Lesson 4 Handout v0.1

1 Terminology

1.1 **loop** – Any code that is intended to execute repetitively. Each pass through the loop is referred to an iteration.

1.2 for loop – A type of loop that executes once for each element or item in a sequence. Some programmin language have several different types of for-loops. Python has only the one type, but allows the use of the range() function to allow fo the behavior of a loop that is base purely on some kind of counter. For loops are most often used when the number of iterations the loop will make is known immediately prior to its execution.

1.3 while loop – A loop that executes as long as some expression is still true. While-loops are most often used when the number of iterations is not certain immediately prior to its execution.

1.4 lists – A type of sequential data structure used heavily in programming. Most programming languages have support available for lists, but do not include them as a native data type. Python is an exeption in that it has direct support for lists as a native data type. Strangely, Python does not support the 'array' data type natively and you must make special provisions to use them in a Python program.

1.5 format – The act of adjusting something to meet certain requirements. In particular, strings/text can be 'formatted' to ensure they appear in a very precise configuration. Many programming languages have formatting facilities available either as native functions or as easily accessible libraries.

1.6 target-variable – The variable specified by a for-loop statement to hold the current element to be used for any particular iteration of the loop.

1.7 iteration, iterating – When code execution passes repeatedly through a loop, each pass is considered an iteration. While execution is passing repeatedly through the loop, it is said to be iterating.

1.8 iterable – (*This topic was not discussed directly in lesson 4)* A iterable is a special type of programming element that is capable of being executed multiple times to generate a sequence in a peice-by-peice fashion. Python's range() function is an iterable.

1.9 string-specifier – Many programming languages have formatting capabilities. They often make use of a string of text which specifies what should be done to format soemthing.

1.10 **mini-language** – Many programming languages have advanced facilities available to the programmer. Sometimes these advanced facilities necessitate the creation of a kind of mini-lanaguge to help specify the details of what these facilities should do. A good example is the rich set of symbolgy available in Python's format function that is used to communicate how to format text.

1.11 multi-line mode – The Python interactive console can execute multi-line pieces of code. This is accomplished by starting the first line off with a statement that ends in a colon, which means that more

lines must follow. Once the final line is entered, the user then hits the ENTER key on a blank line to cause execution to begin.

2 Programming Elements

2.1 Operators:

2.1.1 **:** – 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'.

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

2.1.3 *list_variable***[***index***]** – List variable make us of brackets and an index number to specify a specific element of the list.

2.1.4 **+=** - This combines a plus sign and equal sign into the assignment by addition operator. This assignment operator adds whatever is on the right-hand side to the variable on the left side and then assigns the results to the variable on the left.

2.1.5 **-=** - This combines a minus sign and equal sign into the assignment by subtraction operator. This assignment operator subtracts whatever is on the right-hand side from the variable on the left side and then assigns the results to the variable on the left.

2.2 Python Keywords:

2.2.1 **for** *target_variable* **in** *sequence***:** - The for loop statement begins a executable loop. The code will execute once for each element within *sequence*. The *target_variable* will ge used to reference the current element during each iteration of the loop.

2.2.2 **while** *expression***:** - The while loop statement begins the code for an executable loop. It will continue to execute for as long as the expression evaluates as 'True'.

2.2.3 **continue** – The 'continue' keyword cause execution to bypass the rest of a loop for the current iteration.

2.2.4 **break** – The 'break' keyword causes execution of a loop to end and continue with the code that follows the loop. It is used to 'break-out' of a loop.

2.3 Python Functions:

2.3.1 **len(***list_variable***)** – A function which will retunr the length of a data item. In the case of a list, it will return the number of elements currently contained within the list.

2.3.2 **format(***data***,** *string_specifier***)** – This function will format *data* in accordance with *string_specifier* and return the formatted string.

2.3.3 **range(***end_of_range_exclusive***)** – This function is an iterable which can be used for , maong other things, to cause a for loop to execute a certain number of times. The sequence of numbers that

it generates when used like this will start with 0 and end with the number that comes directly before *end_of_range_exclusive.* (The term ‘exclusive’ is used in mathematics to specify that a number is not part of a set of numbers.)

2.3.4 *list_variable***.append(***data_item***)** – This built-in function is used to add *data_item* to the end of a list specified by *list_variable.*

2.3.5 *list_variable*.**pop(***[index]***)** – This built-in function is used to remove an element from *list_variable* and return it. If *index* is sent as an argument, then the element at that position is removed and returned. If no index is sent then the very last element is ‘popped’.

2.3.6 **del(***variable_reference***)** – Deletes a variable or element(s) of a list. The variable/elemnt to delete is specified as the *variable_reference* argument.