Functions – Quick Notes

Concept

Functions are also called methods, subprograms, procedures, routines, or subroutines in various languages and depending on purpose.

No matter what name you use, functions group together some code and organize it so it can be used from multiple places in your program and possibly by other's programs. When you use a function ("call" the function) you can give it information to work on and it can optionally return some value to you ("return value").

One reason to define functions is to take code that will be used multiple times and write it just once, even if there are differences in what you need it to do each time. Functions can make code easier to read and understand.

Terms to learn and use carefully:

Function name: an identifier used to refer to the function, usually a verb or verb phrase, e.g. `singHappyBirthday`

Formal parameter: an identifier (and type) used in the function to get information passed in from the calling program. A function can have zero or more formal parameters. Parameters may also be called arguments or input parameters; they are defined similarly to variables with a type and a name, e.g. `String name`

Argument value: the information which the calling program passes to the function in one of the formal parameters. Values must be in the same order as in the function declaration. Arguments can be any expression but must match the type of the formal parameter, e.g. `"William" + "  " + "Honig"`

Return value: The information sent back to the caller as the value of the function. Some functions are defined to give return values, some are defined without return values (void). Even if the function returns a value the calling program can ignore it. The type of the return value (or void) is given in the definition e.g. `String`

Declaration of an Function

The declaration tells the compiler the name of the function, the name and type of each of its formal parameters, and the type of its return value if it has one. The declaration begins with the function's "signature" and then the "body" which includes the code of the function. The body can refer to any of the formal parameters by name but is not required to use them and can ignore one or all of its parameters. The body can include one or more return statements which, when executed, will end the function and return to the calling program (possibly with a return value). If execution of the function reaches the end of the body without encountering any return, a return is assumed (in this case no value can be returned).

```
public static String singHappyBirthday(String name, int age){
      String song;
      //To Do: set up the song
      return song;
}
```

Calling a Function

A function is called by writing its name followed by (). Inside the parentheses are zero or more argument value expressions. Generally, the number and type of these expressions must match the number and types of the formal parameters in the function declaration.

If the function returns a value it becomes the value of the function call and expression evaluation continues. A function call acts similarly to any variable name in an expression except instead of simply accessing the current value of the variable, the function is called, inputs are passed to the function's code, and the return value is calculated at run time and returned.

```
…      singHappyBirthday( " William " + "   " + " Honig " , 21)   …
```

Rules and Recommendations

At least for freshers or newbies, *please* (it will help you learn and prevent troublesome problems):

1. Be very careful to distinguish between getting input or printing output to the user and functions. These are two completely different things. Functions get input from their parameters and return values; these interactions are all inside the program, inside the computer, and do not involve the user.

2. For the reason given above, it is often best to have all user input and output in one place, often in main. When a function is called it should not get input from the user or print out things to the user. (This is a standard programming pattern called façade). Functions should use their inputs from parameters and return results as a return value.

3. It is best to have just one return point in a function, often at the bottom of the code. The return statement must be used to return a value but is optional for void functions.

4. Never, ever, not even once, change a value of an formal parameter inside the function.

5. Never, ever, not even once, have a local variable inside the function with the same name as a formal parameter. Formal parameters are defined in the function signature, not as separate variable declarations.

6. Methods are the usual name for Object Oriented functions that work with objects of a given class.

Want to know more?

Call by Value and Call By Reference explained
http://www.javawithus.com/tutorial/call-by-value-and-call-by-reference

Variable number of formal parameters (although passing and returning a collection is often better style)
http://www.javawithus.com/tutorial/using-ellipsis-to-accept-variable-number-of-arguments

Verrsion 1.0 (java edition) October, 2016, for Comp170, Introduction to Object Oriented Programming, a CS1 first serious programming course. Contact whonig AT luc.edu