Lesson 3 Handout v0.1

1 Terminology

1.1 **flow-control** – Most programming languages possess language elements which allow for the branching of program execution. Selection of execution paths is often refered to as ‘flow-control’.

1.2 **boolean** – In programming, a boolean data type can either be ‘true’ or ‘false’.

1.3 **logical expression** – Logical expressions are made up of comparisons and logical operators. They will evaluate to either ‘true’ or ‘false’.

1.4 code-block or **suite** – Python is an indentation-aware programming language. This means that groups of lines of code that are related to each other are indented the same. Many other programming languages use the curly braces ({ }) to isolate code blocks.

1.5 nesting – When code blocks are placed inside other code blocks, this is sometimes called ‘nesting’. This can be seen in lesson 3 when an ‘if’ structure is placed within another ‘if’ structure.

1.6 constant – In programming syntax, a ‘constant’ is a variable or other data object which is assigned a value once, and only once, and then cannot be changed. Many programming languages support the constant, but Python does not offer this support out of the box. It is a common convention in programming to name a constant using all capital letters and to seperate individula words which make up the variable name with the underscore character.

2 Programming Elements

2.1 Operators:

2.1.1 **==** - Used to check if two data are equal or equivalent. Normally one might expect a single eaul sign should be used for this purpose. However, that is already in use for the assignment operator.

2.1.2 **>, >=, <, <=** - Greater-Than, Greater-Than-or-Equal, Less-Than, Less-Than-or-Equal, respectively. These are the main comparison operators used in logical expressions.

2.1.3 **and** – The ‘and’ logical operator. When placed between two logical data or logical expressions it will result in true if, and only if, both are true. Otherwise it will produce a false result.

2.1.4 **or** – The ‘or’ logical operator. When placed between two logical data or logical expressions it will result in true if either are true. It will only result in flase if both are false.

2.1.5 **not** – The logical ‘not’ operator is placed before a logical data or expression. It has the effect of reversing the logical value of whatever it acts upon. True becomes false, and false becomes true.

2.1.6 **!=** - The ‘not equal to’ operator.

2.1.7 **:** – The colon is used to end a line which immediately precedes a indented code-block. The Python documentation sometimes refers to a group of lines of code as a ‘suite’.

2.1.8 **# -** The number sign, or hash-tag as some people know it as, is used to place comments within a Python script. The ‘#’ must come before the comment text, which is largely ignored by Python.

2.2 Python Keywords:

2.2.1 **True** – The Python keyword for the logical value ‘true’.

2.2.2 **False** – The Python keyword for the logical value ‘false’.

2.2.3 **None** – The Python keyword for the state of ‘nothing’. When a variable is set to this it is, essentially, empty.

2.2.4 **if, elif, else** – The three types of branches available for building a ‘if’ type conditional structure in Python. The structure must start with an ‘if’ clause and contain a logical expression. It may then contain zero or more ‘elif’ clauses – which must also have a logical expression. It may also, optionally, contain one ‘else’ clause – which should not have any expression. Each clause should end in a colon ‘:’. As the ‘if’ structure is evaluated one, and only one, clause (or branch) will be selected for execution. The other clauses (or branches) will be ignored.

2.3 Python Functions:

This section intentionally left blank.