Previous Lecture:
- Iteration using `for`

Today’s Lecture:
- Iteration using `while`
- Review loops, conditionals using graphics

Announcements:
- Did you watch MatTV episode “Troubleshooting Loops”? Available on course website
- Project 2 due Monday, 2/29
- We do not use `break` in this course
- Read Insight Section 3.2 before your discussion section next week

Syntax of the `for` loop

```
for <var> = <start value>:<incr>:<end bound>
    statements to be executed repeatedly
end
```

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

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

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

```
Example: n-gon $\rightarrow$ circle

- Inscribed hexagon
  \[
  \frac{n}{2} \sin(\frac{2\pi}{n})
  \]
- Circumscribed hexagon
  \[
  n \tan(\frac{\pi}{n})
  \]

As `n` approaches infinity, the inscribed and circumscribed areas approach the area of a circle.
When will `|OuterA - InnerA| <= 0.00001`?
Find \( n \) such that \( \text{outer}A \) and \( \text{inner}A \) converge

First, itemize the tasks:
- define how close is close enough
- select an initial \( n \)
- calculate \( \text{inner}A \), \( \text{outer}A \) for current \( n \)
- \( \text{diff} = \text{outer}A - \text{inner}A \)
- close enough?
- if not, increase \( n \), repeat above tasks

Find \( n \) such that \( \text{outer}A \) and \( \text{inner}A \) converge

Now organize the tasks → algorithm:

\( n \) gets initial value
\( \text{inner}A \), \( \text{outer}A \) get initial values

Repeat until difference is small:
- increase \( n \)
- calculate \( \text{inner}A \), \( \text{outer}A \) for current \( n \)
- \( \text{diff} = \text{outer}A - \text{inner}A \)

Guard against infinite loop

Use a loop guard that guarantees termination of the loop. Or just limit the number of iterations.

\[
\text{while} \ (B_{n} - A_{n} > \delta \text{ && \ } n < n_{\text{Max}})
\]

Another use of the while-loop: user interaction

- Example: Allow a user to repeatedly calculate the inscribed and circumscribed areas of \( n \)-gons on a unit circle.
- Need to define a “stopping signal”

Common loop patterns

Do something \( n \) times

\[
\text{for } k = 1:1:n \\%
\text{Do something}
\]

Do something an indefinite number of times

\[
\text{while} \ (\text{not stopping signal}) \\%
\text{Do something}
\text{Update loop variables}
\]

end
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

In Matlab, which claim is true? (without break)

A: for-loop can do anything while-loop can do
B: while-loop can do anything for-loop can do
C: for- and while-loops can do the same things

for-loop or while-loop: that is the question

- for-loop: loop body repeats a fixed (predetermined) number of times.
- while-loop: loop body repeats an indefinite number of times under the control of the "loop guard."

Review loops/conditionals using user-defined graphics function

Draw a black square; then draw a magenta disk; then draw a yellow star.

x and y coordinates of lower left corner width height color

DrawRect(-1,-2,6,3,'y')

DrawDisk(1,3,4,'r')
**DrawStar(1,3,4,'g')**

- **x and y coordinates of the center**
- **“radius”**

**Color Options**

- White: \( 'w' \)
- Black: \( 'k' \)
- Red: \( 'r' \)
- Blue: \( 'b' \)
- Green: \( 'g' \)
- Yellow: \( 'y' \)
- Magenta: \( 'm' \)
- Cyan: \( 'c' \)

**A general graphics framework**

- Code fragment to draw the objects (rectangle, disk, star)

**Example: Nested Stars**