# Common Confusion 11
# Functions vs. Anonymous Functions

Matthew Woodring

In this class, we will use both functions and anonymous functions extensively. With this in mind, it is obvious why we need to understand the differences and best use cases for both types.

A function is essentially a block of code that can be reused. It takes a set of inputs, performs some operations on those inputs, and then produces some outputs. In this class, functions will be created in their own separate function file.

Imagine that you wanted a user to be able to enter a value in inches and have the program convert it to centimeters. Note, we want the user to individually enter the values; they cannot input a vector. Without functions, your code would likely look something like this for three inputs:

```
‘inches = input(‘Enter a value in inches: ‘)’
‘centimeters = 2.54 * inches’
‘fprintf(‘The value in centimeters is %0.2f’, centimeters)’

‘inches = input(‘Enter a value in inches: ‘)’
‘centimeters = 2.54 * inches’
‘fprintf(‘The value in centimeters is %0.2f’, centimeters)’

‘inches = input(‘Enter a value in inches: ‘)’
‘centimeters = 2.54 * inches’
‘fprintf(‘The value in centimeters is %0.2f’, centimeters)’
```

When you run your code, MATLAB compiles the code and begins executing it from top to bottom. Since we are not using functions, we have to ask for the input, convert it to centimeters, then output it. With a function named ‘printInchesToCentimeters’, that takes an input in inches and converts it to centimeters and prints it, we can reduce the above code to:

```
‘inches = input(‘Enter a value in inches: ‘)’
‘printInchesToCentimeters(inches)’

‘inches = input(‘Enter a value in inches: ‘)’
‘printInchesToCentimeters(inches)’

‘inches = input(‘Enter a value in inches: ‘)’
‘printInchesToCentimeters(inches)’
```

# Common Confusion 11
# Functions vs. Anonymous Functions
Matthew Woodring

Clearly, this is much easier since we had to write significantly less code. In practice, functions are often even more useful than this example since most functions will contain more code than this simple example. There are many more examples in this class and online that highlight the usefulness of functions.

Functions are a powerful tool and you must know how to use them to do well in this class.

Anonymous functions, while not used as much as functions, are something you need to understand well. Most functions, in their own function file, contain more than one line of code. Anonymous functions are used to account for when you only need a single line of code and do not want to create a whole separate function file.

Usually, anonymous functions are used to define a mathematical formula. For example, the anonymous function below defines a horizontally stretched 'cos' function:

'cosFunction = @(x) cos(2 * x)'

In the above example, the '@' symbol defines the line as an anonymous function, the 'x' acts as the input, and 'cosFunction' acts as the function declaration. For example, if we pass the 'cosFunction' function the input 'pi', we will get an answer of '1':

'y = cosFunction(pi)'

Anonymous functions operate exactly the same as normal functions, but they do not require an additional function file to be made. In the above line of code, we called the 'cosFunction', passed it an argument of 'pi', and stored the output of '1' in 'y'. This is the same way we call a normal function. The only difference is that an extra function file was not required to create the 'cosFunction' function.

Functions and anonymous functions really are pretty similar. However, they both have their own best use cases. Remember, functions typically contain more than one line of code while anonymous functions only contain one line of code. Additionally, remember that functions require an additional file to be made while an anonymous function does not.