

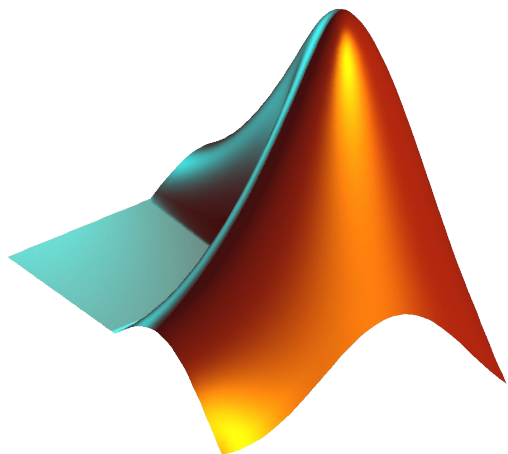
CS 1112 Introduction to Computing Using MATLAB

Instructor: Dominic Diaz

Website:

<https://www.cs.cornell.edu/courses/cs1112/2022fa/>

Today: Vectors (1D arrays)

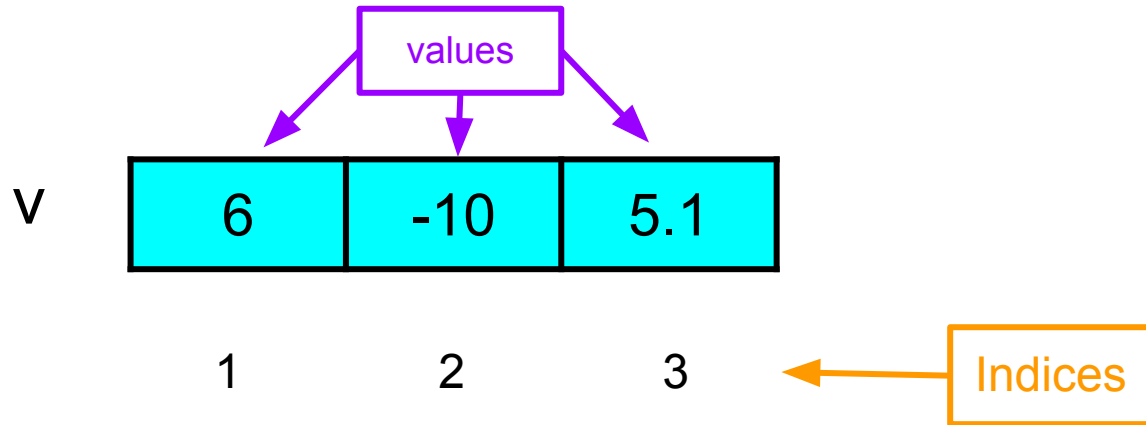


Agenda and announcements

- Last time
 - Started vectors (1D arrays)
- Today
 - More vectors (1D arrays)
- Announcements
 - Project 3 released last Friday and due Wednesday 10/5
 - Each partner is responsible for the whole project, from working on it to submitting it
 - Project 3 partners released
 - Tomorrow's discussion exercises will have two parts:
 - first part needs to be checked off by TA
 - Second part submit on MATLAB grader

Vector recap

A vector (or 1D array) is a collection of like data organized into rows or columns



- Index i ranges from $1 \leq i \leq \text{length}(v)$
- Accessing the i th element: `disp(v(i))`
- Changing the i th element: `v(i) = 10;`

```
% example: disp(v(1))  
% example: v(1) = 10;
```

