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:
- Project 1 due tonight at 11pm; late submission accepted until Friday 1pm sharp with 10% penalty
- Due to Feb Break, attendance at discussion next week is optional but you are responsible for the contents of the exercise to be posted. Attend any Wednesday sections (10:10-4:25) if you like. See Syllabus for locations.

### Top-Down Design

1. State problem
2. Define inputs & outputs
3. Design algorithm
4. Convert algorithm to program

An algorithm is an idea. To use an algorithm you must choose a programming language and implement the algorithm.

### 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!

### Simulation:

use code to imitate the physical experiment

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

```matlab
% 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.

```matlab
Repeat n times

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
', 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
', ave)

How many passes through the loop will be completed?
A: 0
B: 1
C: 9
D: 10
E: 11

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
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
', ave)

Monte Carlo Approximation of \( \pi \)

For each of \( N \) trials
Throw a dart
If it lands in circle
add 1 to total # of hits
\[ \pi = 4 \frac{N_{\text{in}}}{N} \]
Monte Carlo $\pi$ with $N$ darts on $L$-by-$L$ board

$N = _{\text{__}}$;

\begin{verbatim}
for k = 1:N
    \% Throw kth dart
    \% Count it if it is in the circle
\end{verbatim}

\texttt{myPi} = $4 \times \text{hits}/N$;

Syntax of the \texttt{for} loop

\begin{verbatim}
for <var> = <start value>:<incr>:<end bound>
    \text{statements to be executed repeatedly}
end
\end{verbatim}

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

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

Pattern for doing something $n$ times

\begin{verbatim}
n = _____
for k = 1:n
    \% code to do
    \% that something
end
\end{verbatim}

\textbf{Definite iteration}

\texttt{for} loop examples

\begin{verbatim}
for k = 2:0.5:3
    \texttt{disp}(k)
end
\end{verbatim}

$k$ takes on the values \underline{__________}

Non-integer increment is OK

\begin{verbatim}
for k = 1:4
    \texttt{disp}(k)
end
\end{verbatim}

$k$ takes on the values \underline{__________}

Default increment is 1

\begin{verbatim}
for k = 0:-2:-6
    \texttt{disp}(k)
end
\end{verbatim}

\,$"\text{Increment}'$\, may be negative

\begin{verbatim}
for k = 0:-2:-7
    \texttt{disp}(k)
end
\end{verbatim}

$k$ takes on the values \underline{__________}

Colon expression specifies \underline{bounds}

\begin{verbatim}
for k = 5:2:1
    \texttt{disp}(k)
end
\end{verbatim}

\texttt{disp}(k)

\textbf{Colon expression specifies bounds}

% What will be printed?
for k = 1:2:6
    \texttt{fprintf}\%d ', k)
end

A: 123456
B: 1356
C: 135
D: error
(incorrect bounds)

% What will be printed?
for k = 10:-1:14
    \texttt{fprintf}\%d ', k)
\end
\texttt{fprintf}('!')

A: error
(incorrect bounds)
B: 10 (then error)
C: 10 !
D: 14 !
E: !