1. **Get 1 to 255** - Write a function that returns an array with all the numbers from 1 to 255.
2. **Get even 1000** - Write a function that would get the sum of all the even numbers from 1 to 1000.  You may use a modulus operator for this exercise.
3. **Sum odd 5000** - Write a function that returns the sum of all the odd numbers from 1 to 5000. (e.g. 1+3+5+...+4997+4999).
4. **Iterate an array** - Write a function that returns the sum of all the values within an array. (e.g. [1,2,5] returns 8. [-5,2,5,12] returns 14).
5. **Find max** - Given an array with multiple values, write a function that returns the maximum number in the array. (e.g. for [-3,3,5,7] max is 7)
6. **Find average** - Given an array with multiple values, write a function that returns the average of the values in the array. (e.g. for [1,3,5,7,20] average is 7.2)
7. **Array odd** - Write a function that would return an array of all the odd numbers between 1 to 50. (ex. [1,3,5, .... , 47,49]). Hint: Use 'push' method.
8. **Greater than Y** - Given value of Y, write a function that takes an array and returns the number of values that are greater than Y. For example if arr = [1, 3, 5, 7] and Y = 3, your function will return 2. (There are two values in the array greater than 3, which are 5, 7).
9. **Squares** - Given an array with multiple values, write a function that replaces each value in the array with the value squared by itself. (e.g. [1,5,10,-2] will become [1,25,100,4])
10. **Negatives** - Given an array with multiple values, write a function that replaces any negative numbers within the array with the value of 0. When the program is done the array should contain no negative values. (e.g. [1,5,10,-2] will become [1,5,10,0])
11. **Max/Min/Avg** - Given an array with multiple values, write a function that returns a new array that only contains the maximum, minimum, and average values of the original array. (e.g. [1,5,10,-2] will return [10,-2,3.5])
12. **Swap Values** - Write a function that will swap the first and last values of any given array. The default minimum length of the array is 2. (e.g. [1,5,10,-2] will become [-2,5,10,1]).
13. **Number to String** - Write a function that takes an array of numbers and replaces any negative values within the array with the string 'Dojo'. For example if array = [-1,-3,2], your function will return ['Dojo','Dojo',2].

**Basic Foundation II**

1. **Biggie Size** - Given an array, write a function that changes all positive numbers in the array to the string "big".  Example: makeItBig([-1,3,5,-5]) returns that same array, changed to [-1, "big", "big", -5].
2. **Print Low, Return High** - Create a function that takes in an array of numbers.  The function should **print**the lowest value in the array, and **return**the highest value in the array.
3. **Print One, Return Another** - Build a function that takes in an array of numbers.  The function should **print**the *second-to-last*value in the array, and **return**the *first odd* value in the array.
4. **Double Vision** - Given an array (similar to saying 'takes in an array'), create a function that returns a *new*array where each value in the original array has been doubled.  Calling double([1,2,3]) should return [2,4,6] without changing the original array.
5. **Count Positives** - Given an array of numbers, create a function to replace the last value with the number of positive values found in the array.  Example, countPositives([-1,1,1,1]) changes the original array to [-1,1,1,3] and returns it.
6. **Evens and Odds** - Create a function that accepts an array.  Every time that array has three odd values in a row, print "That's odd!".  Every time the array has three evens in a row, print "Even more so!".
7. **Increment the Seconds** - Given an array of numbers **arr**, add 1 to every other element, specifically those whose index is odd (arr[1], arr[3], arr[5], etc).  Afterward, console.log each array value and return **arr**.
8. **Previous Lengths** - You are passed an array (similar to saying 'takes in an array' or 'given an array') containing **strings**.  Working within that same array, replace each string with a number - the *length*of the string at the previous array index - and return the array.  For example, previousLengths(["hello", "dojo", "awesome"]) should return ["hello", 5, 4]. **Hint:** Can for loops only go forward?
9. **Add Seven** - Build a function that accepts an array. Return a **new**array with all the values of the original, but add 7 to each. Do not alter the original array.  Example, addSeven([1,2,3]) should return [8,9,10] in a new array.
10. **Reverse Array**- Given an array, write a function that reverses its values, in-place.  Example: reverse([3,1,6,4,2]) returns the same array, but now contains values reversed like so... [2,4,6,1,3].  Do this **without**creating an empty temporary array.  (Hint: you'll need to swap values).
11. **Outlook: Negative** - Given an array, **create and return a new one** containing all the values of the original array, but make them all negative (not simply multiplied by -1). Given [1,-3,5], return [-1,-3,-5].
12. **Always Hungry -**Create a function that accepts an array, and prints "yummy" each time one of the values is equal to "food".  If no array values are "food", then print "I'm hungry" once.
13. **Swap Toward the Center -** Given an array, swap the first and last values, third and third-to-last values, etc.  Example: swapTowardCenter([true,42,"Ada",2,"pizza"]) turns the array into ["pizza", 42, "Ada", 2, true].  swapTowardCenter([1,2,3,4,5,6]) turns the array into [6,2,4,3,5,1].  No need to return the array this time.
14. **Scale the Array -** Given an array **arr**and a number **num**, multiply all values in the array **arr**by the number **num**, and return the changed array **arr**.  For example, scaleArray([1,2,3], 3) should return [3,6,9].