# Activity 1.4 – Practice with git and GitHub Desktop

### Part 1 – Practice with Test Driven Design

First, we will practice creating function on Colab, but this time using [Test-driven design](https://en.wikipedia.org/wiki/Test-driven_development), which involves creating our test function first, then designing the simplest function that passes all tests.

**Tasks.** Open a new Python 3 notebook on Colab and perform the following steps. Include screenshots of your code and the resulting output for each step.

1. Suppose we plan to create a function names triangle\_area to compute the area of a right triangle with base b and height h. Let’s define an “empty” version of this function using …

A black background with blue text

Description automatically generated

1. Before you write this function, create a test function using assert statements that checks if a function with this name correctly computes the area for several cases. Be sure to cover some *edge cases*, e.g., having a side of length zero.

A computer screen with a question mark

Description automatically generated

1. Verify that the “empty” function fails your tests.
2. Now edit our lambda function so that it solves this problem and use the test function to verify the correctness.
3. Convert the lambda function definition of your function to a def statement definition and add an appropriate doc string.
4. Test the newest version of your function as follows.
   1. Use the test function to test the correctness of this new version.
   2. Verify that your doc string prints when calling help.

### Part 2 – Practice creating a new repository

Next, we will house our new function in a repository.

**Tasks.** Open GitHub Desktop and complete the following tasks. Include screenshots for each part that asks for verification.

1. If you haven’t already, please download and install Visual Studio Code from [https://code.visualstudio.com/.](https://code.visualstudio.com/)
2. Create a new local repository using GitHub Desktop on your Desktop named act1\_3\_repo. Initialize a README.md when creating the repository.
3. Verify that this local repository corresponds to a folder on your desktop that contains the README.md file.
4. Use GitHub Desktop to initialize a remote GitHub repository. Verify the contents on GitHub.
5. Open the README.md in Visual Studio Code and make some edits to reflect the nature of the code base.
6. Now return to GitHub Desktop and verify that the changes we made to the repository are marked in the list of changed files. Commit these changes.
7. We want to download two versions of the notebook from part 1 into this repository. Starting at your Colab notebook, go to **File > Download** and download both notebook (.ipynb) and python script file (.py) version of your notebook.
   1. Open each in Visual Studio Code and verify the contents.
   2. Commit each file to the repository.
8. Push the current repo to GitHub and verify that the newest changes now reside on GitHub.

### Part 3 – Multiple local copies

Next, we will explore a few more features of git. Use your OS to copy and paste the folder containing your repository and perform the following tasks.

1. Edit and make changes to the README.md file in one of the local repositories. Commit the changes. Verify that neither the remote/GitHub repo nor the copy of the local repo have these changes.
2. Push the changes to GitHub. Verify that the remote repository is now up to date, but the second local copy is not.
3. Add the second copy of the repo to GitHub Desktop and pull the changes from the previous part from GitHub. Verify that your changes are now reflected in this repository.
4. Finally, use GitHub Desktop to *clone* a third copy of the repository. To do this, you should
   1. Click on the **Code** button and copy the repo address from GitHub by clicking on .

A screenshot of a computer

Description automatically generated

* 1. Go to **Add**, select **Clone Repository**.

A black and orange arrow pointing to a black bar

Description automatically generated

* 1. Switch to the **URL** tab and insert the copied URL from part **i.**

A screenshot of a computer

Description automatically generated

* 1. Use Finder/Files to verify the contents of this third local copy.

### Part 4 – The hidden .git folder

Finally, we will show off where the git history is stored.

1. Unless you have changed your settings, you probably can’t see some hidden files. Go to your local repositories and determine if you can see a sub-folder labeled .git. Report what you find.
2. By default, both PCs and Macs hide some files, particularly those that start with a dot. Perform a web search to determine how to show hidden files on your Mac/PC. Make these changes to your settings.
3. Inspect one of your local repositories and verify that there is a folder named .git. Take a peek at the contents of this folder and report what you found.
4. Note that it is the .git file that MAKES a folder a repository. If you ever want to remove the “git stuff”, maybe to (A) move the content of a folder to another repository or (B) make this folder a subfolder of a larger repository; you can simply delete this folder. You will retain the current contents of this folder but lose the history. *Be sure things are backed elsewhere to be safe.* To test this out, you should
   1. Delete the .git folder from one of your copies.
   2. Go to GitHub desktop and see what happened to that repository. Report what you find.
5. Finally, let’s clean things up by deleting the two extra local copies of the repository (be sure to keep the first!).

### Part 5 – The .gitignore file

Another useful hidden file is the .gitignore file, which is used to determine which files, and folder should be ignored when listing changes. To see how this works, perform the following tasks.

1. Use Visual Studio Code to create a new text file in your first repo named ignore\_this\_file.txt.
2. Create a subfolder named ignore\_this\_folder.
3. Verify that both items are currently listed as changes on GitHub Desktop.
4. Use Visual Studio Code to create a new text file in your first repo named .gitignore. Save this file in the root directory of your local repository.

A screenshot of a computer

Description automatically generated

1. Add the following two lines to your .gitignore and save the changes. DONNOT add a file extension!



1. Commit and push your .gitignore file to GitHub.
2. Go back to GitHub Desktop and verify that the file and folder created above are no longer in the list of changes.

### Part 5 – Report what you have learned

Write a paragraph or two discussing what you have learned about git after working through this exercise. Finally, write a paragraph discussing anything about git or GitHub that you still find confusing or don’t understand.