Drawing a line segment

```
% position of coordinate 1
```

```
x1 = 1;
```

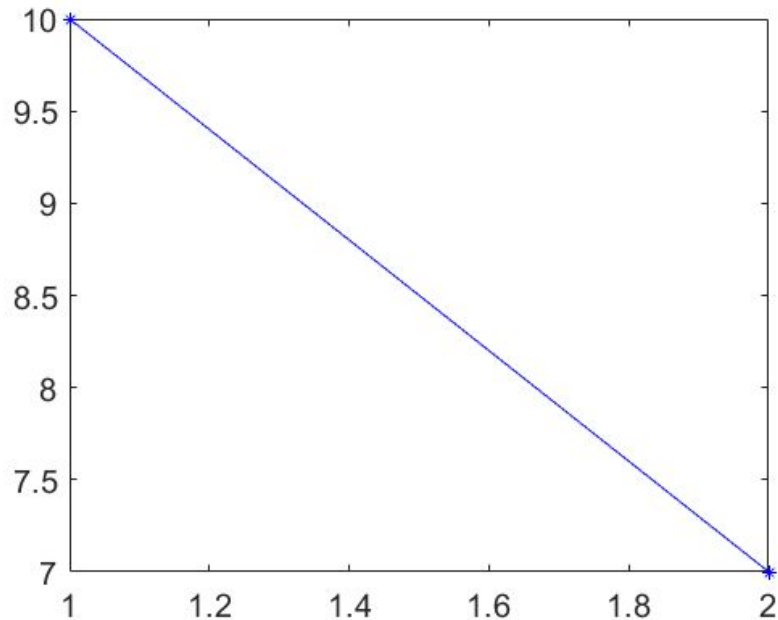
```
y1 = 10;
```

```
% position of coordinate 2
```

```
x2 = 2;
```

```
y2 = 7;
```

```
plot([x1 x2], [y1 y2], 'b-*')
```



Line and marker format

Here, we essentially have created two vectors!

Drawing more complicated plots with more points

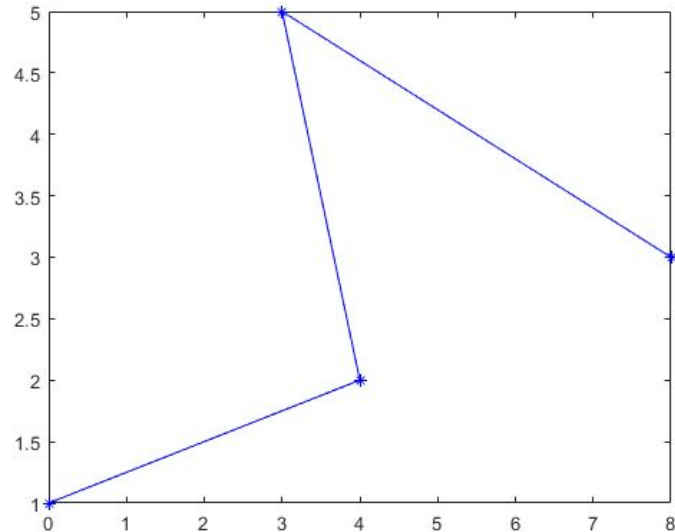
```
x = [0 4 3 8]; % x-coordinates  
y = [1 2 5 3]; % y-coordinates
```

```
plot(x, y, 'b-*')
```

Second input: y coordinates

Third input: line/marker format

First input: x coordinates



Drawing multiple lines on one plot

```
% Draw two different line
```

```
close all
```

```
figure
```

```
hold on
```

```
x = [0 4 3 8]; % first x-coordinates
```

```
y = [1 2 5 3]; % first y-coordinates
```

```
plot(x, y, 'b-*', 'LineWidth', 3)
```

