**INF 1511 – Revision notes – Chapter 4**

1. **Functions**

A **function** is a part of the code that is called by name (as defined in chapter 3), the code calling the function is called a function call. Refer back to chapter 3 for more features of a function.

When defining a function in Python, the following syntax is used:

def function-name(parameters):

statement(s)

where parameters are the values the function will receive from the function call and statements are the tasks the function will perform.

For example, to create a function that calculates interest on a given amount for a given period, the function would be defined as follows:

Def calc\_interest (principal\_amt, num\_months):

int\_rate = 11

Interest\_amt = (principal\_amt \* num\_months \* int\_rate)/100

return interest\_amt

Note that the principal\_amt and the num\_months in the function definition are called **parameters.**

The function call would be calc\_interest (10000, 24), note that 10000 the principal amount and 24 the period, the **arguments** can be variables instead of values.

In the function call, number of **arguments** must always equal number of **parameters**.

**Default value parameters** are optional parameters that do not have to be supplied in the arguments.

The calc\_interect function above can be defined with a default value parameter as shown below.

Def calc\_interest (principal\_amt, num\_months, int\_rate = 11):

Interest\_amt = (principal\_amt \* num\_months \* int\_rate)/100

return interest\_amt

Then the function call can be;

calc\_interest (1000, 24) in which case 11 will be used as the interest.

Or it can be calc\_interest (1000,24,12) in which case the interest will be 12.

In the above examples the order of the arguments must be the same as the order of the parameters. If **keyword parameters** are used, the order of arguments and parameters is no longer important. Keyword parameters can be used by defining **global parameters** as described on page 109 of the textbook. The parameters defined in the examples above are **local** to the function and can only be used within the function.

**Lambda functions** are single use functions that do can take any number of parameters, they do not have a name.

1. **Applying Functions to Sequences**

**For this module were are going to apply the following functions to sequences**

1. filter function – select elements from a sequence that satisfy the criteria defined in the function

filter (function,sequence)where the function specifies the criteria.

1. The map function – calls the given function for all elements in a sequence and returns a list of the returned values.

map (function,sequence)

1. The reduce function – calls the given function using the first and second element, then the result of the first call and the third element are used to make the next call and that result and the fourth element are used to make the next call until the end of the sequence.

reduce (function, sequence)

1. **Function Attributes**

Note the attributes of a function as described on page 115 of the textbook.

1. **Recursion**

Recursion occurs when a function calls itself. To ensure that the call does not result in an endless loop, an exit condition must be included in the function, through the use of a the return statement.

1. **Iterators**

Iterators are used to loop through data, an iter object is created as follows:

Iter (object)

The next () method of the object is then used to loop through elements in the original object.

1. **Generators**

A generator is a function that returns a value using the yield statement. The generator iterator can be called using the \_next\_ () method.

1. **Generator Expression**

Is created by enclosing an expression in parenthesis. The expression will contain either an if statement or a for loop.

e.g.(squarenum(x) for x in range(6))

The \_next\_() method can be used to loop through the elements in the object.

1. **Modules**

A module is a file containing several functions that can be imported into a program using the import statement.

A particular function in a module can be imported as follows:

from module import function

Or all the functions in a module can be imported using:

from module import \*

If a different name is required for the module in the current program it can be imported as follows:

import module as newname

1. **The math Module**

The math Module contains common trigonometric functions as well as mathematical constants. It is imported as follows:

import math

1. **The dir () function**

The dir () function is used to display the identifiers (classes, functions and variables) in a module.

1. **Command-line arguments**

Command-line arguments are used to pass arguments to a program during execution. They are stored in the sys.argv variable