- Previous Lecture:
  - Iteration using `while`
- Today's Lecture:
  - Nested loops
  - Developing algorithms
- Announcements:
  - Discussion this week will be in Upson 225 computer lab
  - Project 1 grades released after lecture
  - Project 2 due Monday 2/17 at 11pm
    - Part B posted after lecture
  - Thanks for filling out survey!

"I had to learn how to study differently – by practicing every day rather than cramming before. I wish that we could have been told earlier in the year to practice like 30 minutes per day..." – FA19 student

Important Features of Iteration

- Task-to-be-repeated forms the loop body
- Need a starting point
- Need to know when to stop
- Need to keep track of (and measure) progress

for vs. while

```
N= ___;  L= ___;  hits= 0;

for k = 1:1:N
    % Throw kth dart
    x = rand()*L - L/2;
    y = rand()*L - L/2;
    % Count if in circle
    if sqrt(x^2+y^2) <= L/2
        hits = hits + 1;
    end

end
myPi = 4*hits/N;
```

```
N= ___;  L= ___;  hits= 0;
k= 1;
while k <= N
    % Throw kth dart
    x = rand()*L - L/2;
    y = rand()*L - L/2;
    % Count if in circle
    if sqrt(x^2+y^2) <= L/2
        hits = hits + 1;
    end
    k = k+1;
end
myPi = 4*hits/N;
```

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."

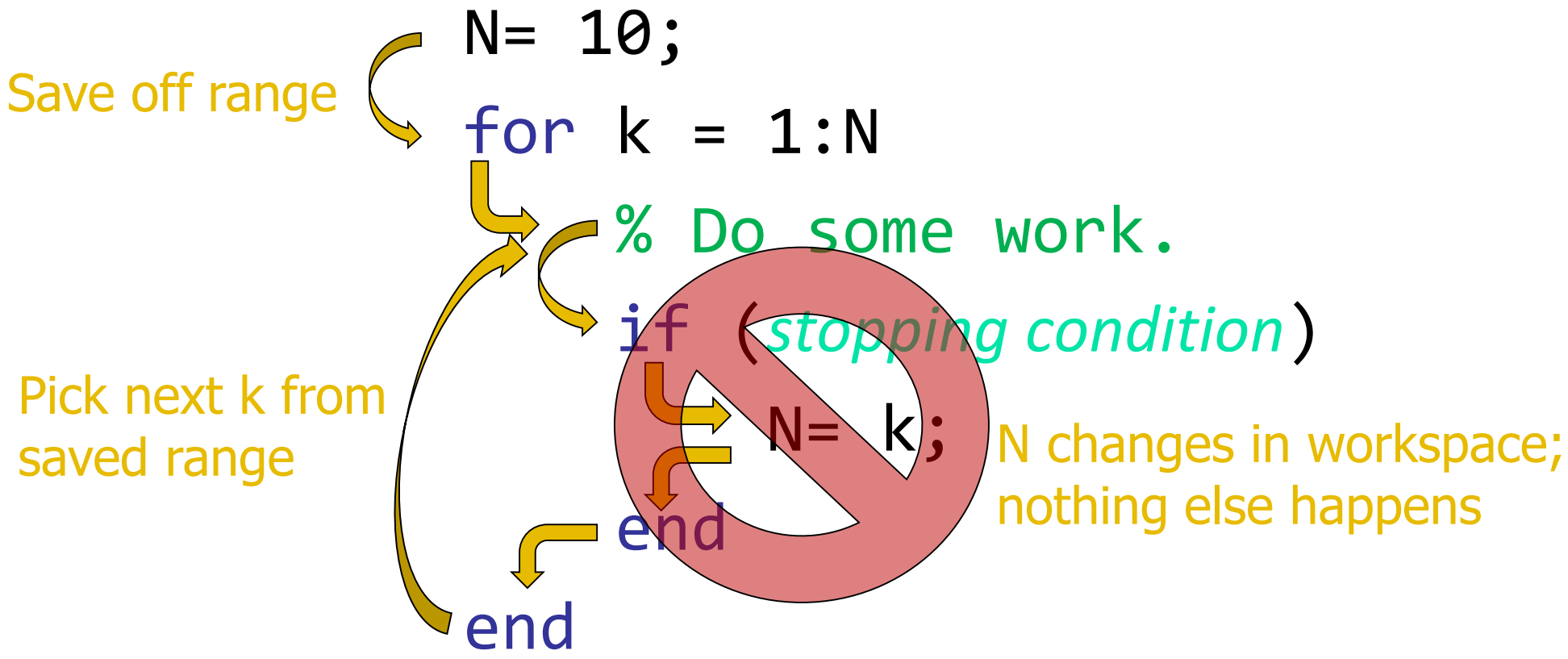
In Matlab, which claim is true? (without **`break`**)

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

Can we cheat?

