

- Previous Lecture:
 - Iteration using `for`

- Today's Lecture:
 - Details on `for`-loop
 - Iteration using `while`
 - Review loop & conditionals using graphics

- Announcements:
 - P2 will be due Thurs Sept 18 at 11pm
 - We do not use `break` in this course

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

Lecture 5 5

Pattern for doing something *n* times

```

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

Definite iteration

Lecture 5 6

```

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

`k`

With this loop header, `k` "promises" to be these values, one at a time

Output in Command Window

Lecture 6 13

```

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

It is an expression that specifies values:

4

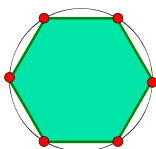
5

6

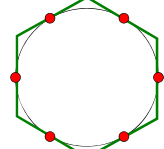
for-loop header is executed only once!
(Loop body is may be executed multiple times)

Lecture 6 30

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 $|OuterA - InnerA| \leq .000001$?

Lecture 6 31

Find n such that $outerA$ and $innerA$ converge

First, itemize the tasks:

- *define how close is close enough*
- *select an initial n*
- *calculate $innerA$, $outerA$ for current n*
- *$diff = outerA - innerA$*
- *close enough?*
- *if not, increase n , repeat above tasks*

Lecture 6

32

Find n such that $outerA$ and $innerA$ converge

Now organize the tasks \rightarrow algorithm:

n gets initial value

Repeat until *difference is small:*

increase n
calculate $innerA$, $outerA$ for current n
 $diff = outerA - innerA$

Lecture 6

33

Find n such that $outerA$ and $innerA$ converge

Now organize the tasks \rightarrow algorithm:

n gets initial value

$innerA$, $outerA$ get initial values

Repeat until *difference is small:*

increase n
calculate $innerA$, $outerA$ for current n
 $diff = outerA - innerA$

Lecture 6

34

Find n such that $outerA$ and $innerA$ converge

n gets initial value

calculate $innerA$, $outerA$ for current n

while *<difference is not small enough>*

increase n
calculate $innerA$, $outerA$ for current n
 $diff = outerA - innerA$

end

Indefinite iteration

areaCircle.m

Lecture 6

35

Guard against **infinite** loop

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

```
while (B_n - A_n > delta && n < nMax)
```

Eg2_2.m

Lecture 6

37

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”

areaIndef.m

Lecture 6

38

Common loop patterns

Do something *n* times

```

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

Do something an indefinite number of times

```

%Initialize loop variables
while ( not stopping signal )
    % Do something
    % Update loop variables
end
                    
```

Lecture 6 39

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

Lecture 6 41

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

Lecture 6 43

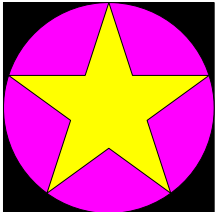
for-loop or while-loop: that is the question

- **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.”

Lecture 6 46

Review loops/conditionals using user-defined graphics function

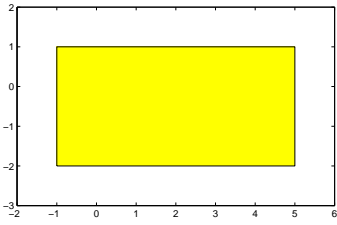
Draw a black square;
then draw a magenta disk;
then draw a yellow star.



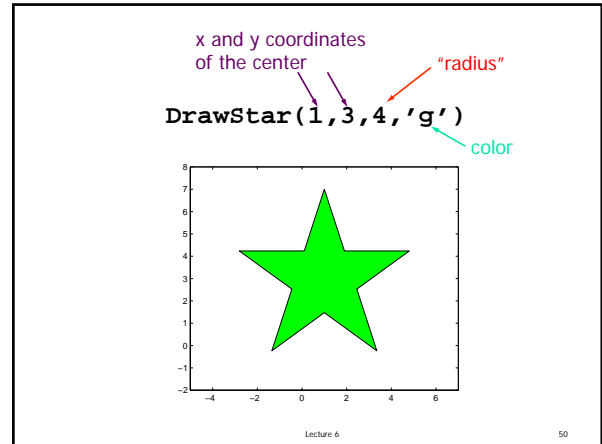
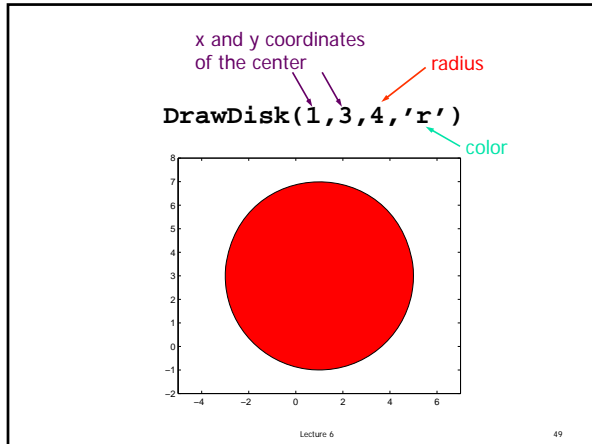
Lecture 6 47

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

x and y coordinates of lower left corner
 width
 height
 color



Lecture 6 48



Color Options

White	'w'	
Black	'k'	
Red	'r'	
Blue	'b'	
Green	'g'	
Yellow	'y'	
Magenta	'm'	
Cyan	'c'	

Lecture 6 51

```

% 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

```

% drawDemo
close all
figure
axis equal off
hold on

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

hold off
    
```

Lecture 6 54

Example: Nested Stars

