EMAT10007 – Introduction to Computer Programming

Exercises – Week 6: File I/O and program arguments

Part 1. File I/O

File input and output (I/O) is an important part of any program, as it allows you to load data into your program and to export data from your program.

Exercise 1 - File input and output

The goal of this exercise is to write a program that reads in a high-score list and then asks the user to enter a new score. Your program should add this score to the list and then reorganise the list such that the scores are in descending order. Then, your program should save the new high-score list to the same file.

1. Start by creating a program which reads a list of high-scores from a file called `HighScoreList.txt`. Print out the list of high-scores in the following format:
   `Name:  Jim Score:  108`
2. Then ask the user to `input` a new name and score (try to accept both arguments in one line).
3. Place the new user and their score in the high-score list, do one of the following two actions:
   1. If the new score is the lowest score in the high-score list, then simply **append** it to the `HighScoreList.txt` file.
   2. If the new score is not the lowest in the list, then sort the list of high-scores into descending order, so that it is once again correctly ordered. Then **overwrite** the old high-score list with the new high-score list.
      That way, when you run your program multiple times, your program will always present you with the most up-to-date high-score list.

   If the user does not enter anything (`input(...)  == ""`) then exit the program.

Part 2. Command line

Exercise 2 - Program parameters (a.k.a command-line arguments)

Using the same

1. Modify the program above to accept program `parameters` of the following format:
   `-a [`*`length`*`]`
   `-d [`*`length`*`]`
   where `-a` here is short for "ascending" and `-d` for "descending".
2. You should then sort your list depending on **ascending** or **descending** order. The second arguments in the above examples are **optional**, and should be integer values which specify the *maximum number of scores to keep in the list.*
   If there are fewer scores than `[`*`length`*`]`, then you should do nothing. Otherwise, you should remove scores from the bottom until your high-score list contains `[`*`length`*`]` high-scores.
   Note: You should only accept sensible values of these arguments, e.g., `[`*`length`*`]` should not be negative, or 0. Any other argument values should be ignored.