)

*Repeat many times:*

```
% one trial of the experiment
breakPt= rand();
shortPiece= min(breakPt, 1-breakPt);
```

*Take average*

*Print result*

```
n= 10000;    % number of trials
total= 0;    % accumulated length so far

for k = 1:1:n   % Repeat many times
   % one trial of the experiment
   breakPt= rand();
   shortPiece= min(breakPt, 1-breakPt);
   total= total + shortPiece;
end

avgLength= total/n;   % Take average
fprintf('Average length is %f\n', ...
        avgLength)   % Print result
```

See `stickExp.m` , `showForLoop.m`

# Syntax of the `for` loop

```
for <var>= <start value>:<incr>:<end bound>

        statements to be executed repeatedly

end
```

*Loop body*

Loop header specifies all the values that the index variable will take on, one for each pass of the loop.

E.g, `k= 3:1:7` means `k` will take on the values 3, 4, 5, 6, 7, one at a time.

# `for` loop examples

```
for k = 2:0.5:3
      disp(k)
end
```

**k** takes on the values 2, 2.5, 3
Non-integer increment is OK

```
for k = 1:4
      disp(k)
end
```

**k** takes on the values 1, 2, 3, 4
Default increment is 1

```
for k = 0:-2:-6
      disp(k)
end
```

**k** takes on the values 0, -2, -4, -6
"Increment" may be negative

```
for k = 0:-2:-7
      disp(k)
end
```

**k** takes on the values 0, -2, -4, -6
Colon expression specifies *bounds*

```
for k = 5:2:1
      disp(k)
end
```

The set of values for k is the empty set: the loop body won't execute

# Pattern for doing something *n* times

```
n= ______
for k= 1:n

    % code to do
    % that something

end
```

Definite iteration

# Accumulation Pattern

Accumulator variable

```
% Average 10 numbers from user input

n= 10;       % number of data values
total= 0;    % current sum (initialized to zero)
for k = 1:n
  % read and process input value
    num= input('Enter a number: ');
    total= total + num;
end
avg= total/n;  % average of n numbers
fprintf('Average is %f\n', avg)
```

# Example: “Accumulate” a solution

```
% Average 10 numbers from user input
clear        % clear workspace
n= 10;       % number of data values

for k = 1:n
  % read and process input value
    num= input('Enter a number: ');
    total= total + num;
end
ave= total/n;  % average of n numbers
fprintf('Average is %f\n', ave)
```

How many passes through the loop will be completed?

