- Previous Lecture:
  - Iteration using `while`

- Today's Lecture:
  - Nested loops
  - Developing algorithms

- Announcements:
  - Due to Feb Break, attendance at discussion next week is optional but you are responsible for the contents of the exercise to be posted. Attend any Wednesday sections (10:10-4:25) if you like. Location is Upson B7 lab. Read Insight §3.2 before discussion.
  - Lecture for Thurs 2/19 will be pre-recorded and put online. TAs will use lecture time to provide help on Project 2
  - Project 2 due Monday 2/23 at 11pm
  - We do not use `break` in this course

---

**What is the last line of output?**

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

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

---

**Example: Nested Stars**

<table>
<thead>
<tr>
<th>Example: Nested Stars</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Nested Stars" /></td>
</tr>
</tbody>
</table>

---

**Knowing how to draw**

```
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
```

---

**Pattern for doing something `n` times**

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

---

**How difficult is it to draw**

<table>
<thead>
<tr>
<th>How difficult is it to draw</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="How difficult is it to draw" /></td>
</tr>
</tbody>
</table>
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 \( a < b \).

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: Times Table

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

<table>
<thead>
<tr>
<th>Row headings</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

- Look for patterns
  - Each entry is \( \text{row#} \times \text{col#} \)
  - \( \text{row#} \), \( \text{col#} \) increase regularly
  - \( \Rightarrow \) Loop!!!

- What kind of loop?
  - for-loop—since the range of the headings will be specified and increment regularly
  - for each row#, get the products with all the col#s. Then go to next row#, 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.

```matlab
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 \( \pi \)

- \( \pi \approx 3.141592653589793 \ldots \)
- Can be closely approximated by fractions, e.g., \( \pi \approx \frac{22}{7} \)
- Rational number: a quotient of two integers
- Approximate \( \pi \) as \( \frac{p}{q} \) where \( p \) and \( q \) are positive integers \( \leq M \)
- Start with a straight forward solution:
  - Get \( M \) from user
  - Calculate quotient \( \frac{p}{q} \) for all combinations of \( p \) and \( q \)
  - Pick best quotient \( \Rightarrow \) smallest error
The savvy programmer...

- Learns useful programming patterns and use them where appropriate.
- Seeks inspiration by working through test data "by hand".
- Asks, "What am I doing?" at each step.
- Sets up a variable for each piece of information maintained when working the problem by hand.
- Decomposes the problem into manageable subtasks.
- Refines the solution iteratively, solving simpler subproblems first.
- Remembers to check the problem’s boundary conditions.
- Validates the solution (program) by trying it on test data.

Algorithm: Finding the best in a set

- Init bestSoFar
- Loop over set
  - if current is better than bestSoFar
    - bestSoFar <- current
- end
- end

```
% Rational approximation of pi
M = input('Enter M: ');

% Best q, p, and error so far
qBest=1;  pBest=1;
err_pq = abs(pBest/qBest - pi);

% Check all possible denominators
for q = 1:M
    % Best p, q, and error so far
    qBest=1;  pBest=1;
    err_pq = abs(pBest/qBest - pi);

    % At this q, check all possible numerators
    for p = 1:M
        % At this p/q, check all possible denominators
        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;

Analyze the program for efficiency

- See Eg3_1 and FasterEg3_1 in the book

```
for a = 1:n
disp('alpha')
for b = 1:m
disp('beta')
end
end

How many times are "alpha" and "beta" displayed?

- A: n, m
- B: m, n
- C: n, n+m
- D: n, n^2m
- E: m^2n, m