

■ Previous Lecture:

- Nesting **if**-statements
- Logical operators short-circuit
- Top-down design

■ Today's Lecture:

- Iteration using **for**
- Watch MatTV episode “Troubleshooting for-loops”

■ Announcements:

- Discussion this week in the classrooms as listed in Student Center
- CS1112 “**Partner Search Mixer**” on Thursday, 9/8, 5:30-6:30pm, Gates Hall atrium in front of G01, sponsored by WICC, ACSU, URM, and the CS Dept
- Last call to register your clickers—use the link on the course website

Question

A stick of unit length is split into two pieces. The breakpoint is randomly selected. On average, how long is the shorter piece?

Physical experiment? ♦

Thought experiment? → analysis

Computational experiment! → simulation♦

♦ Need to repeat many trials!

Question

A stick of unit length is split into two pieces. The breakpoint is randomly selected. On average, how long is the shorter piece?

A: .000001

B: .25

C: .333333

D: .499999

E: none of the above

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
```

```
% 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.

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

aveLength= total/n;
fprintf('Average length is %f\n', ...
        aveLength)
```

Example: “Accumulate” a solution

```
% Average 10 numbers from user input
```

```
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)
```

Important Features of Iteration

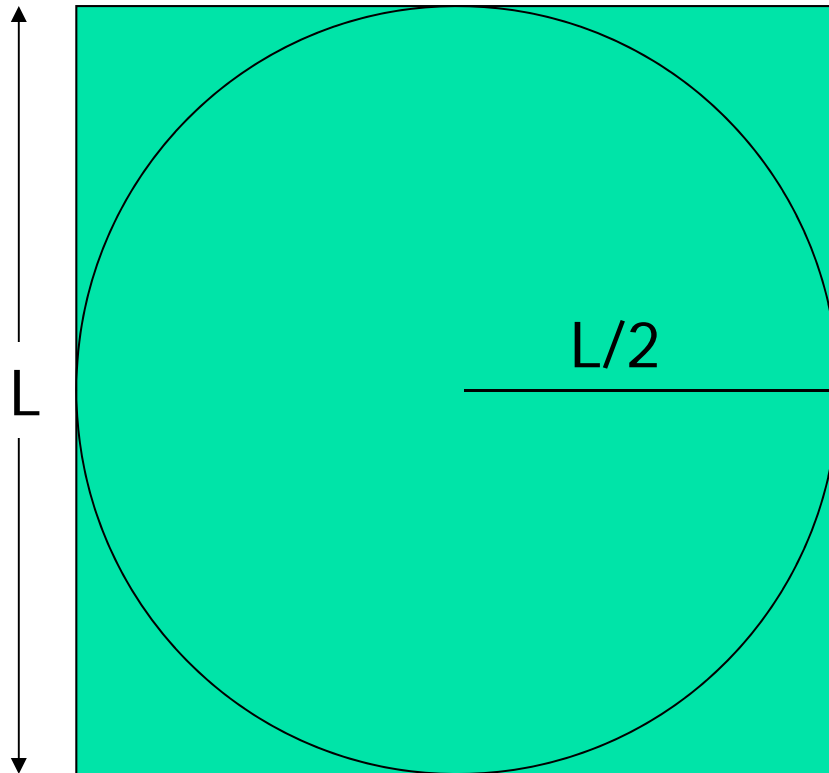
- A task can be accomplished if some steps are repeated; these steps form the **loop body**
- Need a **starting point**
- Need to know **when to stop**
- Need to keep track of (and measure) progress—**update**

Example: “Accumulate” a solution

```
% Average 10 numbers from user input

n= 10;      % number of data values
total= 0;   % current sum (initialized to zero)
for k= 1:1: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 Approximation of π

