1010110101010101101101101001100101010101101000101010010100010001101010101010010101010101010101010010101010101010111000101010101111010100101010101010101010101010100010101010101010101010101000111010101010101001010101010101001111001101001010101011101010101110010101010101010100101010101010100101001010001110101001101010101010100111100101111001100101001100001110000110010110101111001010101101000010101010001100100101010111010010011011001011001010101010100101010101010101010101101010001010011001010qwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnm

|  |
| --- |
| A453: Programming Project  Controlled Assessment Material 1  10/21/2015  11braniffn106  Sandringham School, Centre 17535 |

Table of Contents

[Task 1: To test the basic arithmetic skills 4](#_Toc445674222)

[Task Overview 4](#_Toc445674223)

[Requirements 4](#_Toc445674224)

[Analysis of Requirements 4](#_Toc445674225)

[Success Criteria 4](#_Toc445674226)

[Data Dictionary and Validation 5](#_Toc445674227)

[Flowchart 6](#_Toc445674228)

[Pseudocode Algorithm 7](#_Toc445674229)

[Annotated Code Listings 8](#_Toc445674230)

[Version 1 8](#_Toc445674231)

[Version 2 10](#_Toc445674232)

[Version 3 12](#_Toc445674233)

[Test plan 14](#_Toc445674234)

[Task 1 Evaluation 15](#_Toc445674235)

[Did I reach all my success criteria’s? 15](#_Toc445674236)

[Table 1 – Program results from sample run 17](#_Toc445674237)

[Task 2: Record and store the data for three separate classes of students using the arithmetic quiz 18](#_Toc445674238)

[Task Overview 18](#_Toc445674239)

[Requirements 18](#_Toc445674240)

[Analysis of Requirements 18](#_Toc445674241)

[Success Criteria 18](#_Toc445674242)

[Dictionary and Validation 19](#_Toc445674243)

[Flowchart 20](#_Toc445674244)

[Pseudocode Algorithm 21](#_Toc445674245)

[Annotated Code Listings 21](#_Toc445674246)

[Version 1 21](#_Toc445674247)

[Version 2 23](#_Toc445674248)

[Version 3 25](#_Toc445674249)

[Version 4 27](#_Toc445674250)

[Test plan 29](#_Toc445674251)

[Task 2 Evaluation 30](#_Toc445674252)

[Did I reach all my success criteria’s? 30](#_Toc445674253)

[Task 3: Outputting the results of the quiz for a particular class in specific ways 31](#_Toc445674254)

[Task Overview 31](#_Toc445674255)

[Requirements 31](#_Toc445674256)

[Analysis of Requirements 31](#_Toc445674257)

[Success Criteria 32](#_Toc445674258)

[Data Dictionary and Validation 32](#_Toc445674259)

[Flowchart 34](#_Toc445674260)

[Pseudocode Algorithm 35](#_Toc445674261)

[Annotated Code Listings 36](#_Toc445674262)

[Version 1 36](#_Toc445674263)

[Version 2 42](#_Toc445674264)

[Version 3 45](#_Toc445674265)

[Test plan 48](#_Toc445674266)

[Task 3 Evaluation 50](#_Toc445674267)

[Did I reach all my success criteria’s? 50](#_Toc445674268)

# Task 1: To test the basic arithmetic skills

## Task Overview

### Requirements

A primary school teacher wants a computer program to test the basic arithmetic skills of her students. The program should generate a quiz consisting of a series of random questions, using in each case any two numbers and addition, subtraction and multiplication. The system should ask the student’s name, then ask 10 questions, output if the answer to each question is correct or not and produce a final score out of 10.

### Analysis of Requirements

The task is to create a program to test the arithmetic skills of a primary school class. The program will create random questions made out of two numbers and either add, subtract or multiply them. As the requirement is to test basic arithmetic skills, I have decided to base the program on what we would expect year 3 to year 4 children to know. Therefore the program will never ask a question where the answer is less than 0. It will also only deal with numbers from 0 to 12 as I would not expect 7 and 8 year olds to multiply numbers greater than 12.

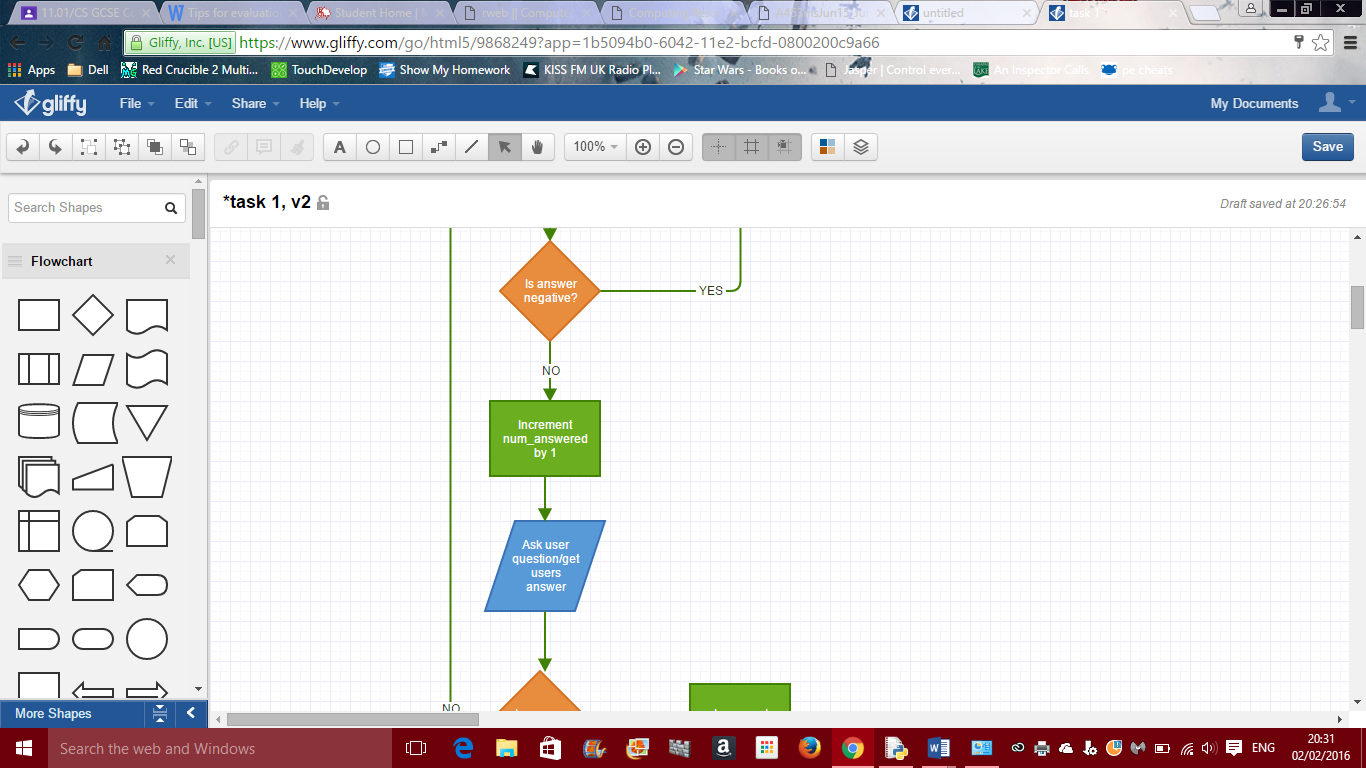
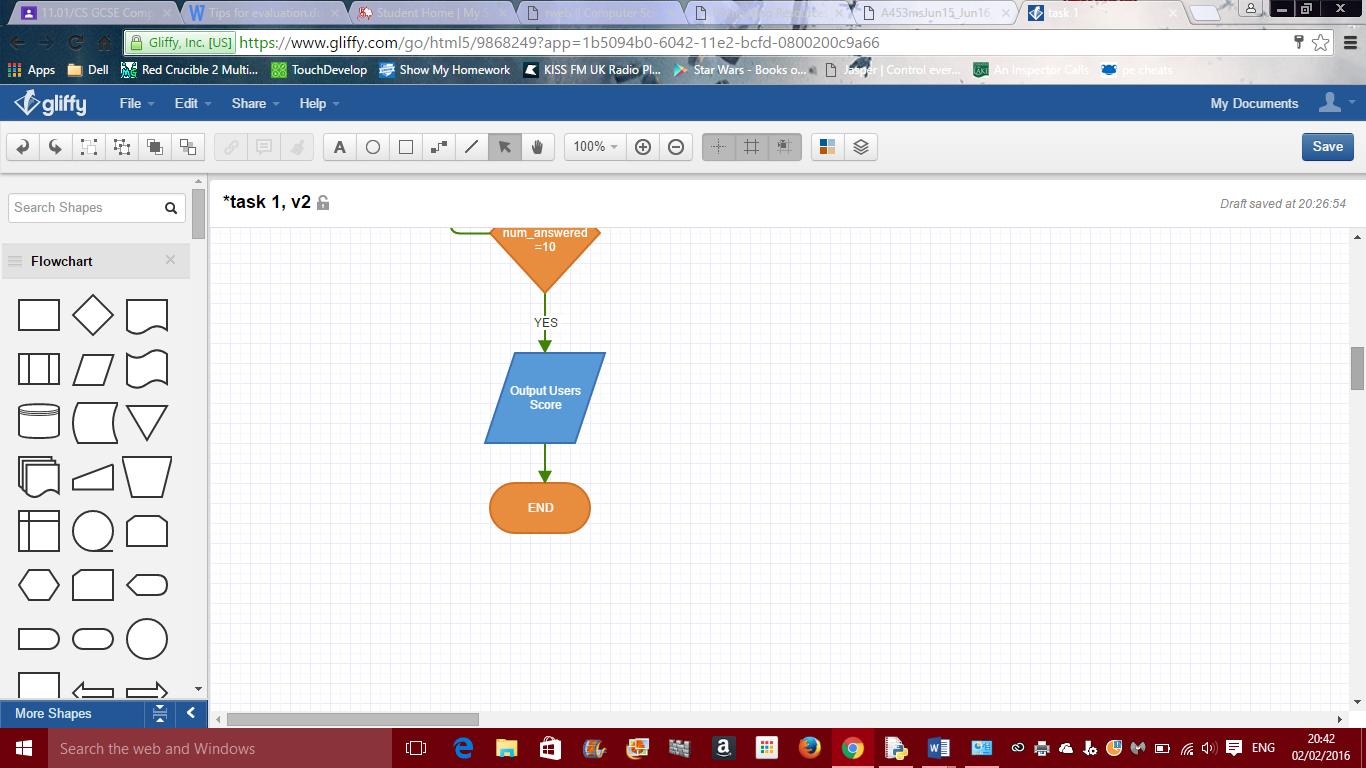
## Success Criteria

* Asks user for their name
* User’s name will be validated, checking that it is no more than 50 characters and that it does not contain numbers.
* Randomly chooses two numbers between 0 and 12.
* Questions with subtraction must not result in a negative number.
* Only whole numbers will be used (i.e. No fractions)
* Randomly selects one operator from +, \*, -
* Creates a question from selected numbers and the operator
* Presents the user with the question in a logical format
* Accepts and validates the input from the user
* Calculates the correct answer for the question
* Compares users answer to the correct answer and displays a message informing them if they are correct or wrong.
* Asks ten questions only
* Shows score out of 10

## Data Dictionary and Validation

|  |  |  |  |
| --- | --- | --- | --- |
| Variable Name | Description | Type | Validation |
| user\_answer | Answer provided by student | Integer | Must be a number. When a non-integer is entered an exception is thrown and the user will be asked to retry. |
| num\_answered | The number of questions answered. This is initialised to 0 and incremented every time a question is asked. | Integer | N |
| Score | The number of correct answers. This is initialised to 0 and incremented each time the player types in a correct answer. | Integer | N |
| high\_number | This contains the biggest number the question can use | Constant Integer | N |
| low\_number | This contains the smallest number the question can use | Constant Integer | N |
| computer\_answer | This contains the correct answer for the question | Integer | N |
| username | The name of the student taking the test | String | Name must not be longer than 50 characters. |
| First\_number | The first number in the equation | Integer | N |
| operators | The mathematical operators which can be used in the calculation | Constant list of characters | N |
| operator | This contains a randomly selected operator from the operators list | Character | N |
| last\_number | The last number in the equation | Integer | N |
| question | This contains the question the user has to answer | String | N |
| Valid | This contains True or False depending if the answer given is correct | Boolean | N |

## Flowchart



## Pseudocode Algorithm

Input player’s name

While length of player’s name is 0 or over 50 characters

Report error stating it has to be between 1 and 50 characters long.  
Ask players name again

Answer = 0

Num\_answered = 0

Score = 0

Set first number to a random number between 1 and 12

Set operator to a random operator, one of +, -, \*

Set second number to a random number between 1 and 12

Generate question as below:

(First number) (Random operator) (Second number) equals symbol

Calculate correct answer to question

While answer to question is less than 0

Set first number to random number between 1 and 12

Generate question as below:

(First number) (Random operator) (Second number)

Calculate correct answer to question

Output the question

Input users answer

If users answer is correct

Output "That is the correct answer"

Add 1 to total score

Add 1 to num\_answered

Otherwise

Output "That is not the correct answer."

If total score is less than 5

Output "Nice try. You got [total score] out of 10, better luck next time"

If total score equals 10

Output "congrats, you got [total score] out of 10, you win"

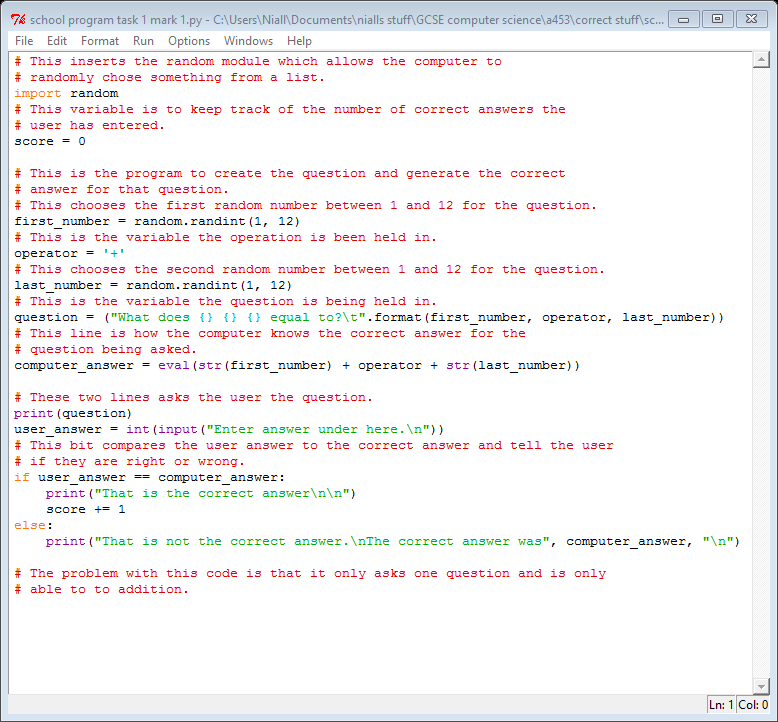
If total score is between 5 and 9

Output "Well done. You got [total score] out of 10, Next time aim for 10 out of 10"