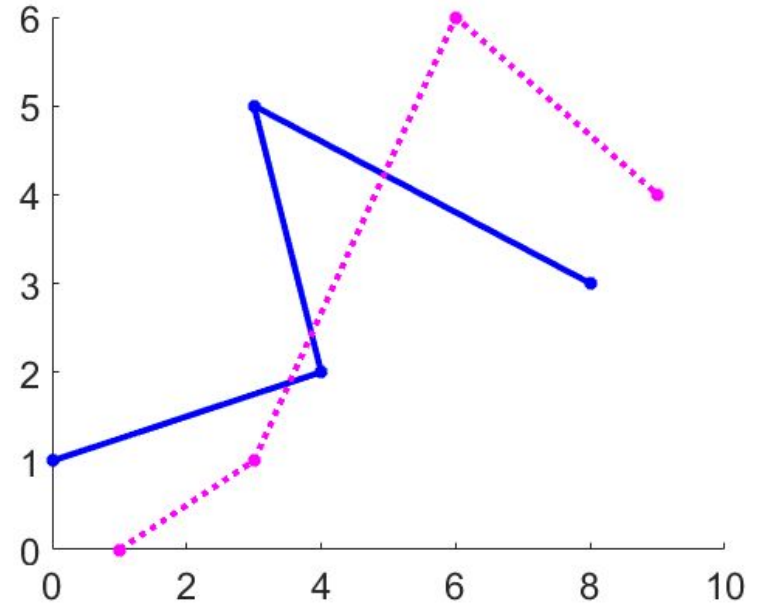
```
a = [1 3 6 9]; % second x-coordinates
```

```
b = [0 1 6 4]; % second y-coordinates
```

```
plot(a, b, 'm:*', 'LineWidth', 3)
```

```
set(gca, 'fontsize', 18);
```

```
hold off
```



Drawing multiple lines on one plot

```
% Draw two different line
```

```
close all
```

```
figure
```

```
hold on
```

```
x = [0 4 3 8]; % first x-coordinates
```

```
y = [1 2 5 3]; % first y-coordinates
```

```
plot(x, y, 'b-*', 'LineWidth', 3)
```

```
a = [1 3 6 9]; % second x-coordinates
```

```
b = [0 1 6 4]; % second y-coordinates
```

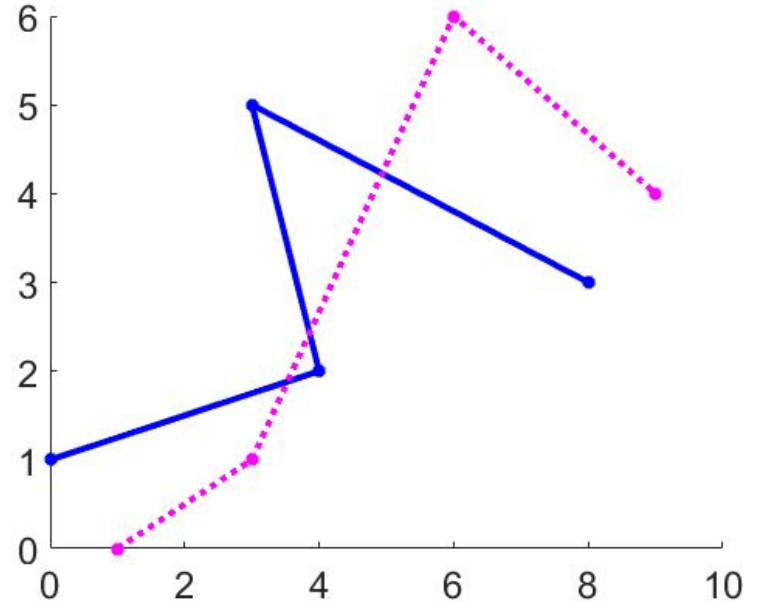
```
plot(a, b, 'm:*', 'LineWidth', 3)
```

```
set(gca, 'fontsize', 18);
```

```
hold off
```