A: 0

B: 1

C: 9

D: 10

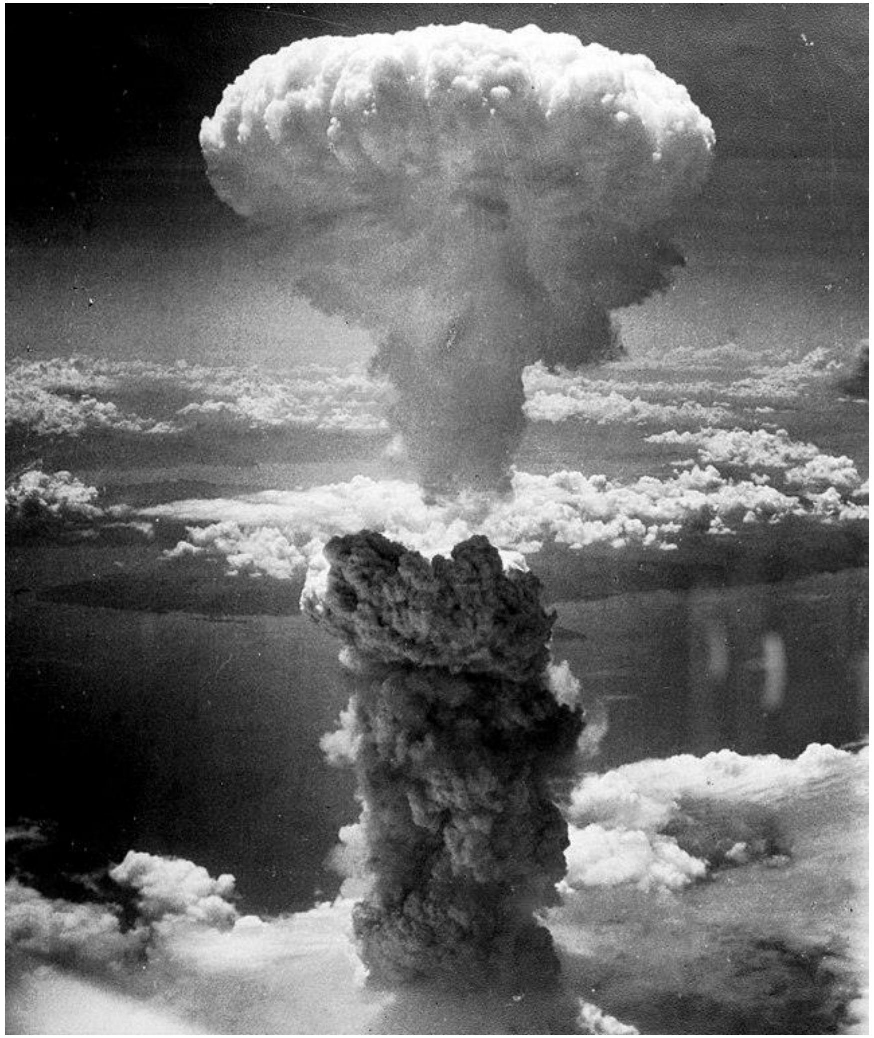
E: 11

# Remember to initialize

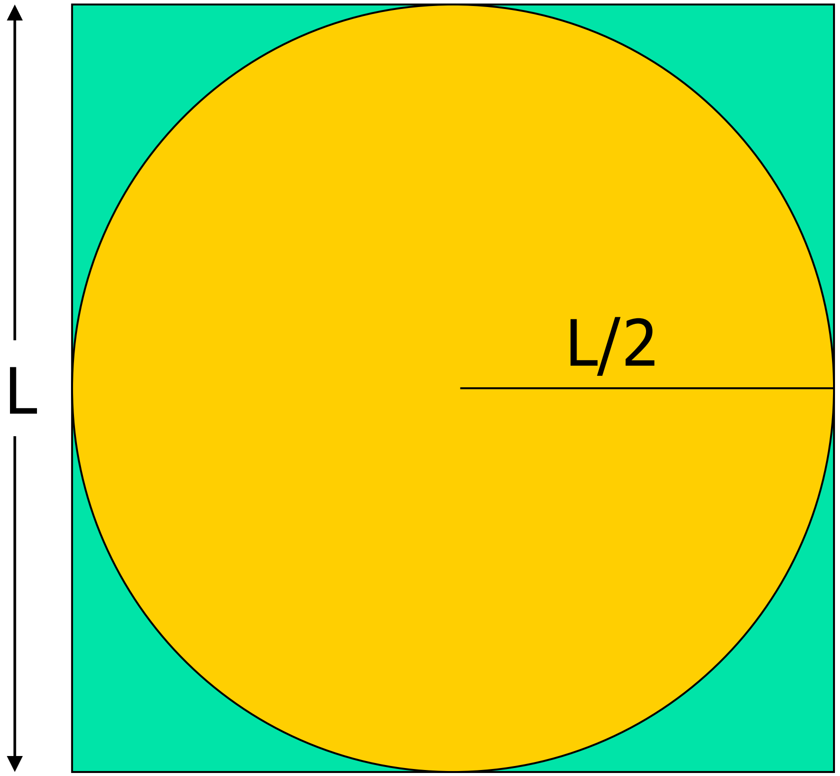
```
% Average 10 numbers from user input

n= 10;      % number of data values
total= 0;   % current sum (initialized to zero)
for k = 1:n
  % read and process input value
    num= input('Enter a number: ');
    total= total + num;
end
ave= total/n;  % average of n numbers
fprintf('Average is %f\n', ave)
```

# Monte Carlo methods

1. Derive a relationship between some *desired quantity* and a *probability*
2. Use simulation to estimate the probability
   - Computer-generated random numbers
3. Approximate desired quantity based on prob. estimate



