Creating a Local Git Repository

In this activity, we will be walking you through creating and using a local Git repo. You will be asked to add screenshots for each question, and you can include one full screenshot per question, but grading will be based on the specific commands used for each sub-question.

Question 1 (2pts)

[1-1] Make sure you've completed the Git set-up steps! Check if Git is installed on your system by typing **(1pt)**

```
git --version
```

```
(base) sandrawiktor@Lotus ~ % git --version
git version 2.33.0
```

[1-2] If this is your first time using Git, configure git with your name and email.

```
git config --global user.name "<your_name_here>"
```

```
git config --global user.email "<your_email_here>"
```

[1-3] Create a folder on your system by navigating to your intended directory with "cd <directory_name>" and typing "mkdir github_practice." Then, navigate to your folder with "cd github_practice." **(1pt)**

```
(base) sandrawiktor@Lotus ~ % cd Documents
(base) sandrawiktor@Lotus Documents % mkdir github_practice
(base) sandrawiktor@Lotus Documents % cd github_practice
```

Provide a screenshot of the above steps for your submission!

```
C:\Users\Caleb Kronstad>git --version
git version 2.50.1.windows.1

C:\Users\Caleb Kronstad>cd ..

C:\Users>cd ..

C:\>cd Development

C:\Development>cd Classes

C:\Development\Classes>cd ITSC-3155

C:\Development\Classes\ITSC-3155>mkdir github_practice

C:\Development\Classes\ITSC-3155>cd github_practice

C:\Development\Classes\ITSC-3155\github_practice>
```

Question 2 (2pts)

[2-1] Initialize a Git repository with the "git init" command. **(1pt)**

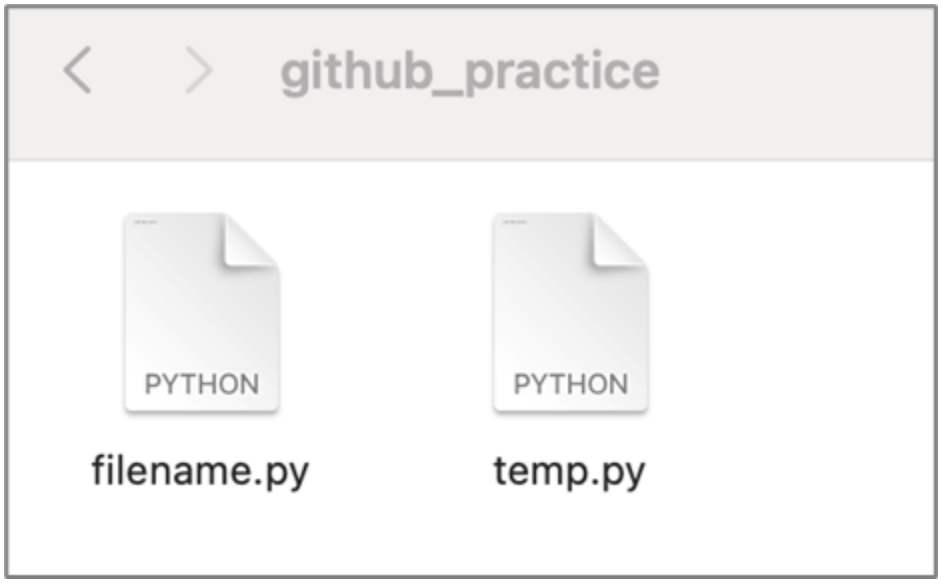
```
(base) sandrawiktor@Lotus github_practice % git init
hint: Using 'master' as the name for the initial branch. This default branch nam
e
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /Users/sandrawiktor/Documents/github_practic
e/.git/
```

[2-2] Create a .py file using "touch filename.py" in your directory. Open the file and write a basic Python command like print("hello world"). Save the file and close out.

[2-3] Create a .py file using "touch temp.py" in your directory. Open the file and write another basic Python command like print("hello world2"). Save the file and close out.

