## OCR A453 Programming Project

## Controlled Assessment Material 2

### Notes For Task 3

* Ask the user for the class name. In the solution I have fixed this to 4TC for testing purposes. This is the text file produced in Task 2.
* Readfile() - Read in ALL names and scores for that class. (Now stored in results)
* List names in file() - Go through all the results and create a dictionary of scores with the student’s name as the key. As dictionaries do not allow duplicate keys, at the end of the process the keys that you have is a list of pupil names without duplicates. The dictionary itself isn’t used for anything other than to provide a unique list of student names.
* Strip old results() - For each ‘key name’, go through the results list from back to front (the newest scores are at the end of the file). Keep a count each time a result is found for the student being searched for. After a 3rd result has been found for the student, delete any further ones found from the results list. The list ‘results’ now contains a maximum of 3 results for each student.
* High score and average() – For each ‘key name’ find that pupil’s highest score and calculate their average score. Add the name and highest score to the GLOBAL list ‘high\_score\_list’ and add the name and average score to another GLOBAL list ‘ave\_score\_list’.
* Print the information in the format required by the class teacher.
	+ Alpha highest score high to low() – Sort the ‘high score list’ by name, alphabetically. Print suitable headings and then print the list.
	+ Highest score high to low() - Sort the ‘high score list’ by high score. Print suitable headings and then print the list in reverse order.
	+ Average score high to low() - Sort the ‘ave score list’ by average score. Print suitable headings and then print the list in reverse order.

#-------------------------------------------------------------------------------

# Name: A453 Programming Project Sept 2014

# CA Material 2 - Task 3

# Purpose: Simple Maths quiz

#

# Author: Andrew Baker

#

# Created: 24/02/2015

# Copyright: (c) Andrew Baker 2015

# Licence: <your licence>

#-------------------------------------------------------------------------------#----------------------------------------

def readfile(classname):

 filename=classname+".txt"

 file = open(filename,"a+")

 file.seek(0) # to make sure the pointer is at the beginning

 results=file.readlines()

 file.close()

 return results

#----------------------------------------

def list\_names\_in\_file():

 # Get names in class who have results

 for x in results:

 (name,score)=x.split(":") #takes every result and splits the name and score into the variables name and score

 res\_dict[name]=score # name is the key in the dictionary res\_dict[]

 # keys are unique. If it reads in a result for the same person, the score is overwritten

 # the net result is the keys provide a list of names, without duplication

#----------------------------------------

def strip\_old\_results():

 for i in res\_dict.keys(): #go through all names

 count=0

 for y in reversed(results): #work backwards through results file

 (name,score)=y.split(":") #takes every result and splits the name and score into the variables name and score

 if (i)==name: # compares the key name with the name read in from the results file

 count +=1 # keeps a count of how many results it has read in (working from the back of the file) for the key\_name

 if count>3: # delete from results if the key\_name has more than 3 results

 results.remove(y)

#results now contains a maximum of 3 results for each pupil. They are their latest scores

#----------------------------------------

def high\_score\_and\_average():

 for i in res\_dict: #go through all key\_names

 high\_score=0

 total=0

 count=0

# for each key\_name find their score in the results

 for x in results: # go through every result

 (name,score)=x.split(":") #takes every result and splits the name and score into the variables name and score

 if (i)==name: # compares the key name with the name read in from the results file

 count +=1 #keeps a count of the number of scores for that student

 total=total+int(score) #this is a running total of the scores that have been read in for that student

 if (int(score)>int(high\_score)): # if the score read in is great than the highest score so far for the student, make this the new high score

 high\_score=int(score)

 ave=total/count #calulate the average score for the student

 high\_score\_list.append((i,high\_score)) #add this student's name and highest score to the GLOBAL high\_score list

 ave\_score\_list.append((i,ave)) #add this student's name and average score to the GLOBAL ave\_score list

#----------------------------------------

def alpha\_highest\_score\_high\_to\_low():

 high\_score\_list.sort(key=lambda s: s[0]) #sorts the GLOBAL high score list using the student name as the key

 print() #prints a blank line

 print("Student's Highest Score") #prints a heading

 print("Name Score") #prints a heading

 #prints the list

 for name, score in (high\_score\_list):

 print("{} {}".format(name,score))

#----------------------------------------

def highest\_score\_high\_to\_low():

 high\_score\_list.sort(key=lambda s: s[1]) #sorts the GLOBAL high score list using the highest score as the key

 print() #prints a blank line

 print("Highest Score") #prints a heading

 print("Score Name") #prints a heading

 #prints the list

 for name, score in reversed(high\_score\_list):

 print(" {} {}".format(score,name))

#----------------------------------------

def average\_score\_high\_to\_low():

 ave\_score\_list.sort(key=lambda s: s[1]) #sorts the GLOBAL average score list using the average score as the key

 print() #prints a blank line

 print("Average Score") #prints a heading

 print("Ave Score Name") #prints a heading

 #prints the list

 for name, ave in reversed(ave\_score\_list):

 print(" {:.2f} {}".format(ave,name))

#----------------------------------------

classname="4TC" #input("Which class do you want to see?")

results=readfile(classname) #Read in ALL the results for the class

#Define Globally, Dictionary and Lists

res\_dict={}

high\_score\_list=[]

ave\_score\_list=[]

#Analyse Data

list\_names\_in\_file()

strip\_old\_results()

high\_score\_and\_average()

#Outputs

alpha\_highest\_score\_high\_to\_low()

highest\_score\_high\_to\_low()

average\_score\_high\_to\_low()