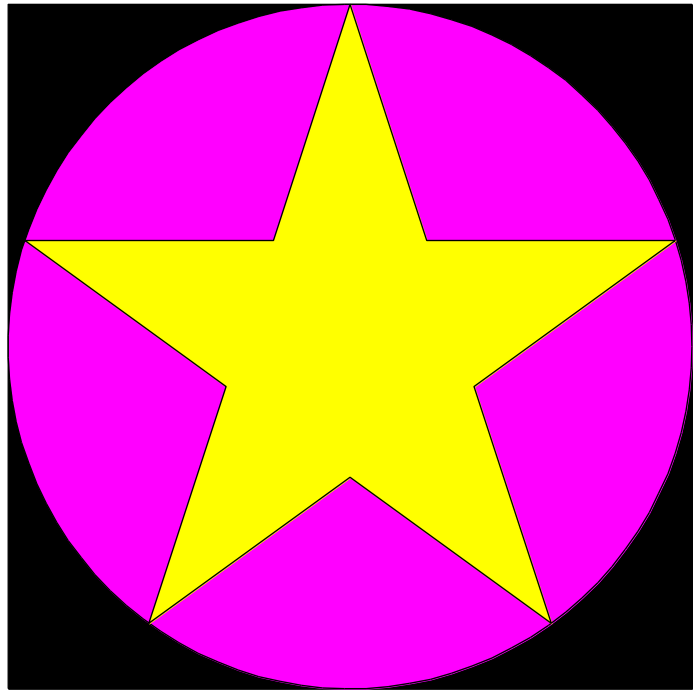

Save off range
N= 10;
for k = 1:N
% Do some work.
if (stopping condition)
N= k;
end
end
Pick next k from saved range
N changes in workspace; nothing else happens

Review loops/conditionals using user-defined graphics function

Draw a black square;

then draw a magenta disk;

then draw a yellow star.



Refinement tip: Survey your tools

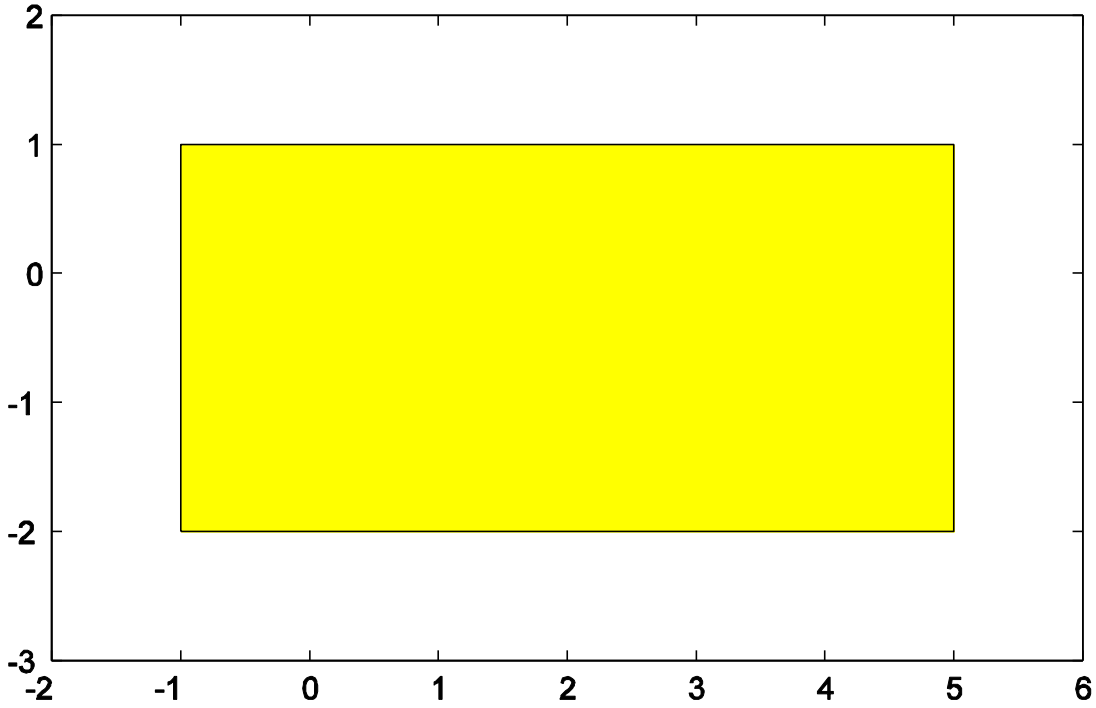
x and y coordinates of lower left corner