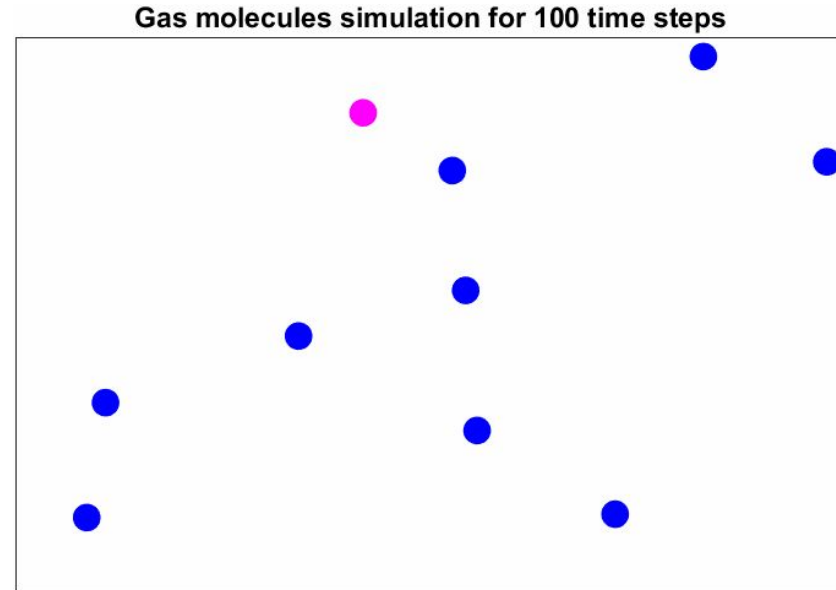
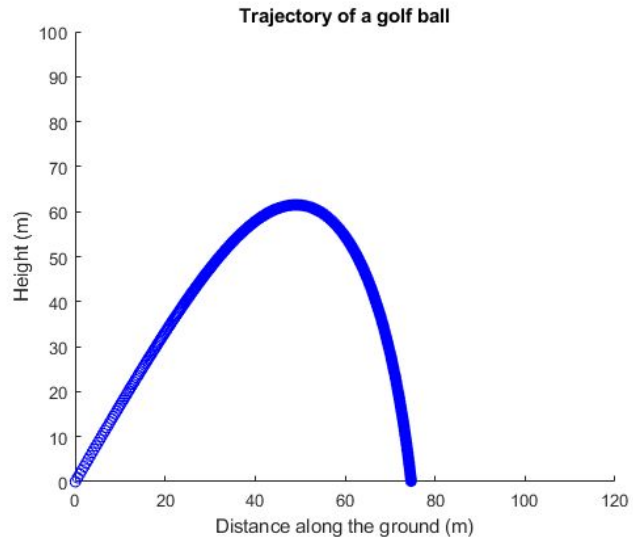
Sets the width of the line

Sets the size of the font



Simulation

- Imitates real system
- Often requires the use of random numbers
- May require many trials
 - This is a great opportunity to practice working with vectors!



Random numbers

- Random number in programming are typically **pseudorandom**
 - Pseudorandom number generator refers to an algorithm that uses mathematical formulas the produce random numbers
 - Not possible to generate truly random numbers from a deterministic thing like a computer
- Function rand generates random real numbers in the interval (0, 1). All numbers in this interval are equally likely to occur—uniform probability distribution

```
rand()           % one random number in (0,1)
6*rand()        % one random number in (0,6)
6*rand() + 1    % one random number in (1,7)
```

Width of interval

Base of interval

```
floor(6*rand() + 1) % one random integer in [1, 2, .., 6]
```

floor: rounds to nearest integer less than or equal to the input (rounds down)
ceil: rounds to nearest integer greater than or equal to the input (rounds up)

Poll Everywhere

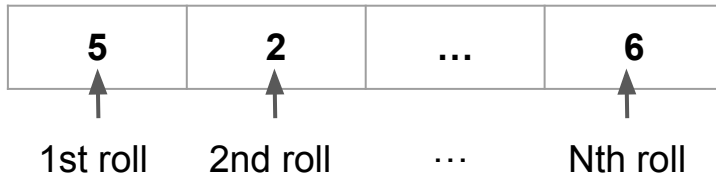
```
rand           % output is real number in interval (0, 1)
rand*6        % output is real number in interval (0, 6)
ceil(rand*6)  % output is integer in [1, 2, ..., 6]
```

Initializing a vector

When using vectors in MATLAB you should ask yourself “Do I know how long the vector should be?”

