- 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= 1:1:n
    % Do something

end
```

Do something an indefinite number of times

```
%Initialize loop variables

while ( not stopping signal )
    % Do something

    % 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;
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

What will be displayed when you run the following script?

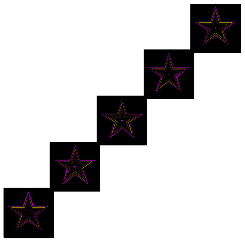
```
for k = 4:6
    disp(k)
    k= 9;
    disp(k)
end
```

A	or	B	or	C
4 9		4 4		*Something else …*

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
```



```
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: Are they prime?

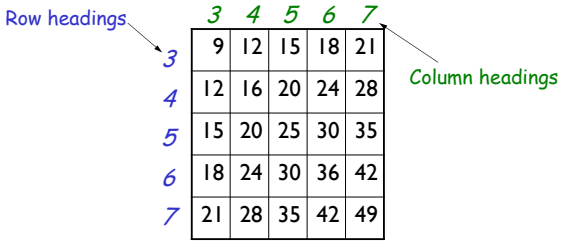
- Given integers a and b, write a program that lists all the prime numbers in the range [a, b].
- Assume a>1, b>1 and a<b.

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: 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.



	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

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

```
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 π

- π = 3.141592653589793…
- Can be closely approximated by fractions, e.g., π ≈ 22/7
- Rational number: a quotient of two integers
- Approximate π as p/q where p and q are positive integers ≤M
- Start with a straight forward solution:
  - Get M from user
  - Calculate quotient p/q for all combinations of p and q
  - Pick best quotient → 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



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




   end
end

myPi = pBest/qBest;
```

```
% Complicated version in the book

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
   % 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: ');
% 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
   % 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