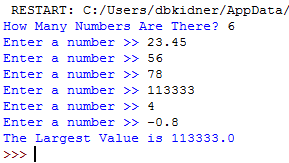
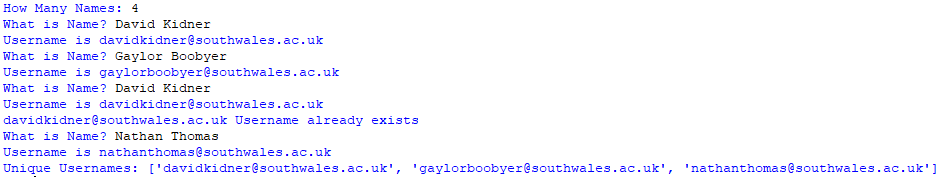
**PYTHON Programming Exercises 4**

The solutions to the Exercises in the last tutorial (Python Programming Exercises 3) are now up on Blackboard. Take a look at these and make sure you understand the logic behind the solutions. In most cases, you can probably identify where the code can be improved? (e.g. use of main() or better print formatting).

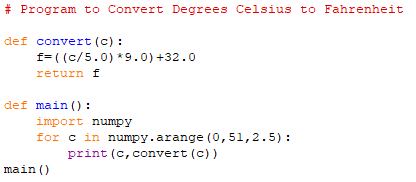
**EXERCISE 1:** To dust off the cobwebs, let’s start by re-considering the code (and solutions) for finding the largest of 3 numbers. Extend one of the program solutions to consider any number (defined by the user) of inputs to find the largest. So, the input and output might look like:



**EXERCISE 2:** Write a program to read (from the keyboard) a user-defined number of first names and last names (e.g. 4, 5, 6, 7, etc.) including spaces, e.g. “David Kidner”, “Gaylor Boobyer”, etc. These names will be used (within a function) to generate computer usernames (for output to the screen), which will be in lower case with spaces removed and ending in “@southwales.ac.uk”. Any duplicate names which are input will be identified and an appropriate message output. For example, the input and output might look like (with a summary of all unique usernames):



**EXERCISE 3:** Using the code in the lecture as a starting point, re-write the code to generate a series of Fahrenheit to Celsius conversions at (a) 1 Degree Intervals and (b) 0.5 Degree Intervals between 0 and 100 Degrees Fahrenheit.



**Task 1: Reading a Text File**

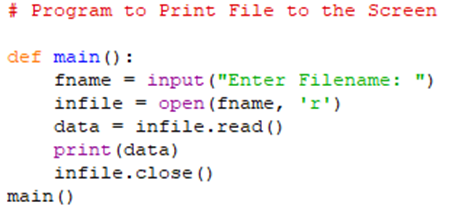
OK, let’s get started with reading some text files. Open up a text editor (i.e. Notepad) and type the following 4 lines of sample text and save as ***test.txt*** in your workspace (or USB drive). Create the python program to read this text file as one data object:

abcd

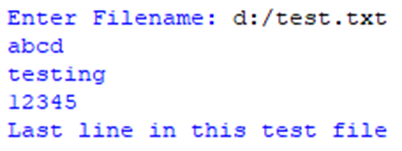
testing

12345

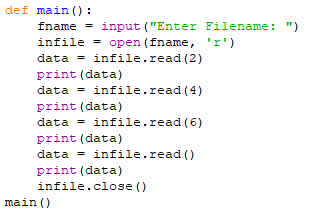
Last line in this test file



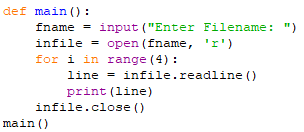
The filename should include the full path to your test.txt file.