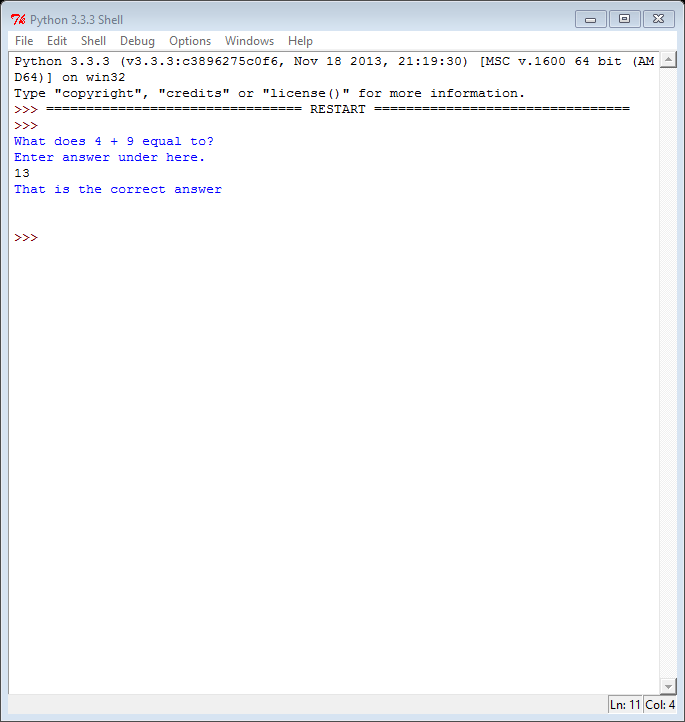
## Annotated Code Listings

### Version 1

#### The program

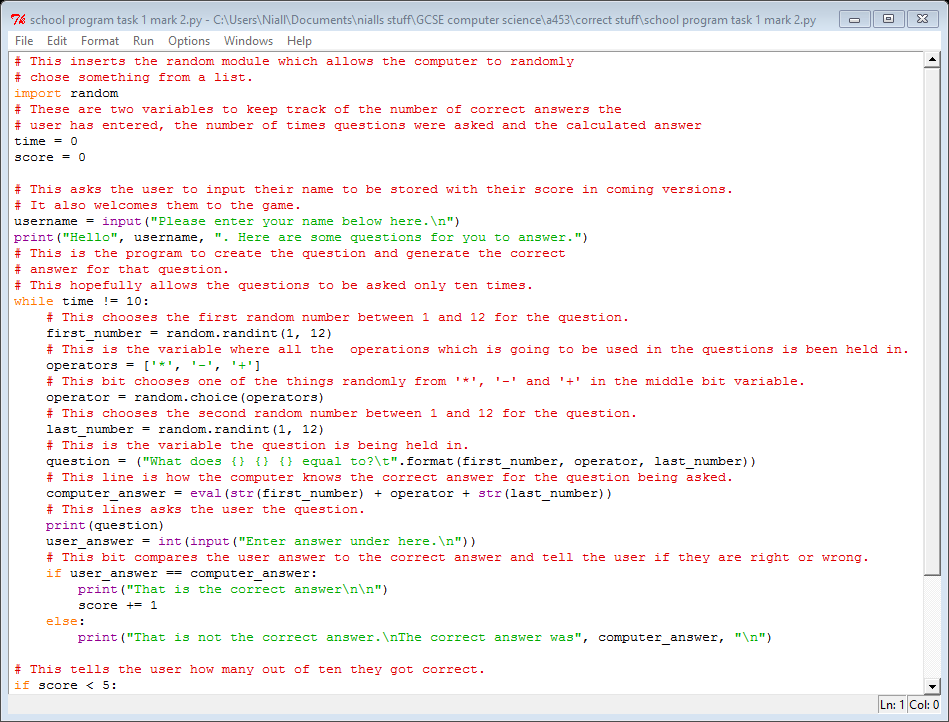


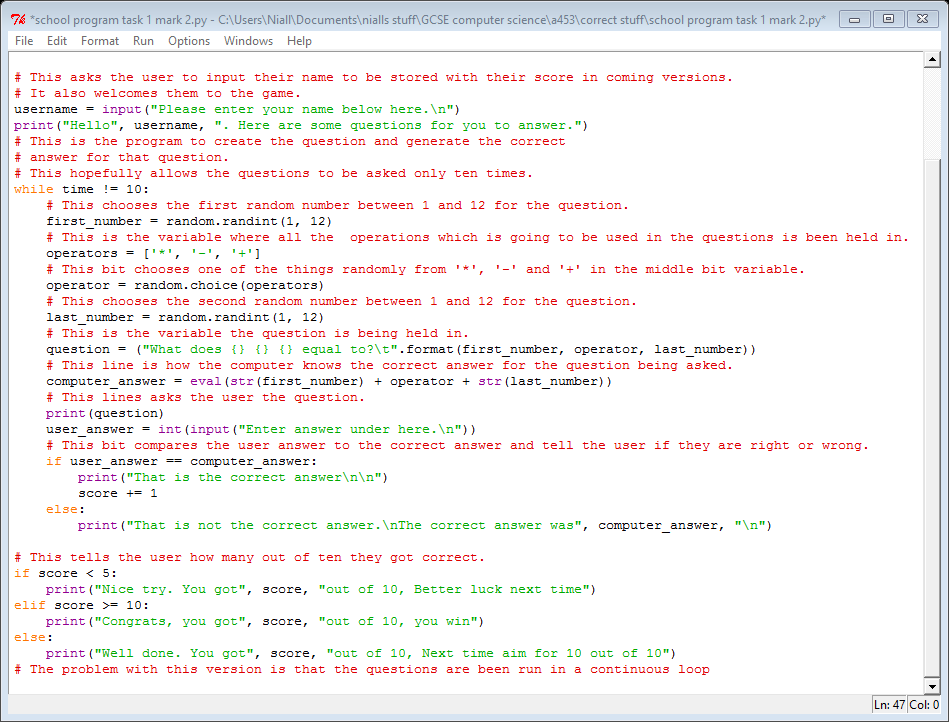
#### The program running



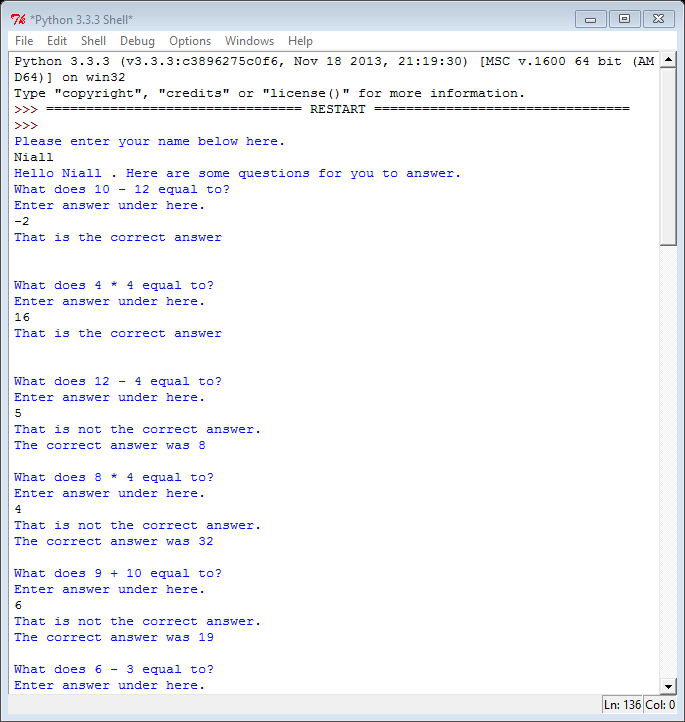
### Version 2

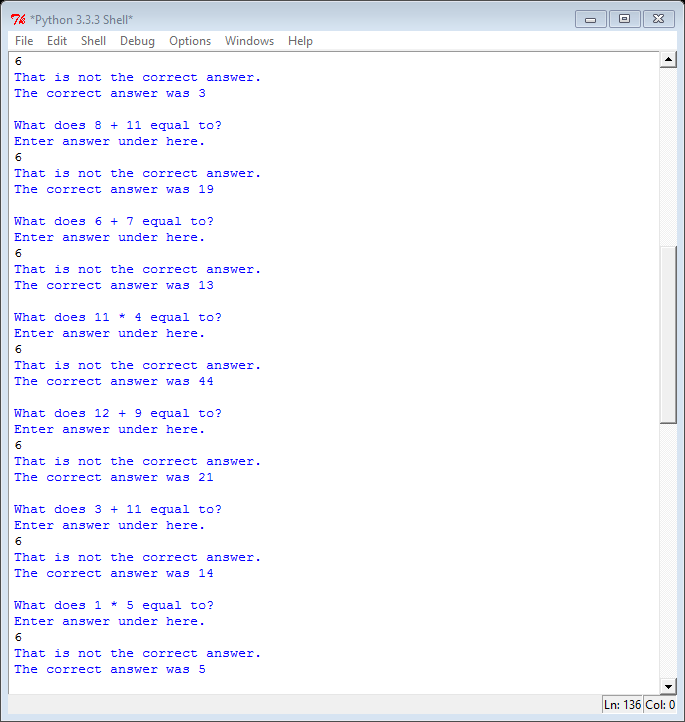
#### The program





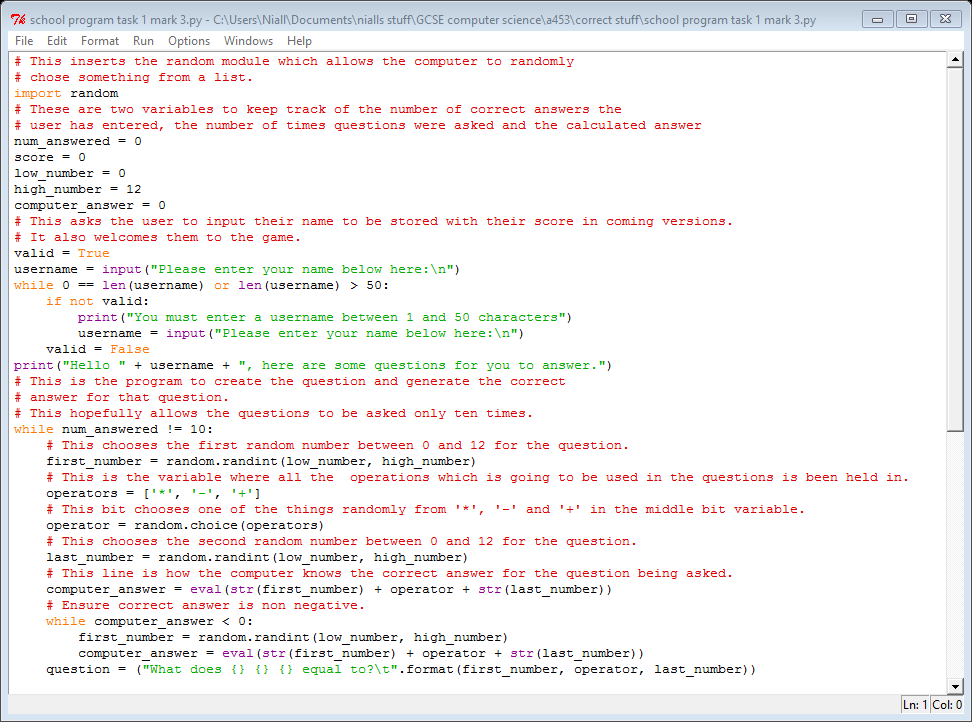
#### The program running

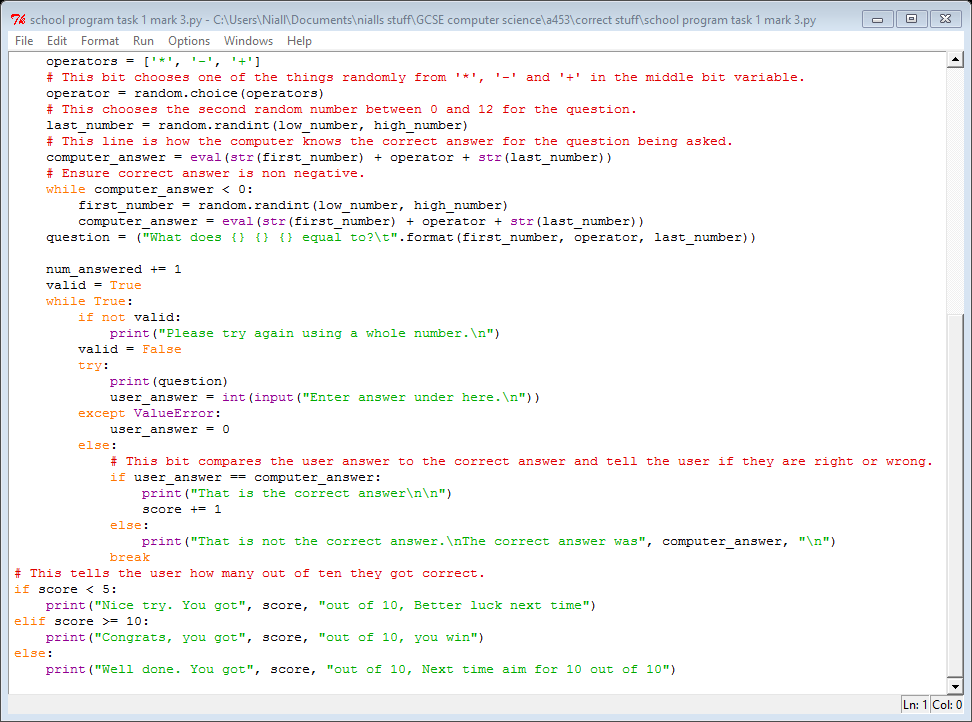




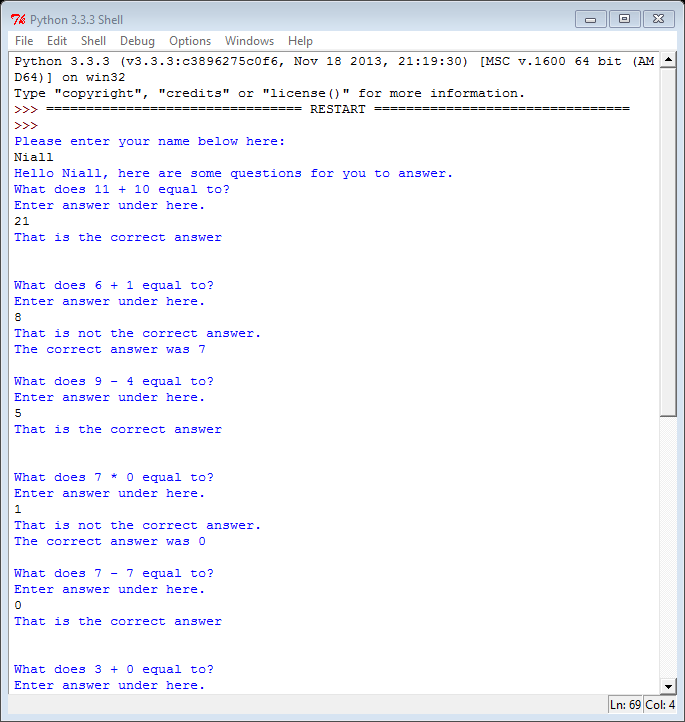
### Version 3

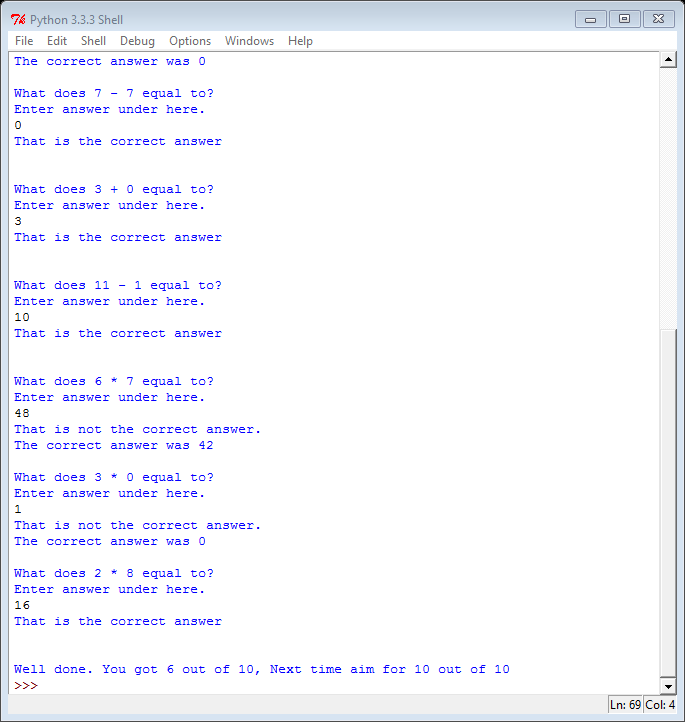
#### The program





#### The program running





## Test plan

The test plan for this task consists of seven specific tests. These cover validation of all user input. It also confirms that when a correct answer is given that the total score increases. All success criteria can be validated using the test below.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test ID | Test description | Input data | Input type | Expected outcomes | Actual outcome | Corrective action | Retest outcome |
| 01 | Confirm that the users name is less than 50 characters | Niall | Normal | Data accepted | Data accepted | N/A | N/A |
| 02 | Confirm that the users name is less than 50 characters | Bobbbbbbbby  Bobbbbbbbbbby  Bbobbbbbbbbbby  Bbbobbbbbbbbbby | Invalid | Error message displayed (please try again) | Data accepted | Change validation rule to  while 0 == len(username) or len(username) > 50:  print error message | Error message displayed (please try again) |
| 03 | Checks if users answer is an integer | 50 | Normal | Data accepted | Data accepted | N/A | N/A |
| 04 | Checks if user answer is an integer | Dfcghj | invalid | Error message displayed (please try again) | Error message displayed (please try again) | N/A | N/A |
| 05 | Confirm that when a correct answer is given that the users score increases | 10 Correct answers to the quiz | normal | Data accepted final score is 10 out of 10 | Data accepted final score is 10 out of 10 | N/A | N/A |
| 06 | Confirm that the only numbers are between 0 and 12 | 10 Correct answers to the quiz | normal | All numbers in sums are between 0 and 12 | All numbers in sums are between 0 and 12 | N/A | N/A |
| 07 | Confirm that the answers are never less than 0 | 10 Correct answers to the quiz | normal | All answers to the 10 sums were not less than 0 | All answers to the 10 sums were not less than 0 | N/A | N/A |

## Task 1 Evaluation

### Did I reach all my success criteria’s?

|  |  |  |
| --- | --- | --- |
| Success criteria | Success criteria satisfied | Evidence |
| Asks user for their name | Yes |  |
| User’s name will be validated, checking that it is no more than 50 characters and that it does not contain numbers. | yes |  |
| Randomly chooses two numbers between 0 and 12. | Yes | Having carried out ten tests all numbers in the questions are between 0 and 12.  See test results in table 1 below. |
| Questions with subtraction must not result in a negative number. | Yes | In all of the subtraction questions I have tested, none of them had an answer less than 0.  See table 1 below for proof. |
| Only whole numbers will be used (i.e. No fractions) | Yes | I have tested the program numerous of times and I have not yet come across any decimal or fractions questions and answers.  See table 1. |
| Randomly selects one operator from +, \*, - | Yes | The only operators I have set it to use is the +, - and the \* symbols. This meant the only questions generated would have the either +, - or \*.  See evidence in table 1. |
| Creates a question from selected numbers and the operator | Yes | I have set the program to only use the numbers between 0 and 12 meaning for each question it can only use those numbers. I have tested it numerous times to prove that it only use those numbers as shown in table 1. |
| Presents the user with the question in a logical format | Yes |  |
| Accepts and validates the input from the user | Yes |  |
| Calculates the correct answer for the question | Yes | The program is able to calculate the correct answer as it is able to tell whenever the user has typed in the wrong answer and correct them as shown in table 1. |
| Compares users answer to the correct answer and displays a message informing them if they are correct or wrong. | Yes | As shown in table 1, if the user types in the wrong answer to the question, the program will say that it is wrong and outputs the correct answer. This proves that the program calculates the correct answer to compare the users answer to and correct them if necessary. |
| Asks ten questions only | Yes | I have set the program to only ask 10 questions each time. I have ran the program a couple of times to test if the program only asks ten questions, which it does, like in table 1. |
| Shows score out of 10 | Yes |  |

### Table 1 – Program results from sample run

|  |  |  |  |
| --- | --- | --- | --- |
| Question number | Question | User answer | Program response |
| 01 | What does 9 + 9 equal to? | 18 | That is the correct answer |
| 02 | What does 12 - 12 equal to? | 0 | That is the correct answer |
| 03 | What does 3 \* 9 equal to? | 17 | That is the correct answer |
| 04 | What does 8 - 7 equal to? | 1 | That is the correct answer |
| 05 | What does 2 - 2 equal to? | 0 | That is the correct answer |
| 06 | What does 9 \* 6 equal to? | 2 | That is not the correct answer.  The correct answer was 54 |
| 07 | What does 12 - 1 equal to? | 2 | That is not the correct answer.  The correct answer was 11 |
| 08 | What does 6 \* 11 equal to? | 2 | That is not the correct answer.  The correct answer was 66 |
| 09 | What does 8 + 12 equal to? | 2 | That is not the correct answer.  The correct answer was 20 |
| 10 | What does 0 + 2 equal to? | 2 | That is the correct answer |

When I was writing the program for task 1, I found it pretty easy to make the program ask ten questions. This was because I only needed to use commands I was already familiar with. Another reason why I found it straight forward was because I did not need to use any functions and the program was written in a simple sequential manner.

I could have used functions to improve my program because it makes it easier to extend and support the program. I could have added more blank lines so that it would be easier to read. It could also be improved by adding difficulty levels which could either be set by the user or automatically increase as the user gets answers correct. The program is currently limited to ten questions, but to improve it in the future I could allow the user to set the amount of questions they would like to answer. If I had more time I would have split it up into functions and possibly displayed all the questions at the end with the correct answer and the answer the user entered.

