Exercise – Functions

First are quick review questions that you should write down the answers for in a document. The second part is practical exercises. Both are important for your learning and to help you retain the concepts.

Review Questions

1. What does the word “public” in a function mean?
   Public means other classes/files can use this function
2. What is the purpose of giving a function a name?
   So you can easily identify the function in all your code and call it.
3. What can you put in the **() parentheses** of the function?
   Parentheses are for arguments, it can also be left empty.
4. What does the **return value** do in a function?
   It returns some data back to the code that called the function.
5. How would you describe a functions purpose?
   Is to bundle a chunk of code together so it can be used elsewhere in the code.
6. What can you do with values that you **return** from functions?
   Use the values in other parts of the code.
7. What is the naming convention called for functions?
   Pascal Case
8. What functions might you add to a game object (i.e., a class) representing the player?
   DamageCalc(), PlayerAttack(), Item()

Practical Exercise

Make a new C# console project in Visual Studio called **FunctionsExercise.**

1. Make a new class called **Game**
2. In your Game class, make a variable called **score** and set it to 0
3. Then in the Game class make a new function called **Start**
4. In Start, print out the starting value of your **score** variable
5. From your **Program** class, in the **Main** function, create a new instance of your **Game** class
6. Access the **score** variable of the instance of your game class and set it to 100
7. Run the **Start** function belonging to the instance of your game class. If it prints out the text and you get no errors, it was successful. If not, consider the privacy value of your function
8. In your Game class, create a new function called **AddToScore**
   a. Give the function a return type of **int**
   b. In the **parentheses**, create a temporary int called **add**
   c. In the body of the function, add the value of **add** to **score**
   d. return the final value of **score**
9. In the Start function of your game class, on a new line

   a. Print out text telling the player they scored, and then print out their score.
   b. Instead of printing out the value of the **score** variable directly, insert the **AddToScore** function and pass in how much you want to increase the score by, so that you can increase the score and display the new value at the same time
10. On a new line, do the same thing you just did but increase the score by a different value
11. Repeat that process one more time, increasing the score again and printing out the value
12. Create a new function called **PrintScore**
   a. Create a temporary int called **add** in the **parentheses**, also called passing in a variable
   b. Copy and paste the text you used earlier for printing the players score, and insert it into this function.
   c. Replace the value you used to increase the score by with the new temporary int **add** from the PrintScore function
13. Replace the code you copy and pasted earlier for printing the score and replace it by using this function 3 times, passing in different values

This process of quickly getting the code to work by copy and pasting, then once it works and seeing where code is repeated, creating functions to replace that repeated code, is a useful process.

   a. *First focus on quickly making something work*
   b. *Test it*
   c. *Then look for areas of code that are messy and could be cleaned up or areas of code that are repeated and see if a function can replace repeated code.*

Leaving repeated code, or leaving code that has been copy and pasted multiple times, is bad practice. Always replace repeated code with functions. You can see how much easier it will be to change the code now if we wanted to do something different when we printed the score or added to the score.

But doing some quick copy and pasting to make something work initially is okay, so long as it doesn't stay that way. Always aim for clean code. It will make your life as a programmer easier in the long run.