Functions

A **function** is a set of code within a program that performs a specific task. They are also reusable and can be used over and over again. Functions are a powerful programming tool that make programs easier to read and change. You may see them referred to as "Procedures", "Methods", or "Subroutines" but we will stick with calling them functions.

A function can be imagined as a box with mechanisms inside that act on data. There is typically an input into the box and code inside that utilizes the input to produce an output.

A function will be defined by three parts:

1. Return type
2. Function name
3. Arguments

```
returnType functionName(arguments){
    //Code
}
```

Processing has two main functions that form the backbone of the language: `setup()` and `draw()`.

The code inside `setup()` runs once when the program first starts. The size of the display window size is set, along with initial drawing conditions, and the assignment of global variables.

The code inside `draw()` runs continuously in a loop and draws one image to the display window at the end of each loop.

We've used functions each time we call a draw primitive, `point()`, `line()`, `triangle()`, `quad()` are all functions.

Function declarations

Draw and setup are preceded by the keyword void. This is a Java programming keyword to specify that the function does not return a value.

Functions can be preceded by other keywords including boolean, int, float, and all other primitive data types(Functions can also return objects but we will discuss that later.) These keywords indicate what value the function returns.

For example:

```
int addNumbers(){
    int n = i + j;
    return n;
}
```

The addNumbers function will return an integer value.

Or:

```
boolean isTrue(){
    decision = true;
    return decision;
}
```

The isTrue function will return a boolean value.

Arguments and Parameters.

Arguments are the variables that live inside the parentheses in a function. Parameters are the values that are passed to the function. It's a

Using the addNumbers function example, we can pass the variables i and j as arguments. The values that i and j contain are the parameters.

```
int addNumbers(int i, int j){
    int n = i + j;
    return n;
}

addNumbers(5,6);
```

Passing the parameter values 5 and 6 to the arguments of the addNumbers function.

The difference between arguments and parameters is not overly important for our current work. If you confuse them do not worry. Understanding how to create a function that excepts values and returns values is the knowledge to take away.