# Task 2: Record and store the data for three separate classes of students using the arithmetic quiz

## Task Overview

### Requirements

The teacher wants to keep track of the scores each member of the class obtains in the quiz. There are three classes in the school and the data should be kept separately for each class.

### Analysis of Requirements

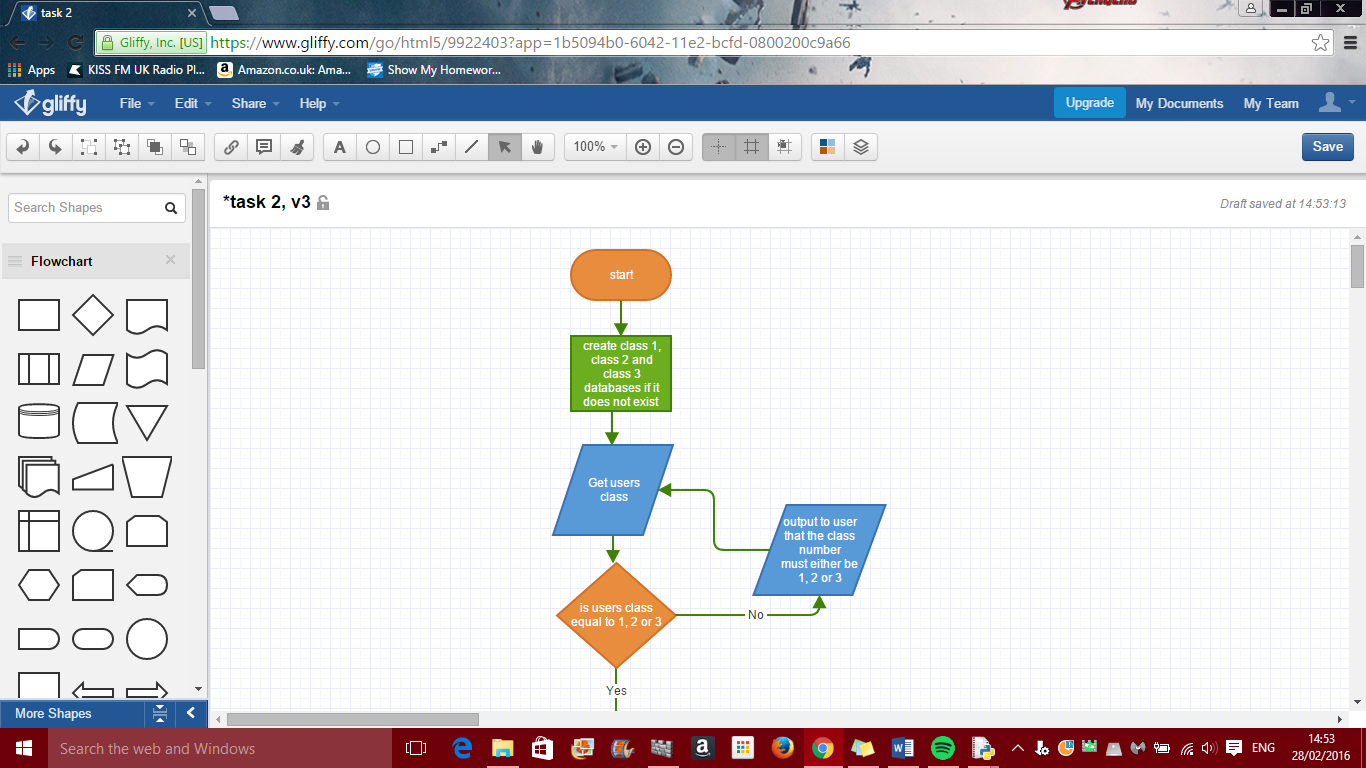
The task this time is to design, code and test a program which will be able to store three different primary school classes’ high scores. The program is to take in the information the teacher enters and be able to store it in a file that can opened in the future. The data for each class will be stored as a csv file as this can easily be viewed in Microsoft Excel which the teachers are most likely to be familiar with.

## Success Criteria

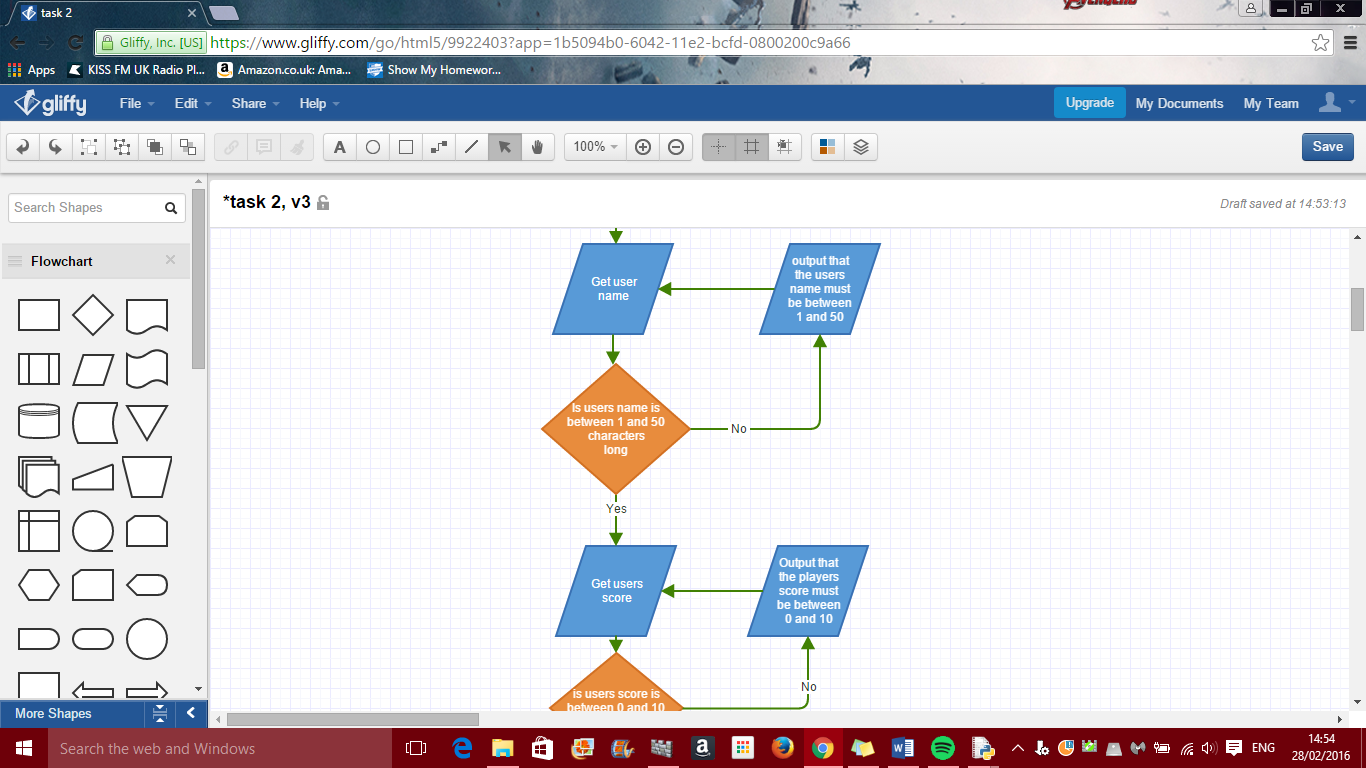
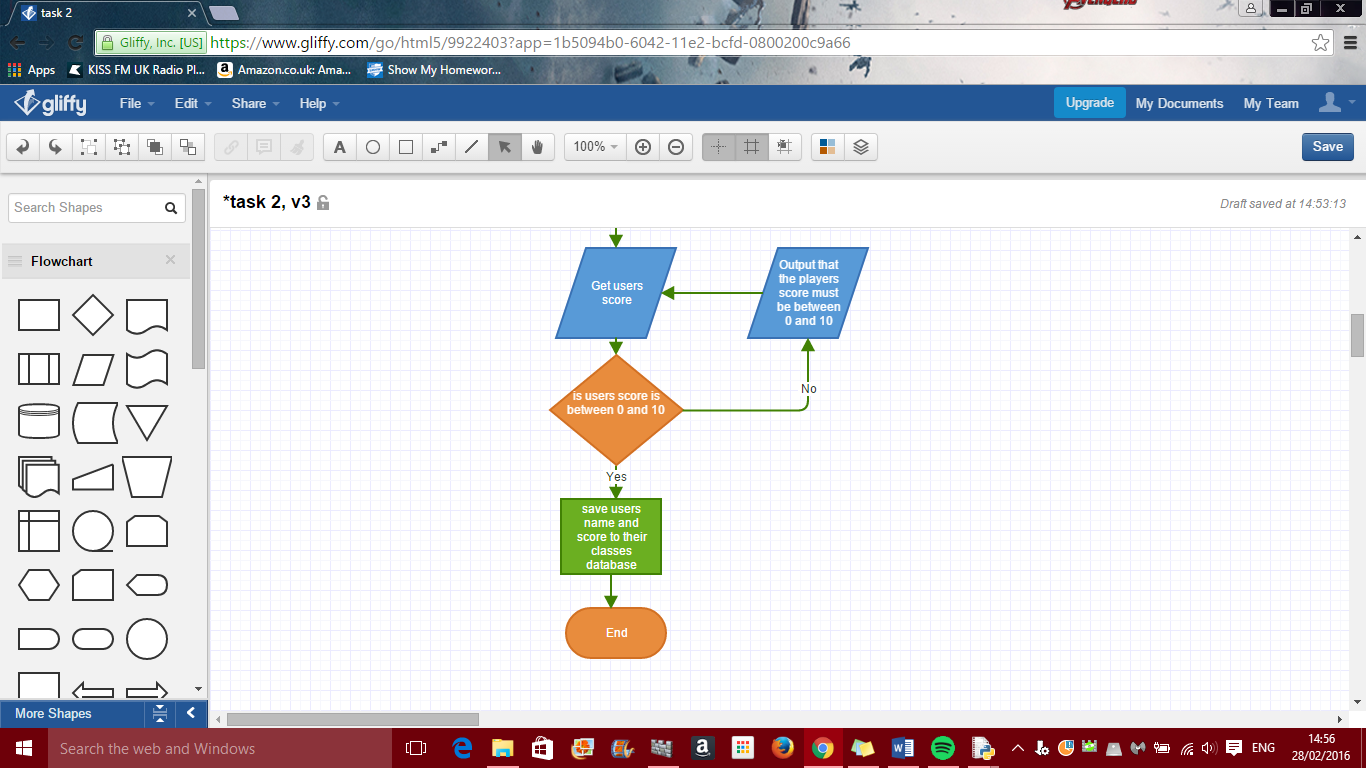
* Asks the user which class’s file they want to add the scores too
* Asks the user for their score and name
* Successfully saves the data to the correct file

## Dictionary and Validation

|  |  |  |  |
| --- | --- | --- | --- |
| Variable Name | Description | Type | Validation |
| Number\_of\_classes | This variable contains the maximum number of class data files the program will create. | Constant Integer | N |
| PlayersNames | This will contain all the players names in the chosen data file | Array of strings | N |
| Classes\_num | This contains the number of the class the user is updating | Integer | The class number the user inputs must be between 1 and Number\_of\_classes |
| Users\_score | This variable contains the score the user has input | Integer | The users score must be between 0 to 10 because 0 is the lowest score they can get and 10 is the highest |
| Name | This variable contains the users name | string | The users name must be between 1 and 50 characters long |
| Scores\_file | This references the scores file to be opened/closed/updated | File object | N |
| Result\_file | This references the result file to be opened/closed/updated | File object | N |



## Flowchart



## Pseudocode Algorithm

If data file for class 1, 2, 3 does not exist

Then create data files for those which do not exist

Input player’s class

While player’s class is not between 1 and 3

Then input player’s class again after showing error message

Input player’s name

While length of player’s name equals 0 or is over 50 characters

Then input player’s name again after showing error message

Input player’s score

While player’s score is not between 0 and 10

Then input player’s score again after showing error message

If player’s name is already in the data file of the player’s class

Then tell the user that the player’s name is already in use in that data file

Go back to start of algorithm

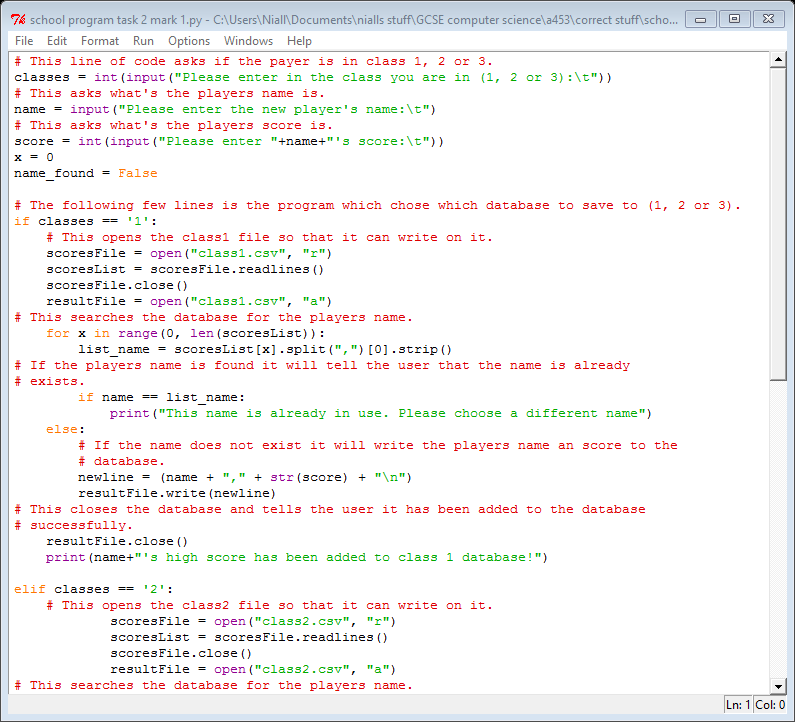
Save player’s name and score to their class’s file

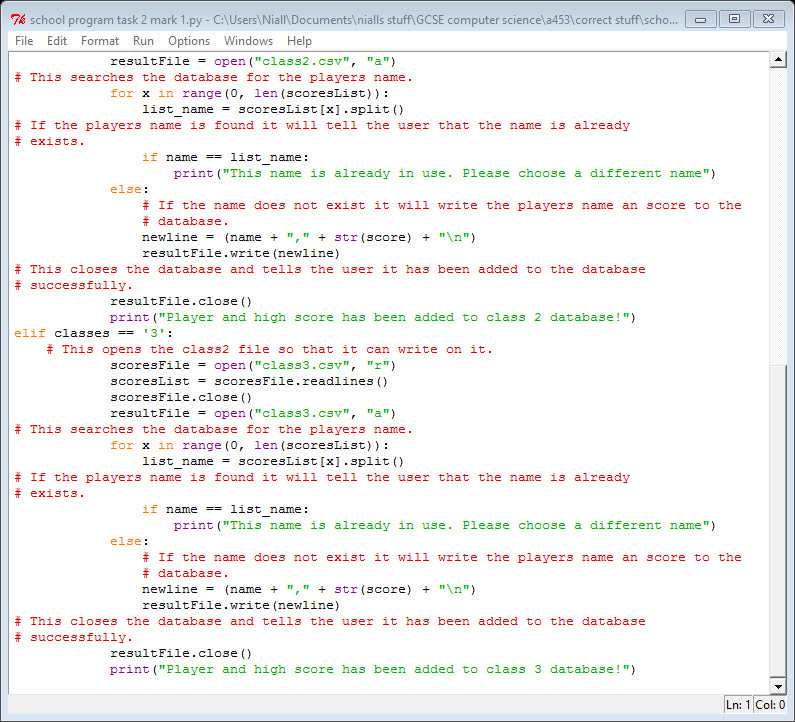
Output to the user that the player’s score has been saved

## Annotated Code Listings

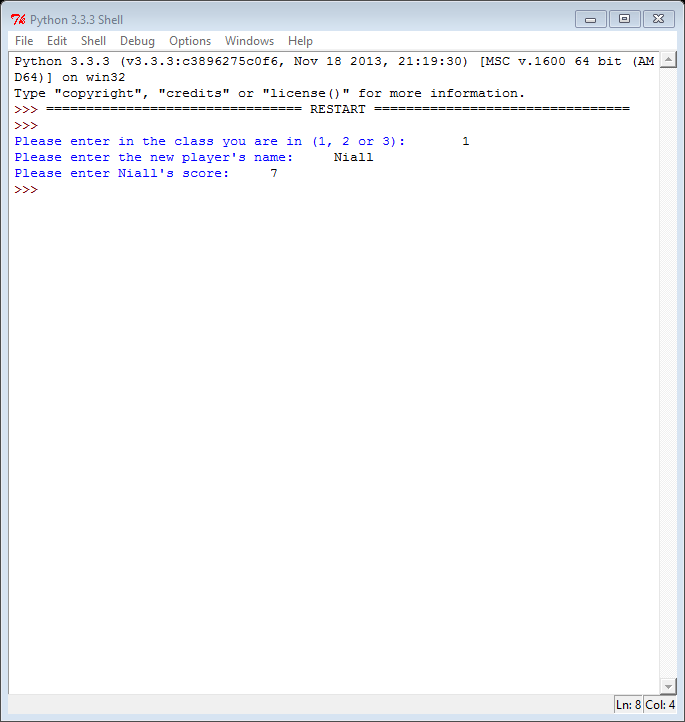
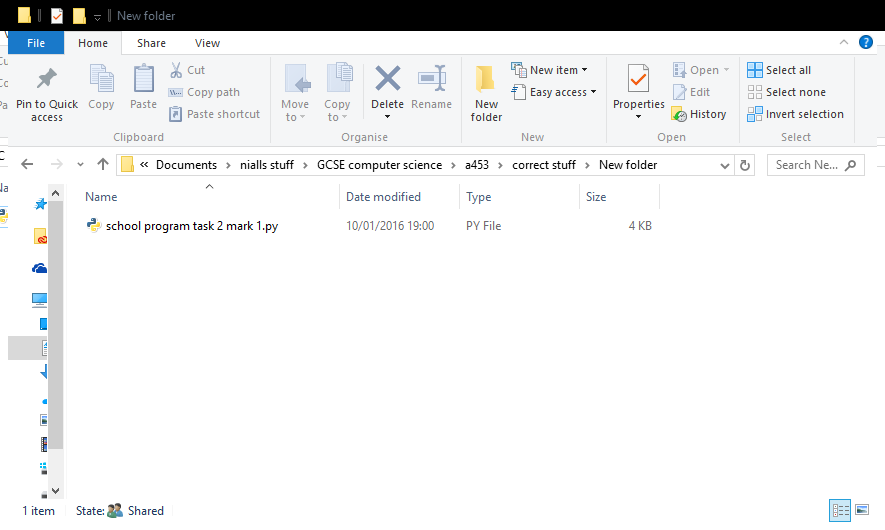
### Version 1

