PROJECT

Programming and Scripting

Due: last commit on or before April 29th, 2022.

This document contains instructions for Project for Programming and Scripting. You are not expected to know how to do the whole project from the beginning. Rather, I expect that you research ways to tackle the project and formulate your own submission based on your investigations. Remember, all students are bound by the GMIT's Quality Framework [2] including the Code of Student Conduct and the Policy on Plagiarism.

Problem statement

This project concerns the well-known Fisher's Iris data set [3]. You must research the data set and write documentation and code (in Python [1]) to investigate it. An online search for information on the data set will convince you that many people have investigated it previously. You are expected to be able to break this project into several smaller tasks that are easier to solve, and to plug these together after they have been completed.

You might do that for this project as follows:

1. Research the data set online and write a summary about it in your README.
2. Download the data set and add it to your repository.
3. Write a program called analysis.py that:
    1. Outputs a summary of each variable to a single text file,
    2. Saves a histogram of each variable to png files, and
    3. Outputs a scatter plot of each pair of variables.
    4. Performs any other analysis you think is appropriate

You may produce a Jupyter notebook as well containing all your comment. This notebook should only contain text that you have written yourself, (it may contain referenced code from other sources). I will harshly mark any text (not code) that I feel is not written directly by you. I want to know what YOU think, not some third party website.

It might help to suppose that your manager has asked you to investigate the data set, with a view to explaining it to your colleagues. Imagine that you are to give a presentation on the data set in a few weeks' time, where you explain what investigating a data set entails and how Python can be used to do it. You have not been asked to create a deck of presentation slides, but rather to present your code and its output to them.

Minimum Viable Project

The minimum standard is a GitHub repository containing a README, a Python script, a generated summary text file, and images. The README (and/or notebook) should contain a summary of the data set and your investigations into it. It should also clearly document how to run the Python code and what that code does. Furthermore, it should list all references used in completing the project.

A better project will be well organised and contain detailed explanations. The analysis will be well conceived, and examples of interesting analyses that others have pursued based on the data set will be discussed. Note that the point of this project is to use Python. You may use any Python libraries that you wish, whether they have been discussed in class or not. You should not be thinking of using spreadsheet software like Excel to do your calculations.

Submissions

GitHub must be used to manage your project submission. Your GitHub repository will form the main submission of the project. You must submit the URL of your GitHub repository using the link on the Learnonline 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 project.

Any submission that does not have a full and incremental git history with informative commit messages over the course of the project timeline will be accorded a proportionate mark. It is expected that your repository will have lots 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 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 an incremental commit history and by being able to explain your code.

Marking scheme

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

Mark	Heading	Description
25%	Research	Investigation of each notebook and script as demonstrated by references, background information, and approach.
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. (I will be looking at the commit history in depth)
25%	Documentation	Concise descriptions and explanations of theoretical and practical aspects of problems. (This can be in the readme and/or notebooks and/or comments in the code

Advice for students

1. 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.
2. Using information, code and data from outside sources is sometimes acceptable — so long as it is licensed to permit this, you clearly reference the source, and the overall project is substantially your own work. Using a source that does not meet these three conditions could jeopardise your mark.
3. You must be able to explain your project during and after its completion. Bear this in mind when you are writing your README and/or notebook. 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 in your README and/or notebook, so that you can jog your memory later.
4. Everyone is susceptible to procrastination and disorganisation. You are expected to be aware of this and take reasonable measures to avoid them. The best way to do this is to draw up an initial straight-forward project plan and keep it updated. You can show the examiner that you have done this in several ways. The easiest is to summarise the project plan in your README. Another way is to use a to-do list like GitHub Issues.
5. Students have problems with projects from time to time. Some of these are unavoidable, such as external factors relating to 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 it. 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.
6. Go easy on yourself - this is one project in one module. It will not define you or your life. A higher overall course mark should not be determined by a single project, but rather your performance in all your work in all your modules. Here, you are just trying to demonstrate to yourself, to the examiners, and to prospective future employers, that you can take a reasonably straight-forward problem and solve it within a few weeks.

References

[1] Python Software Foundation. Welcome to python.org. https://www.python.org/.

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

[3] UC Irvine Machine Learning Repository. Iris data set.
http://archive.ics.uci.edu/ml/datasets/Iris.s