I need to store a fixed number of values.

- Example: rolling a 6-sided die N times

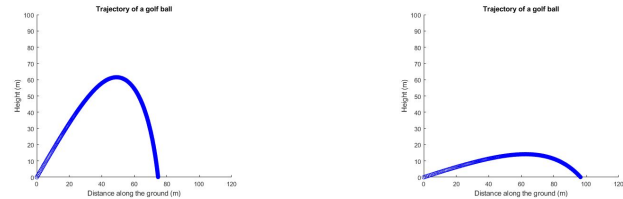


Initialize to be a vector of all zeros

```
dieRolls = zeros(1,N);  
for i = 1:N  
    dieRolls(i) = ceil(rand*6);  
end
```

I need to store values until some condition is met.

- Example: Storing trajectory of a golf ball until it hits the ground



Initialize to be an empty vector

```
xCoords = [];  
i = 0;  
while y > 0  
    i = i + 1;  
    xCoords(i) = x;  
    x = ...  
end
```

Initializing a vector

When using vectors in MATLAB you should ask yourself “Do I know how long the vector should be?”

I need to store a fixed number of values.

```
% Initialize
vec = zeros(1, N);

% update values
for i = 1:N
    vec(i) = [value];
end
```

I need to store values until some condition is met.

```
% Initialize
vec = [];

% update values
i = 0;
while [continueCondition]
    i = i + 1;
    vec(i) = [value]
end
```

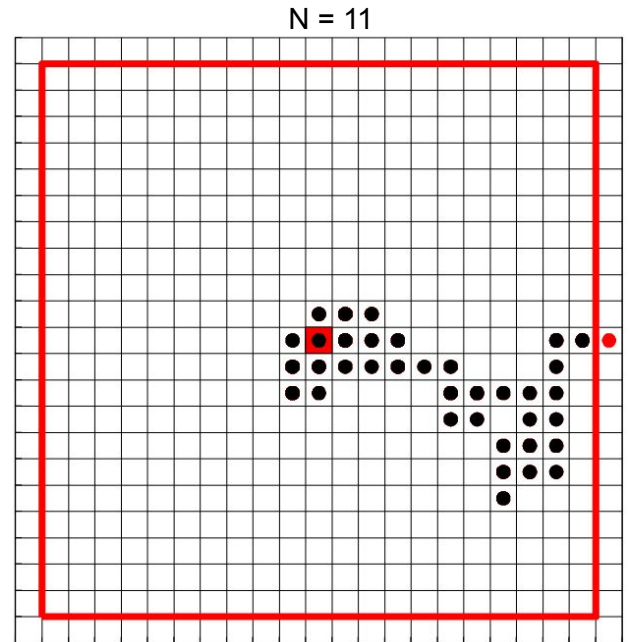
See `growingVec.m` if you would like to see small example in action!

Example 1: 2D random walk

A random walk is a random process that describes a path that consists of a succession of random steps. Scientists use random walks to model share prices, genetic drift, Brownian motion, animal movements, etc.

Complete the function `RandWalk2D_mod` that simulates a random walk as specified below:

```
function [x, y] = RandWalk2D_mod(N)
% Modified 2D random walk in a 2N+1 by 2N+1 grid
% Walk randomly from (0, 0) to an edge.
% At each time step, the walker steps in each
% direction with probability 1/8 and stays in its
% current location with probability 1/2.
% Walking stops when the absolute value of the
% x-coord or y-coord equals N.
% Vectors x and y represent the path.
```



```
function [x, y] = RandWalk2D_mod(N)

k = 0; xcurr = 0; ycurr = 0; % initialize position
x = []; y = [];

while abs(xcurr) < N && abs(ycurr) < N
    k = k + 1;

    x(k) = xcurr; % store current position
    y(k) = ycurr;

    % move or don't move based on random number generator

end
```

```
function [x, y] = RandWalk2D_mod(N)
```

```
k = 0; xcurr = 0; ycurr = 0; % initialize position  
x = []; y = [];
```

```
while abs(xcurr) < N && abs(ycurr) < N
```

```
    k = k + 1;
```

