- 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?

N = 10;
for k = 1:N
    % Do some work.
    if (stopping condition)
        N = k;
    end
end

Save off range
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
DrawRect(-1,-2,6,3,'y')
\textbf{DrawDisk}(1,3,4,\textquoteleft r\textquoteleft)
DrawStar(1,3,4,'g')

- **x and y coordinates of the center**
- **“radius”**
- **color**
<table>
<thead>
<tr>
<th>Color</th>
<th>Code</th>
<th>Color Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>White</td>
<td>'w'</td>
<td><img src="image" alt="White" /></td>
</tr>
<tr>
<td>Black</td>
<td>'k'</td>
<td><img src="image" alt="Black" /></td>
</tr>
<tr>
<td>Red</td>
<td>'r'</td>
<td><img src="image" alt="Red" /></td>
</tr>
<tr>
<td>Blue</td>
<td>'b'</td>
<td><img src="image" alt="Blue" /></td>
</tr>
<tr>
<td>Green</td>
<td>'g'</td>
<td><img src="image" alt="Green" /></td>
</tr>
<tr>
<td>Yellow</td>
<td>'y'</td>
<td><img src="image" alt="Yellow" /></td>
</tr>
<tr>
<td>Magenta</td>
<td>'m'</td>
<td><img src="image" alt="Magenta" /></td>
</tr>
<tr>
<td>Cyan</td>
<td>'c'</td>
<td><img src="image" alt="Cyan" /></td>
</tr>
</tbody>
</table>
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
x = 0; y = 0; \quad \% \text{figure centered at (0,0)}
\r = 1; \quad \% \text{radius of outermost star}
\s = 2\r + 0.1; \quad \% \text{side length of square}
\text{DrawRect}(x-s/2, y-s/2, s, s, 'k')

\% \text{Draw nested stars, smallest } r \text{ 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

\[
n = \underline{\ldots} \\
\text{for } k = 1:n \\
\text{\% code to do} \\
\text{\% that something} \\
\text{end}
\]
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
Example: Times Table

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

<table>
<thead>
<tr>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>12</td>
<td>15</td>
<td>18</td>
<td>21</td>
</tr>
<tr>
<td>12</td>
<td>16</td>
<td>20</td>
<td>24</td>
<td>28</td>
</tr>
<tr>
<td>15</td>
<td>20</td>
<td>25</td>
<td>30</td>
<td>35</td>
</tr>
<tr>
<td>18</td>
<td>24</td>
<td>30</td>
<td>36</td>
<td>42</td>
</tr>
<tr>
<td>21</td>
<td>28</td>
<td>35</td>
<td>42</td>
<td>49</td>
</tr>
</tbody>
</table>
Developing the algorithm for the times table

- 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.

<table>
<thead>
<tr>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>9</td>
<td>12</td>
<td>15</td>
<td>18</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>16</td>
<td>20</td>
<td>24</td>
</tr>
<tr>
<td>5</td>
<td>15</td>
<td>20</td>
<td>25</td>
<td>30</td>
</tr>
<tr>
<td>6</td>
<td>18</td>
<td>24</td>
<td>30</td>
<td>36</td>
</tr>
<tr>
<td>7</td>
<td>21</td>
<td>28</td>
<td>35</td>
<td>42</td>
</tr>
</tbody>
</table>
disp('Show the times table for specified range')
lo= input('What is the lower bound? ');
hi= input('What is the upper bound? ');