Exercise 22 – What do you Know So Far?

Commands, Symbols, and Functions		
Code	**Type**	**Description**
`print`	basic command	Followed by the word print, the user encloses a string in inverted commas 'like so', or using speech marks "like so". The print command will print out on screen everything within the quotations as a string on screen.
#	Comment out	Anything written after the hash tag (without a deliberate line break) is ignored in the source code. This is a great way of narrating through each part of a script. It aids in following complex code, and makes bugs easier to find.
+	Arithmetic/Maths	Plus.
-	Arithmetic/Maths	Minus.
*	Arithmetic/Maths	Multiply.
/	Arithmetic/Maths	Divide.
%	Arithmetic/Maths	Reports the remainder or a division.
<	Boolean	Less than.
>	Boolean	Greater than.
<=	Boolean	Less than or equal to.
>=	Boolean	Greater than or equal to.
`# -*- coding: utf-8 -*-`	ASCII hack	When placed at the top of a script, this allows for the use of non-ASCII characters. This is useful for non-English speakers using alternative alphabets.
%d	Formatter (or variable placeholder)	When writing a string of text, this placeholder can be written as part of the sentence. Outside the string, a % symbol is then used to call up the particular numerical variable of interest. See **Exercise 5** for a first example.
%s	Formatter (or variable placeholder)	As above but for calling up a string variable.
%r	Formatter (or variable placeholder)	As above but good for first draft of a script or during bug fixing. This placeholder says "print this variable no matter what it is".
\n	Escape sequence	Moves to a new line when placed in a string. See **Exercise 9** for an example.
\t	Escape sequence	Similar to \n but used for tabbing the text rather than taking a new line.
""" (i.e. 3 x ")	Escape sequence/documentation quotes.	Within triple quotes, text can be written as desired (i.e. with line breaks, etc.). When the command 'help' is called for a given function, the triple quoted words WITHIN the

		triple quotes will be printed on screen as the help.
\\	Escape sequence	Gives a single backslash in a string.
\’	Escape sequence	Gives a single apostrophe in a string.
\”	Escape sequence	Gives a single speech mark in a string. Note – these last two commands are good for dealing with strings that require quotes within quotes.

Commands, Symbols, and Functions

Code	Type	Description
\f	Escape sequence	ACSII formfeed. When this is used as part of a string, it takes a new line and tabs to being in line with the END of the previous line of text.
\v	Escape sequence	Vertical tab. **NOTE** – other, more specialised escape sequences are detailed in **Exercise 10**.
`raw_input()`	Input command	Reads input from the user as a string.
`from`	Module command	In a Python script, one can import modules (libraries of other code). Sometimes, only part of a given module is required. The command ‘from’ is written immediately before the module name. The command ‘import’ immediately follows the module name. See next table entry.
`import`	Module command	This command immediately precedes a module or sub-module name to be brought into the python script.
`as`	Module command	Long module names can be shortened using the ‘as’ command. For instance, the module NumPy can be imported
`sys`	Module	This contains objects such as ‘argv’ and ‘path’. These can be described as argument variables. See below for individual explanations.
`argv`	Module object	A list of variables (beginning with ‘script’) can be attached to argv. The script name and the following variables are typed on the command line when running the script. In the source code, the variables attached to argv are called in order.
`open(filename, ‘r’)`	File command	Opens a file that is attached to the variable **filename**. The file itself is probably called up at the command line when running the script, as implemented using the **argv** module object. The ‘r’ means open in read only format.

		See **Exercise 15** for an example.
`open(`**`filename`**`, 'w')`	File command	Open and write to this file. **NOTE** – this overwrites existing file contents!
`open(`**`filename`**`, 'a')`	File command	Open file such that it can be appended to.
`+`	Modifier	Add to r, w, or a to open file in read and write mode.
`variable``.read()`	File command	Reads the opened file which is connected to the variable, **variable**. This command can be attached to the **print** command to print the read file on screen.NOTE – the variable name can be any string as long as it's consistent throughout the code.
`variable``.truncate()`	File command	This command empties the opened file. Be careful!
`variable``.close()`	File command	Operates like 'save and close'. Good to have after you're finished with file.

Commands, Symbols, and Functions

Code	Type	Description
`os.path`	Module	Deals with file paths.
`exists`	Module object	Written in the syntax exists(**variable**), it returns 'True' or 'False' as to whether the file attached to **variable** exists or not.
`len(`**`variable`**`)`	File command	For an opened file attached to **variable**, this command reports the number of bytes in the file.
`def` **`function_name`**`(argumnets):`	Function	'def' stands for 'define' and is used to attach a section of code to a shorthand name and one, some, or no arguments. See **Exercise 18** for a clear example. **NOTE**: the colon is used in conjunction with indented code below to show what is attached to the function.
`*args`	Argument variable	Like **argv** but used for functions. See **Exercise 18**.
`int(`**`variable`**`)`	Command	Turn the string inside **variable** into an integer.
`float(`**`variable`**`)`	Command	Turn the string inside **variable** into an floating point number.
`variable``.readline()`	File command	If no number is inside brackets, reads first line of file, then moves placeholder to next line.
`variable``.seek(0)`	File command	Moves back to the start of the file.
`return`	Command	Used within defined functions. This returns values from the temporary variables used within the function.
`help(`**`function`**`)`	Command	Prints the triple quoted words within the function as a means of help in understanding the function.
`list_variable``.pop(x)`	List command	From a separated list of terms, the x^{th} term (starting from zero) will be stored and subsequently removed (or 'popped') from the list. The

		stored list value can then be printed in some way.
list_variable.sorted()	List command	Sorts the given list in alphanumerical order.
variable.split()	List command	Can be used to create a separated list of terms from a given string variable.
getopt	Module	
chmod + **script.py**	script modifier	makes script executable with typing 'python' before the name.