width

height

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

color

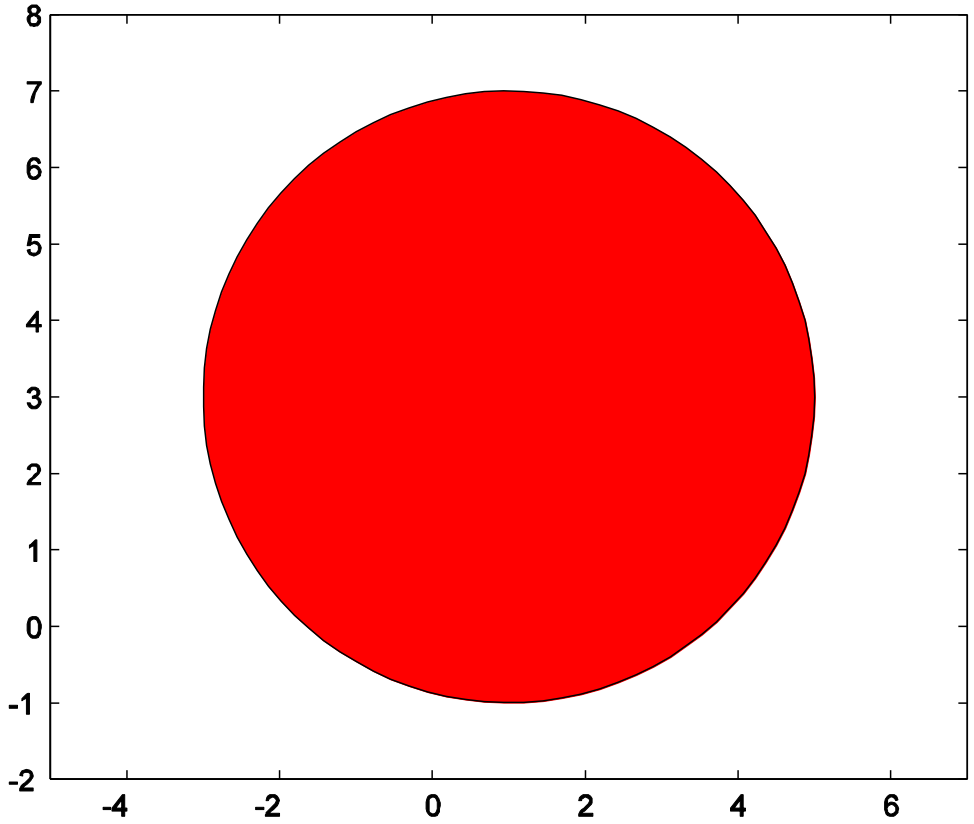


x and y coordinates
of the center

radius

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

color

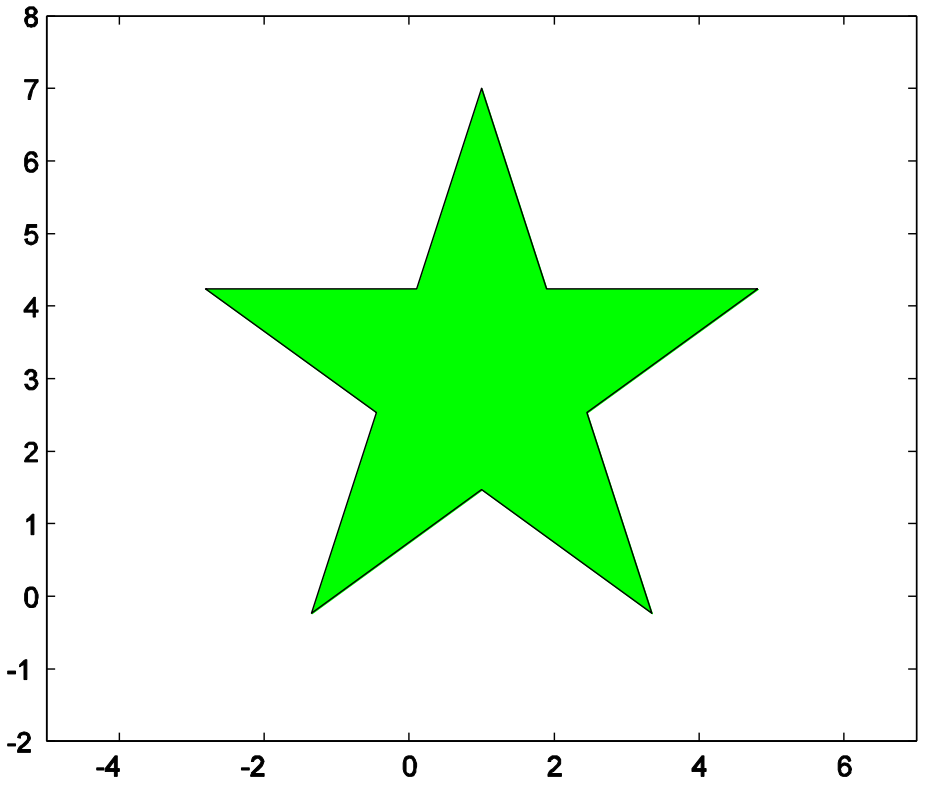


x and y coordinates
of the center

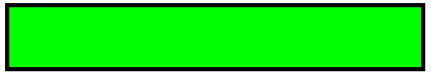