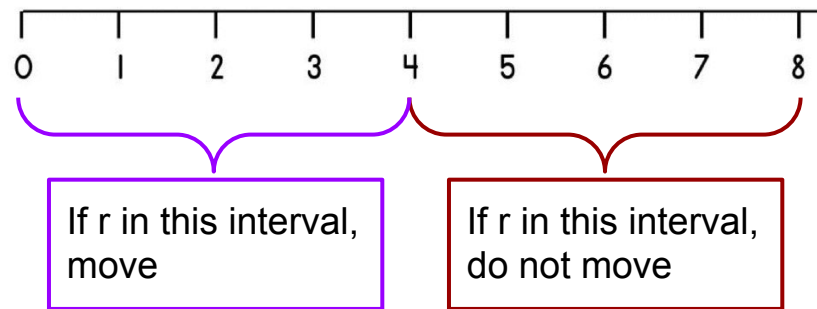
```
    r = rand*8; % random number between (0, 8)
```

```
    x(k) = xcurr; % store current position
```

```
    y(k) = ycurr;
```

```
    % move or don't move based on random number generator
```

```
end
```



```
function [x, y] = RandWalk2D_mod(N)
```

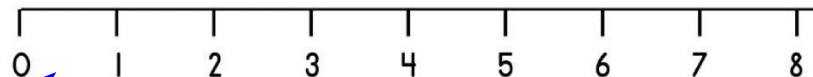
```
k = 0; xcurr = 0; ycurr = 0; % initialize position  
x = []; y = [];
```

```
while abs(xcurr) < N && abs(ycurr) < N
```

```
    k = k + 1;  
    r = rand*8; % random number between (0, 8)
```

```
    x(k) = xcurr; % store current position  
    y(k) = ycurr;
```

```
    if r <= 1  
        xcurr = xcurr + 1; % move right  
    elseif r <= 2  
        xcurr = xcurr - 1; % move left  
    elseif r <= 3  
        ycurr = ycurr + 1; % move up  
    elseif r <= 4  
        ycurr = ycurr - 1; % move down  
    end  
end
```



If r in this interval,
move

If r in this interval,
do not move

Move right if r is in (0, 1].


```

function [x, y] = RandWalk2D_mod(N)

k = 0; xcurr = 0; ycurr = 0; % initialize position
x = []; y = [];

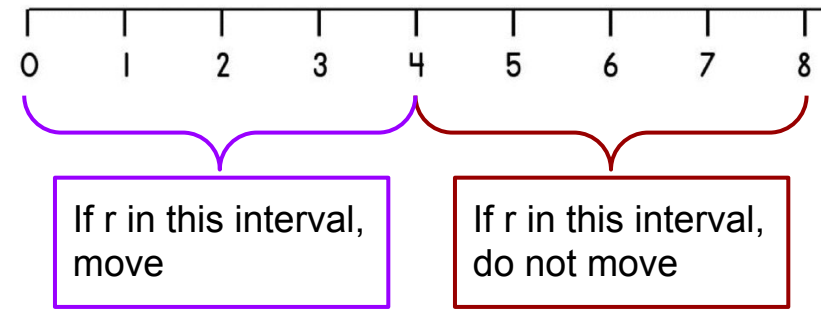
while abs(xcurr) < N && abs(ycurr) < N
    k = k + 1;
    r = rand*8; % random number between (0, 8)

    x(k) = xcurr; % store current position
    y(k) = ycurr;

    if r <= 1
        xcurr = xcurr + 1; % move right
    elseif r <= 2
        xcurr = xcurr - 1; % move left
    elseif r <= 3
        ycurr = ycurr + 1; % move up
    elseif r <= 4
        ycurr = ycurr - 1; % move down
    end
end

x(k+1) = xcurr; % store final position
y(k+1) = ycurr;

