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 Thursday 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
  ```
  for \( k=1:n \)
  \% Do something
  end
  ```
- Do something an indefinite number of times
  ```
  while (not stopping signal)
  \% Do something
  \% Update loop variables
  end
  ```

What is the last line of output?

```matlab
x = 1;
disp(x)
y = x;
while y==x && x<=4 && y<=4
   x = 2*x;
disp(x)
end
```

Options:
- A: 1
- B: 2
- C: 4
- D: 8

What will be displayed when you run the following script?

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

Options:
- A
- B
- C
- D: 8

Example: Nested Stars
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

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 $y$. 

```matlab
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
```
Example: Times Table

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

Row 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>

Column headings

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>

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

Rational approximation of \( \pi \)

- \( \pi = 3.141592653589793... \)
- Can be closely approximated by fractions, e.g., \( \pi \approx 22/7 \)
- Rational number: a quotient of two integers
- Approximate \( \pi \) as \( p/q \) where \( p \) and \( q \) are positive integers \( \leq M \)
- Start with a straightforward solution:
  - Get \( M \) from user
  - Calculate quotient \( p/q \) for all combinations of \( p \) and \( q \)
  - Pick best quotient \( \rightarrow \) smallest error

% Rational approximation of pi

M = input('Enter M: ');

% Check all possible denominators
for q = 1:M
  % For current q find best numerator p...
  % Check all possible numerators
  pBest=1;  qBest=1;
  err_pq = abs(pBest/qBest - pi);
  % At this q, check all possible numerators
  for p = 1:M
    % Check all possible denominators
    end
  end
myPi = pBest/qBest;
% Complicated version in the book

M = input('Enter M: ');
qBest=1;  pBest=1;
err_pq = abs(pBest/qBest - pi);

% Check all possible denominators
for q = 1:M
    % At this q, check all possible numerators
    p0=1;  e0=abs(p0/q - pi);  % best p & error so far
    for p = 1:M
        if abs(p/q - pi) < e0  % new best numerator found
            p0=p;  e0 = abs(p/q - pi);
        end
    end
    % Is best quotient for this q is best over all?
    if e0 < err_pq
        pBest=p0;  qBest=q;  err_pq=e0;
    end
end

myPi = pBest/qBest;

% Rational approximation of pi

M = input('Enter M: ');
qBest=1;  pBest=1;
err_pq = abs(pBest/qBest - pi);

% Check all possible denominators
for q = 1:M
    % At this q, check all possible numerators
    for p = 1:M
        if abs(p/q - pi) < err_pq  % best p/q found
            err_pq = abs(p/q - pi);
            pBest= p;
            qBest= q;
        end
    end
end

myPi = pBest/qBest;

Algorithm: Finding the best in a set

Init bestSoFar
Loop over set
    if current is better than bestSoFar
        bestSoFar = current
    end
end