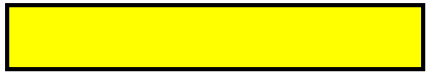
"radius"

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

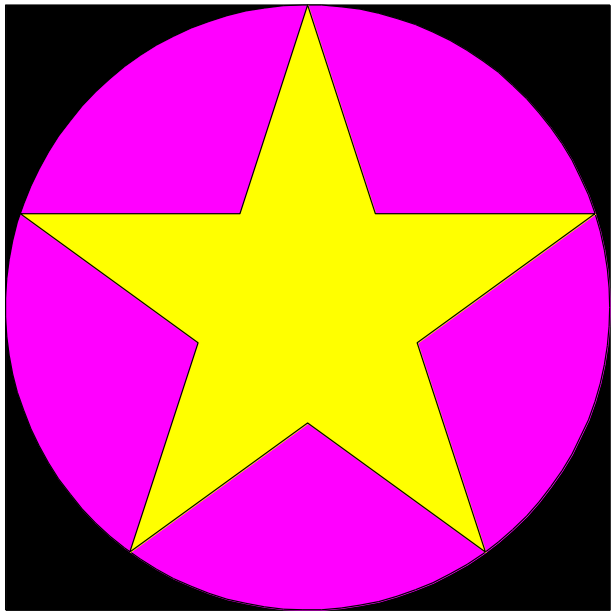
color



Color Options

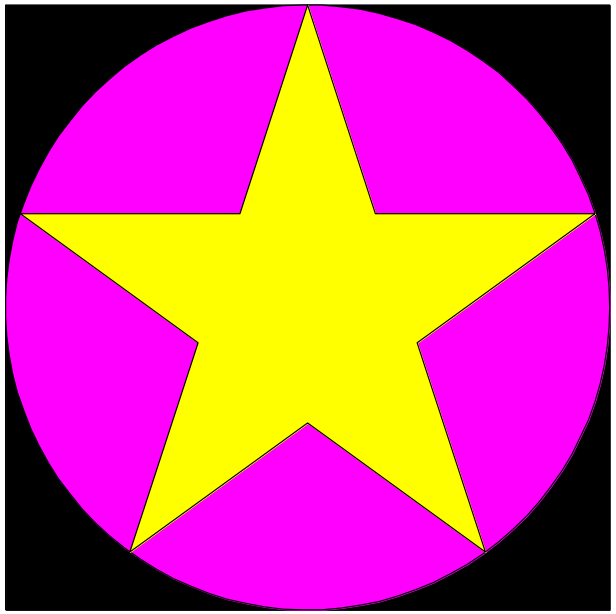
White	`'w'`	
Black	`'k'`	
Red	`'r'`	
Blue	`'b'`	
Green	`'g'`	
Yellow	`'y'`	
Magenta	`'m'`	
Cyan	`'c'`	

