November 2024

DAB410 Assignment

The assessment for this module is in two parts. Part One will be the creation of a Python programme and Part Two is written work, comprising testing and an analytical evaluation.

The deadline for submission can be found on the module's Moodle page.

Part One – Python programming

You will build and application that reads a data file regarding student grades, processes it and displays results. There are some mandatory requirements to fulfil, and optional ones (as suggestions) at your discretion for extra marks.

Mandatory calculation & display (minimum requirements for a pass)

1. Average student grade
2. Average student attendance
3. Number of fails (grades below 40)
4. Number of passes (grades 40 and above)
5. Number of grades - Cs, Bs, As.

Optional display suggestions

1. Any correlation between grade and attendance
2. Average grade by country
    a. Above in a graph
3. The display of a menu to allow the user options.
4. A search facility to find a particular student.

Areas for extra marks

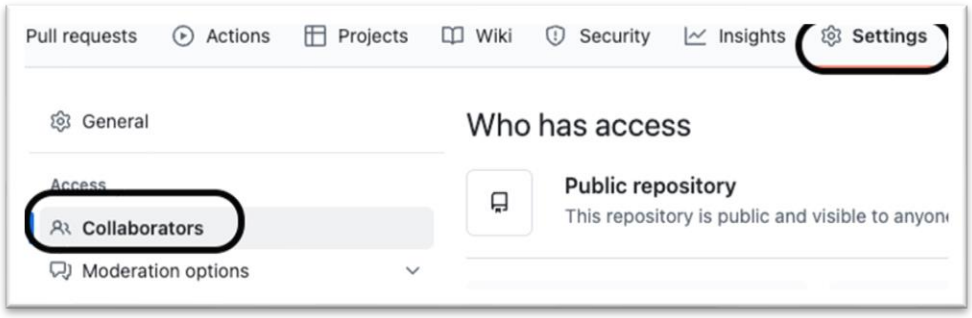
Below are areas to consider for additional marks. You do not need to implement / address all of these and you are not limited to this list. Essentially you are trying to demonstrate your skill as a Python programmer.

1. Good use of comments
2. Use of Python naming conventions
3. Effective modularisation of code using functions
4. Use of import files
5. Use of relevant Python packages (e.g. numpy, matplotlib, pandas, json, csv or others)
6. Effective error handling
7. Effective user interface – is it easy for the user to understand what to do?
8. Good use of general Python-specific techniques

9. Use of built-in Python functions
10. Good referencing or acknowledgements, where appropriate.
11. Use of a database. (only use SQLite – you will not be marked on any database schema, just the Python).

For Degree Apprentices – Git repository

Your code will be uploaded to Github.com and you will add 'roger06' as a Collaborator (Settings > Collaborators | Add) to your repository.



For top marks:

- Repository is well-structured with meaningful folders and files, and the project is easy to navigate.
- The README is well-written, clear, and contains all essential information: project overview, installation instructions, usage examples, and how to contribute.
- Git commits are well-structured, with clear commit messages that follow a logical progression (e.g., feat: add user authentication).
- Evidence of merging and / or rebasing
- Branches

Computer Science students are free to do this for experience, but it will not be marked (thus a poor effort cannot lower the mark, nor an excellent effort increase it)

While the code will be submitted via Moodle, you will also create and online repository (most likely on github.com) which will contain your working code, development branches and a history of commits, all of which will be explained and practised in class. This is optional for Computer Science students.

You will mainly be assessed on the quality and efficiency of your code, using a variety of techniques and specific Python techniques.

Note, if you work on you own computer, you may find yourself extending Python (again, which will be explained) with modules not available on University PCs. Your submitted code, MUST run on a standard University PC.

Pre-submission

Due to the nature of Python, module dependencies etc, you are strongly advised to pre-submit a copy of your project code **no later than a week** before the submission deadline. Your code MUST be able to run on University PCs. There is no obligation for the tutor to 'fix' your code nor to ask you to do so after the submission deadline.