#### The program





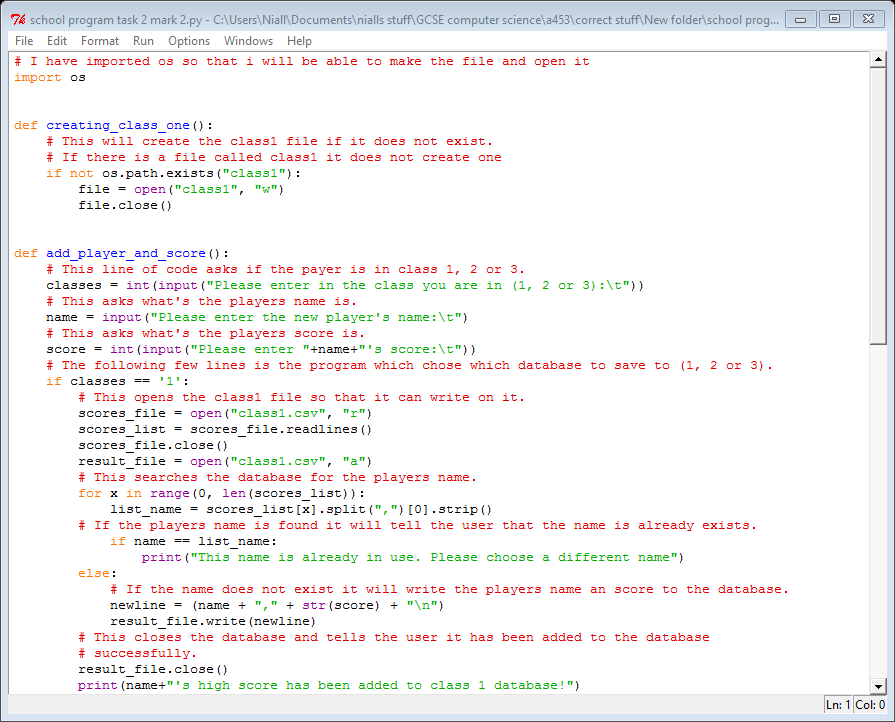
#### The program running

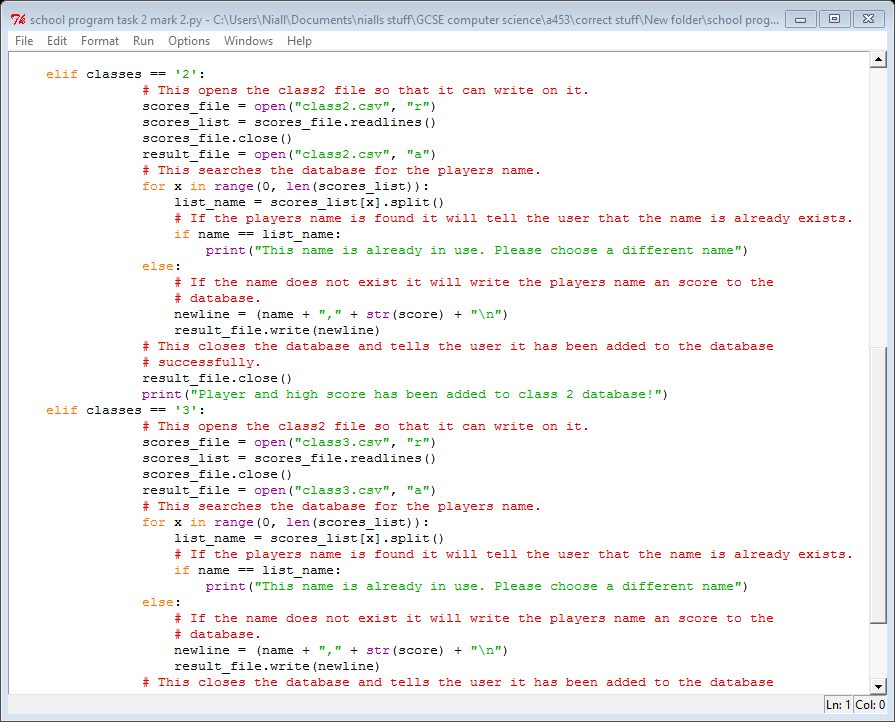


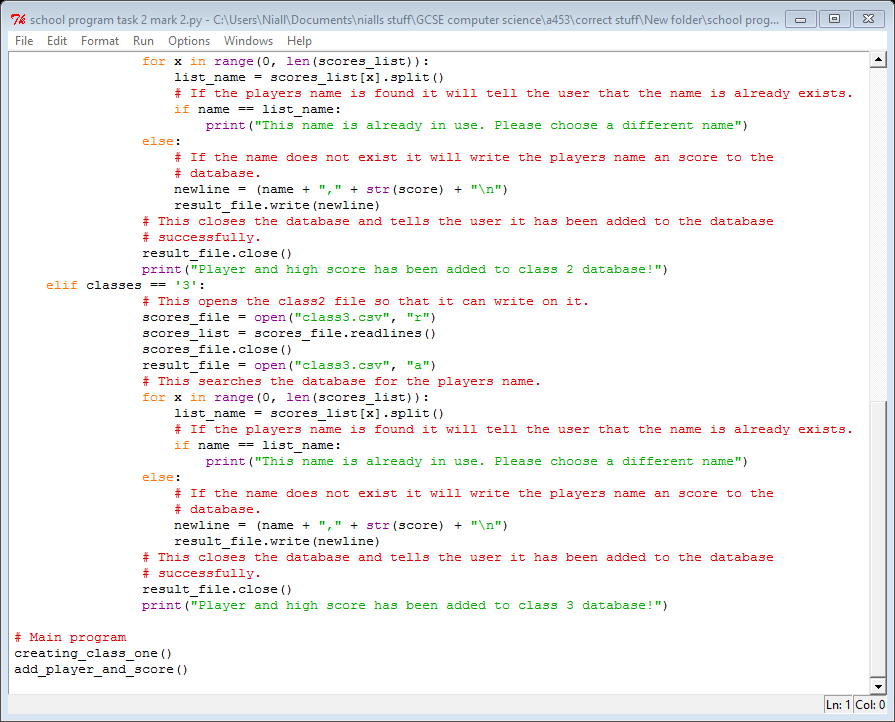
The data was not saved because the program did not create the csv files needed

### Version 2

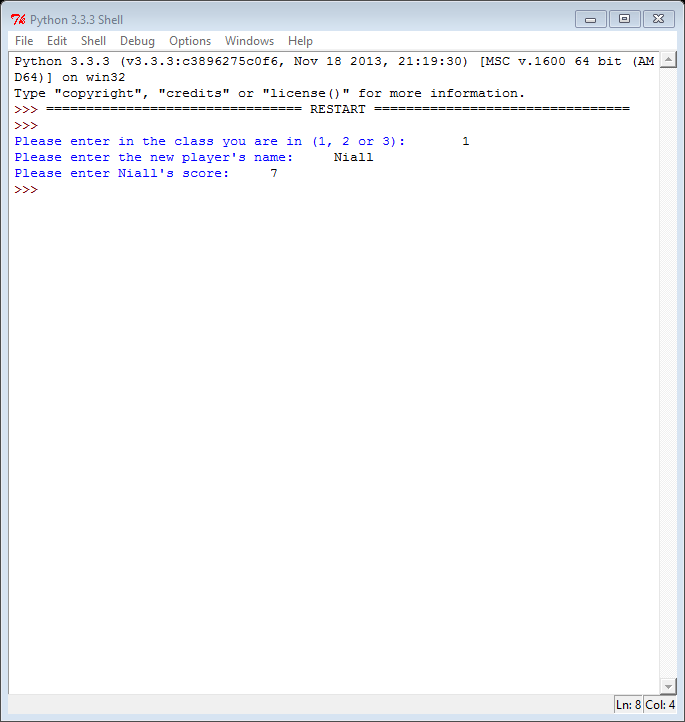
#### The program

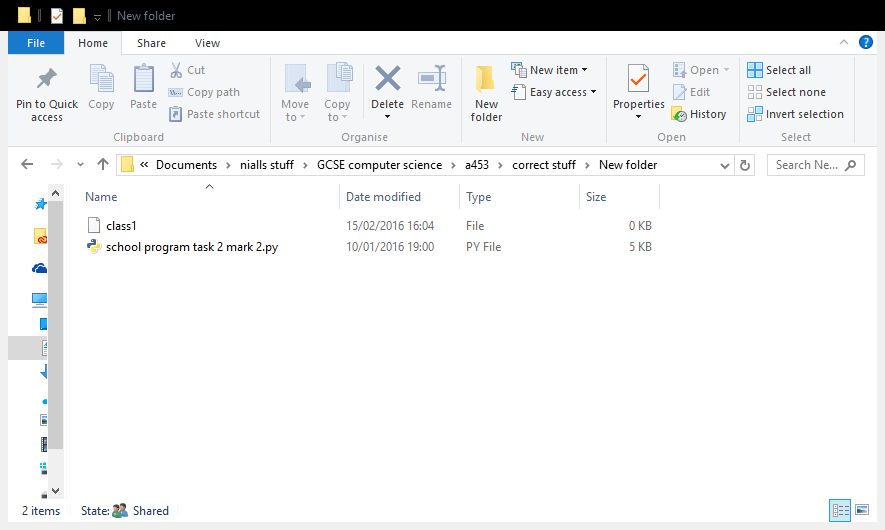


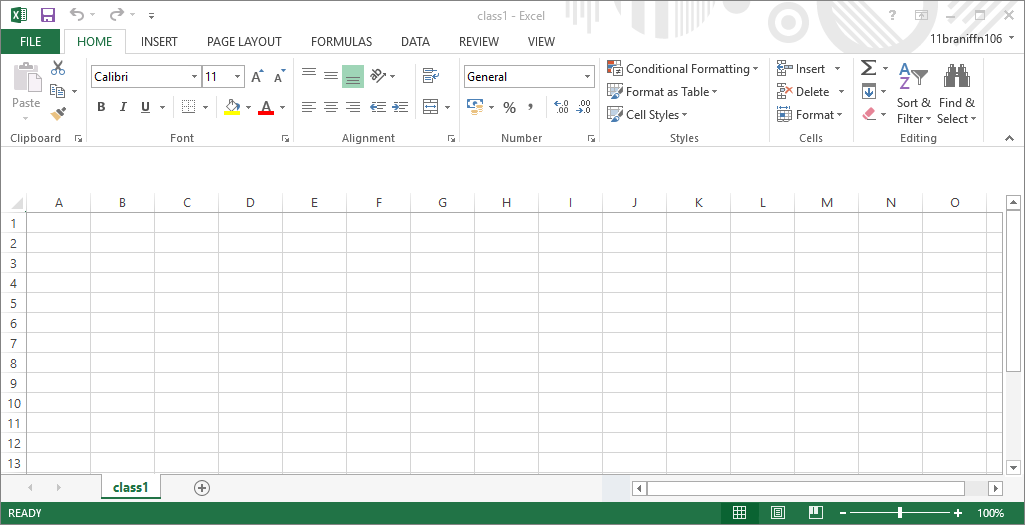




#### The program running



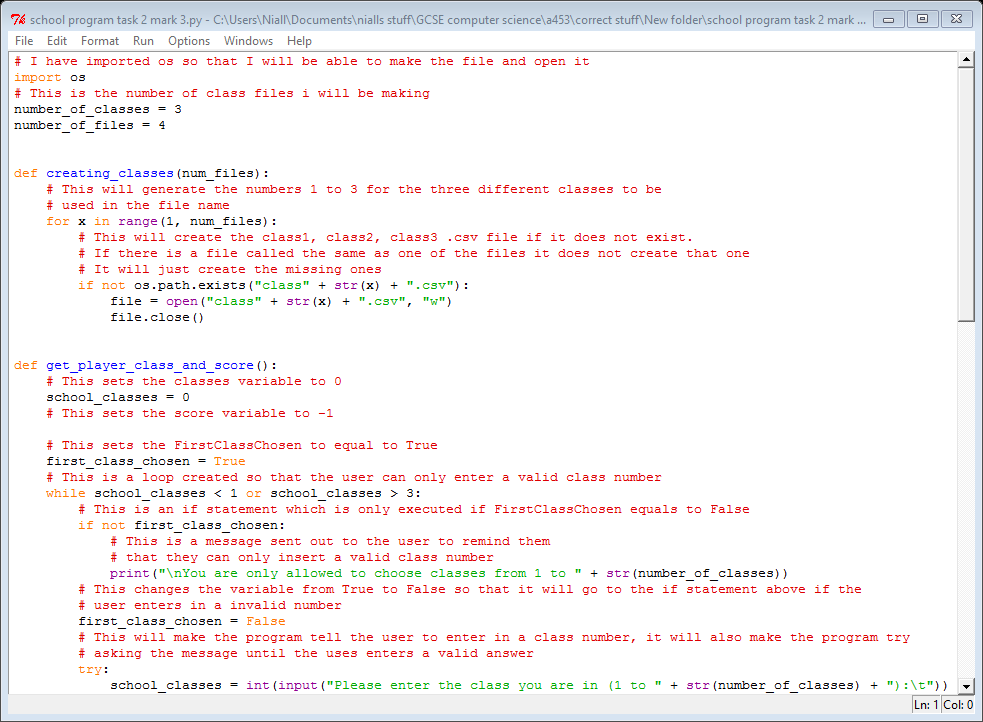


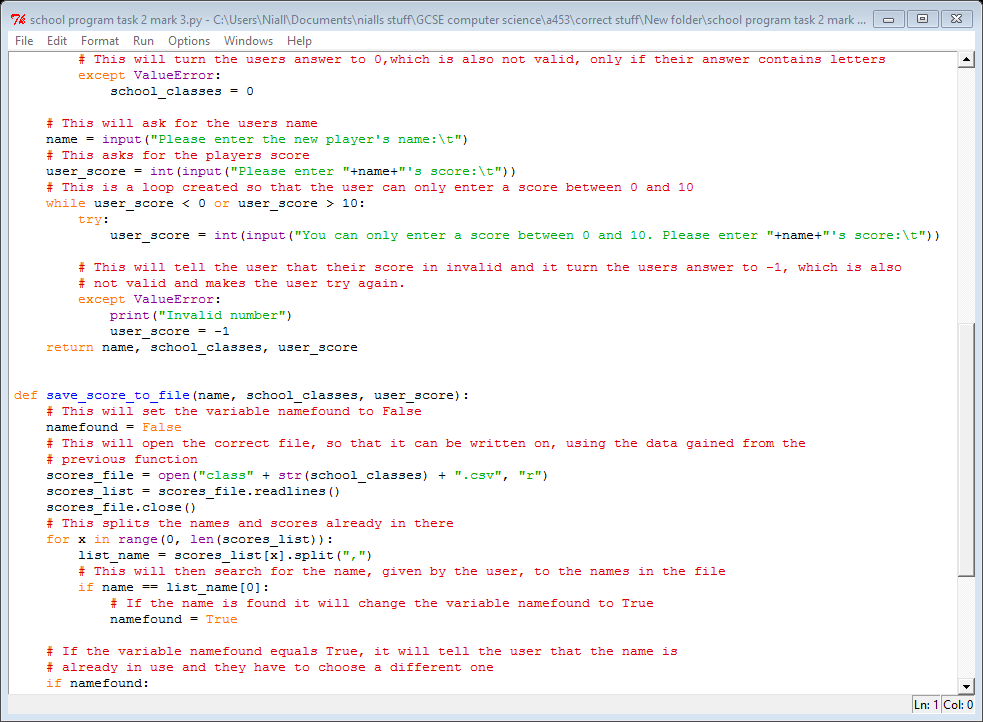


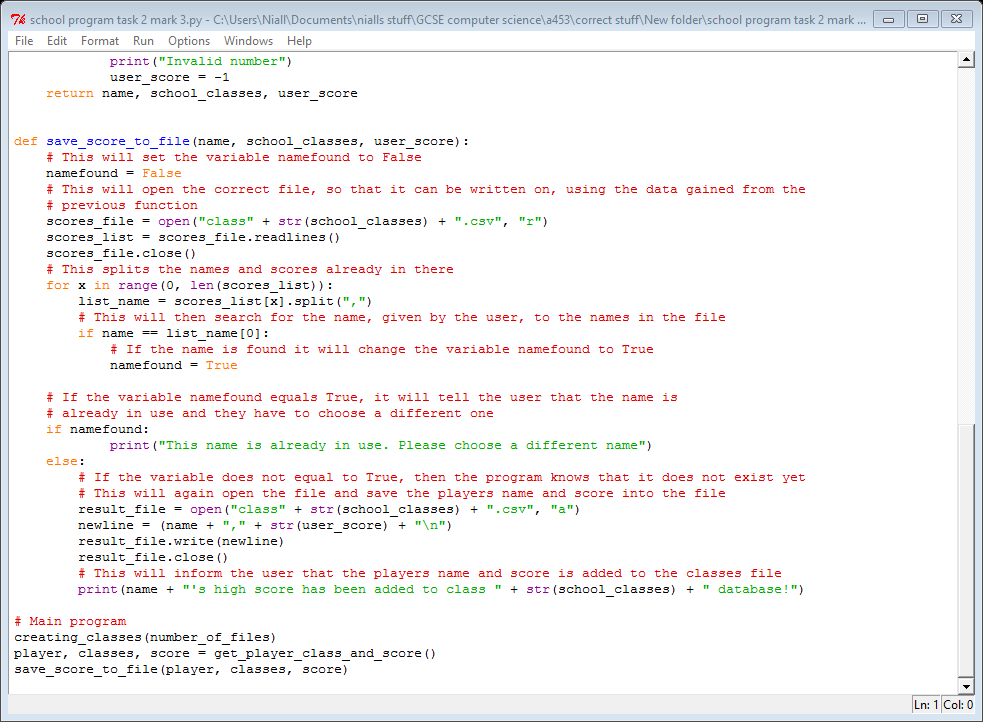
This shows that the data has not been saved correctly

