COMPSCI-2XC3 : Lab 1

Due Friday, January 21st, 11:59pm

Submit the assignment on Avenue. Note, Avenue must receive your LAB by the due date. **Do not leave your submission to the last minute! Late submissions, even by seconds, will not be graded.** You have been warned.

This lab is worth 1% of your final grade in the course. Read this document carefully and completely.

Purpose

This is not meant to be a stressful or intensive lab. The goals of this lab are as follows:

1. Create and register your official lab group for the rest of the term.
2. Set clear expectations between yourself and your group members.
3. Re-familiarize yourself with simple Python functions.
4. Familiarize yourself with Git.

Submission

You will submit your lab via Avenue. You will submit your lab as a .zip. Within your .zip there will be two documents:

- code.py
- report.pdf (or docx, etc.)

Only one member of your lab group will submit to Avenue – see the section below for more details on that. Your report .pdf will contain all information regarding your group members, i.e. name, students number, McMaster email, and enrolled lab section. This will be given on the title page of the report. For the remainder of this document, read carefully to gauge what other material is required in the report. Your report should be professional and free from egregious grammar, spelling, and formatting errors. Moving forward in this course, you may lose grades if this is not the case.

Forming Your Group

The first step to you being successful in this course is creating your lab group. If you have not created a lab group yet, reach out to your colleagues to do so. There is a Group Finder channel on our MS Team which you may wish to use. Consider the following rules when forming groups:

- Groups must have 2-3 members
- Members of a group may be enrolled in different lab sections. Furthermore, members of a group may be also be enrolled in different courses (SE or CS 2XB3).
- Once your group is formed, determine a member of the group to be your "contact member". The contact member will the be person which submits the labs.
- You must officially register with a TA/lab section. You may contact TAs via email or Teams. You are not registered with a TA until they give you a confirmation. Keep a record of this confirmation.
- When registering, give the TA as much information as possible about your group members, sections, ids, emails, etc. Also let them know who the contact member is. If you have already registered with the TA, send them a followup email with this information if necessary.
- The TA you register with must be a TA associated with one of your group members' enrolled lab sections.
- The TA you contact may refuse to register your group due to being at capacity (this is to decrease variance across the different lab sections). In this case they will ask you to register with a different lab section – this different lab section must still be one in which a member of your group is registered.
- If you cannot find a group, or you have a two member group and you wish to potentially have a third member, reach out to your TA. Early this week, they will assign you to a group. **Note: I would really prefer you make groups on your own.**

When first contacting members of your group, use McMaster email. If they do not respond keep a formal record of this. **If it is found that you have not been responding to legitimate communications from your group members you will receive a 0 on this lab, as well as any other labs in the future with the same issue.**

Now that your group is formed and registered you can begin the actual lab. Within your lab report create an informal contract between you and your members. This contract will state any expectations you have on your members, what times each of you are available to work on the lab, any tools you will be using, etc. For example, the contract could contain something like:

- We will primarily be using Teams to communicate (as a side note here you can use whatever you want for communication, Discord, emails, etc.).
- All members must respond to a post which they are *mentioned* in within 24 hours.
- During 8:30am-11:30am on Mondays, all members must respond to a post they are *mentioned* in within 10 minutes

Have all members sign or agree to this contract. Keep some sort of formal record of this agreement. It could be digital signatures directly on the lab report, saving an email chain, etc.

Version Control

Remember, this is an experiential learning course. That is, you are meant to encounter some problems and solve them on your own with actual hands on *experience*. Having said that, your TAs are there for a reason. Feel free to ask them questions.

Create a Git repository. This can be done via Bitbucket (if you wish) like we saw in lecture. Ensure all members have access and can push/pull from the repository. Have each member create a file `<name>.txt` write something in it and push it to the repository. Via screen-shots, show that each member can push their local file, and pull their colleagues files.

Do some research on the git commands revert and reset. Using the files you pushed, run some experiments with these commands. In your report explain the functionality of each and show via screen shots some of the experiments you ran. In what sort of a scenario would you want to use revert? When would you use reset?

Have one member create the (for now) empty file code.py and push it. Have all members pull from the repository and be up-to-date before completing the next step. Each member will locally work on code.py by implementing the following function (note, some things are left intentionally vague).

- Create a function named: `are_valid_groups`
- This function will take in a list of student numbers and a list of groups, and return True if and only if every student number in the list appears in one of the groups.
- Groups will be represented as a nested list of student numbers.

Once each group member has completed the function, have them push the file. There should be conflicts here. Depending what you wrote, you may have to manually resolve them. Resolve them. Document your merge and include it in your report – describe what you did. Once the conflicts are resolved have all members pull and be up-to-date.

You will now take turns modifying the code. Choose a member to modify the code, this member will be called the "player" the other member(s) will be their adversary(ies). The player will change `are_valid_groups` so that all student numbers are represented by Strings (if this is not yet the case), and also change the functionality such that:

- The function returns True if and only if every student number in the list appears **exactly** once **and** all groups are between 2-3 members in size.

The adversaries' job is to also change this file in anyway they wish to annoy the player when they try to push it. The adversaries will always get to push their code before the player does. The player must resolve the potential conflicts the adversaries made for them. Document all these steps and include it in the report. Once the player has resolved the conflict, reset to

when the player was selected. Select a new player and repeat until all members have been players and adversaries.

Write a paragraph in the report discussing what your learned about conflict. Which ones are easy to resolve, which are difficult, what are some different ways you can resolve them, etc. Congratulations, you have completed the first lab. Remember to submit before the deadline.