**Functions**

A function is a set of statements that perform a particular task.

**Why functions?**

* Code is developed in a modular fashion.
* Able to use existing functions as building blocks for new programs
* Abstraction; able to hide the function source code.

**Naming conventions**

For this course use a lowercase letter for function name.

**Variable scope**

A variable declared within a function has a local scope.

A variable declared outside a function at the top of your program has a global scope.

**Format of functions**

type name(param1, param2, …){

statements

}

Where:

* **type –** refers to the type of data the function will return, however return type is not necessary in the pseudocode.
* **name –** uniquely identifies the function, making it possible for it to be called.
* **parameters -**  act as local variables of the function, allowing the user to pass arguments to the function when it is called, making the function dynamic.
* **statements –** the block of code that will be executed when the function is called.

The first line of a function definition is called a ***header*** and the following lines is called the ***body.***

***Return* statement**

In a value-returning function, ***return does two distinct things***:

* specifies the value returned after the function is called.
* terminates that execution of the callee and transfer control back to the caller.

A ***function can only return one value***:

* The value can be any expression matching the return type.
* But it can contain more than one return statement.

In a ***void function:***

* return is optional.
* return may also be used to terminate further execution of the function explicitly.
* No return value should appear following return.

**Calling a function**

A function is called by the function name and its parameters (if any were defined).

* The function call must include an argument for each parameter.
* Arguments should match to each corresponding formal parameter definition.
* The arguments should also match the data types in the parameter definition.

**Function mechanism**

* Function in a program is not executed until called.
* Function can take arguments within the function call, even if it’s not mandatory.
* Argument is a program data needed by the function to execute.
* When the function is done executing, programs returns to the same location which called the function.

Parameter list:

* When the function is invoked, the parameters can be a variable, constant or expression e.g. play(7, 6 + a, dollars)
* Where ***7 is put into the local variable*** **a**, ***6 + a is put into the local variable b*** and the ***value of the variable dollars in put into c.***

**Uses of functions**

* used to avoid repetition of same code/logic in the program.
* functions have no limit in the number of times it can be called.
* large programs are easily comprehensible if they are divided into functions.

**Advantages of functions**

* **Reusability –** able to reuse the same code in different places in the program.
* **Abstraction –** functions can help hide the complexity of tasks executed in the function.
* **Testing –** functions can be tested independently of the rest of the program, making it easier to find and fix bugs.

**Cohesion**

Cohesion measures the extent to which a set of statements in a function is related to a single purpose or task. If code in a function are all related to one task then it is said to have a strong cohesion. If multiple statements in the function perform unrelated tasks then it is a weak cohesion.

**Coupling**

Looks at the degree of dependency of modules, function or other components towards each other.

* **Tight coupling –** two functions are strongly dependent of each other, where one change in a function requires a corresponding change in the other to maintain intended functionality.
* **Loose coupling –** two functions are independent of each other, and a change in one function does not require changes in the second function. This result in a more flexible and modular code that is easier to maintain.