### Version 3

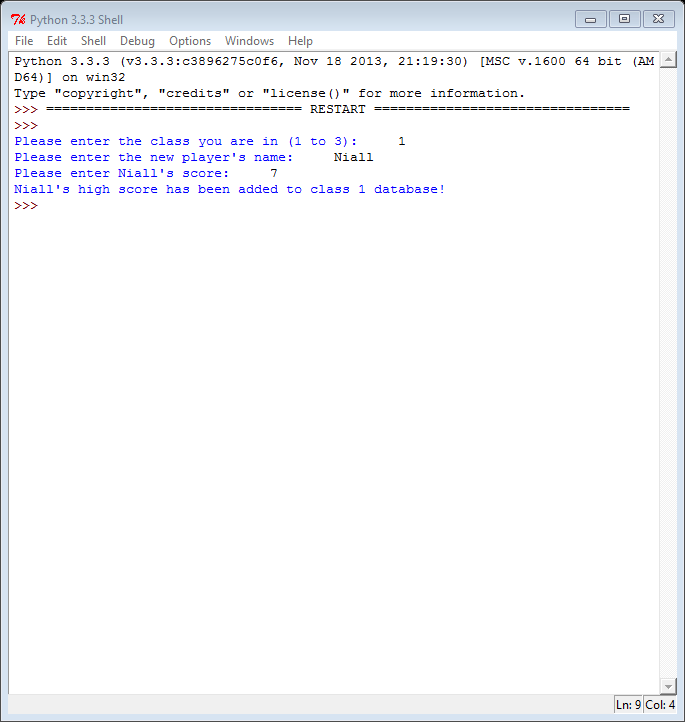
#### The program

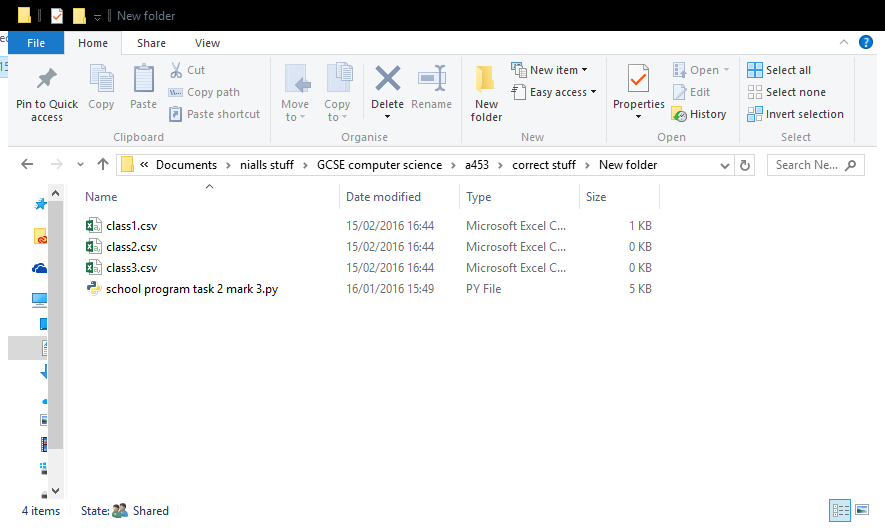


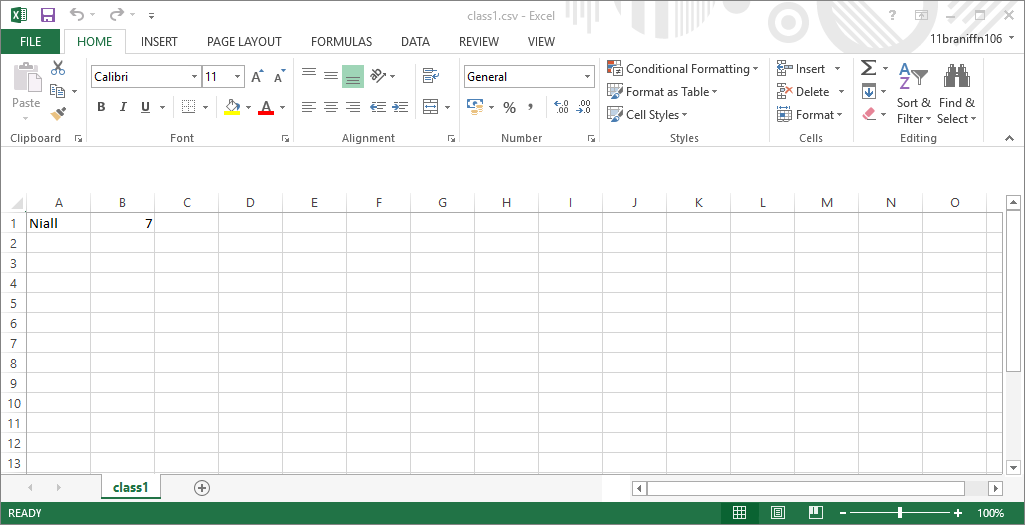




#### The program running

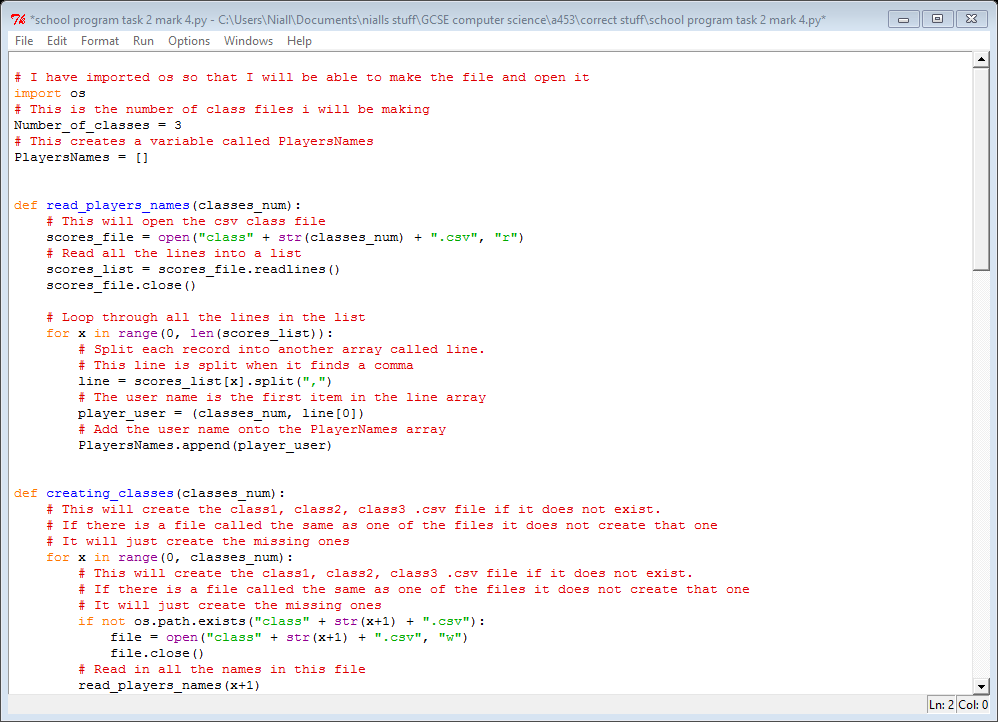


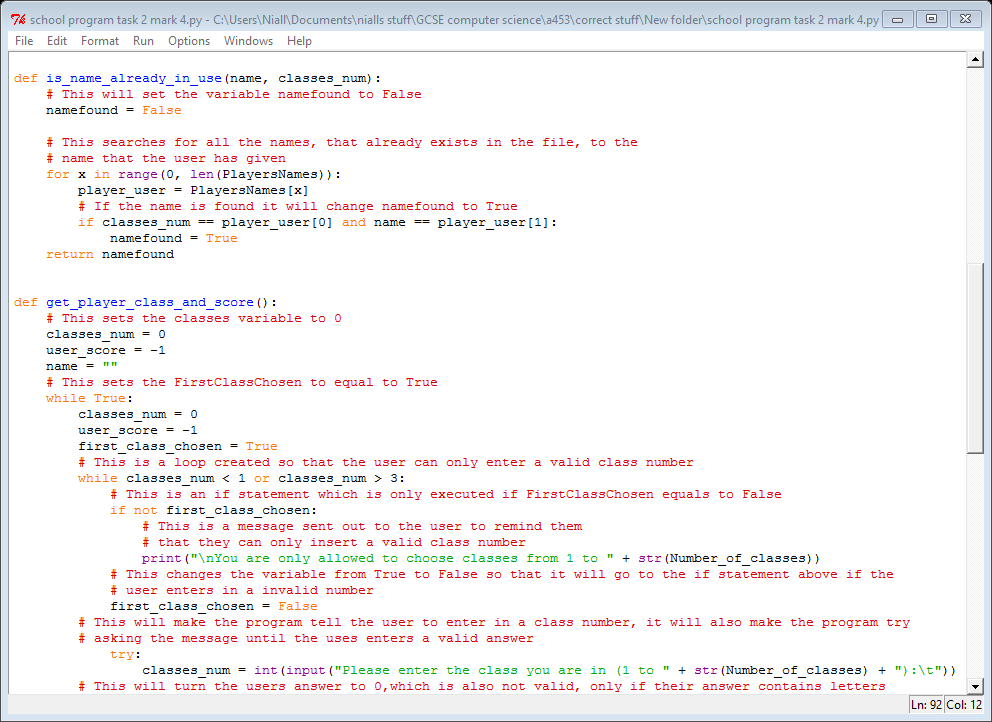


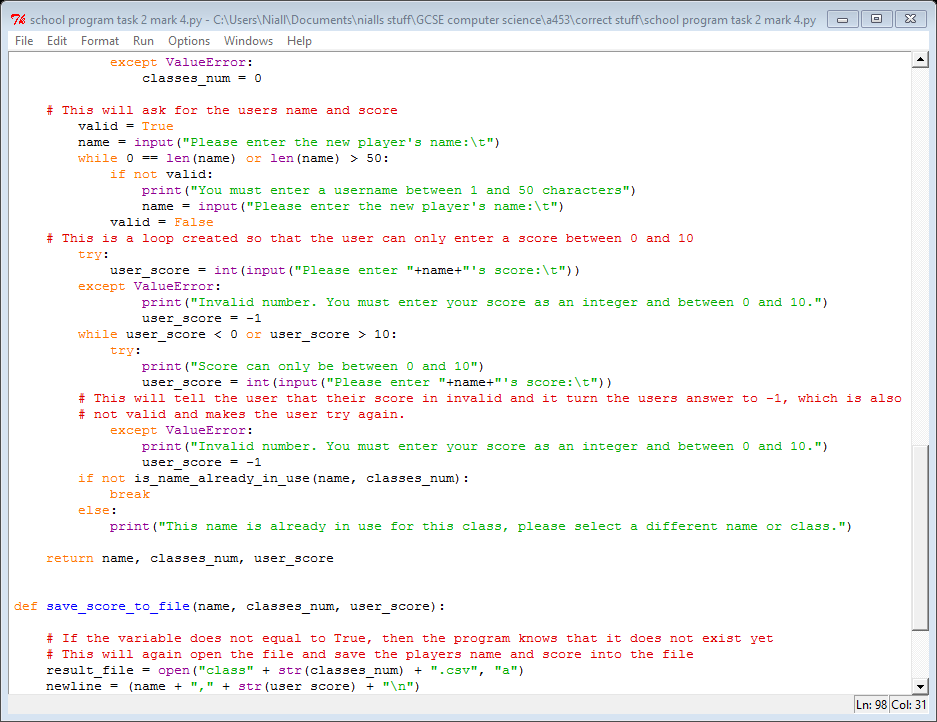


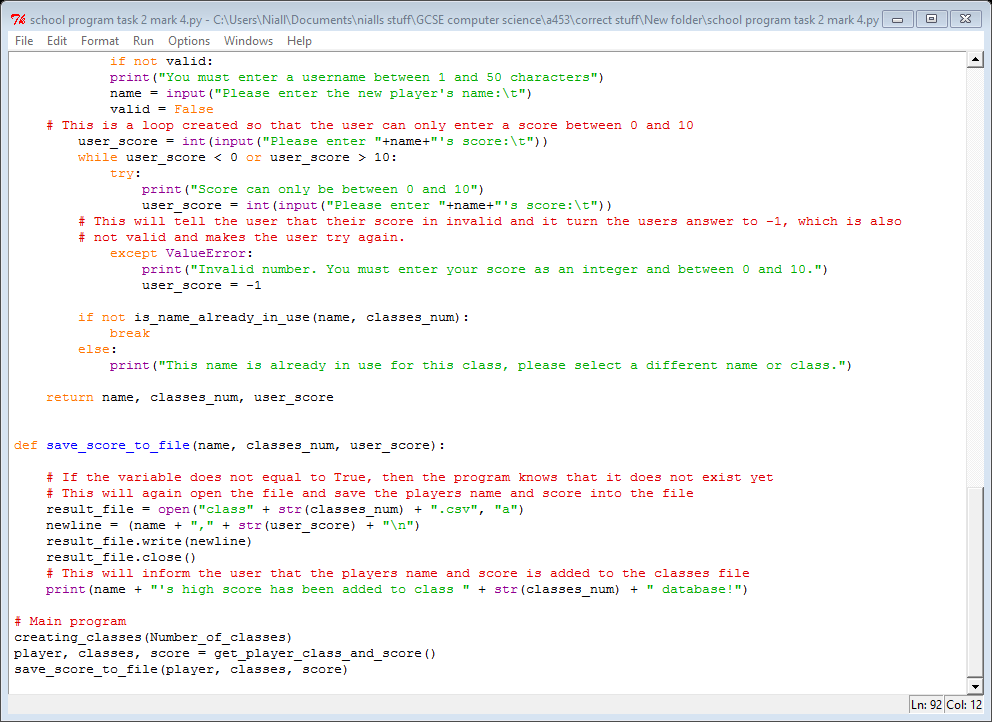
### Version 4

#### The program

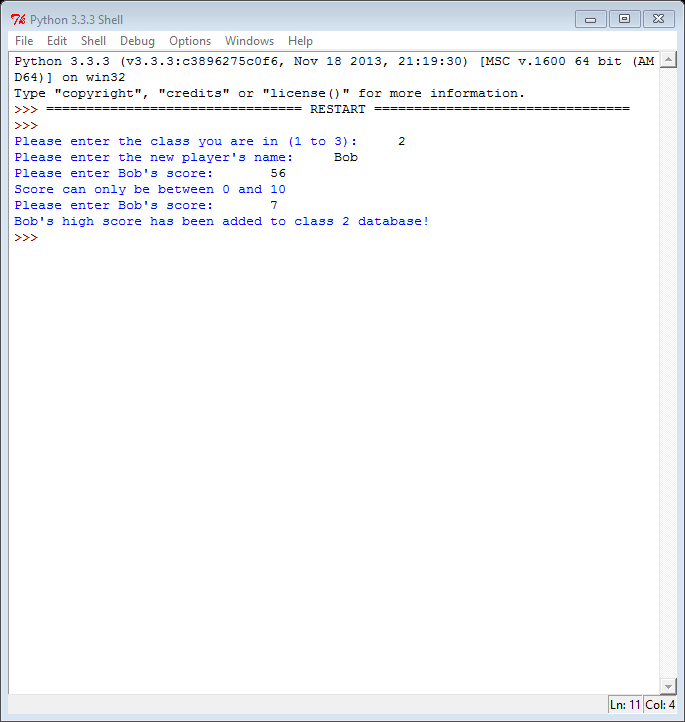


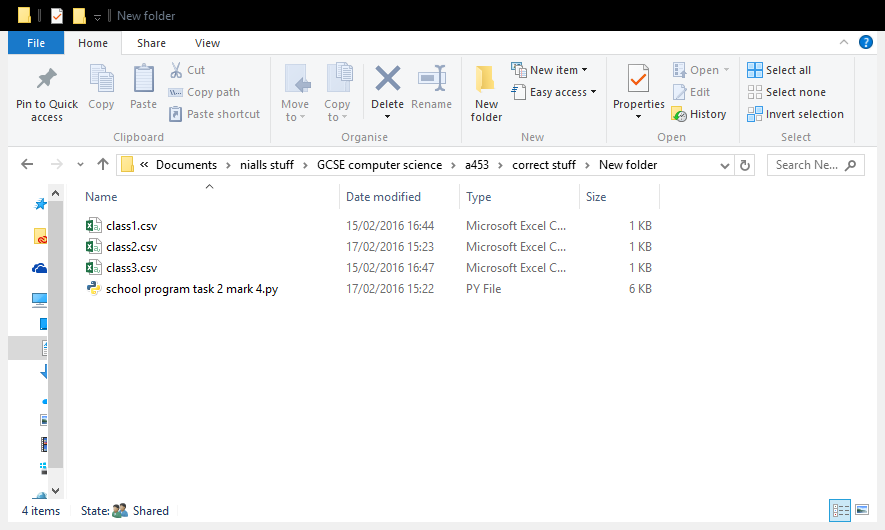


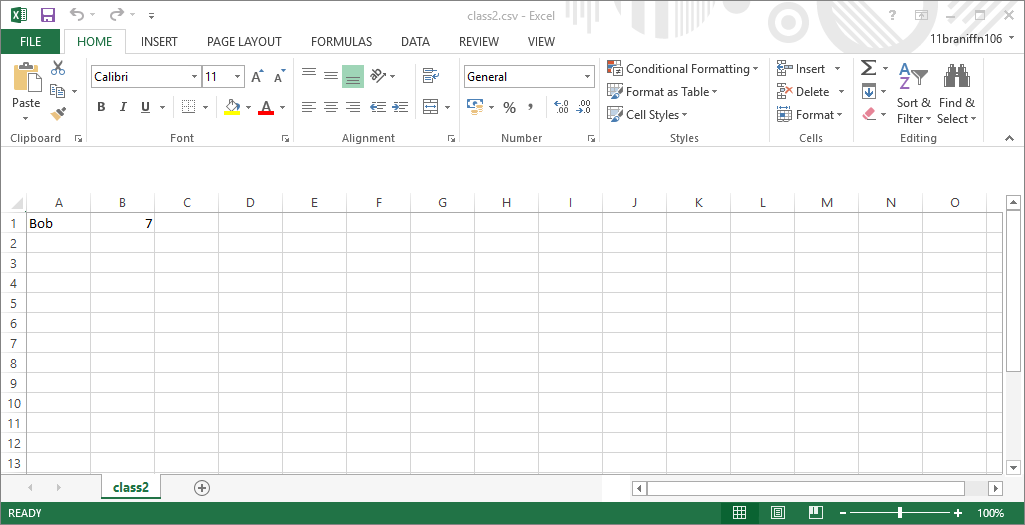




#### The program running







## Test plan

The test plan consists of eleven tests which covers all of the success criteria’s and tests validation of all users’ inputs.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test ID | Test description | Input data | Input type | Expected outcomes | Actual outcome | Corrective action | Retest outcome |
| 01 | Check if class number is valid | 1 | Normal | Data accepted | Data accepted | N/A | N/A |
| 02 | Check if class number is valid | 2 | Normal | Data accepted | Data accepted | N/A | N/A |
| 03 | Check if class number is valid | 3 | Normal | Data accepted | Data accepted | N/A | N/A |
| 04 | Check if class number is valid | 7 | Invalid | Data not accepted | Data not accepted | N/A | N/A |
| 05 | Check if class number is valid | fsfsfs | Invalid | Data not accepted | Data not accepted | N/A | N/A |
| 06 | Check if characters in name is between 0 and 50 | fsfsfs | Normal | Data accepted | Data accepted | N/A | N/A |
| 07 | Check if characters in name is between 0 and 50 | Fsfsfsfsfsfsfsfsfsfsfsfsfsfsfsfsfsfsfsfsfsfsfsfsfsfsfs | Invalid | Data not accepted | Data accepted | Change validation rule to  while 0 == len(username) or len(username) > 50:  print error message | Data not accepted |
| 08 | Check if players score is between 0 and 10 | 7 | Normal | Data accepted | Data accepted | N/A | N/A |
| 09 | Check if players score is between 0 and 10 | 16 | Invalid | Data not accepted | Data not accepted | N/A | N/A |
| 10 | Check if players score is between 0 and 10 | -7 | Invalid | Data not accepted | Data not accepted | N/A | N/A |
| 11 | Check if players score is a valid score | jhbjh | Invalid | Data not accepted | Data not accepted | N/A | N/A |

## Task 2 Evaluation

### Did I reach all my success criteria’s?

|  |  |  |
| --- | --- | --- |
| Success criteria | Success criteria satisfied | Evidence |
| Asks the user what class’s csv file they want to add the scores too | Yes |  |
| Asks the user what’s their score is and their name | Yes |  |
| Correctly saves the data to the correct csv file | Yes |  |

When writing the program I found it fairly straight forward to code the functions that get the user input data, i.e. class number, name and score. This was because I only needed to use input and validation commands, which I was already familiar with. What I found a little bit tricky was writing the code that saves the results to the correct csv files.

