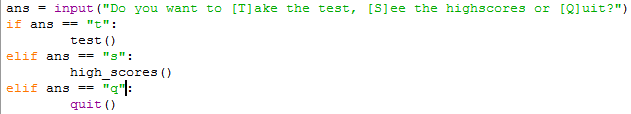
|  |  |
| --- | --- |
| Explanation of the test | Screenshots |
| You can run the test |  |
| You can quit the program |  |
| You can see the high scores |  |
| It marks the questions correctly |  |
| It records the mark correctly |  |
| It orders the scores first by mark (from highest to lowest), and then alphabetically if they have the same mark |  |
| It is not case sensitive |  |

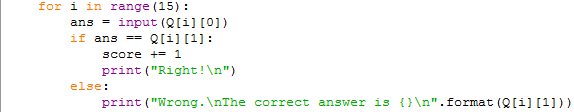
This term our task in computing was to create a gamebook (a game that allows you to make decisions in a story to affect the outcome) but I had a slightly different task. This time I had to create a maths test that can record people’s marks. I had a few problems with saving the scores but otherwise it was pretty straightforward.

I planned to create a maths test that my friends could have a go at and that it would evaluate their maths ability. What I actually made was slightly different: I made a maths questionnaire that tests you on your general knowledge of maths, rather than your ability.

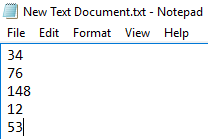
To start I made a text interface that uses simple input and conditional statements to call different functions. For example, I would use code like:



After that, I worked on the test part. The way I made it meant that I can easily change and improve the questions and answers. All of the questions and answers were stored in a two-dimensional list (a list of lists) composed of the question and its answer. When I wanted to ask the question, the program would print the corresponding value of the list. To do this, I introduced this sort of code:



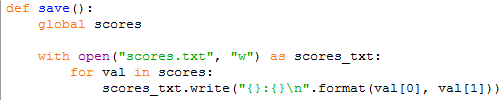
Next, I decided to do the highscore viewing and saving. For this section I encountered a few problems. The highscore viewing was pretty straightforward. I just had to read the highscore file and print it in the right order. To read it was easy, I just had to use the open function. And even though ordering it was a bit more complicated (I had to use the sorted function and the lambda statement, both of which I was unfamiliar with), I managed to get it done by scrolling through tutorials and Stack Overflow questions. However, saving the highscores was more difficult. I had to come up with a way that I could save more than 1 value. The first way was saving it like this:



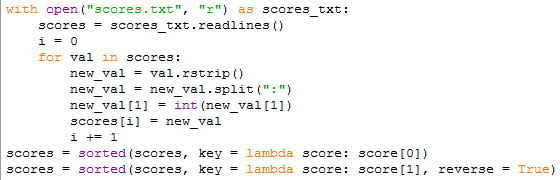
But the problem was that I couldn’t save the corresponding names to the scores. So I ended up saving them like this:



The way I did this was by using the format method to insert the values like this:



This worked fine except now I had to devise a way to read this. At first, I used regular expressions to find where the colon was and then used string slicing to separate the values but this was awfully long for such a simple thing to do. Then I found out about the split method and I used that. This is what it looked like:



The final bit to do was to actually come up with the questions that it was going to ask. For this I searched on the web for maths questions and I looked through them to find my favourites. This was probably the easiest bit of the whole project.

To do the testing I just ran through the different parts of the program that I thought were important and could get me more marks. I can’t really compare other people’s testing because they had a different assignment but I’ll just say that I think my testing might have been more detailed than others’. The tests I did were:

* You can run the test
* You can quit the program
* You can see the high scores
* It marks the questions correctly
* It records the mark correctly
* It orders the scores first by mark (from highest to lowest), and then alphabetically if they have the same mark
* The menus are not case sensitive

These tests pretty much covered what my application was supposed to do and I am quite happy with them.

In conclusion, I think that what went well was:

* The maths questions were good and tested you on your general knowledge of maths
* The program works fine with no bugs
* It saves the marks and you can view other people’s marks
* It sorts the marks in order of marks and then in alphabetical order
* The questions are randomly picked from a selection so you don’t always get the same questions

And even better if was:

* Add different difficulty levels for the test
* Add a wider range of questions
* Make the answers not case sensitive
* Categorise the questions so you get a good range of different types of questions
* If the answer is a number, to only accept numbers so there is no confusion between “9” and “NINE”

