**Interpreting results**

The *analyse\_multiple\_files.py* script works by automatically playing hangman on each of the students scripts 20 times, it outputs a summary of the results into a csv file which will be discussed here.

* Column **file\_name** is the name of the students Python script being tested.
* Column **No. wins** is the number of times (out of 20) that my algorithm was able to guess the correct word before being hung.
* Column **No. losses** is the number of times (out of 20) that my algorithm was not able to guess the correct word before being hung.
* Column **List of errors** is the error returned for each of the 20 tests (if an error occurred). It is worth noting here that these are my own custom errors and are Python strings not actual Python errors – if a python error actually occurs during the testing process then everything will stop and the results.csv file is unlikely to be created (off the top of my head this should not ever happen). There are 7 different types of return errors. To see how these errors are returned find the return statements in the *solveGame* function of the *python\_module3\_marker.py* module but here is written description:
  + A ‘Spawn error’ occurs when our Python program attempts to run the students script but cannot (*python student\_script.py* should recreate the problem in this case but in essence probably means that their script isn’t a valid Python script and so probably an auto-fail).
  + An ‘Expect error’ occurs if the students script stops without either 'Please enter your next guess: ' or *pexpect.EOF* (end of file indicator according to the *pexpect* library). This should not occur if the student has correctly followed our description of how the script should run. However, it’s possible that their script is completely functional and so could be viewed as a harsh auto-fail.
  + A ‘bfr decode error’ is likely to be caused if the last thing printed to screen before stopping was not a string. I expect that this is likely to only be caused by a significant error and so probably an auto-fail but should be watched closely in alpha testing just to be sure.
  + A ‘afr decode error’ is similar to a ‘bfr decode error’ except it is refering to the output of the script after stopping and that it is neither a string nor a pexpect.EOF object. Again I expect that this is likely to only be caused by a significant error and so probably an auto-fail but should be watched closely in alpha testing just to be sure.
  + A ‘Return error’ is caused if the output returned after a stop is a pexpect.EOF object and output before the stop is not a string that contains either ‘congratulations you win’ or ‘you lose’. This could be caused by some that would be a harsh auto-fail (e.g. misspelling congratulations) or by an acceptable auto-fail. Alpha testing should reveal our best next steps.
  + A ‘Vowel error’ should only be returned if my algorithm has tried all vowels without correctly guessing a letter. This means that either the word-list used by the student is wrong or the students script isn’t checking the user input correctly. This should be an easy auto-fail but should be confirmed in alpha testing.
  + A ‘EOF/input error’ occurs if the output after a stop in the students program is not a pexpect.EOF object or is not a string that contains ‘please enter your next guess’. This is probably an auto-fail but should be checked in alpha testing.
* Column **Result** is the result returned by the algorithm. This can be ‘pass’, ‘fail’, or ‘error’. Pass means that the students submission passed every single test carried out. Error means that the script did not perform as expected (i.e. as described in the instructions). Fail means that the students script performed as expected except the rate at which my algorithm solved the problems. This could be because the student used the wrong word list, was not picking a word randomly from a uniform distribution, or there is a small chance that this unlikely event happened by chance (‘probability that this result occurred by chance not error’ (see bullet point below) will give you the probability of this – a good way to test this is to run the test again as it would be very unlikely to happen twice in a row by chance). The easiest interpretation is to class pass as pass and fail and error as fail but could result in some harsh decisions, thus, these results do not have to be taken at face value. For example, pass could be interpreted as a good submission but some more subtle things need to be checked manually first before giving a final decision, and fail or error could be interpreted as a flag that strange behaviour occurred that needs to be manually investigated. What these mean in terms of final result need to be discussed and tested over alpha and beta testing.
* Column **Probability that this result occurred by chance not error** calculates the probability that a fail result occurred by random chance (i.e. the students submission is correct and is being unfairly failed).