Previous Lecture:
- Iteration using `while`

Today's Lecture:
- Nested loops
- Developing algorithms

Announcements:
- Discussion this week in computer labs. Read *Insight* §3.2 before discussion section.
- **Project 2** due Monday at 11pm
- We do not use `break` in this course
- Make use of Piazza, office hrs, and consulting hrs
Common loop patterns

Do something \( n \) times

\[
\text{for } k = 1:1:n \\
\quad \frac{\% \text{ Do something}}{}
\]

end

Do something an indefinite number of times

\[
\text{while} \ ( \text{not stopping signal} ) \\
\quad \frac{\% \text{ Initialize loop variables}}{} \\
\quad \frac{\% \text{ Do something}}{} \\
\quad \frac{\% \text{ Update loop variables}}{}
\]

end
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.”
What is the last line of output?

\[ x = 1; \]
\[ \text{disp}(x) \]
\[ y = x; \]
\[ \textbf{while } y == x \textbf{ && } x \leq 4 \textbf{ && } y \leq 4 \]
\[ \quad x = 2 \times x; \]
\[ \quad \text{disp}(x) \]
\[ \textbf{end} \]

\[ \begin{array}{cccc}
\text{A: 1} & \text{B: 2} & \text{C: 4} & \text{D: 8}
\end{array} \]
What will be displayed when you run the following script?

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

4
9

or

4
4

or

Something else …
Review loops/conditionals using user-defined graphics function

Draw a black square;
then draw a magenta disk;
then draw a yellow star.
DrawRect(-1, -2, 6, 3, 'y')

x and y coordinates of lower left corner
width
height

color
DrawDisk(1,3,4,\textquotesingle r\textquotesingle)

- **x and y coordinates of the center**
- **radius**
- **color**
DrawStar(1, 3, 4, 'g')

x and y coordinates of the center

“radius”

color
## Color Options

<table>
<thead>
<tr>
<th>Color</th>
<th>Code</th>
<th>Color Block</th>
</tr>
</thead>
<tbody>
<tr>
<td>White</td>
<td>'w'</td>
<td><img src="white.png" alt="White" /></td>
</tr>
<tr>
<td>Black</td>
<td>'k'</td>
<td><img src="black.png" alt="Black" /></td>
</tr>
<tr>
<td>Red</td>
<td>'r'</td>
<td><img src="red.png" alt="Red" /></td>
</tr>
<tr>
<td>Blue</td>
<td>'b'</td>
<td><img src="blue.png" alt="Blue" /></td>
</tr>
<tr>
<td>Green</td>
<td>'g'</td>
<td><img src="green.png" alt="Green" /></td>
</tr>
<tr>
<td>Yellow</td>
<td>'y'</td>
<td><img src="yellow.png" alt="Yellow" /></td>
</tr>
<tr>
<td>Magenta</td>
<td>'m'</td>
<td><img src="magenta.png" alt="Magenta" /></td>
</tr>
<tr>
<td>Cyan</td>
<td>'c'</td>
<td><img src="cyan.png" alt="Cyan" /></td>
</tr>
</tbody>
</table>
A simple 3-line script

\begin{align*}
\text{DrawRect(...)} \\
\text{DrawDisk(...)} \\
\text{DrawStar(...)}
\end{align*}
% 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
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`
Knowing how to draw

How difficult is it to draw
Pattern for doing something $n$ times

\[ n = \_\_\_\_\_ \]

\[ \text{for } k = 1:n \]

\[ \% \text{ code to do} \]
\[ \% \text{ that something} \]

\[ \text{end} \]
x = 0; y = 0; % figure centered at (0,0)

s = 2.1; % side length of square
DrawRect(x-s/2,y-s/2,s,s,'k')

r = 1; k = 1;
while r > 0.1 %r still big
  % draw a star
  if rem(k,2) == 1 %odd number
    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 (c,c)

s = 2.1; % side length of square
DrawRect(x-s/2,y-s/2,s,s,'k')

r = 1; k = 1;
while r > 0.1 % r still big
    % draw a star
    if rem(k,2) == 1 % odd number
        DrawStar(x,y,r,'m') % magenta
    else
        DrawStar(x,y,r,'y') % yellow
    end
    % reduce r
    r = r/1.2;
k = k + 1;
end
Pattern for doing something $n$ times

$n = \underline{\text{______}}$

for $k = 1:n$

% code to do
% that something

end
Example: Are they prime?

- Given integers $a$ and $b$, write a program that lists all the prime numbers in the range $[a, b]$.
- Assume $a > 1$, $b > 1$ and $a < b$. 
Example: Are they prime?
Subproblem: Is it prime?

Given integers $a$ and $b$, write a program that lists all the prime numbers in the range $[a, b]$.

Assume $a>1$, $b>1$ and $a<b$.

Write a program fragment to determine whether a given integer $n$ is prime, $n>1$.

Reminder: $\text{rem}(x,y)$ returns the remainder of $x$ divided by $y$. 
Example: Are they prime?
Subproblem: Is it prime?

* Given integers \( a \) and \( b \), write a program that lists all the prime numbers in the range \([a, b]\).
* Assume \( a>1 \), \( b>1 \) and \( a<b \).
* Write a program fragment to determine whether a given integer \( n \) is prime, \( n>1 \).
* Reminder: \( \text{rem}(x,y) \) returns the remainder of \( x \) divided by \( y \).
Start:
   divisor = 2

Repeat:
   rem (n, divisor) = divisor + 1

End:
   rem (n, divisor) = = 0
   divisor <= n ?

   divisor = 2;
   while (rem (n, divisor) = = 0)
      divisor = divisor + 1;
      end
   if (divisor = = n)
      disp ('prime')
      else
      disp ('composite')
      end
   end
%Given n, display whether it is prime

divisor = 2;
while ( rem(n,divisor) ~= 0 )
    divisor = divisor + 1;
end
if (divisor == n)
    fprintf(‘%d is prime\n’, n)
else
    fprintf(‘%d is composite\n’, n)
end
for n = a:b

%Given n, display whether it is prime
divisor = 2;
while ( rem(n,divisor) ~= 0 )
    divisor = divisor + 1;
end
if (divisor==n)
    fprintf('%d is prime\n', n)
else
    fprintf('%d is composite\n', n)
end
end
Example: Times Table

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

Row headings

Column headings

<table>
<thead>
<tr>
<th></th>
<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>
<td>21</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>16</td>
<td>20</td>
<td>24</td>
<td>28</td>
</tr>
<tr>
<td>5</td>
<td>15</td>
<td>20</td>
<td>25</td>
<td>30</td>
<td>35</td>
</tr>
<tr>
<td>6</td>
<td>18</td>
<td>24</td>
<td>30</td>
<td>36</td>
<td>42</td>
</tr>
<tr>
<td>7</td>
<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

<table>
<thead>
<tr>
<th></th>
<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>
<td>21</td>
</tr>
<tr>
<td>4</td>
<td>12</td>
<td>16</td>
<td>20</td>
<td>24</td>
<td>28</td>
</tr>
<tr>
<td>5</td>
<td>15</td>
<td>20</td>
<td>25</td>
<td>30</td>
<td>35</td>
</tr>
<tr>
<td>6</td>
<td>18</td>
<td>24</td>
<td>30</td>
<td>36</td>
<td>42</td>
</tr>
<tr>
<td>7</td>
<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# \times col#
  - Row#, col# increase regularly
- \Rightarrow 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, …
- \Rightarrow 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? ')