```



Example 2: All Larger

Task: Write a function `allLarger` that takes two vectors `x`, `y` as inputs. Assume `x` and `y` have the same length. The function should return `true` if for each index `k`, $x(k) > y(k)$ and `false` otherwise.

x	6	-10	5.1
y	5	-11	0

$$\begin{aligned}6 &> 5 \\ -10 &> -11 \\ 5.1 &> 0\end{aligned}$$

Output: true!

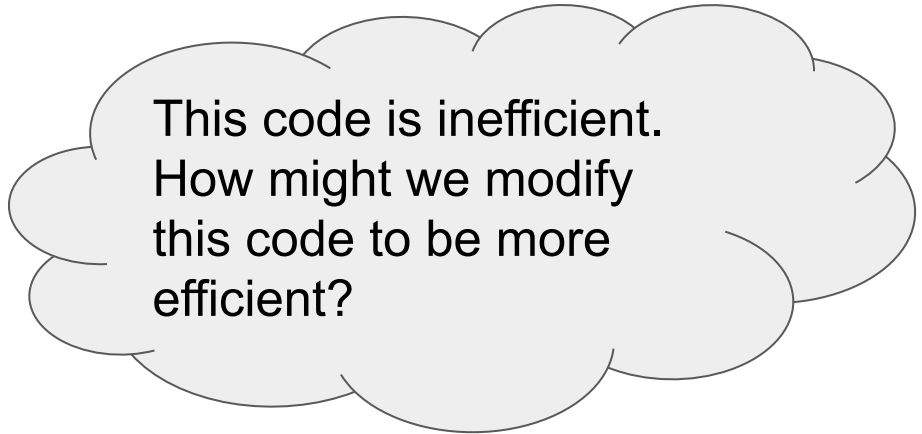
x	6	-10	100	5.1
y	5	-11	0	200

$$\begin{aligned}6 &> 5 \\ -10 &> -11 \\ 100 &> 0 \\ 5.1 &< 200\end{aligned}$$

Output: false!

```
function tf = allLarger(x, y)
% Sets tf = true if all elements of x are larger than
% corresponding elements in y, false otherwise.
% Assumes x and y are the same length.
```

```
tf = true;
for i = 1:length(x)
    if x(i) <= y(i)
        tf = false;
    end
end
```

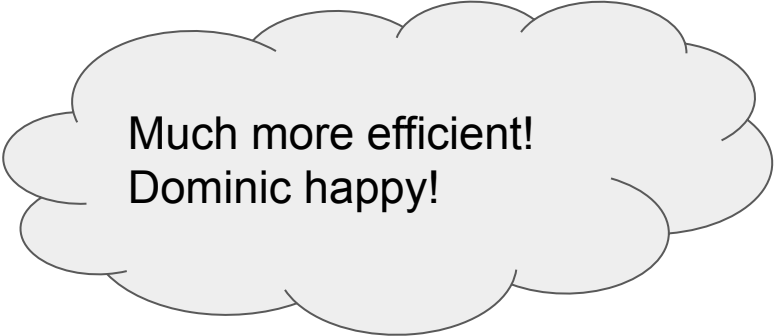


x	1	421	2	-300	90	4	1.3	3	-7	10
y	2	56	8	-10	0	0	3	10	65	18

After checking the first elements, the loop can terminate!

```
function tf = allLarger(x, y)
% Sets tf = true if all elements of x are larger than
% corresponding elements in y, false otherwise.
% Assumes x and y are the same length.
```

```
tf = true;
i = 1;
while i < length(x) && tf == true
    if x(i) <= y(i)
        tf = false;
    end
    i = i + 1;
end
```



Much more efficient!
Dominic happy!