Previous Lecture:
- Iteration using for

Today's Lecture:
- Iteration using while
- Review loop & conditionals using graphics

Announcements:
- We do not use break in this course
- Register your iClicker this week! See registration link on course website.

---

**Syntax of the for loop**

```
for <var> = <start value>:<increment>:<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:n
  % code to do
  % that something
end
```

Definite iteration

---

% 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: ~

---

What will be displayed when you run the following script?

```
for k = 4:6
  disp(k)
k = 9;
  disp(k)
end
```

```
4
9
```

A or B or C

---

% What will be printed?
```
for k = 4:6
  disp(k)
k = 9;
  disp(k)
end
```

```
4
4
```

With this loop header, \( k \) "promises" to be these values, one at a time.

---

Output in Command Window
for k = 4:6
    disp(k)
k= 9;
disp(k)
end

not a condition (boolean expression)
that checks whether k==6.

it is an expression that specifies values:

4 5 6

for-loop header is executed only once!
(Loop body is may be executed multiple times)

Example: n-gon → circle

Inscribed hexagon
\((n/2) \sin(2\pi/n)\)
Circumscribed hexagon
\(n \tan(\pi/n)\)

As \(n\) approaches infinity, the inscribed and circumscribed areas
approach the area of a circle.
When will \(|\text{OuterA} - \text{InnerA}| \leq .000001\)?

Find \(n\) such that \(\text{outerA}\) and \(\text{innerA}\) converge

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

Find \(n\) such that \(\text{outerA}\) and \(\text{innerA}\) converge

Now organize the tasks \(\rightarrow\) algorithm:

\(n\) gets initial value

Repeat until difference is small:

increase \(n\)

calculate \(\text{innerA}\), \(\text{outerA}\) for current \(n\)

diff= \(\text{outerA} - \text{innerA}\)

Find \(n\) such that \(\text{outerA}\) and \(\text{innerA}\) converge

Now organize the tasks \(\rightarrow\) algorithm:

\(n\) gets initial value

\(\text{innerA}, \text{outerA}\) get initial values

Repeat until difference is small:

increase \(n\)

calculate \(\text{innerA}, \text{outerA}\) for current \(n\)

diff= \(\text{outerA} - \text{innerA}\)

Find \(n\) such that \(\text{outerA}\) and \(\text{innerA}\) converge

\(n\) gets initial value

calculate \(\text{innerA}, \text{outerA}\) for current \(n\)

\(\text{while}\) <difference is not small enough>

increase \(n\)

calculate \(\text{innerA}, \text{outerA}\) for current \(n\)

diff= \(\text{outerA} - \text{innerA}\)

end

Indefinite iteration
Guard against infinite loop

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

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

See Eg2_2.m

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

```
for k = 1:1:n
    % Do something
end
```

Do something an indefinite number of times

```
while (not stopping signal)
    % Do something
    % 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

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

```
DrawRect(-1,-2,6,3,'y')
DrawDisk(1,3,4,'r')
DrawStar(1,3,4,'g')
```

Color Options

- White: ‘w’
- Black: ‘k’
- Red: ‘r’
- Blue: ‘b’
- Green: ‘g’
- Yellow: ‘y’
- Magenta: ‘m’
- Cyan: ‘c’
A simple 3-line script

```matlab
% drawDemo
close all
figure
axis equal off
hold on
DrawRect(0,0,2,2,'k')
DrawDisk(1,1,1,'m')
DrawStar(1,1,1,'y')
hold off
```

A general graphics framework

```matlab
% drawDemo
close all
figure
axis equal off
hold on

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

hold off
```

Example: Nested Stars

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