Word equivalent: 2,400.

Part Two

Deliverables

Analytical evaluation of the coding techniques and syntax used.

Screen shots of the various system interactions, i.e. showing how the application works.

Testing Log

One of the crucial outcomes of this coursework is that you understand the importance of testing. This coursework will follow a test-driven development (TDD) cycle. That means you will write functions, then fully test them before you write any more. This process makes it easier to find bugs; you know that any bugs must have been part of the work you did since the last test. You are required to create a log of all your errors during the development of your application. This log should also identify nature of error and your fix.

Word limit: 1,600.

Evaluation of the techniques used to generate code to meet the system specification. 60%	
70 and above	Fully working, error-free application covering all aspects of the requirements. Making use of excellent programming practices and structures, (efficient code, well commented etc)

	Excellent, creative or innovative types of additional advanced features and reasoning for their choice. Evidence of an excellent understanding of online Git repositories (most likely github.com).
60-69	Working, error-free application covering most aspects of the requirements. Making use of good programming practices and structures. All requirements are addressed. Good range of additional advanced features and reasoning for their choice. Evidence of a good understanding of online Git repositories (most likely github.com).
50-59	Modest working application. One or two errors are permitted but explanation for them is required. Many requirements are addressed. Basic or very limited range of additional advanced features and reasoning for their choice. Evidence of a reasonable understanding of online Git repositories (most likely github.com).
40-49	Modest working application. Numerous errors are evident. Few requirements are addressed. Poor range of visual types of additional advanced features and reasoning for their choice. Evidence of a basic understanding of online Git repositories (most likely github.com).
<40 (fail)	Faulty code, minimal, no explanations for problems. No attempt at handling requirements is positively evident.

	Seriously flawed, absent or irrelevant additional advanced features or no reasoning (or relevant reasoning) for their choice. Little or no evidence of any effective use of online Git repositories (most likely github.com).

Part two – report and testing (40%)

Quality of documentation including lessons learnt, issues addressed and suggestions for future developments 50%	
70 and above	An excellent report, well written and backed by deep, relevant research, providing a thoroughly analytical discourse on the development process. Correct referencing used.
60-69	A well written report backed by relevant research, providing analytical discourse on the development process. Correct referencing mostly used.
50-59	A report backed by some relevant research with limited analytical discourse on the development process.
40-49	A descriptive report with limited research and limited analytical discourse on the development process.
<40 (fail)	A poorly written report, missing detail or irrelevant. No, or very limited, evidence of reading or research and lacking analysis.

Suitability and comprehensiveness and rigour of testing used 50%	
70 and above	Evidence of relevant, comprehensive, diverse testing, demonstrating an excellent understanding of Unit, Integration, System and Acceptance Testing
60-69	Evidence of relevant and diverse testing, demonstrating a good understanding of Unit, Integration, System and Acceptance Testing
50-59	Evidence of some testing, demonstrating some understanding of Unit, Integration, System and Acceptance Testing
40-49	Minimal evidence of a testing, demonstrating only a basic understanding of Unit, Integration, System and Acceptance Testing
<40 (fail)	The log fails to demonstrate sufficient understanding of any or all the main software testing elements.

Further Issues/Considerations

Your application must run on PyCharm on University computers. The inability of the tutor to execute submitted code could have an impact on the mark.

There is no need to create a GUI for this application, but if you do, that could score extra marks for advanced additional functionality.

Word limit does not include source code.

*** Important ***

This is an individual piece of work.
Whilst you are permitted to work together you MUST, however, ensure your work is a result of your own individual effort. Plagiarism when detected will be reported and action taken.
Your report must include your student number in the file name. Failure to do so will result in a 5% penalty.

Use of AI

Any use of AI generated code must be clearly stated, and, where appropriate, evidence given that is it fully understood.

Hand in date: See Moodle
Use the submission link provided on Moodle to upload your work.