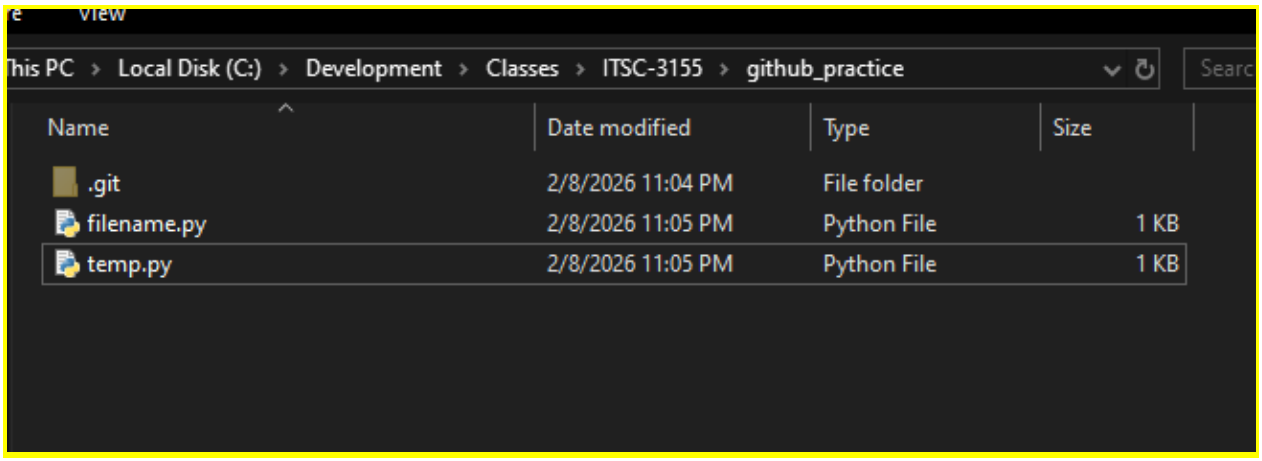
```
(base) sandrawiktor@Lotus github_practice % touch filename.py
(base) sandrawiktor@Lotus github_practice % touch temp.py
```

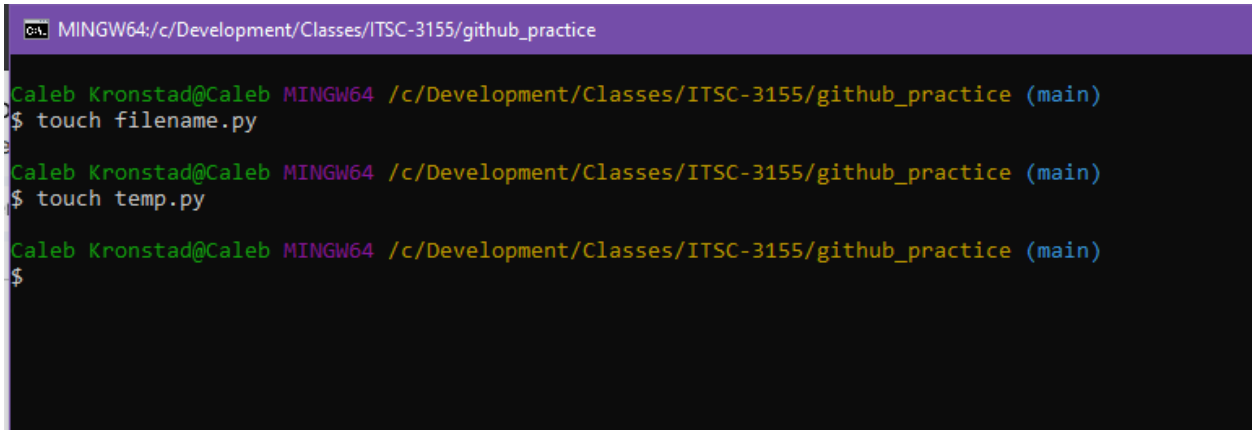
Your folder should look like this.



