Lesson 5 Handout v0.1

1 Terminology

1.1 **function** – Functions, in programming, are blocks of code which are associated with a function name and are able to return one or more values or references using a return statement.

1.2 **function definition** – The entire definition of a function includes its definition statement and the code that will be executed.

1.3 function definition statement - The function definition statement defines the name and other characteristics of a function.

1.4 module – In Python any file containing definitions and statements is technically a “module”.

1.5 return statement – A statement that starts with the “return” keyword and contains one or more data to be returned to the function call. When a value is returned the function call is treated as if it were “equal” to the returned value(s).

1.6 import – Other modules may be imported into a Python script file/module.

1.7 docstrings – A useful feature of the Python language. A docstring is the first string literal that follows the beginning of a module, function definition, or class definition. These docstrings are associated with the defined items and stand as language supported documentation.

1.8 time module – The Python standard library of modules includes a “time” module which supports access to many time-related functions.

1.9 standard library – Most programming languages have a standard set of libraries which cover a wide range of commonly needed functions.

1.10 **application** – There are many definitions for “application” within Computer Science. As a noun, “application” can mean the set of files that make up a program which serves a specific function directly accessed by a human user.

1.11 orphaned strings – When a string literal definition is not associated with an expression or asignemnt statement it can referred to as an “orphan”. The Python language supports using strings literals as commentary in that it does not throw an error when they are encountered. Also, see “docstrings” above.

1.12 namespace and **scope** – Namespace and scope cover the mechanism by which a label used for a variable, module, or function can be reused for different purposes within different contexts of a program. The details of this are beyond the scope of this lesson, but they wil be covered in supplemental material.

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’.

2.2 Python Keywords:

2.2.1 **def** *function_name* **(***optional_argument(s)***):** – A function definition statement. *optional_arguments* can include both the local names to assign arguments sent to the function, as well as assignments to use when arguments are not sent.

2.2.2 **return** *optional_return_value(s)* – The return keyword is used to terminate the execution of a function and return any value to the function call.

2.2.3 **import** *module_name* – Used to import the functions and other features from an external module. This supports reusability and modularity across multiple source files.

2.2.4 **from** *module_name* **import** *function_name* – This is used to import one or more function from an external module. This method causes the imported functions to be available by referencing them *without* the module name.

2.3 Python Functions:

2.3.1 *string*.**isdigit()** – A function that is part of Python’s built-in string library. It returns ‘True’ for strings which contain digits, and only digits. Otherwise it returns a ‘False’.

2.3.2 **time.time()** - When used in this form time.time() returns a floating point value that is the number of seconds that has elapsed since some specific date/time. The exact date/time used can vary from system to system but this is generally not an issue since it is normally used for differential calculations for time measurement in real-time.