- Previous Lecture:
  - Iteration using **for**
- Today's Lecture:
  - Detail on **for**-loop
  - Iteration using **while**
  - Review loops, conditionals using graphics
- Announcements:
  - Discussion this week in classrooms, not lab
  - Project 2 posted, due Monday, 9/20
  - We do not use **break** in this course

```
% 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
```

A:
```
4
9
```

or

B:
```
4
4
```

or

C: *Something else …*


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

4 5 6

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

Output in Command Window

k 4

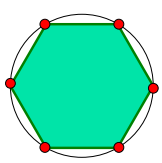

```
for k = 4:6
    disp(k)
    k= 9;
    disp(k)
end
```

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

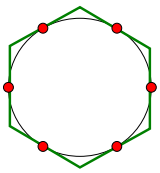
It is an expression that specifies values:

4 5 6

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. How big should *n* be?

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*

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

Now organize the tasks → algorithm:

n gets initial value
Repeat until difference is small:
    calculate innerA, outerA for current n
    diff= outerA – innerA
    increase n

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

n gets initial value
while *<difference is not small enough>*
    calculate innerA, outerA for current n
    diff= outerA – innerA
    increase n
end

Indefinite iteration

areaCircle.m

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)
```

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"

areaIndef.m

Common loop patterns

Do something *n* times

```
for k= 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
```


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

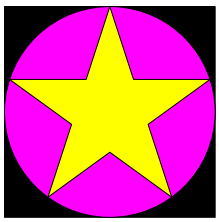
- for-loop: loop body repeats a *fixed* (predetermined) number of times.
- while-loop: loop body repeats an *indefinite* number of times under the control of the "loop guard."

Review loops/conditionals using user-defined graphics function

Draw a black square;

then draw a magenta disk;

then draw a yellow star.



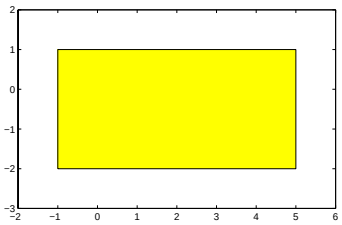
x and y coordinates of lower left corner

width

height

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

color

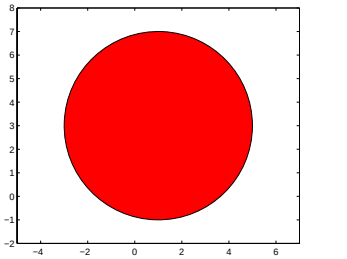


x and y coordinates of the center

radius

```
DrawDisk(1,3,4,'r')
```

color



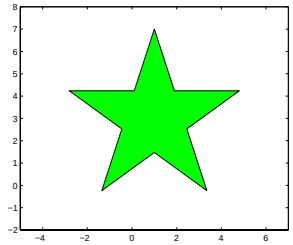
45

x and y coordinates of the center

"radius"

```
DrawStar(1,3,4,'g')
```

color



Lecture 6 46

Color Options

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

Lecture 6 47

```
% 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 50

Example: Nested Stars