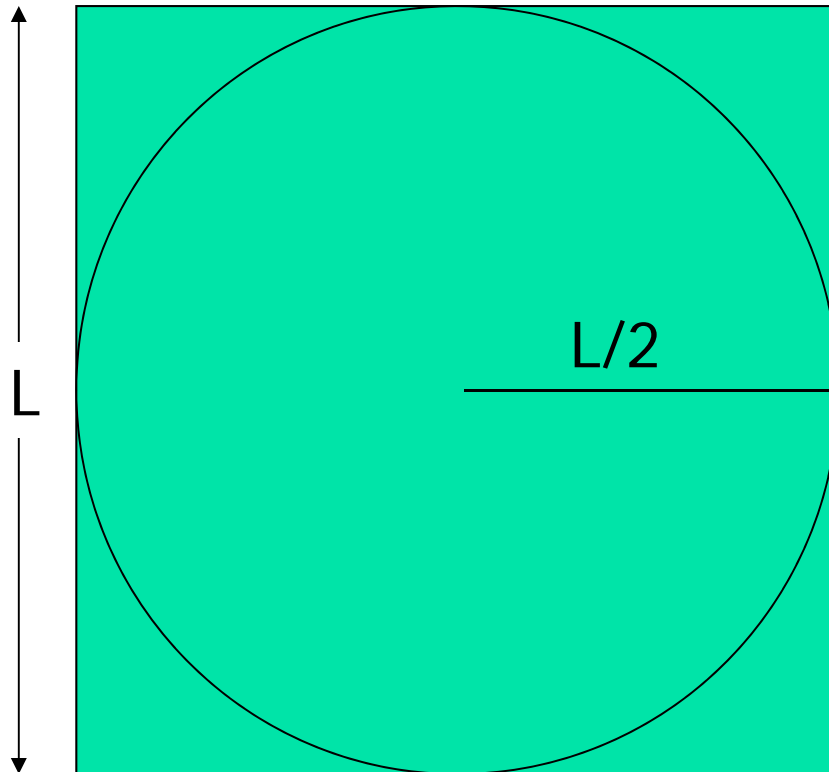


Throw N darts

$$\text{Sq. area} = N = L \times L$$

$$\begin{aligned}\text{Circle area} &= N_{in} \\ &= \pi L^2 / 4\end{aligned}$$

Monte Carlo Approximation of π



Throw N darts

Sq. area = $N = L \times L$

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

$$\pi = 4 N_{in} / N$$

Monte Carlo Approximation of π

For each of N trials

Throw a dart

If it lands in circle

add 1 to total # of hits

π is $4 \cdot \text{hits} / N$

Monte Carlo π with N darts on L-by-L board

```
N=___;
```

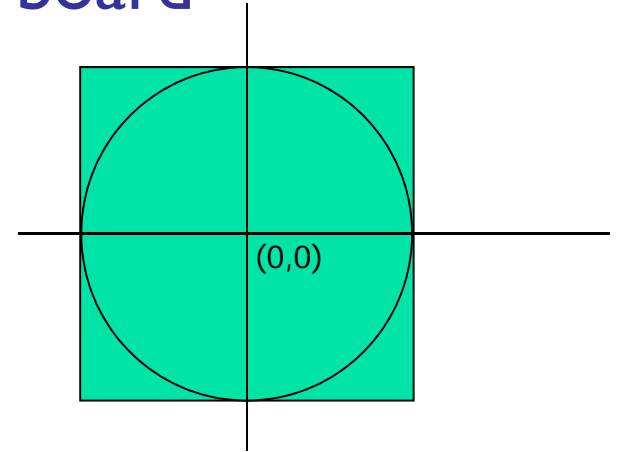
```
for k = 1:N
```

```
end
```

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

Monte Carlo π with N darts on L-by-L board

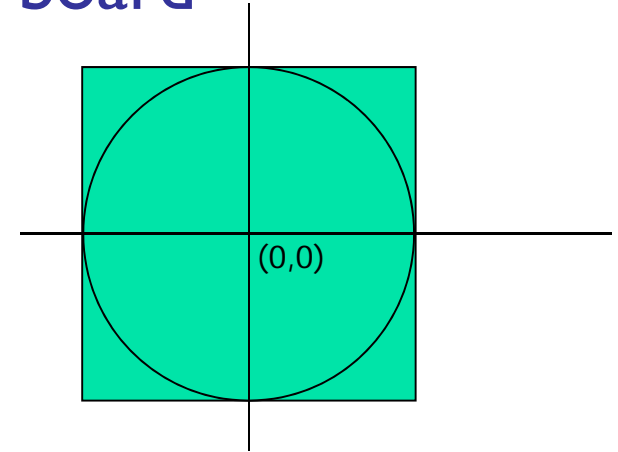
```
N=___;  
for k = 1:N  
    % Throw kth dart
```



```
    % Count it if it is in the circle
```

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

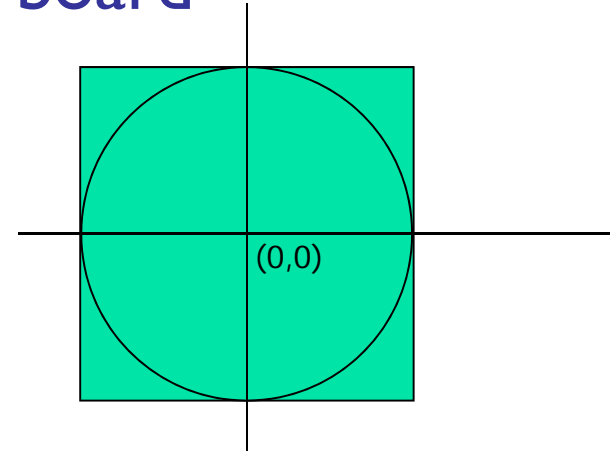

Monte Carlo π with N darts on L-by-L board



```
N=___;  
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;
```

Monte Carlo π 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;
```



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**.

Pattern for doing something n times

```
n= _____  
for k= 1:n  
  
    % code to do  
    % that something  
  
end
```

Definite iteration

for loop examples

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

k takes on the values _____
Non-integer increment is OK

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

k takes on the values _____
Default increment is 1

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

k takes on the values _____
“Increment” may be negative

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

k takes on the values _____
Colon expression specifies *bounds*

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

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