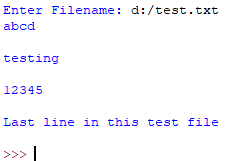


All the data is output as one object. By default the read() method returns the whole text, but you can also specify how many characters you want to return. Try the following:



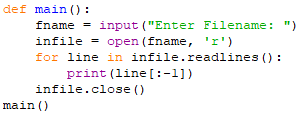
Now, amend the code to read and print the file line by line:





The output has extra lines between our data, due to the way that the end of line / carriage returns are interpreted. We could strip these out. Amend the print statement above to print(line[:-1])

Alternatively, we can use the readlines function, which returns a list of the remaining lines in the file. Each list item is a single line including the newline characters:



**Task 2: Reading Another Text File**

In this exercise we are going to download a text file from the internet, open it and read it. On Blackboard, locate the file poem.txt (in Python folder) and save it to your workspace. Open it up in Notepad to look at it. It is a poem entitled “Cimmeria” by Robert E. Howard, the homeland of Conan the Barbarian, although sounds like Treforest.

Make a note of the directory in which you have saved the file. In my case, I have saved it to my workspace (which is:



In the Python IDLE, at the command line, define YOUR directory path as the string variable ***path***, so the above path is defined as:

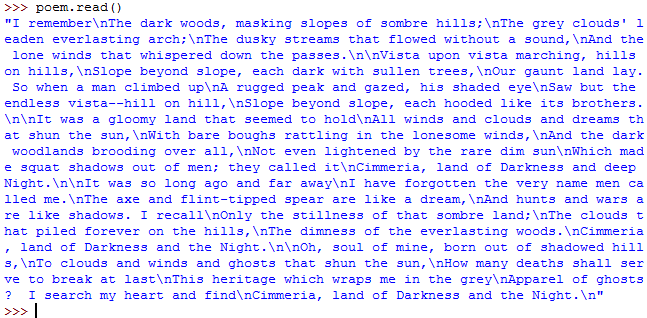


NOTE that the \\ and \ get reversed. Give your tutor a shout if you can’t find the pathname. If you download the file to your local workspace (e.g. c:/users/temp/30xxxxxx) where 30xxxxxx is your enrolment number, the path is simply “c:/users/temp/30xxxxxx”.

To open the file, we can concatenate the path with the file name.



Let’s read the whole file:

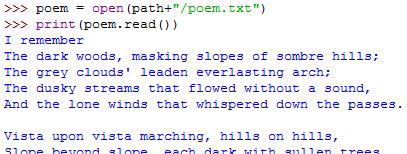


Eh??? Let’s try that again:

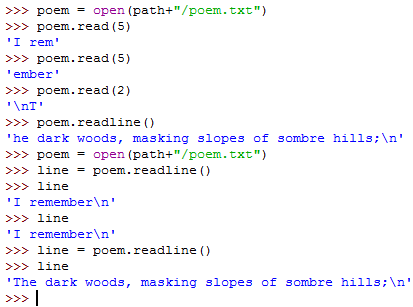


Double Eh??? What happened there?

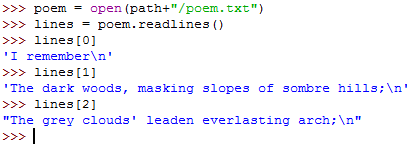
In the first “read”, the whole file was read, including the line breaks (\n = new line), so gets read completely as one big block of data. Upon the second “read” statement, as the data was already read, there’s nothing more to see. Let’s start again by recreating the file and printing as readable text:



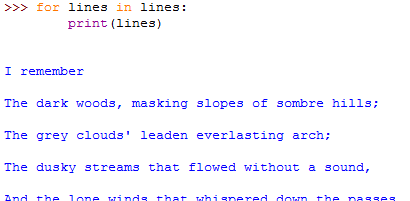
Practice the read commands as before by typing the following (and understanding what is happening):



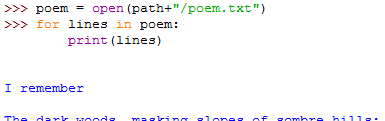
Similarly, we can use the **readlines()** function to grab all the lines of the text and store them as multiple lists. These can then be stored as a variable:



