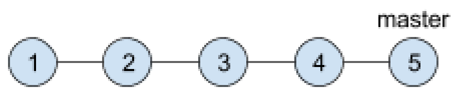
Homework 1

All assignments need to be submitted via github classroom, assignment https://classroom.github.com/a/C8TSJy77

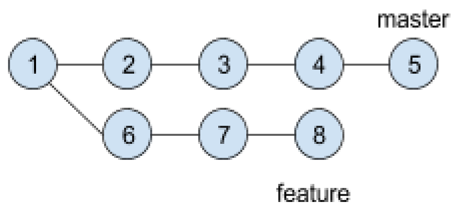
Each assignment needs to be a separate subfolder in the repository called `task1`, `task2` and `task3`. Please add a Readme.md to your repository stating your UNI so that we can identify you. Please also add links to the output of the travis run for task2.

Task 1: Git

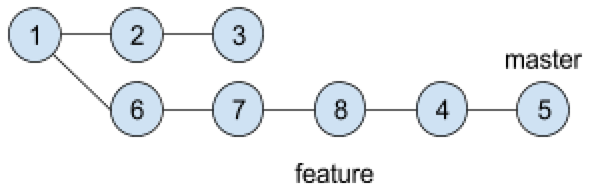
Write a shell script[1] that creates a new folder, a repository in that new folder and a series of commits as described in the following. Each commit should create a new empty file with the number of the commit as file name (1 to 10), using `touch`.
First, create a chain of five commits on the branch `master`:



Then, create three new commits starting from the first commit, on a branch called `feature`:



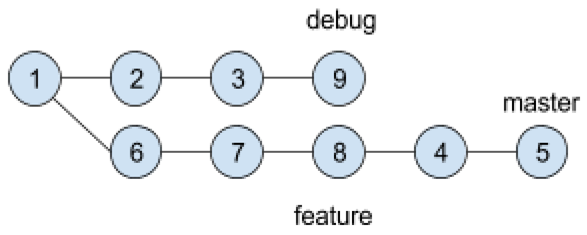
Then, move the commits 4 and 5 to the `feature` branch:



Note that after this transformation, there will be no pointer to commit 3, so "git log" will not show it any more. However, "git reflog" will.

[1] If you don't know what a shell script is, check out https://www.shellscript.sh/. Windows users might find http://www.howtogeek.com/261591/how-to-create-and-run-bash-shell-scripts-on-windows-10/ helpful.

Create a new commit attached to commit 3 in a branch called `debug`



Write the result of ``git reflog`` to ``reflog.txt`` (as part of your script) and include ``reflog.txt`` in your submission.

Task 2: Continuous integration

Ensure that all tests that are written as part of this task are run by continuous integration and pass.

2.1 Set up travisci for your repository, running for Python2.7, Python3.4 and Python3.5.
You need to enable travis for your repository in the repository settings, then you can check the status at https://travis-ci.com/applied-ml-spring-18/<your_repo_name> after pushing a commit.

You need to add at least a single test to the repository for the tests to run successfully, so don't worry if the tests fail at this point.

2.2 Create a test that ensures that two divided by eight (both represented as integers) is 0.25, once using built-in functions, once using numpy arrays of shape `(1,)`. Run the test with py.test. Push the test to travis, and ensure that travis runs and passes on the test. You need to ensure all the requirements are installed. Because travis has a very old version of pip installed, it is a good idea to run "pip -U pip" before you install anything else. See the py.test documentation for details.

2.3 Write a test that reads `input.txt` as provided in your repository and checks that the number of unicode characters in the file is 6 (not including newline). The io module might be helpful. The function should be written so that it works for any text file with a single line in it.

Task 3: Data Visualization and Analysis

The following plots need to be generated by python files within the task3 folder, called task31.py, task32.py and task33.py. Each file show both show the plot and store it as an image file (called task31.png, …). Also include the generated images in your submission.

3.1 Pick a graph from http://www.tylervigen.com/spurious-correlations and recreate it using matplotlib, numpy and pandas. The code should include reading the data from the web if possible, otherwise from a text for csv file (to be included in the repository). Ensure the axes are labeled properly.

3.2 Create a pair-plot of the iris dataset similar to Figure 1-3 in IMLP using only numpy and matplotlib (you can use scikit-learn to load the data with sklearn.datasets.load_iris, you are not allowed to use pandas). Ensure all axes are labeled. The diagonals need to contain histograms, the different species need to be distinguished by color or glyph, and there needs to be a legend for the species.

3.3 Create an array of scatter plots on the boston housing dataset (sklearn.datasets.load_boston). For each feature, plot this feature against the target MEDV. Use alpha to cope with overplotting. Ensure everything is labeled properly and the resulting png can easily be read and understood.