- Previous Lecture:
  - Iteration using `for`
- Today's Lecture:
  - Iteration using `while`
  - Calling given (not built-in) functions
- Announcements:
  - Watch MatTV episode "*Troubleshooting Loops.*" Available on course website
  - Project 2 due Thursday 9/15
  - We do not use `break` in this course
  - Read *Insight* Section 3.2 before your discussion section next week
  - Come to office/consulting hours to get help!

Pattern for doing something *n* times

```
n= _____
for k= 1:1:n

   % code to do
   % that something

end
```

Definite iteration


```
% What will be printed?
for k= 1:2:6
    fprintf('%d ', k)
end
```

A: 1 2 3 4 5 6

B: 1 3 5 6

C: 1 3 5

D: *error (incorrect bounds)*


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

A: *error (incorrect bounds)*

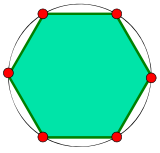
B: 10 *(then error)*

C: 10 !

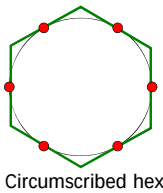
D: 14 !

E: !

Example: *n*-gon → 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

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

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)
```

`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= 1:1: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

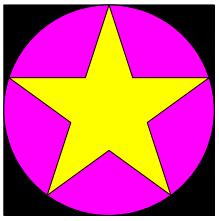
- for-loop: loop body repeats a *fixed* (predetermined) number of times.
- while-loop: loop body repeats an *indefinite* number of times under the control of the "loop guard."

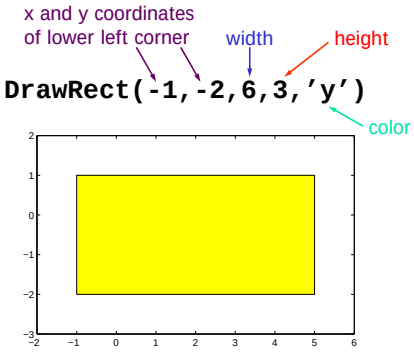
Review loops/conditionals using user-defined graphics function

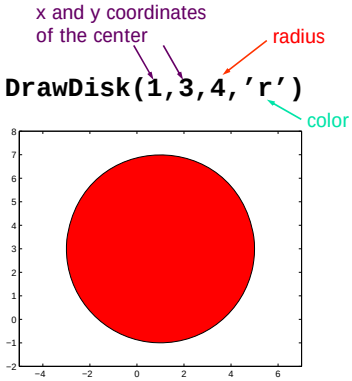
Draw a black square;

then draw a magenta disk;

then draw a yellow star.





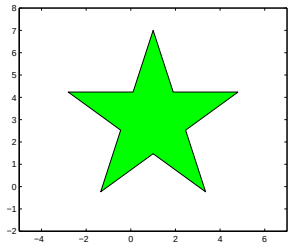


x and y coordinates of the center

"radius"

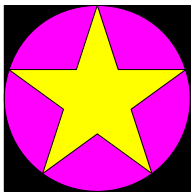
```
DrawStar(1,3,4,'g')
```

color



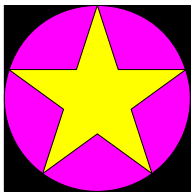
Color Options

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



```
DrawRect( , , , ,   )
DrawDisk( , , ,   )
DrawStar( , , ,   )
```

```
% 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