Module	CS4012 – Representation and Modelling
Day	14
Lab	8
Topic	Functions

Summary

(Function call): A statement in which a named section of code is executed.

Functions are named sections of code that carry out a specific task. To execute a function, its name is called. At the point of a function call, data values can be passed over to the function, and after execution values can be returned. Unlike many other programming languages, functions in Lua allow for multiple return values. With this in mind, the following focuses on declaring and calling functions.

Exercise 1: Declaring and calling functions

A function may be passed zero or more values as arguments when it is called. Similarly, a function can return zero or more values to the site of a call. The syntax of a function declaration has the following form:

```
function name(par1, par2)          -- parameters, a comma separated list of  variables

        -- statement(s)

        return val1, val2          --  values, a comma separated list of expressions
end
```

The function declaration is comprised of the function signature (underlined) and the function body. Enclosed by parentheses, the parameter variables are local to the scope of the function body – just like a "do" block. The key word "return" is optional, but if present, must be stated on the last line within the function body or a conditional statement. Zero or more values can be returned from a function. For instance, the following code fragment declares a function that takes one parameter and returns its data type:

```
function findType(dataValue)

        var = type(dataValue)

        return var
end

aNum = 7
aNumType = findType(aNum)
print(aNumType)                    -- outputs "number"
```

Before it can be used, a function must first be declared; note the order of the declaration and call above. The "findType(aNum)" call passes the variable "aNum" as an argument, and returns its data type, namely, number.

Tasks:

1. Write (declare) a function called “helloWorld” that takes zero parameters and returns zero values. The function should output the string “Hello, Word!” when called. Test the function by calling it.

2. Modify the above function so that the string “Hello, Word!” is returned as a value. Test the function by assigning the return value to a variable, whose value is then printed.

3. Write a function called “sumOfTwo” which takes two parameters and returns one value. The function should return the sum of the two numbers arguments passed to it. Test the function by calling it with values 14 and 44 and outputting the return value as above.

4. Write a function called “largestOfThree” which takes three parameters and returns one value. The function should return the largest of the three parameters passed to it. Test the function by calling it with values 14, 7, and 21, and then output the return value.

Hint: Consider the ternary operator used (in Session 03) to determine the larger of *two* numbers.

Exercise 2: Using predefined functions

Programming languages usually have predefined procedures or functions for tasks that are commonly performed, like text manipulation. Thus a user defined function can make use of those already defined.

Tasks:

1. Write a function called “myFormat” that takes three parameters and returns zero values. The first and second string arguments passed to the function are your forename and surname, and the third is a format string. The function should insert these names into the format string and output the new string. The output should look something like:

   “My first name is *Sam* and my last name is *Jones*.”

Hint: Consider the predfined string.format function.

2. Write a function called “myReplace” that takes three parameters and returns one value. The first string argument passed to the function is a sentence, the second is a word to be replaced and the third is another word with which to replace it. The function should output the new string, and then return how many times the word was replaced. Test the function by outputting the return value for a given call.

Hint: Consider the predfined string.gsub function.

3. Write a function called “myFind” that takes two parameters and returns two values. The first string argument passed to the function is a sentence, and the second is a word to find. The function should return the start and end indices within the sentence of the first occurrence of the word. Test the function by outputting the return values for a given call.

Hint: Consider the predfined string.find function.