(1pt)

Provide a screenshot that shows your files (1pt) and use of the commands (1pt)





Question 3 (4pts)

[3-1] Type "git status." With this command, we can check if any of our files are part of our Git repo. **(1pt)**

```
(base) sandrawiktor@Lotus github_practice % git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        filename.py
        temp.py

nothing added to commit but untracked files present (use "git add" to track)
```

> Hmm... It doesn't seem that way. However, we can see that Git is AWARE of the files. We haven't added it to the repo yet though. The files in our working directory that have not been added to the repo are considered *untracked.* The files that Git knows about *and* has them added to the repository are considered *tracked.* How can we track our files?
>
> We need to add them to the staging environment.

[3-2] Read through this section to learn more about the Git staging area and commits.

[3-3]

Add the "filename.py" file to the staging area using the command "git add filename.py." **(1pt)**
Then, to check what's in our staging area, we can use the command "git status" again. **(1pt)**

```
(base) sandrawiktor@Lotus github_practice % git add filename.py
(base) sandrawiktor@Lotus github_practice % git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   filename.py

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        temp.py
```

[3-4] Now, we want to *commit* our new file. Each commit is like a save point we can go back to if we want to make any changes. We should also always include a message describing the type of commit we made.

We can commit with the following command: **(1pt)**

```
(base) sandrawiktor@Lotus github_practice % git commit -m "Added filename.py"
[master (root-commit) 39eb0c9] Added filename.py
 1 file changed, 1 insertion(+)
 create mode 100644 filename.py
```

Go ahead and commit the new file.

Provide screenshots of each step in this part for credit!

```
Caleb Kronstad@Caleb MINGW64 /c/Development/Classes/ITSC-3155/github_practice (main)
$ git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        filename.py
        temp.py

nothing added to commit but untracked files present (use "git add" to track)

Caleb Kronstad@Caleb MINGW64 /c/Development/Classes/ITSC-3155/github_practice (main)
$ git add .

Caleb Kronstad@Caleb MINGW64 /c/Development/Classes/ITSC-3155/github_practice (main)
$ git commit -m "added files"
[main (root-commit) 9387212] added files
 2 files changed, 2 insertions(+)
 create mode 100644 filename.py
 create mode 100644 temp.py

Caleb Kronstad@Caleb MINGW64 /c/Development/Classes/ITSC-3155/github_practice (main)
$
```

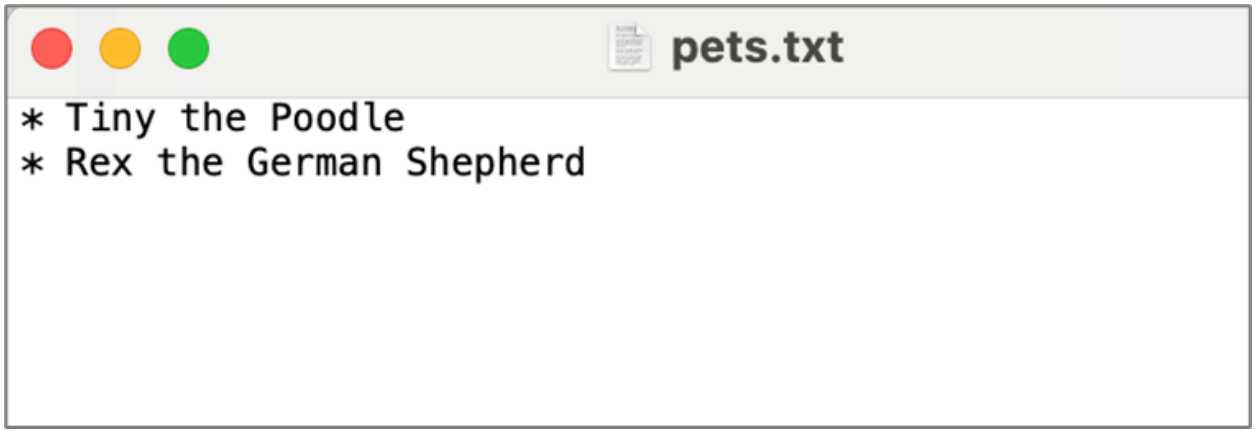
Question 4 (10pts)

Branching allows you to create different versions of your repository. We can use branches to work on fixes or changes in our code without affecting the main codebase before merging the changes back into the main branch.