# Monte Carlo Approximation of $\pi$




Throw $N$ darts

Sq. area = $\boldsymbol{L \times L}$

Circle area = $\boldsymbol{\pi L^2/4}$

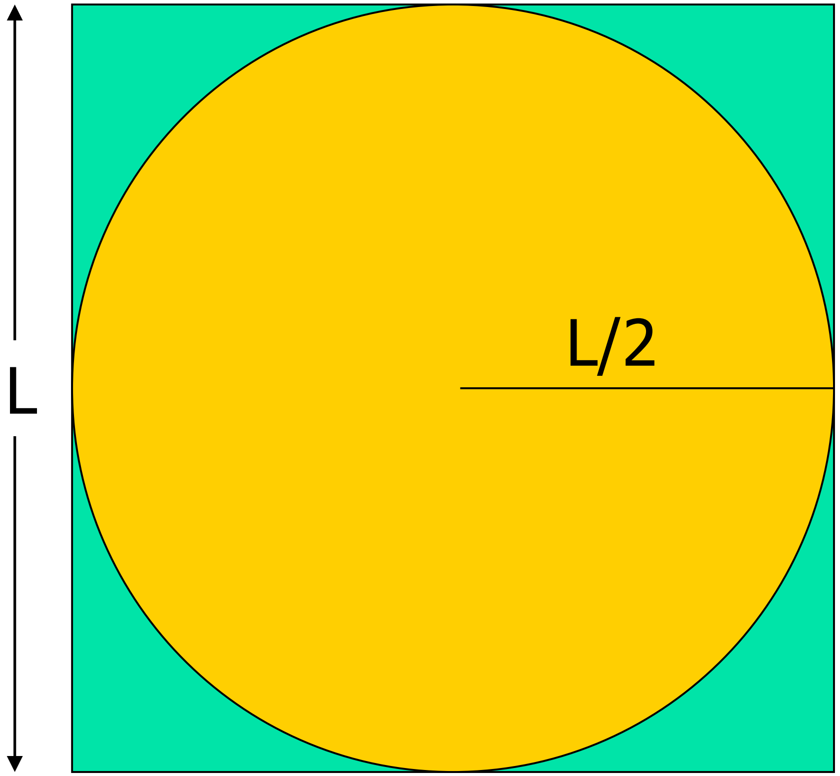
Prob. landing in circle

= (circle area)/(sq. area)

= $\boldsymbol{\pi/4}$

$\cong \boldsymbol{N_{in}/N}$

# Monte Carlo Approximation of $\pi$




Throw $N$ darts

$\pi \cong \mathbf{4}\ \boldsymbol{N_{in}} / \boldsymbol{N}$

# Monte Carlo Approximation of $\pi$

For each of N trials

Throw a dart

If it lands in circle

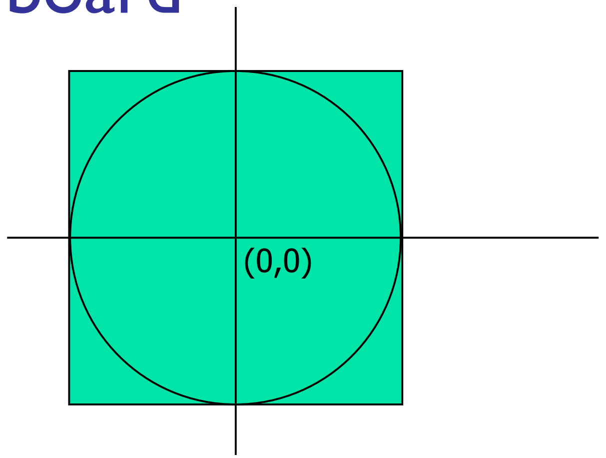
add 1 to total # of hits

Pi is 4*hits/N

# Monte Carlo $\pi$ with N darts on L-by-L board

```
N=__;
for k = 1:N







end
myPi= 4*hits/N;
```

# Monte Carlo $\pi$ with N darts on L-by-L board




```
N=__; L=__; hits= 0;
for k = 1:N
    % Throw kth dart
    x= rand()*L - L/2;
    y= rand()*L - L/2;
    % Count it if it is in the circle
    if sqrt(x^2 + y^2) <= L/2
        hits= hits + 1;
    end
end
myPi= 4*hits/N;
```