I could have improved my program in the future by allowing the user to input a class number greater than 3. This would allow other classes to save their scores from the test. If I had more time I could have allowed the user to play the game and save whatever score they got, instead of needing to remember it and run a different program.

# Task 3: Outputting the results of the quiz for a particular class in specific ways

## 

## Task Overview

### Requirements

The teacher wants to use the results from students taking the quiz to log their performance. The program should store the last three scores for each student. The teacher would also like to be able to output the results of the quiz for a particular class, sorted like this: in alphabetical order with each student’s highest score for the tests, by the highest score (highest to lowest) and by the average score (highest to lowest).

### Analysis of Requirements

The task this time is to design, code and test a program which will be able to view the scores for the arithmetic skills test of primary school classes. The program needs to be able to display the class scores either in alphabetical order, in order of highest score to lowest or, in order of highest to lowest average score. The program also needs to take in the information from the test and be able to store it in a file that can opened in the future. The file must only store the three latest scores from the test. The data for each class will be stored as a csv file as this can easily be viewed in Microsoft Excel which the teachers are most likely to be familiar with.

## Success Criteria

* View class in alphabetical order with each student’s highest score for the tests
* View class using their highest score, highest to lowest
* View class by their average score, highest to lowest
* The program should store the last three scores for each student

## Data Dictionary and Validation

|  |  |  |  |
| --- | --- | --- | --- |
| Variable Name | Description | Type | Validation |
| MAX\_NUMBER\_OF\_CLASSES | This contains how many classes there are in the program | Constant integer | N |
| Choice | This contains the users choice from the menu | integer | Must be a number. When a non-integer is entered an exception is thrown and the user will be asked to retry. |
| class\_num | This holds the number of the class the user has entered in | integer | If the user does not input an integer or a valid integer, then they will be asked to try again. |
| data\_list | This holds the list of players names and scores of the users chosen class |  | N |
| user\_answer | Answer provided by student | Integer | Must be a number. When a non-integer is entered an exception is thrown and the user will be asked to retry. |
| Tries | The number of questions answered. This is initialised to 0 and incremented every time a question is asked. | Integer | N |
| Score | The number of correct answers. This is initialised to 0 and incremented each time the player types in a correct answer. | Integer | N |
| high\_number | This contains the biggest number the question can use | Constant Integer | N |
| low\_number | This contains the smallest number the question can use | Constant Integer | N |
| computer\_answer | This contains the correct answer for the question | Integer | N |
| Username | The name of the student taking the test | String | Name must not be longer than 50 characters. |
| first\_number | The first number in the equation | Integer | N |
| Operators | The mathematical operators which can be used in the calculation | Constant list of characters | N |
| Operator | This contains the randomly selected operator from the variable operators | String | N |
| last\_number | The last number in the equation | Integer | N |
| Question | This contains the question the user has to answer | String | N |
| Valid | This contains True or False depending if the answer given is correct | Boolean | N |
| average\_score | This contains the average score of the players scores | integer | N |
| number\_of\_scores | This counts how many scores are saved for the player so it can accurately work out their average score | integer | N |
| max\_score | This holds the biggest score out of three for a player | integer | N |

## Flowchart

## Pseudocode Algorithm

Input player’s class

Show user the 5 options

Option 1 = to play the game

Option 2 = to view in alphabetical order with each student’s highest score for the tests

Option 3 = to view scores by highest, highest to lowest

Option 4 = to view scores by average, highest to lowest

Option 5 = to exit

Input users choice from the options above

If users choice equals option 1

Then Input Users Name

If player’s name equal to 0 characters long or over 50 characters long

Then input player’s name after stating it has to be over 0 and same/under 50 characters long

Answer = 0

Num\_answered = 0

Score = 0

Set first number to a random number between 1 and 12

Set operator to a random operator, one of +, -, \*

Set second number to a random number between 1 and 12

Generate question as below:

(First number) (Random operator) (Second number) equals symbol

Calculate correct answer to question

While answer to question is less than 0

Set first number to random number between 1 and 12

Generate question as below:

(First number) (Random operator) (Second number)

Calculate correct answer to question

Output the question

Input users answer

If users answer is correct

Output "That is the correct answer"

Add 1 to total score

Add 1 to num\_answered

Otherwise

Output "That is not the correct answer."

If total score is less than 5

Output "Nice try. You got [total score] out of 10, better luck next time"

If total score equals 10

Output "congrats, you got [total score] out of 10, you win"

If total score is between 5 and 9

Output "Well done. You got [total score] out of 10, Next time aim for 10 out of 10"

Save player’s name and score to their class’s csv file

Output to the user that the player’s score has been saved

If users choice = 2

Then read all players from users chosen class csv file

Sort list in alphabetical order the names read from csv file

For each player find their highest score

Output to user the alphabetical order and highest score of the players read from the csv file