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



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

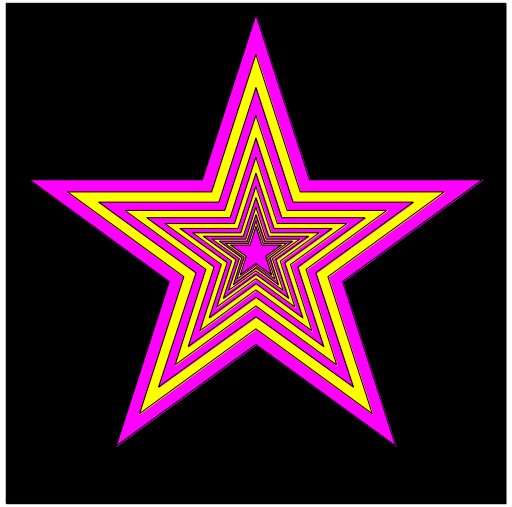
Draw a black square;
then draw a magenta disk;
then draw a yellow star.



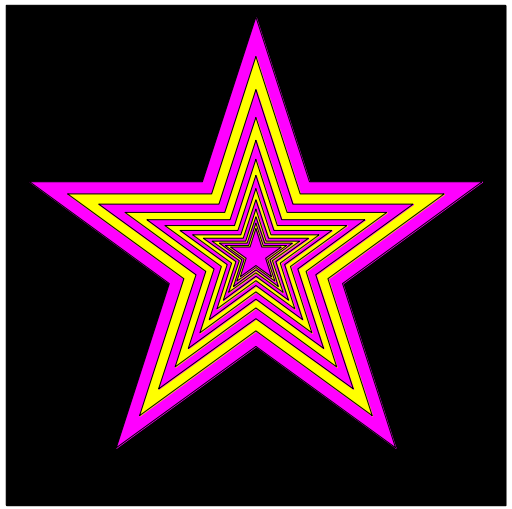
```
DrawRect(0,0,2,2,'k')
DrawDisk(1,1,1,'m')
DrawStar(1,1,1,'y')
```

```
% 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
```

Example: Nested Stars



Example: Nested Stars



Draw a black square
-Bigger than the biggest star
(at least 2 times radius of star)
- Center at (0,0)

Draw a sequence of stars
-Stars alternate in color
-Stars get smaller
    - radius r=1 to start
-1st star smaller than the sqr
-When to stop?
    - when r is small

`nestedStars.m`

```
x= 0; y= 0;    % figure centered at (0,0)
r= 1;          % radius of outermost star
s= 2*r + 0.1; % side length of square
DrawRect(x-s/2, y-s/2, s, s, 'k')

% Draw nested stars, smallest r at least 0.1
```

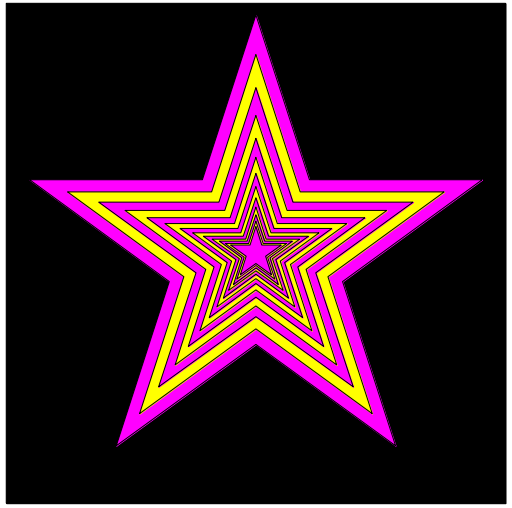
```
x= 0; y= 0;    % figure centered at (0,0)
r= 1;          % radius of outermost star
s= 2*r + 0.1; % side length of square
DrawRect(x-s/2, y-s/2, s, s, 'k')

% Draw nested stars, smallest r at least 0.1

while  r >= 0.1
   % Draw a star with radius r




   % Reduce r


end
```

```
x= 0; y= 0;    % figure centered at (0,0)
r= 1;          % radius of outermost star
s= 2*r + 0.1; % side length of square
DrawRect(x-s/2, y-s/2, s, s, 'k')

% Draw nested stars, smallest r at least 0.1

while  r >= 0.1
   % Draw a star with radius r




   % Reduce r
   r= r/1.2;

end
```