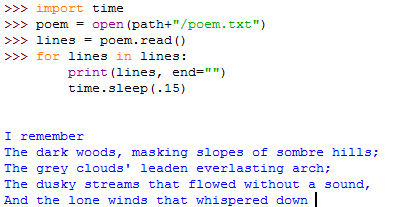
Which lends itself well to being output in a loop:



Or:



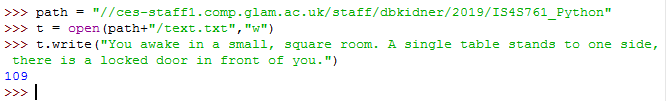
Now, let’s imagine that you want to print the text one character at a time (like the Final Score Vidiprinter, or an old dot-matrix printer). We can make use of a time delay. Try this:



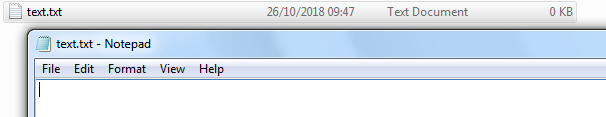
The time module has some very useful functions which we can incorporate into our programs. Check out or Google the time module to see what it can offer us.

**Task 3: Writing To A Text File**

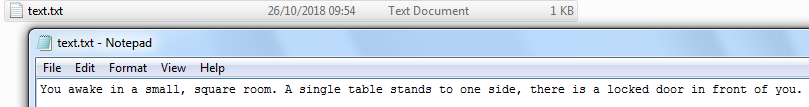
In the Python IDLE, set up the path as before and open a text file entitled “text.txt” in that location with “write” or “w” permissions. Try this:



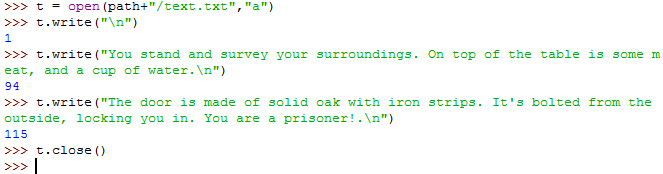
Note that the “109” is the number of characters entered. However, the actual file is still blank (check it out by opening it up in Notepad).



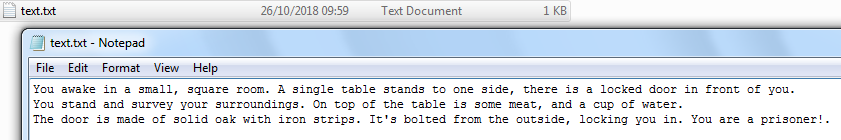
This is because we’ve written the line of text to the file object but not committed it to the file itself. We can’t do this until we close the file:



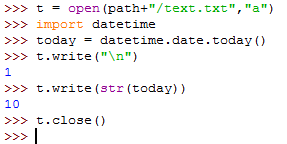
The line of text is now there. To expand this code, we can re-open the file using the “a” (append) mode. This will add any text at the end of the original line instead of wiping the file and creating a new one. For example:



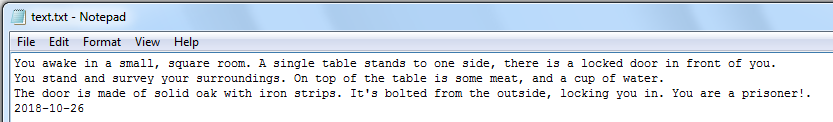
Check out the file:



As well as text, we can output or pass variables to a file we’ve created in Python. For example, we might want to add the date or time to our file.



And check out the file:



Have a go at adding the exact time as well. Use the knowledge of the ***time*** module to help you!

**EXERCISE 4:** Amend your code from EXERCISE 2 to read a text file (people.txt) of 10 names and output a new text file (usernames.txt) of unique usernames. Produce some screen output to inform the user whether a username is NOT unique.