# Introduction to Computers for Engineers:

Recitation #5

# Learning Objectives

- Continue to develop an understanding of arrays and loops
- Understand how loops and arrays can be used together
- Understand the potential relationship between counting variables and indexing

# Pre-Activity: Factorial Function

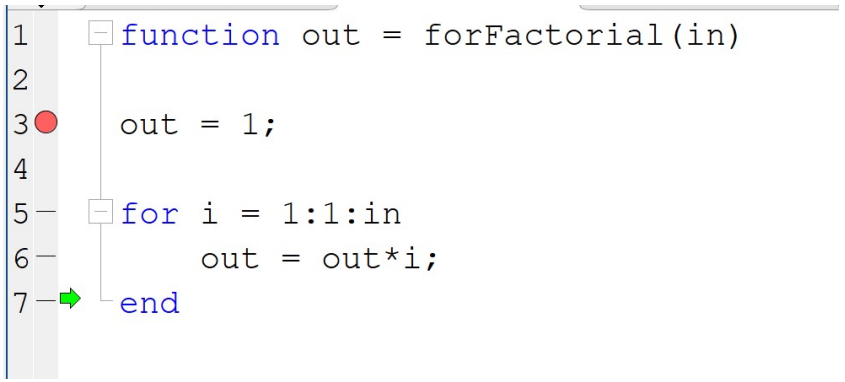
- Recall our factorial example from last week:

```
function [factorial_value] = factorialFunction(input_number)

factorial_value = 1;
counter = 1;

while counter <= input_number
    factorial_value = factorial_value * counter;
    counter = counter + 1;

end
```

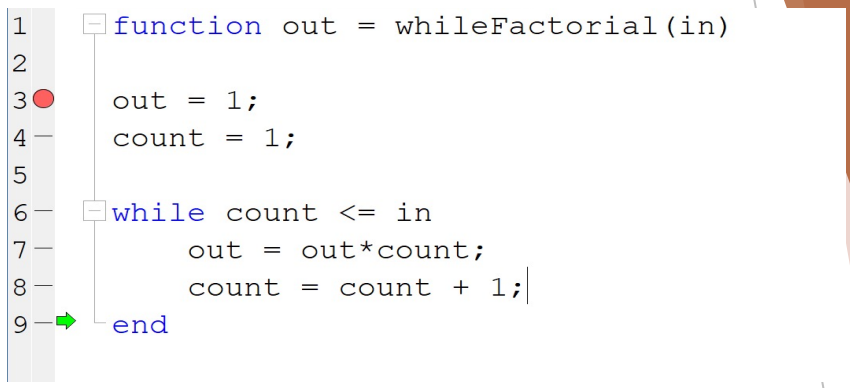
```
function [factorial_value] = factorialFunction(input_number)

factorial_value = 1;

for i=1:input_number
    factorial_value = factorial_value * i;

end
```

- Go on the following website to get these two functions:
  - https://soominkwon.github.io/matlab.html

# Activity 1: Visualization with Breakpoints



```
1   function out = forFactorial(in)
2
3     out = 1;
4
5   for i = 1:1:in
6       out = out*i;
7   end
```



```
1   function out = whileFactorial(in)
2
3   out = 1;
4   count = 1;
5
6   while count <= in
7       out = out*count;
8       count = count + 1;
9   end
```

1. In order to better understand how these loops work, we will use a **breakpoint** and fill out a **table.**

2. To place a breakpoint, click on the dashed line next to a line number. In the images above, the orange dots are breakpoints. **Your function must be saved in order for you to be able to place a breakpoint**

3. Call each function using **5** as the input value. Use the "step" button to step through the code until the green arrow is on the same line number as the image above. Use the workspace to get the values of all variable to fill out the table in your breakout room sheet. Then repeat, recording the values of all variables each time the green arrow is on the "end" keyword.

