Problem Set 2019

Programming and Scripting

Due: last commit on or before March 31st

This document contains the instructions for Problem Set 2019 for Programming and Scripting. Please note that that all students are bound by the Quality Assurance Framework [3] at GMIT which includes the Code of Student Conduct and the Policy on Plagiarism. The onus is on the student to ensure they do not, even inadvertently, break the rules. A clean and comprehensive git **??** history (see below) is the best way to demonstrate that your submission is your own work. It is, however, expected that you draw on works that are not your own and you should systematically reference those works to enhance your submission.

Instructions

The following ten problems should be completed by you over the course of the semester.

1. Write a program that takes any positive integer as input and outputs the successive values of the Collatz conjecture.
2. Write a program that prints all numbers between 1,000 and 10,000 that are divisible by 6 but not 12.
3. Write a program that reads in a text file and outputs every second line.
4. Write a program that tells the user whether or not today is a day that begins with the letter T.
5. Write a program that takes any positive integer and tells the user whether or not the number is a prime.
6. Write a program that takes any positive integer and outputs its factorial.
7. Write a program that displays a plot of the functions x, x^2 and 2^x in the range $[0, 4]$.
8. Write a program that takes a user input string and outputs every second word.
9. Write a program that that takes a floating point number as input and approximates its square root.
10. Write a program that outputs today's date and time in the format "Monday, January 10th 2019 at 1:15pm".

Submission

You must use the version control software git [1] to track your work and you will submit your problem set solutions by providing a URL to your git repository. It is suggested you use GitHub [2] for this purpose and that you consider making your repository publicly available so that prospective employers may view it. However, should you wish to, you may restrict general public access to your repository so long as you give permission to the lecturer to view it. Furthermore, any git repository URL to which you provide access to the lecturer will suffice – you don't have to use GitHub. You must submit the URL of your git repository using the link on the course Moodle page before the deadline. You can do this at any time, as the last commit before the deadline will be used as your submission for this assignment.

Any submission that does not have a full and incremental git history with informative commit messages over the course of the assignment timeline will be accorded a proportionate mark. It is expected that your repository will have at least tens of commits, with each commit relating to a reasonably small unit of work. In the last week of term, or at any other time, you may be asked by the lecturer to explain the contents of your git repository. While it is encouraged that students will engage in peer learning, any unreferenced documentation and software that is contained in your submission must have been written by you. You can show this by having a long incremental commit history and by being able to explain your code.

Minimum standard

The minimum standard for this assignment is a git repository containing a README, a gitignore file and ten Python scripts. The README need only contain an explanation of what is contained in the repository and how to run the Python scripts. A good submission will be clearly organised and contain concise explanations in the comments of each script.

Marking scheme

This problem set will be worth 50% of your mark for this module. The following marking scheme will be used to mark the assignment out of 100%. Students should note, however, that in certain circumstances the examiner's overall impression of the assignment may influence marks in each individual component.

25%	**Research**	Investigation of each problem and its solution, as evidenced by clean, efficient solutions.
25%	**Development**	Clear, well-written, and efficient code with appropriate comments.
25%	**Consistency**	Good planning and pragmatic attitude to work as evidenced by commit history.
25%	**Documentation**	Concise descriptions of solutions.

Advice for students

- Your git commit history should be extensive. A reasonable unit of work for a single commit is a small function, or a handful of comments, or a small change that fixes a bug. If you are well organised you will find it easier to determine the size of a reasonable commit, and it will show in your git history.
- You must be able to explain your assignment during and after its completion. Bear this in mind when you are writing your README. If you had trouble understanding something in the first place, you will likely have trouble explaining it a couple of weeks later. Write a short explanation of it into your submission, so that you can jog your memory later.
- Everyone is susceptible to procrastination and disorganisation. You are expected to be aware of this and take reasonable measures to avoid them.
- Students have problems with assignments from time to time. Some of these are unavoidable, such as acute family issues or illness. In such cases allowances can sometimes be made. Other problems are preventable, such as missing the submission deadline because you are having internet connectivity issues five minutes before the deadline. Students should be able to show that up until an issue arose they had completed a reasonable and proportionate amount of work and took reasonable steps to avoid preventable issues.
- Go easy on yourself — this is one assignment in one module. It will not define you or your life. A higher overall course mark should not be determined by a single assignment, but rather your performance in all your work in all your modules.

References

[1] Software Freedom Conservancy. Git.
`https://git-scm.com/`.

[2] Inc. GitHub. Github.
`https://github.com/`.

[3] GMIT. Quality assurance framework.
`https://www.gmit.ie/general/quality-assurance-framework`.