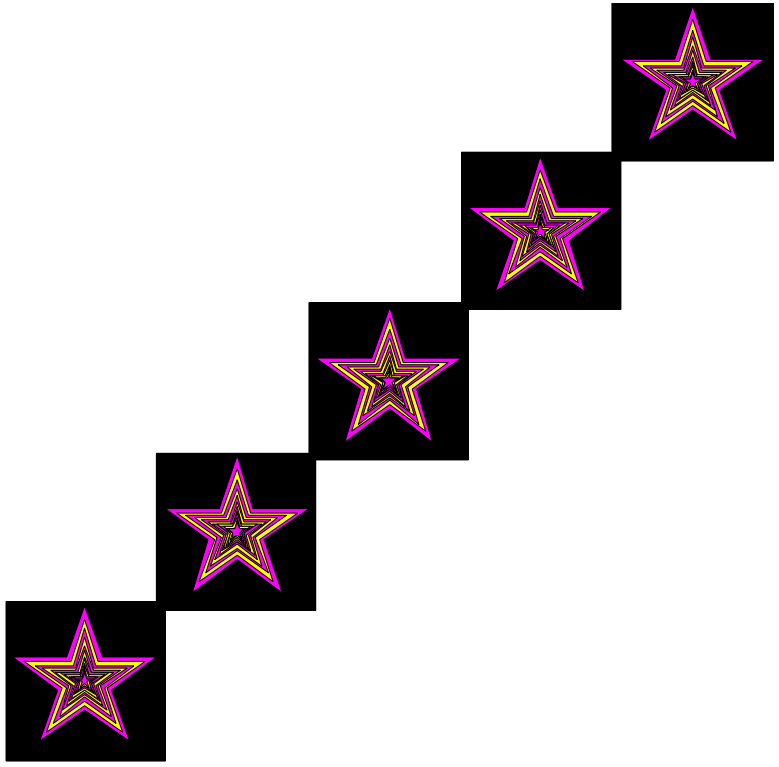
```
x= 0; y= 0;    % figure centered at (0,0)
r= 1;          % radius of outermost star
s= 2*r + 0.1; % side length of square
DrawRect(x-s/2, y-s/2, s, s, 'k')

% Draw nested stars, smallest r at least 0.1

while  r >= 0.1
    % Draw a star with radius r
    if
        % magenta
    else
        % yellow
    end
    % Reduce r
    r= r/1.2;

end
```

```
x= 0; y= 0;    % figure centered at (0,0)
r= 1;          % radius of outermost star
s= 2*r + 0.1; % side length of square
DrawRect(x-s/2, y-s/2, s, s, 'k')

% Draw nested stars, smallest r at least 0.1
k= 1;
while  r >= 0.1
    % Draw a star with radius r
    if rem(k,2)==1  % odd k
       DrawStar(x, y, r, 'm')  % magenta
    else
       DrawStar(x, y, r, 'y')  % yellow
    end
    % Reduce r
    r= r/1.2;
    k= k + 1;
end
```

Knowing how to draw, …



how difficult is it to draw… ?