**What did this activity demonstrate about the differences and similarities between for and while loops?**

# Activity 2: Array-related functions

- Create a new script
- Create an array of random numbers with a random length in MATLAB
  - y = randn(randi(10),1);
- Use the **length** command to find the length of the array
- Use the **mean** command to find the mean value of the array
- Use the **median** command to find the median value of the array
- Use the **std** command to find the standard deviation of the array
- In your groups, discuss how you would do the following:
  - Use the zeros and ones command to create two arrays that are the same length as y, one filled entirely with ones and one filled entirely with zeros
  - Create another array that is the same length as y, with the same **mean** value, but with different numbers.
  - Create another array that is the same length as y, with the same **median** value, but with different numbers.

# Activity 2: Solution

```
% create an array of random numbers with a random length
rand_len = randi(10);
y = randn(rand_len, 1);

% compute the length, mean, median, and std of the array
len_y = length(y);
mean_y = mean(y);
median_y = median(y);
std_y = std(y);

% create an array of ones and zeros with the same length
zeros_y = zeros(rand_len, 1);
ones_y = ones(rand_len, 1);

% create an array with the same mean as y
x = randn(rand_len, 1);
mean_x = mean(x);

new_x = x / mean_x;
new_x = new_x * mean_y;

if mean(new_x) == mean_y
    disp('Yay!')
end
```

```
>> activity_2
Yay!
```

# Activity 3: Thinking with Loops

- Look at the script on the right
- This is a very long script with a lot of repetition
- **Rewrite this script using a loop!**

```
myData = randn(8,1);
numberPositive = 0;
if myData(1) > 0
    numberPositive = numberPositive + 1;
end
if myData(2) > 0
    numberPositive = numberPositive + 1;
end
if myData(3) > 0
    numberPositive = numberPositive + 1;
end
if myData(4) > 0
    numberPositive = numberPositive + 1;
end
if myData(5) > 0
    numberPositive = numberPositive + 1;
end
if myData(6) > 0
    numberPositive = numberPositive + 1;
end
if myData(7) > 0
    numberPositive = numberPositive + 1;
end
if myData(8) > 0
    numberPositive = numberPositive + 1;
end
```

# Activity 3: Solution

```
myData = randn(8,1);
numberPositive = 0;
if myData(1) > 0
    numberPositive = numberPositive + 1;
end
if myData(2) > 0
    numberPositive = numberPositive + 1;
end
if myData(3) > 0
    numberPositive = numberPositive + 1;
end
if myData(4) > 0
    numberPositive = numberPositive + 1;
end
if myData(5) > 0
    numberPositive = numberPositive + 1;
end
if myData(6) > 0
    numberPositive = numberPositive + 1;
end
if myData(7) > 0
    numberPositive = numberPositive + 1;
end
if myData(8) > 0
    numberPositive = numberPositive + 1;
end
```

```
myData = randn(8, 1);

number_positive = 0;

for i=1:length(myData)

    if myData(i) > 0
        number_positive = number_positive + 1;
    end

end
```

# Activity 4: Function with Loops

- We want to generalize our previous example by writing a function that takes an array as an input and counts how many positive numbers are in the array.
- Write a function called **arrayCounter**:
  - One input: **my_array**
  - One output: **num_positive**
- Hint: you want to loop through your array and increment a count every time you see a positive value (like the last activity)

# Activity 4: Solution

```
function [num_positive] = arrayCounter(my_array)

num_positive = 0;

for i=1:length(my_array)
    if my_array(i) > 0
        num_positive = num_positive + 1;
    end
end

end
```

# Activity 5: More Functions with Loops

- Write a function called meanThreshold that takes one input **my_array**
- The function should have 2 outputs:
  - above_mean: The number of elements with a value **above or equal to** the mean value of the array
  - below_mean: The number of elements with a value **below** the mean value of the array
- To do this
  1. Find the length of the array
  2. Find the mean value of the array (you can use the “mean” function to do this)
  3. Use a loop to count through each element of the array using a counting variable as the index
  4. Set up a conditional statement to determine whether values are above or below the mean
- **Come up with test cases together and see if your function is correct!**

# Activity 5: Solution

```
function [above_mean, below_mean] = meanThreshold(my_array)

above_mean = 0;
below_mean = 0;
mean_threshold = mean(my_array);

for i=1:length(my_array)
    if my_array(i) >= mean_threshold
        above_mean = above_mean + 1;
    else
        below_mean = below_mean + 1;
    end
end


end
```