If users choice = 3

Then read all players from users chosen class csv file

For each player find their highest score

Sort list of players by highest score

Output list

If user choice = 4

Then read all players from users chosen class csv file

Add all scores and divide it by how many scores are saved for each player read

Sort list of read players by highest average score

Output list

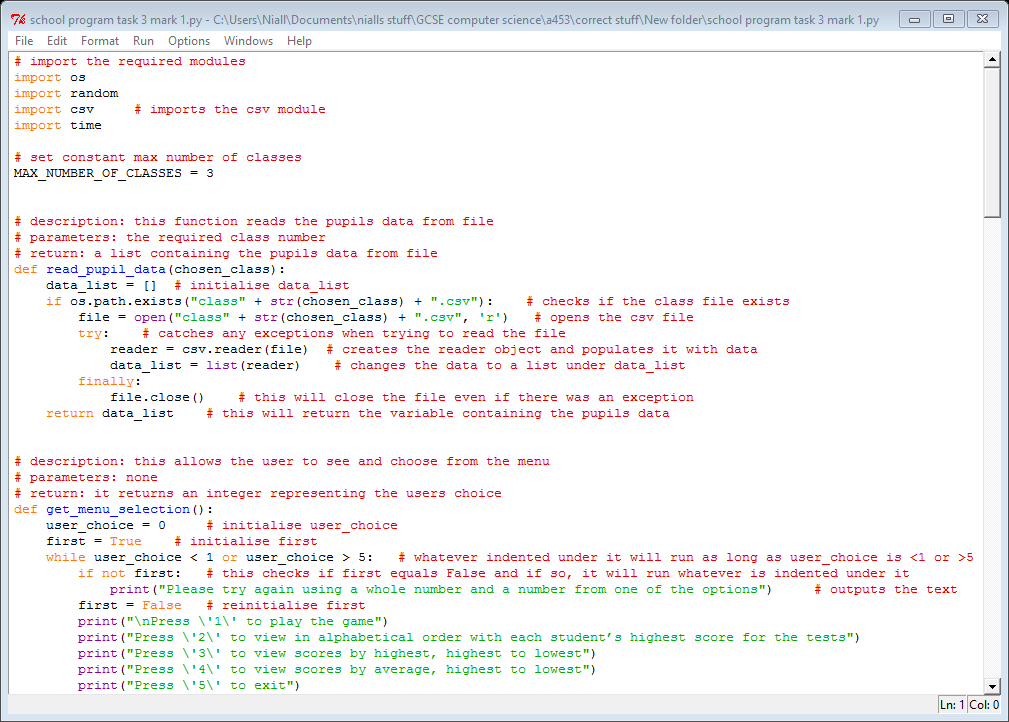
If user choice = 5

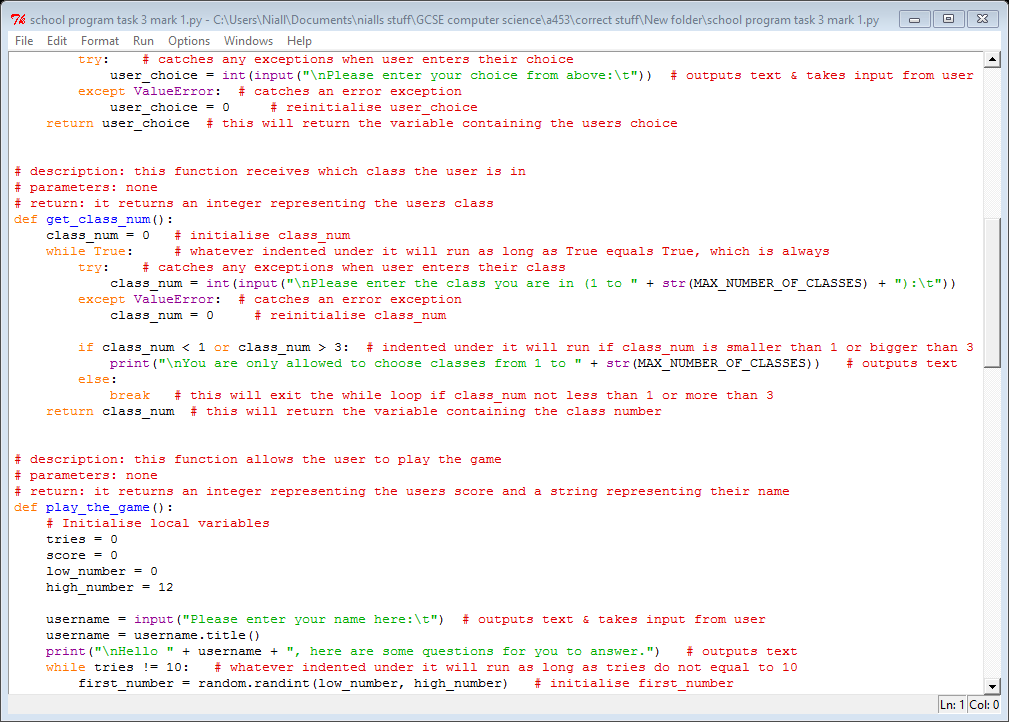
Then close program

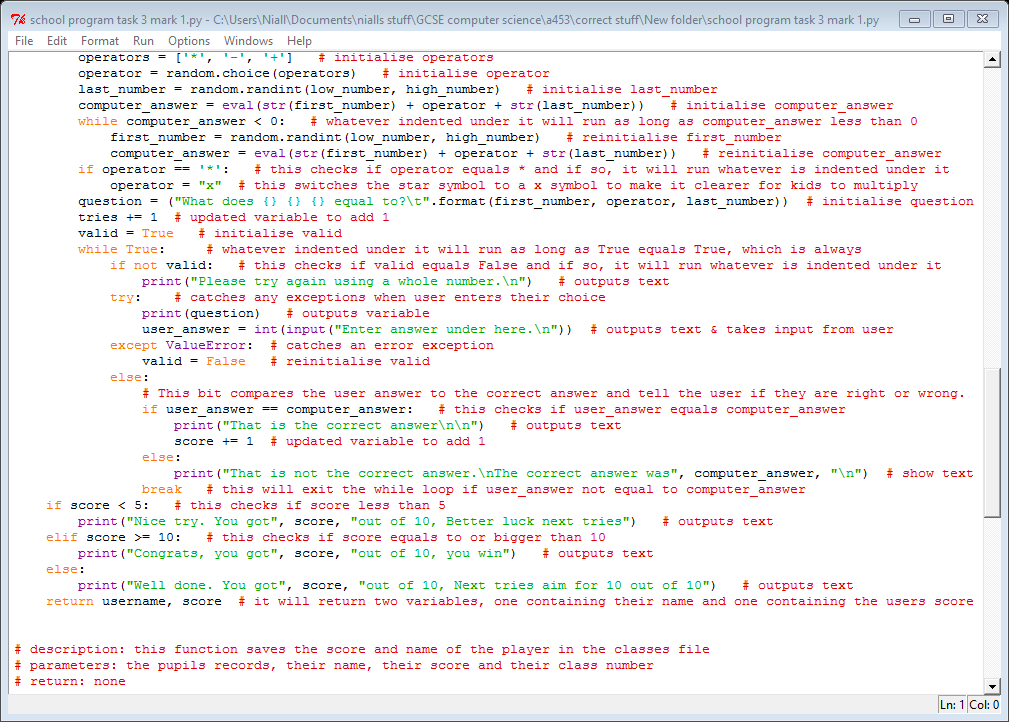
## Annotated Code Listings

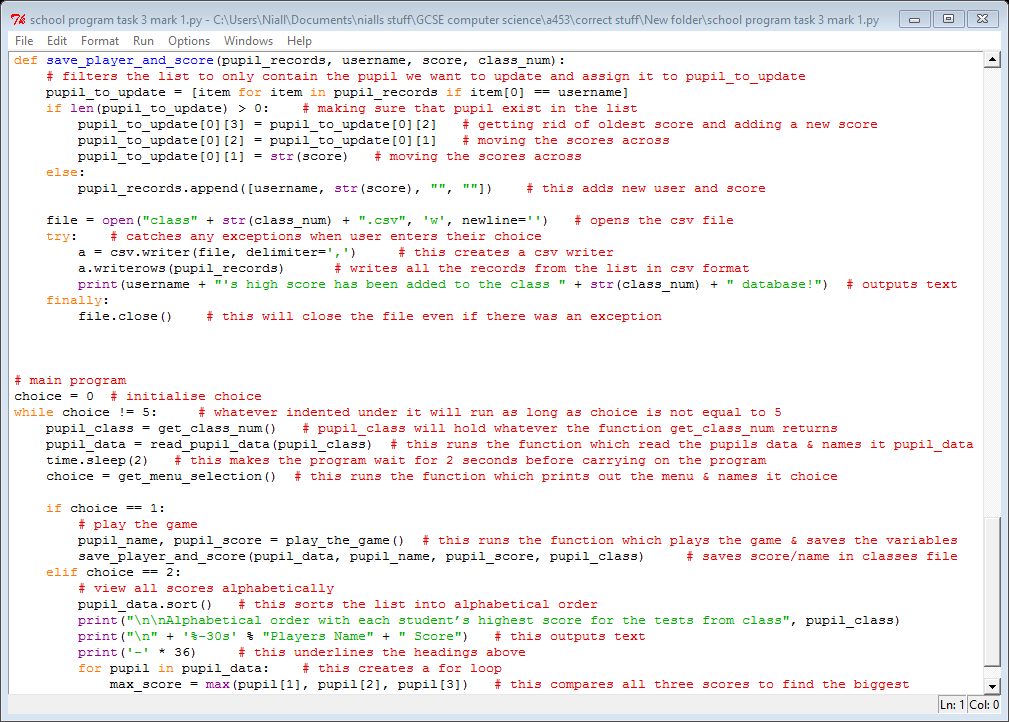
### Version 1

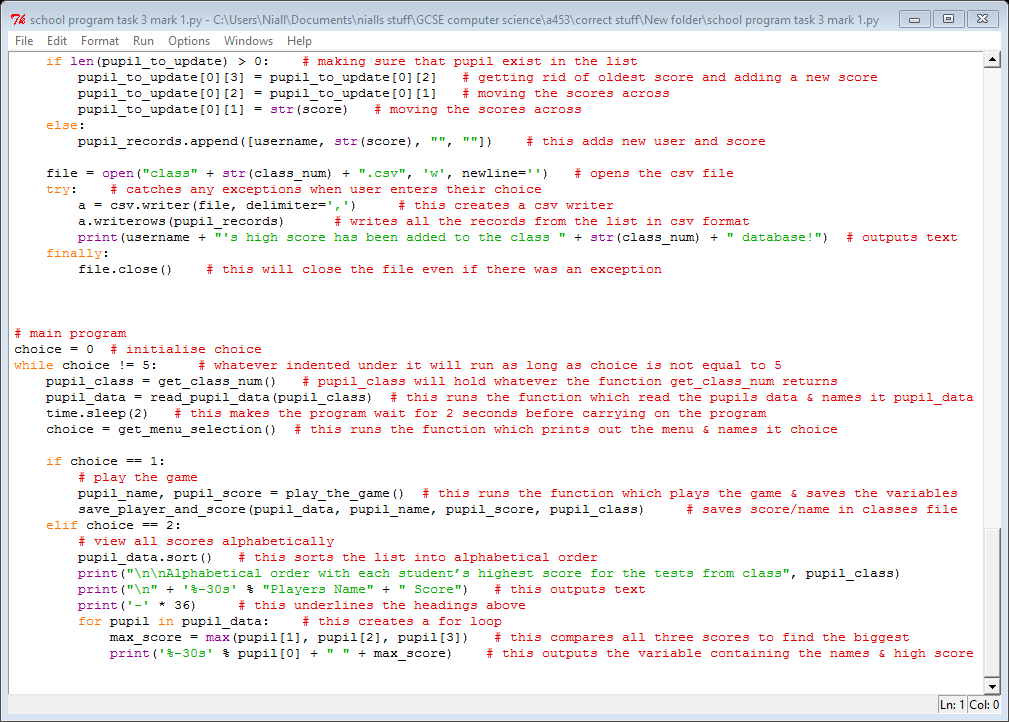
#### The program



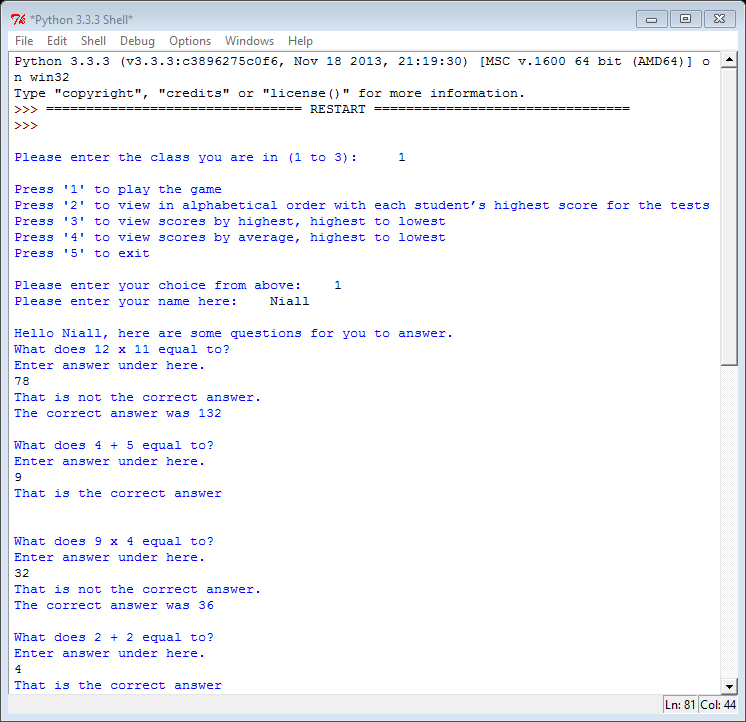




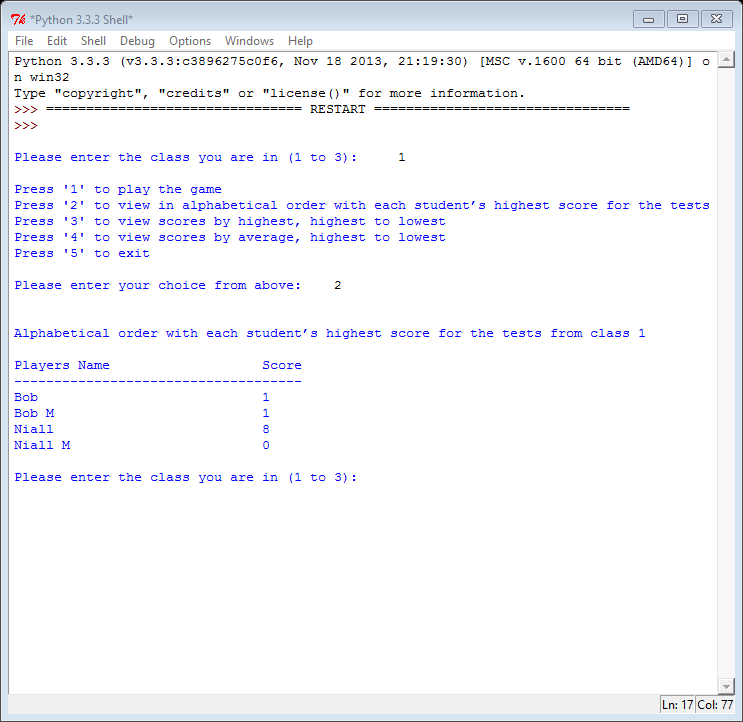




#### The program running

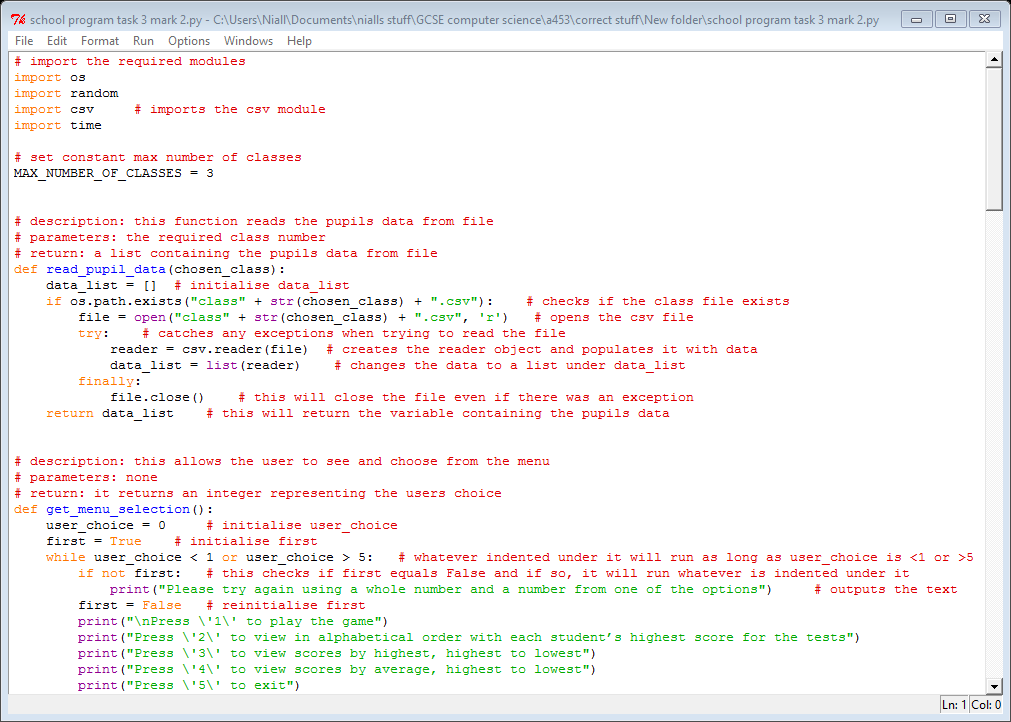


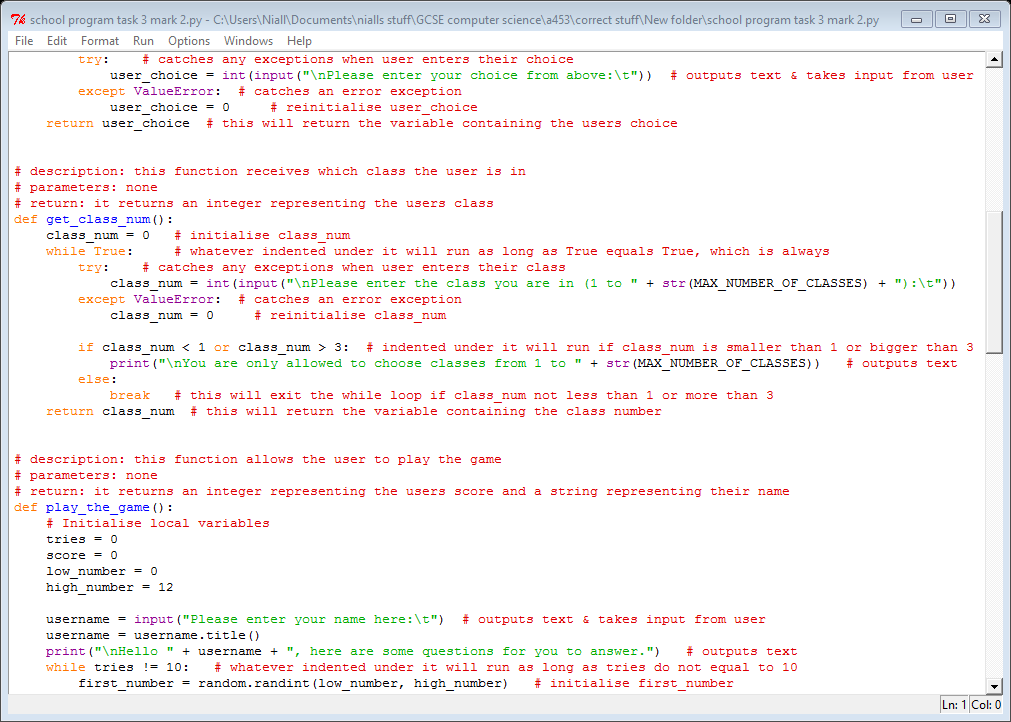


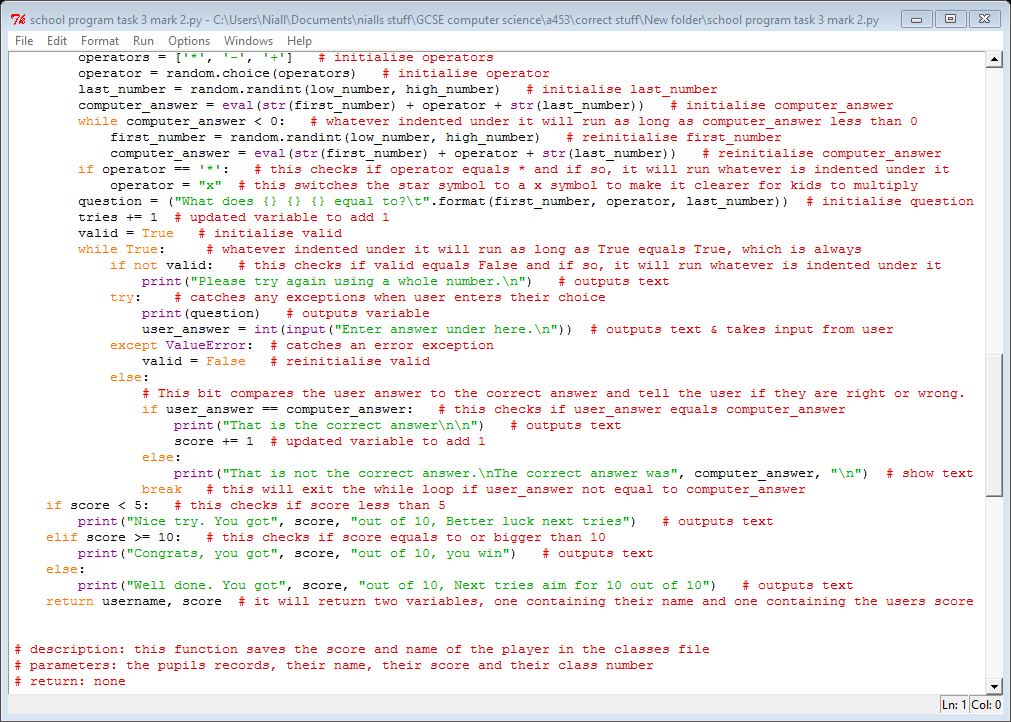


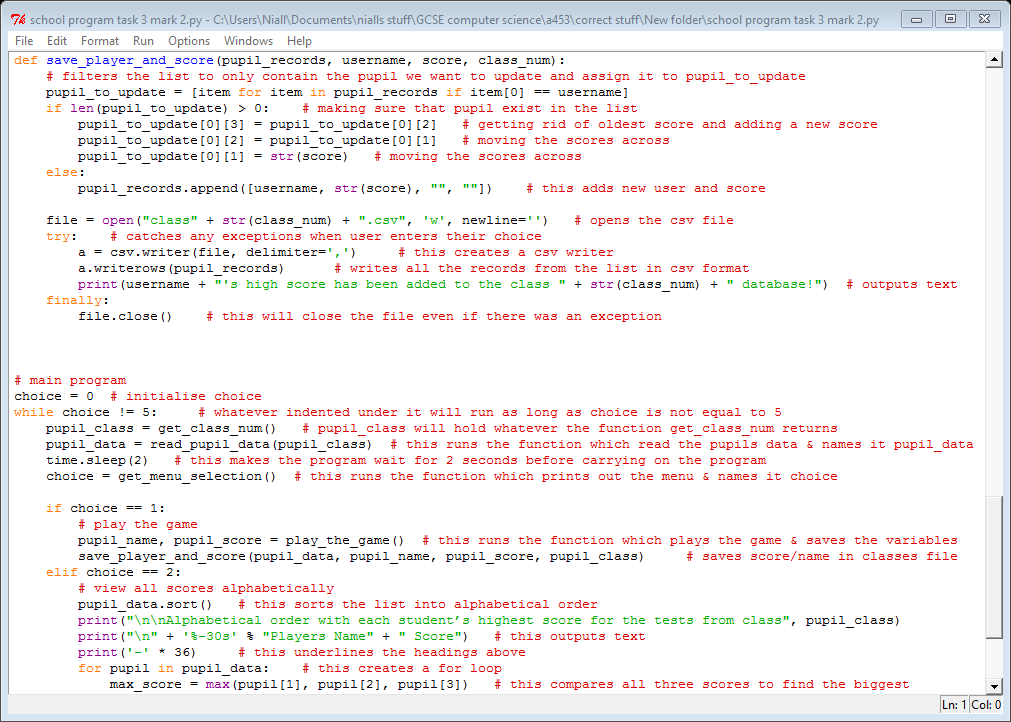
### Version 2

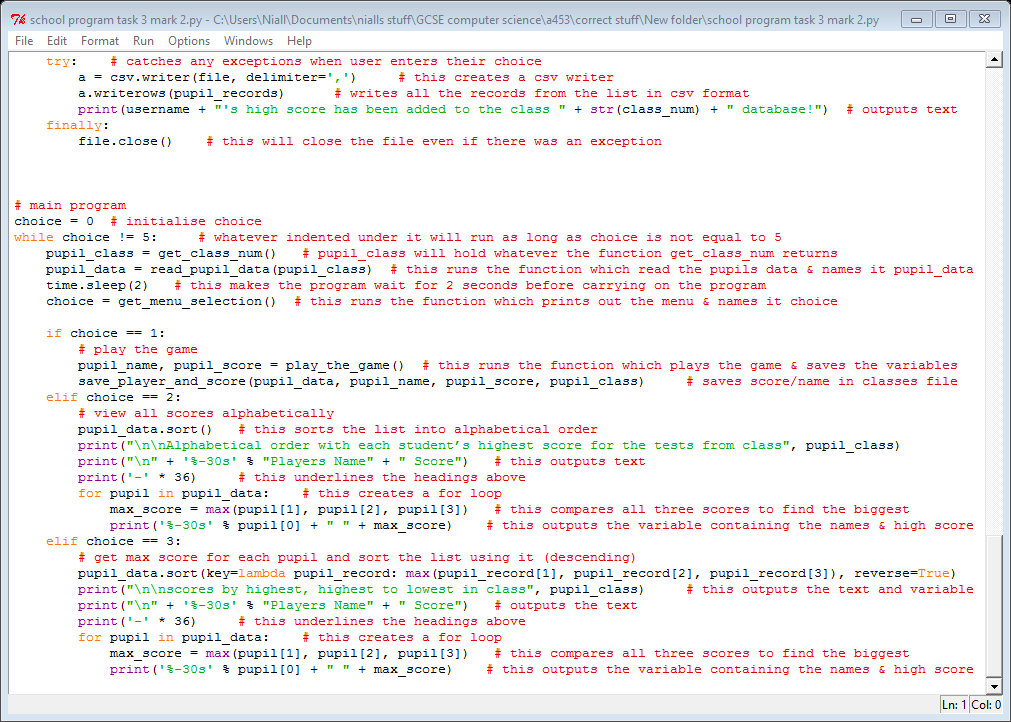
#### The program



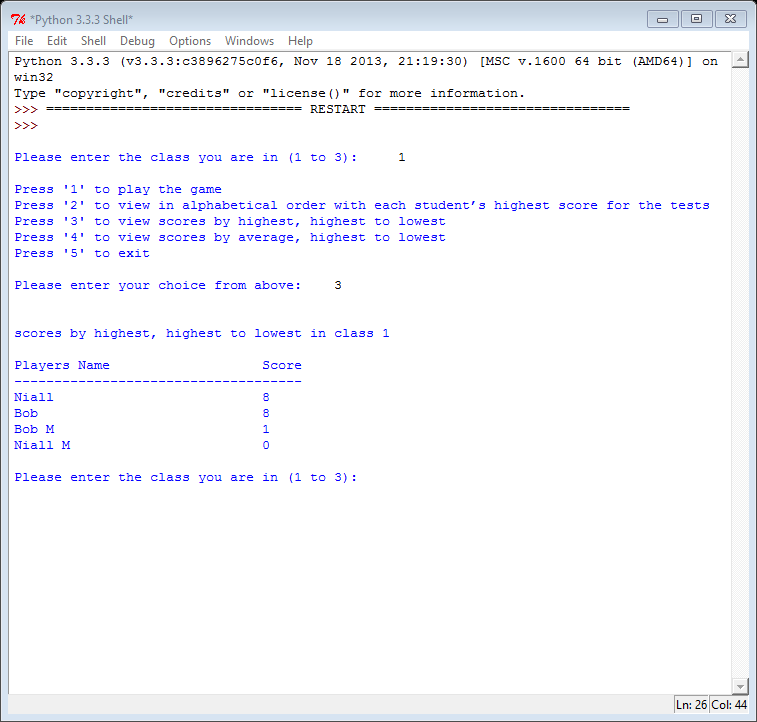






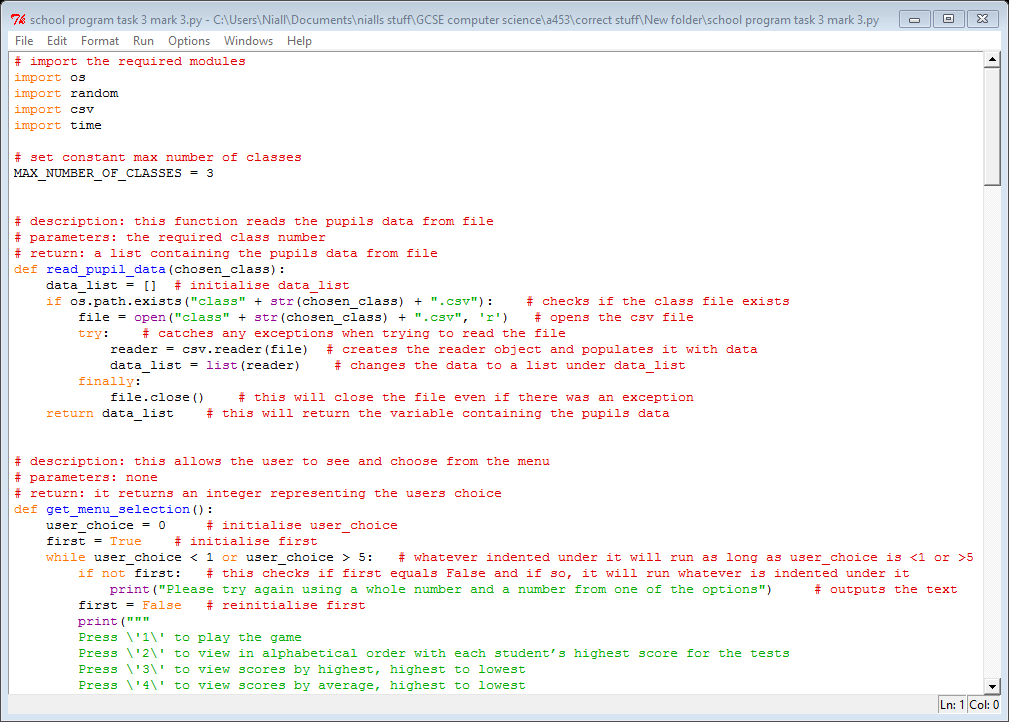


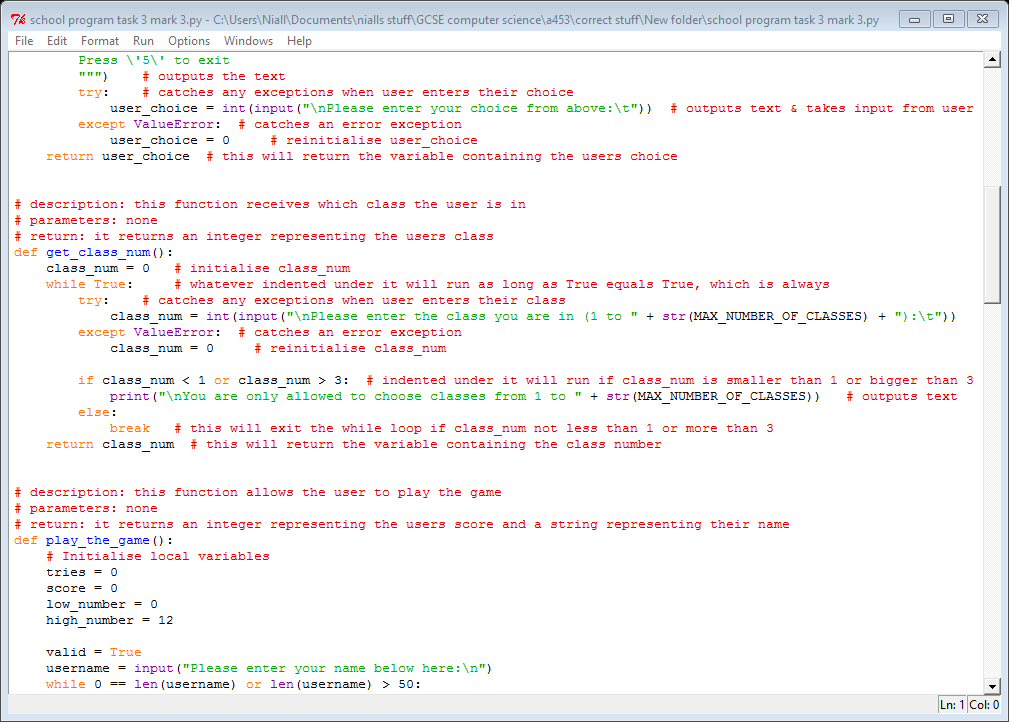
#### The program running

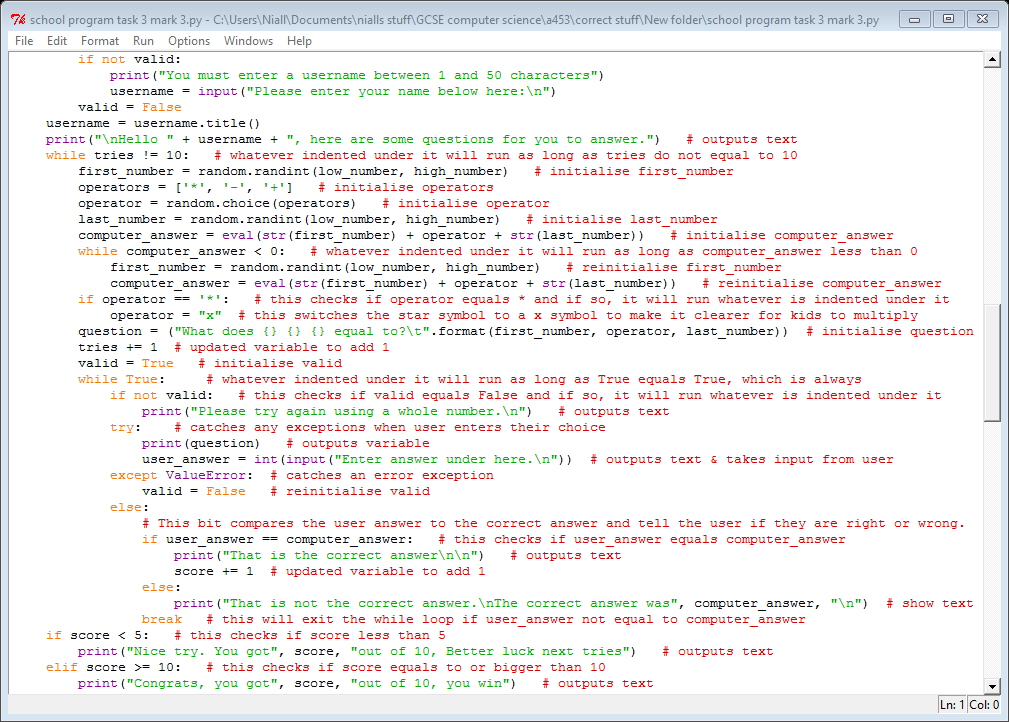


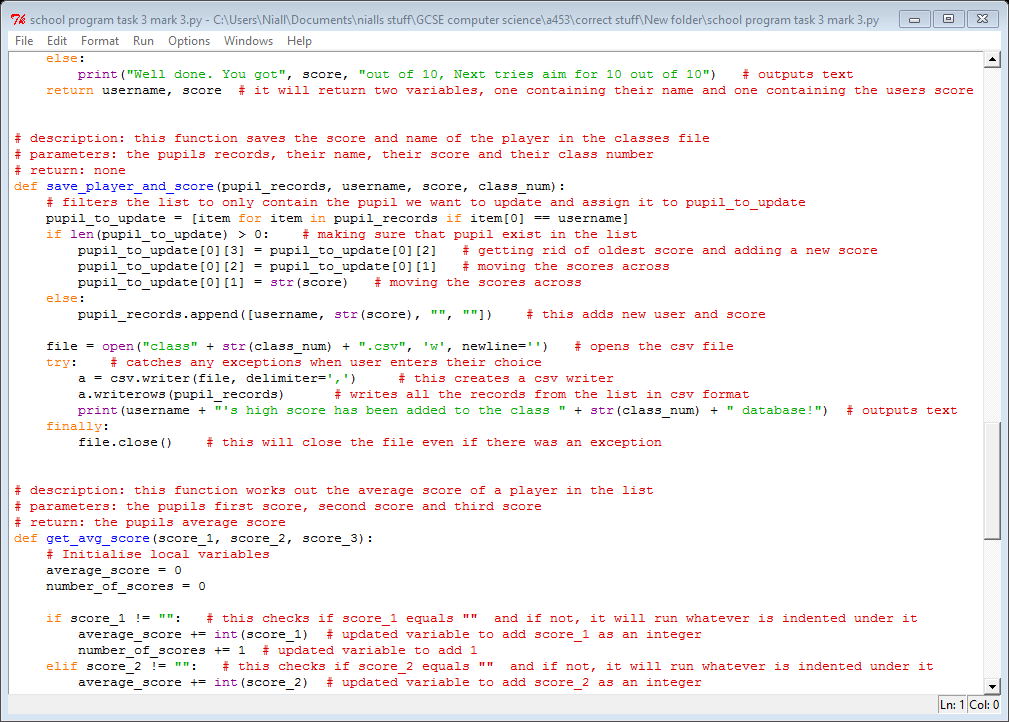
### Version 3

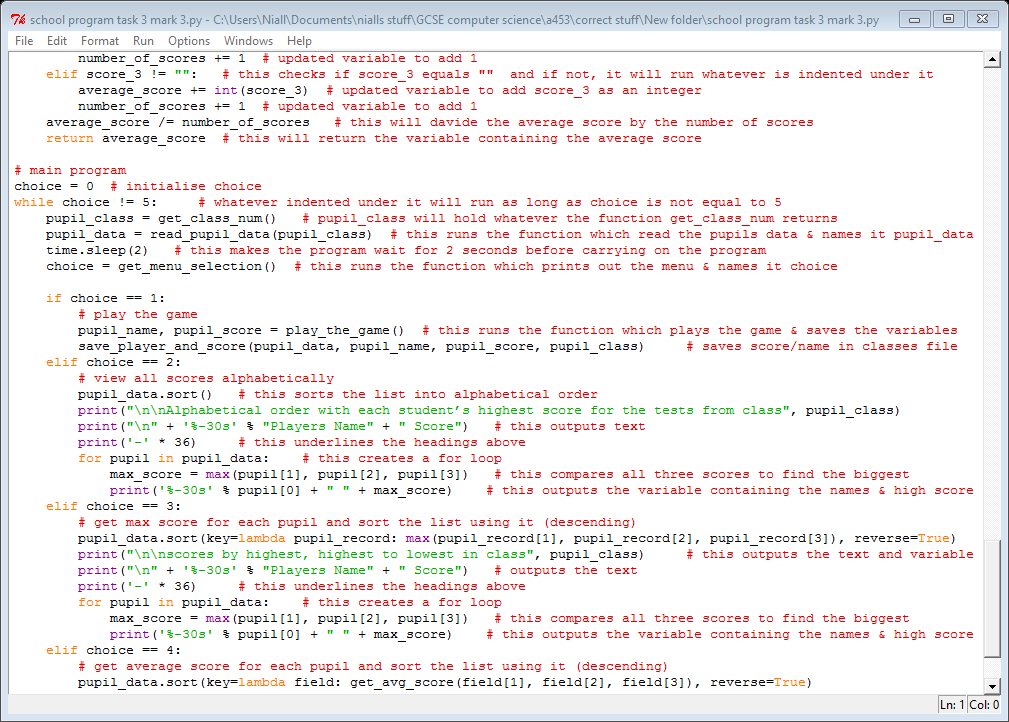
#### The program

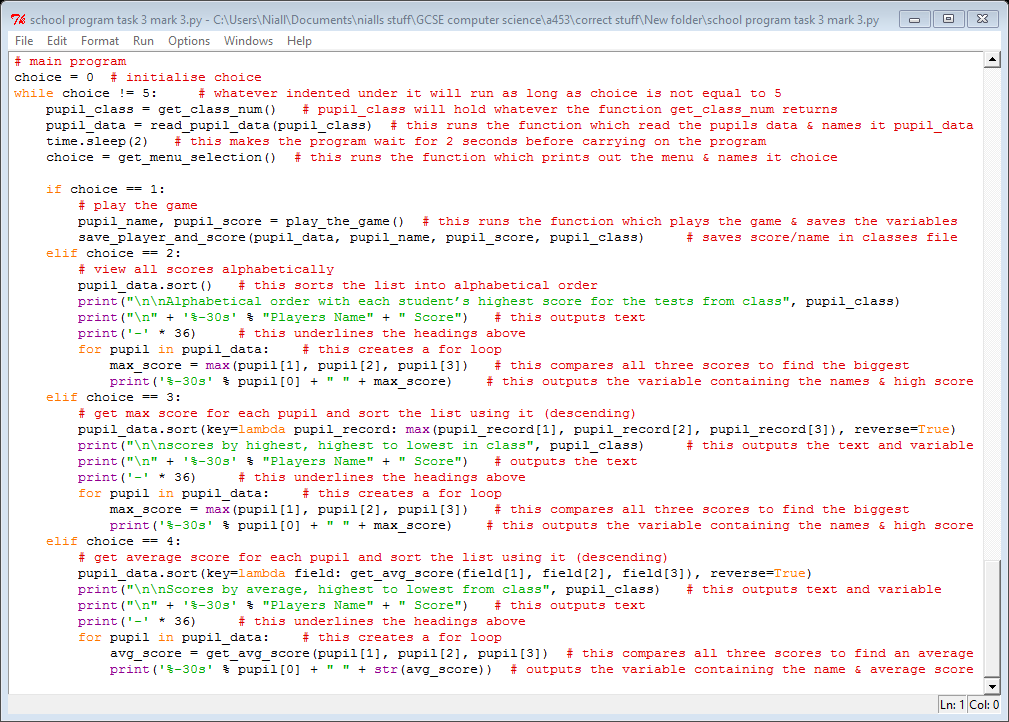




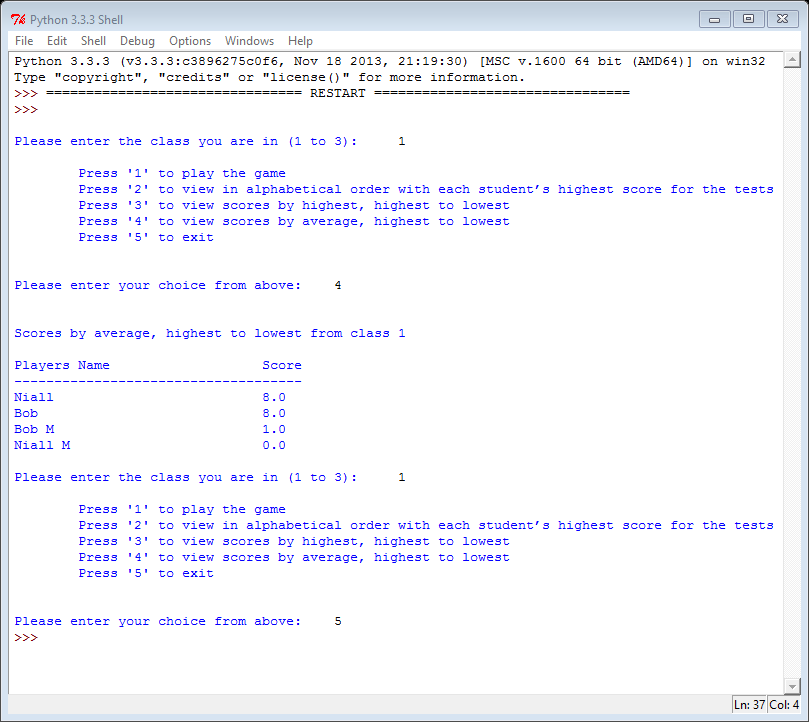








#### The program running



## Test plan

The test plan below consists of twenty separate tests. They verify all the success criteria are satisfied, that all user input is validated and results output are correct.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Test ID | Test description | Input data | Input type | Expected outcomes | Actual outcome | Corrective action | Retest outcome |
| 01 | Check if class number is valid | -1 | invalid | Data not accepted | Data not accepted | N/A | N/A |
| 02 | Check if class number is valid | 1 | valid | Data accepted | Data accepted | N/A | N/A |
| 03 | Check if class number is valid | 2 | valid | Data accepted | Data accepted | N/A | N/A |
| 04 | Check if class number is valid | 3 | Valid | Data accepted | Data accepted | N/A | N/A |
| 05 | Check if class number is valid | 7 | Invalid | Data not accepted | Data not accepted | N/A | N/A |
| 06 | Check if class number is valid | fggb | Invalid | Data not accepted | Data not accepted | N/A | N/A |
| 07 | Check if users choice is valid | 1 | Valid | Data accepted | Data accepted | N/A | N/A |
| 08 | Check if users choice is valid | 2 | Valid | Data accepted | Data accepted | N/A | N/A |
| 09 | Check if users choice is valid | 3 | Valid | Data accepted | Data accepted | N/A | N/A |
| 10 | Check if users choice is valid | 4 | Valid | Data accepted | Data accepted | N/A | N/A |
| 11 | Check if users choice is valid | 5 | Valid | Data accepted | Data accepted | N/A | N/A |
| 12 | Check if users choice is valid | 7 | Invalid | Data not accepted | Data not accepted | N/A | N/A |
| 13 | Check if users choice is valid | Jghgf | Invalid | Data not accepted | Data not accepted | N/A | N/A |
