- Previous Lecture:
  - Iteration using `for`
- Today's Lecture:
  - Iteration using `while`
  - Review loops, conditionals using graphics
- Announcements:
  - Discussion this week in classrooms as listed in Student Ctr., not in the lab
  - We do not use `break` in this course
  - Register your iClicker this week! See *Syllabus* link on course website.

`for` loop examples

```
for k= 2:0.5:3
    disp(k)
end
```
`k` takes on the values _____
Non-integer increment is OK

```
for k= 1:4
    disp(k)
end
```
`k` takes on the values _____
Default increment is 1

```
for k= 0:-2:-6
    disp(k)
end
```
`k` takes on the values _____
"Increment" may be negative

```
for k= 0:-2:-7
    disp(k)
end
```
`k` takes on the values _____
Colon expression specifies a *bound*

```
for k= 5:2:1
    disp(k)
end
```


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

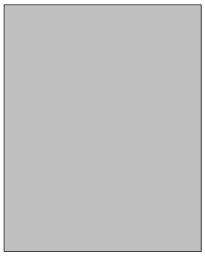
B:
```
4
4
```
or

C: *Something else …*


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

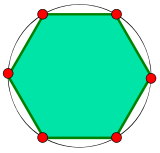
With this loop header, `k` "promises" to be these values, one at a time

Output in Command Window

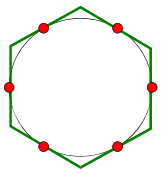


`k`

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| <= .000001?

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
innerA, outerA get initial values
Repeat until difference is small:
    increase n
    calculate innerA, outerA for current n
    diff= outerA - innerA

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

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

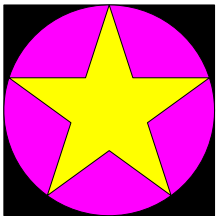
- for-loop: loop body repeats a *fixed* (predetermined) number of times.
- while-loop: loop body repeats an *indefinite* number of times under the control of the "loop guard."

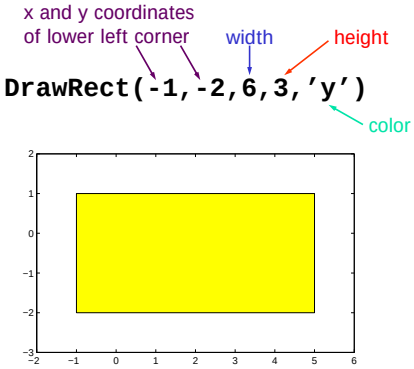
Review loops/conditionals using user-defined graphics function

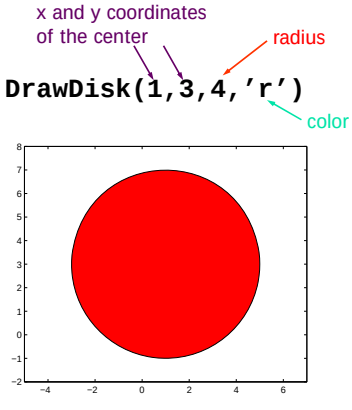
Draw a black square;

then draw a magenta disk;

then draw a yellow star.





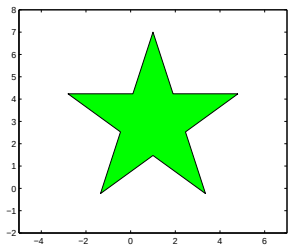


x and y coordinates of the center

"radius"

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

color



Color Options

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


```
% 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
```

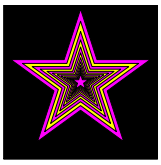
A general graphics framework

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

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

```
hold off
```


Example: Nested Stars



nestedStars.m