According to Dr. Charles Severance of University of Michigan School of Information, in the context of programming, a <strong>function</strong> is a named sequence of statements that perform a computation. You define a function by using the keyword <code>def</code> followed by the name if the function and then the sequence of statements as seen in previous articles. You call a function by specifying the name followed by parenthesis with the parameters inside for example

<code>

>>>type(“Dumebi”)

<class “string”>

</code>

<h3>Built-in functions</h3>

Python has different built-in functions that can be used without needing to define a function. For example, we have seen the use of print(), input() etc. These are built-in functions. You should treat the names of built-in functions as reserved words and avoid naming other functions or variables with these names.

Other examples of built-in functions apart from the ones already seen in previous articles include <code<len()</code> which tells you how many items are in its argument. The argument can be anything inside the parenthesis e.g.

<code>

>>>len(“Dumebi”)

6

</code>

Other built-in functions include type conversion functions which convert one data type to another if the conversion is possible e.g. <code>int(“32”)</code would convert the string “32” into an integer. <code>int(“Dumebi”)</code> would not work. <code>float()</code> converts the argument to floating point numbers, <code>str()</code> converts the argument in to string. Etc.

<h3>Adding new functions</h3>

The aim of adding new functions is to have a DRY (don’t repeat yourself) code. It is so that you can reuse a sequence of statements. You define the function and then you can call it anywhere you want rather than writing the entire code all over again. This would lead to cleaner code and shorter lines of code.

When using functions however, you need to understand the <strong>flow of execution</strong> of your code. Code execution always starts from the beginning of the code i.e. from the first line and then it is run from top to bottom so for instance if you want to print a variable <code>fname = “Dumebi”</code>, you have to first assign that variable something before you can call <code>print()</code>. Your print statement cannot come before the variable is assigned.

When you define functions using the <code>def</code> keyword, the function is not being executed and as such, it does not alter the flow of execution. However, when you call a function, there is a detour in the flow of the code execution. Instead of the code execution to follow the normal top to bottom flow and go to the next line of code, it jumps to where the function is defined and runs the function (executes the statements) and then comes back to pick up where it left off. You can have function calls within functions.

<h3>Types of function</h3>

There are functions that <em>return a value</em> and there are functions that <em>don’t return values</em>. We can call them <strong>fruitful functions</strong> and <strong>void functions</strong>. When you have a fruitful function, you almost always want to do something with the return value for example assign the return value to a variable so you can make use of that return value later. For example

<img src=”/images/function\_add.JPG”>

When the function was called i.e. <code>add(3, 5)</code> it added the numbers and then the result was returned. It is that return value that is being assigned to the <code>new</code> variable as seen in the variable explorer just on the right hand side. If you print the “new” variable, you’d get back eight (8).

For void functions, you do not return a value so even if you try to assign it to a variable, you will get a special value called <code>None</code> if you try to print the variable you’re assigning it to.

<img src=”/images/function\_add.JPG”>