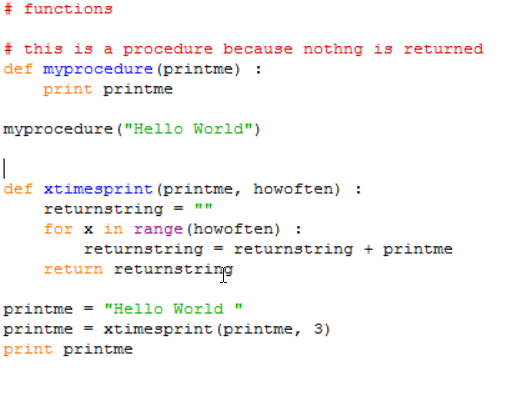
### Lesson 8 – Creating functions and using parameters

All programming language allow developers to create their own functions or procedures. These are sections of code which are written separately from the main program code, but possibly in the same file and used by the main program. They are generally a way of avoiding repetition and making the code more tidy and easier to read.

The only real difference between a function and a procedure is that a function returns a value and a procedure does not.

The following examples show the definition and use of some local functions (i.e. in the same code file as the main program)

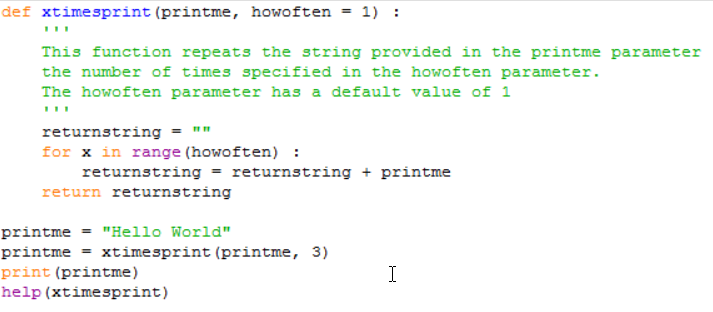


Points to note;

1. The definition of a function (or procedure) starts with the def keyword and is followed by the name of the function with any parameters used by the function in brackets.
2. The definition clause is terminated with a ‘:’ which causes indentation on the next and subsequent lines. All of these lines form the statements which make up the function. The function ends after the indentation is removed.
3. Within the function, the parameters behave just like variables whose initial values will be those that they were given when the function was called.
4. functions have a return statement which specifies the value to be returned. This is the value assigned to the variable on the left-hand side of the call to the function. (printme in the example above)
5. You call (run the code) a function simply by providing its name and values for its parameters just as you would for any builtin function.
6. In the call we pass a variable whose name is printme to the printme parameter, there is no connection between them, the passed variable or string literal could be anything.
7. If you do not specify the name of a parameter, it is assumed that you are providing them in the same order as they appear in the definition.
8. You can however explicitly name the parameter whose value you are passing in. The following would be a valid call : xtimesprint(howoften = 2, printme = printme)

The code is in the file functions.py in the code folder

Here is a modified version of the function



Points to note;

1. In this definition the howoften parameter has been given a default value of 1.
2. This means that if a value for howoften is not provided when the function is called, than the value of 1 will be assumed.
3. This definition also includes a docstring which describes the function. If you call help(xtimesprint) the system will display this text along with definition of the function.

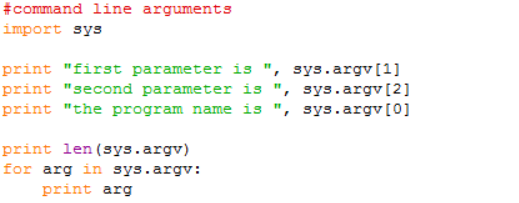
The code is in the file functions2.py in the code folder

### Using parameters in main program

If you want to run your program directly from the command line and don’t want to be prompting the user for input, you can provide values for variables directly on the commandline when you run the program.

You provide the parameters immediately after the program name separating the parameters with a space.

The code below gives an example of how the arguments can be retrieved from the command line call;



It can be called from the command line like this;

>commandlineargs.py "abc" "def"

Points to note;

1. In the call the parameters are separated from the program name and each other by spaces.
2. The arguments are returned a list of strings.
3. The program name is counted as an argument (sys.argv[0])
4. In order to access the arguments from within your program, you need to import the ‘sys’ package.

The code is in the file commandlineargs.py in the code folder.

### Lesson 9 Introduction to Object Oriented Programming

In Notebook & slides

### Lesson 10 Introduction to Pandas and Matplotlib

In Notebook & slides

### Lesson 11 Creating and running Programs

Instructions in Programming Tasks.docx