Pattern for doing something *n* times

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

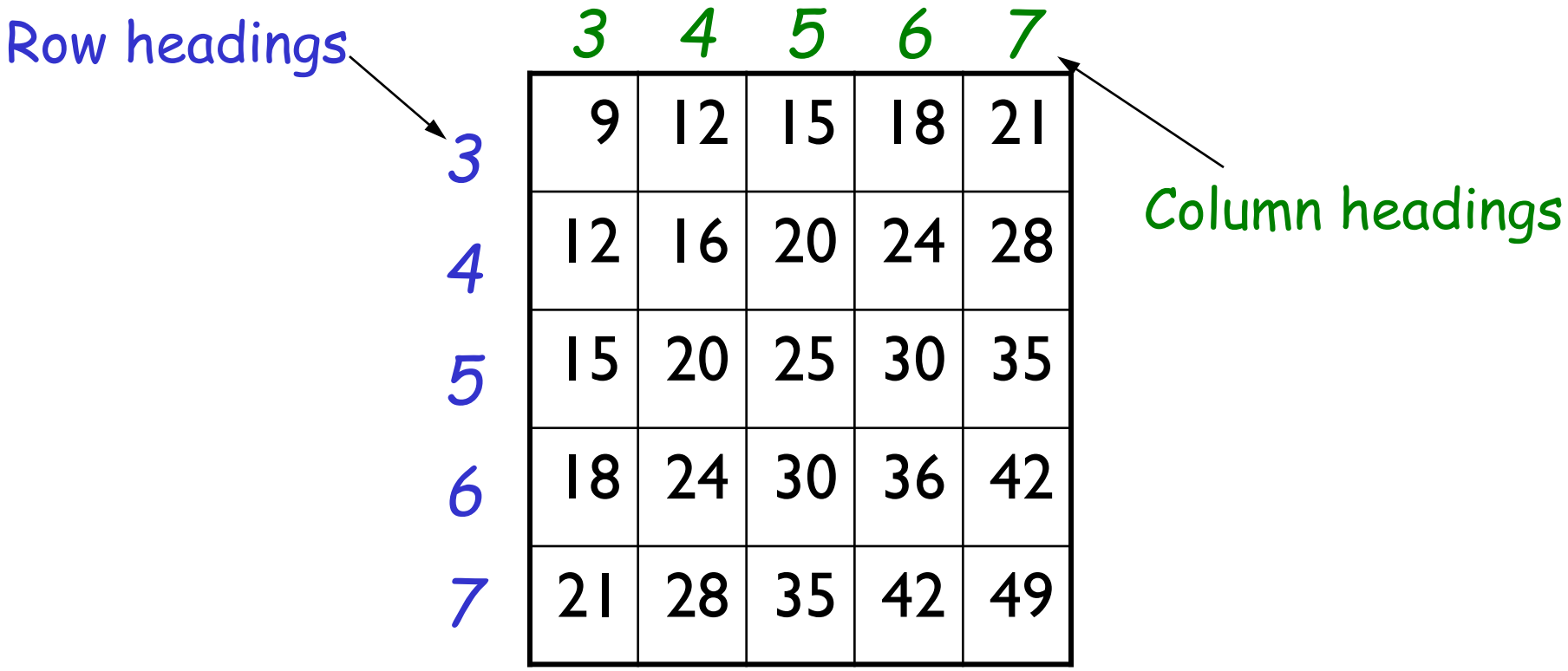
```
x= 0; y= 0;    % figure centered at (0,0)
r= 1;          % radius of outermost star
s= 2*r + 0.1; % side length of square
DrawRect(x-s/2, y-s/2, s, s, 'k')

% Draw nested stars, smallest r at least 0.1
k= 1;
while  r >= 0.1
   % Draw a star with radius r
   if rem(k,2)==1  % odd k
      DrawStar(x, y, r, 'm')  % magenta
   else
      DrawStar(x, y, r, 'y')  % yellow
   end
   % Reduce r
   r= r/1.2;
   k= k + 1;
end
```

```
for c = 0:2:8
    x= c; y= c;    % figure centered at (0,0)
    r= 1;          % radius of outermost star
    s= 2*r + 0.1; % side length of square
    DrawRect(x-s/2, y-s/2, s, s, 'k')

    % Draw nested stars, smallest r at least 0.1
    k= 1;
    while  r >= 0.1
        % Draw a star with radius r
        if rem(k,2)==1   % odd k
            DrawStar(x, y, r, 'm')   % magenta
        else
            DrawStar(x, y, r, 'y')   % yellow
        end
        % Reduce r
        r= r/1.2;
        k= k + 1;
    end
end
```

Nested loop

Example: Times Table

Write a script to print a times table for a specified range.



Developing the algorithm for the times table

	3	4	5	6	7
3	9	12	15	18	21
4	12	16	20	24	28
5	15	20	25	30	35
6	18	24	30	36	42
7	21	28	35	42	49

- Look for patterns
  - Each entry is row# × col#
  - Row#, col# increase regularly
- ⇒ Loop!!!
- What kind of loop?
  - for-loop—since the range of the headings is specified and the increment is regular
  - for each row#, get the products with all the col#s. Then go to next row# and get products with all col#s, …
  - ⇒ Nested loops!
- Details: what will be the print format? Don't forget to start new lines. Also need initial input to specify the range.

```
disp('Show the times table for specified range')
lo= input('What is the lower bound? ');
hi= input('What is the upper bound? ');
```

mTable.m