Completed 15 August 2015, by Eugene Lee, for Mrs Saul, St Olaves Computing Faculty.

Alternatively, all work completed as part of this project can be accessed at github.com/labcoatlazuli/summer-programming-challenges

Programs Used

Python Version 3.4

IDE JetBrains PyCharm Community Edition 4.5.3

Version Control Git 1.9.5 for Windows

PDF Printing / Viewing Foxit PDF Reader

Helpful Resources and Other Links

GitHub.com Online Code Hosting

code2flow.com Interactive C style Pseudocode to Flowchart converter

docs.python.org Python Documentation

Comments:

A useful, if not extremely time-consuming set of challenges to complete. For the work completed (questions 1 through 8 with pseudocode and flowchart for each) I estimate I used 70-80 hours total, a large portion of which was spent debugging and improving the robustness of my code.

In regards to this, code2flow.com was an incredibly useful resource which saved me from having to draw out blocks by hand and arrange them myself, instead using a C-like pseudocode language to create the flowchart from, which is incredibly easy to learn, and a tutorial is provided.

I do however question the resolute need to make a flowchart and pseudocode for every single question, especially since flowcharts are often used to visualize program modules as an organizational tool, rather than to provide an oversight of the whole program. I propose a system of where half the questions require pseudocode and the other half require a flowchart instead. Even better, instead asking for the use of comments would allow students to explain the purpose of each section of code, which would in turn help a student organize their code for efficiency and readability, also encouraging a mindset of module-based programming so that when it comes to fixing problems, especially concerning larger programs with 150 lines of code or more, it is easier to narrow down the problem. In my particular case I did not find a particular need for comments, but that was only down to the way I had structured my code, which allowed me to quickly remember what each little bit of code was doing, no doubt helped by well-chosen variable names.

A large portion of time was also saved through the use of a professional IDE, in this case PyCharm, which has integrated debugging, code execution console, refactoring, intelligent code completion, and even support for adherence to the PEP 8 coding style guide. For such a huge amount of time needed to complete this project, PyCharm is an utterly essential tool.

I also take issue with the quality of the questions themselves. While not a huge issue in terms of ability required, some of the suggestions put forward by the programs are laughable if not downright ludicrous. For example, the password changer program stores usernames and passwords in a single unencrypted text file, which is nowhere near a satisfactory backend solution in a real-life situation. In addition, a secure password is best created using a mixture of random words strung together, creating a long and extremely secure password which is also easy to remember, as opposed to a shorter one with special characters and a mix of upper/lower case, which is harder to remember and hence inherently less secure as it is much more likely to be written down.

However the real-life practicality of the questions notwithstanding, it is understandable that some compromises had to be reached in order to limit the ability required to even complete the questions themselves.

Overall, I consider this to be somewhat of a valuable experience, as I had the chance to practice using functions, error handling, file handling etc, which is ultimately the goal of the Computing GCSE, as well as finding and deciding on a workflow.