| 14 | Confirm that the users name is less than 50 characters | Niall | Normal | Data accepted | Data accepted | N/A | N/A |
| 15 | Confirm that the users name is less than 50 characters | BobbbbbbbbyBobbbbbbbbyBobbbbbbbbbbbbbbbbbbbbbbbbbby | invalid | Data not accepted | Data not accepted | N/A | N/A |
| 16 | Checks if users answer is an integer | 50 | valid | Data accepted | Data accepted | N/A | N/A |
| 17 | Checks if user answer is an integer | dfcghj | invalid | Data not accepted | Data not accepted | N/A | N/A |
| 18 | Confirm that results can be outputted in alphabetical order | 2 | valid | Displays Students names and highest score from chosen class file, with the names output in alphabetical order | Displayed Students names and highest score from chosen class file, with the names output in alphabetical order | N/A | N/A |
| 19 | Confirm that results can be outputted in highest score order | 3 | valid | Displays Students names and highest score from chosen class file, with output in order of highest to lowest score | Displayed Students names and highest score from chosen class file, with output in order of highest to lowest score | N/A | N/A |
| 20 | Confirm that results can be outputted highest to lowest by average score | 4 | valid | Displays Students names and average score from chosen class file, with output in order of highest to lowest average score | Displays Students names and average score from chosen class file, with output in order of highest to lowest average score | N/A | N/A |

## Task 3 Evaluation

### Did I reach all my success criteria’s?

|  |  |  |
| --- | --- | --- |
| Success criteria | Success criteria satisfied | Evidence |
| View class in alphabetical order with each student’s highest score for the tests | Yes |  |
| View class using their highest score, highest to lowest | Yes |  |
| View class by their average score, highest to lowest | Yes |  |
| The program should store the last three scores for each student | Yes |  |

Some of the programing in this task was easy and straight forward. This was because some of the coding was similar/exactly like the previous two tasks. This enabled me to use inputs and functions I was already familiar with. I did had some difficulties when writing the program to always to save the latest 3 scores and find out the highest and averages of scores. I was able to overcome these challenges by learning new commands like Len and max. Also the sorting of the data took some time to work out, and required me to research it and how to use lambda.

I could improve my program, if I had enough time, by creating a special menu for the teachers, which would allow them to set targets and difficulty levels for each student. I could also improve it so that when a student gets a question incorrect, the program would explain to them how to work it out correctly or by giving them a link to YouTube on how to solve the question. An improvement which could be done in the future would be to allow the program to take in scores from many classes instead of just classes 1, 2 and 3.