**COMP491 Assignment HW1: Slack, wiki, Git**

This assignment is worth a small amount of credit (20 points) and will be graded on completion only.

**Part A: Slack**

Slack is “A messaging app for teams”. Slack helps teams working on projects to do all communications within a single app rather than spreading information across e-mail, IM, texts, etc. Everything posted on Slack is indexed and searchable, making it easy to go back and find information and resources or reconstruct ideas and processes. We will be using Slack as our primary communication mechanism for this course. You can install it locally or use it via a browser.

Complete at least the first four tasks below, and use your own initiative and interest to further explore the features of Slack.

1. Skim the [What is Slack?](https://get.slack.help/hc/en-us/articles/115004071768-What-is-Slack-) page and/or watch the video (1:13) to familiarize yourself with Slack.
2. Use the link [comp491.slack.com/signup](https://comp491.slack.com/signup) to sign up directly for the COMP491 Slack Workspace.
3. [Edit Your Profile](https://get.slack.help/hc/en-us/articles/204092246-Edit-your-profile) to include at least your first and last name. You can also upload a photo or avatar if you’d like.
4. Using the links on the [Using Slack](https://get.slack.help/hc/en-us/categories/200111606) page do the following:
   * Join the #tips-and-techniques, #slack-intro and #comp491-announcements channels.
   * Create a *private channel* using your name + “live log”
   * Set the *description* for your channel to be something meaningful.
   * Add the instructor to your privatel live log channel

[remaining activities are optional]

1. Using the links on the [Using Slack](https://get.slack.help/hc/en-us/categories/200111606) page do the following:
   * Post a message to your private channel (only you will be able to see it).
   * Post a message to the #slack-intro channel
   * Edit the message you just posted
   * *Star* your private channel and note where your starred channels appear.
   * Star a message in the #slack-intro channel channel and note where your starred messages appear.
   * Unstar the message you just starred in the #slack-intro channel.
   * Add a *reaction* to a message in the #slack-intro channel.
   * Team up with a classmate or two:
     + *Mention* a member of the class in a message in the #slack-intro channel.
     + Find the list of your *mentions* and *reactions*.
     + Send a *direct message* to someone.
     + Reply to a direct message from someone.
     + Start a *huddle* with someone from a direct message.
2. Using the Slack search feature do the following:
   * Find the message that talks about “Linux Foundation’s Open Source Jobs Report.”

**Part B: wiki**

A wiki provides a way to collaboratively create, modify and structure the content of a web site directly through a web browser. It is very useful when a large group of people need to contribute directly to the creation and maintenance of the information on a site. Many FOSS projects use wikis for design documents and end-user documentation so that lots of people can contribute to these resources. We will use a wiki to maintain information about the projects you are working on in the course and to track your progress on the projects. Complete the following tasks.

1. Log into the Moodle for the course.
2. Open the “COMP491/492 wiki”
3. Go to the “Individual wiki pages” page
4. Edit this page and add a link to your own new Individual wiki page.
5. Return to the View mode and click the link to go to your personal wiki page (it will be blank). Now edit this page. Use the instructor’s page as a guide and consult the nwiki Markup guide linked in the Moodle for formatting information. You must include at least:
   * Your slack handle
   * Your GitHub username
   * Any other contact information you would like others in the class to have.

NOTE: On our wiki, anyone can edit any page. This is great for collaborative creation of the site, but it also comes with great responsibility. Please do not edit anyone else’s personal pages. Keep in mind that the wiki software logs all changes and the user that makes them. Have a look at the “History” tab if you are curious.

**Part C: Git review**

Spend some time reviewing your knowledge of the version control system Git. Complete the first six activities below at a minimum.

1. Fork the *upstream* repo <https://github.com/dickinson-comp491-fall2023/git-practice>. We’ll refer to your forked repo as the *origin* repo.
2. Clone your fork (so now you have a local copy).
3. Create a new feature branch.
4. Add your name to the Python list in the file comp491.py.
5. Create a new Python file that does something simple (one or two lines or even just a print statement would be fine). Include your username in the file name. We’ll refer to this as your *individual* Python file. You can use it for further experiments later, without worrying about conflicting changes from others.
6. Commit your changes, push them to your origin, and create a new pull request back to the upstream repo. If there are merge conflicts or other difficulties, work with the instructor and/or other students to resolve these.

[Remaining activities are optional]

1. Check that the instructor has accepted your pull request within a couple of days.
2. Pull the most recent changes from the upstream to your origin and local repos.
3. Create two different branches and make some non-conflicting changes your individual file, with different changes in each branch. Commit all your changes. Now try merging one branch into the other. Push to origin then create a pull request that includes all of your changes.
4. Repeat the previous exercise, but deliberately create conflicting changes in the two branches. When you merge one branch into the other, you can practice resolving conflicts.
5. [Optional advanced topic: squashing commits] Create a new feature branch and make a sequence of small changes, committing each one. Now create another new feature branch. Merge all of the changes from your first branch into the new feature branch, but squash the commits. Push this to the origin and make a pull request for these changes, noting how the maintainer only has to deal with one commit because you have squashed your commits.
6. [Optional advanced topic: rebasing a branch] Create two new feature branches and make several new commits in each branch. The first time you do this, make sure there are no conflicts between the branches. Read about the git rebase command. Now rebase one of your branches onto the end of the other one. Once this is working correctly, try the exercise again but with a minor conflict in the changes.

Acknowledgment: Parts A and B written by Professor Grant Braught with edits by John MacCormick.