**HOW2**

**SLIDE 1:**

Why should you move away from excel?

Even though you are almost definitely used to excel, this does not mean it is the best tool for the job.

Its not built with actuaries in mind. A better approach would be something less general and more flexible – like coding what you need in python.

Python allows you the ability to write unit tests and be more certain your work is correct.

Python is more powerful than excel, and can be a lot faster. Its also more suited to dealing with larger amounts of data than excel.

Furthermore, more of your work can be automated with functions if you are using python, reducing the opportunity for erroneous formula entry and incorrect copy pastes, and making your job easier.

**SLIDE 2:**

This slide is an excerpt from a paper that explored the use of different software in the actuarial world. This further evidentiates the idea that excel is not the most suitable tool.

**SLIDE 3:**

**Parameter passing**

Parameter passing to functions in python is done by assignment, so changes made inside the function will not affect the value of the variable outside of that function’s scope (unless it is the return value and is assigned outside of it). This is made evident in our example by the fact that var1’s value was not changed after we called exampleFunction(), and we assigned var3 to the return value of the function (which was the changed var1).

**SLIDE 4:**

**Indentation**

A small note about python is that indentation is very important. If you want lines of code to happen “inside” of another line, then you would indent it. The most convenient way is by using the tab button to indent, but as long as you are consistent, the number of spaces you use is not important.

Some examples of indentation

**SLIDE 5:**

**Comments**

Comments in python are prefixed with a #. They are not executable and are purely there for explanation (it is important to aid in readability to sprinkle comments throughout your code). Multi-line comments are surrounded in three quotes (“””) like below

**SLIDE 6:**

**Variable naming conventions**

You are technically free to name your variables however you wish, but it’s likely that you will need to share your code with your co-workers, and you might have to revisit it yourself after you have lost your familiarity of it. In order to make your code as readable, and understandable as possible, you should have sensible variable names. The best practice is to choose a naming convention and stick to it. Some examples of this are:

camelCase (every word but the starting word is capitalised)

snake\_case (words are separated by an underscore)

Hungarian Notation (camelCase, but includes a prefix descriptor for the type of the variable, e.g.

sUnderwritingYear is a string that holds the Underwriting YEAR

These are just conventions though. The most important thing to consider is, (unless its clearly a throwaway variable) that the name of the variable gives some indication as to what it holds!