[Complete levels 1-3 to get a visual demonstration about how Git branching works!]

Upload a screenshot that shows you completed up to level 3. You may need to refresh the page (or maybe type "levels") to view the levels you've completed. **(2pts)**

[4-1] Create a .txt file in your git repository named "pets.txt." Write a list of pets as follows below.



[4-2] Add the file to the staging area. **(1pt)**

```
(base) sandrawiktor@Lotus github_practice % git add pets.txt
(base) sandrawiktor@Lotus github_practice % git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   pets.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        temp.py
```

[4-3]

After adding the pets.txt file into the stage, git commit -m "Added first version of pets.txt" **(1pt)**

[4-4]

Create a new branch called "new_pets" using the command "git branch new_pets." **(1pt)**

We can check which branches we have using the command "git branch." Try it now. **(1pt)**

```
(base) sandrawiktor@Lotus github_practice % git branch new_pets
(base) sandrawiktor@Lotus github_practice % git branch
* master
  new_pets
```

The * indicates that we are on the *master* branch. To change branches, we can use "git checkout <branch-name>." Switch branches and show the new branch that we are on.

```
(base) sandrawiktor@Lotus github_practice % git checkout new_pets
Switched to branch 'new_pets'
(base) sandrawiktor@Lotus github_practice % git branch
  master
* new_pets
```

[4-5] Now, let's make a change to our file in the new branch by adding a new pet.

You can add your own pet if you have one, or professor Wiktor's! Shrimp. She's a blue parakeet.

[4-6] Add your change to the staging area with git add pets.txt. **(1pt)**

[4-7] Check the changes you made in the staging area with the git status command. **(1pt)**

```
(base) sandrawiktor@Lotus github_practice % git status
On branch new_pets
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   pets.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        temp.py
```

[4-8] Commit the change. **(1pt)**

```
(base) sandrawiktor@Lotus github_practice % git commit -m "Added a new pet"
[new_pets 627f06e] Added a new pet
 1 file changed, 2 insertions(+), 1 deletion(-)
```

[4-9] **Checkout to your master branch again**. Do you still see the new line you added? (You can use "cat pets.txt" to view the contents of the file.) **(1pt)**

Provide screenshots of each step in this part for credit!

```
Caleb Kronstad@Caleb MINGW64 /c/Development/Classes/ITSC-3155/github_practice (main)
$ touch pets.txt

Caleb Kronstad@Caleb MINGW64 /c/Development/Classes/ITSC-3155/github_practice (main)
$ git add ..
fatal: ..: '..' is outside repository at 'C:/Development/Classes/ITSC-3155/github_practice'

Caleb Kronstad@Caleb MINGW64 /c/Development/Classes/ITSC-3155/github_practice (main)
$ git add .

Caleb Kronstad@Caleb MINGW64 /c/Development/Classes/ITSC-3155/github_practice (main)
$ git commit -m "Added first version of pets.txt"
[main 59698e0] Added first version of pets.txt
 1 file changed, 2 insertions(+)
 create mode 100644 pets.txt

Caleb Kronstad@Caleb MINGW64 /c/Development/Classes/ITSC-3155/github_practice (main)
$ git branch new_pets

Caleb Kronstad@Caleb MINGW64 /c/Development/Classes/ITSC-3155/github_practice (main)
$ git branch
* main
  new_pets

Caleb Kronstad@Caleb MINGW64 /c/Development/Classes/ITSC-3155/github_practice (main)
$ git checkout new_pets
Switched to branch 'new_pets'

Caleb Kronstad@Caleb MINGW64 /c/Development/Classes/ITSC-3155/github_practice (new_pets)
$ git branch
  main
* new_pets

Caleb Kronstad@Caleb MINGW64 /c/Development/Classes/ITSC-3155/github_practice (new_pets)
$ git add pets.txt

Caleb Kronstad@Caleb MINGW64 /c/Development/Classes/ITSC-3155/github_practice (new_pets)
$ git status
On branch new_pets
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   pets.txt


Caleb Kronstad@Caleb MINGW64 /c/Development/Classes/ITSC-3155/github_practice (new_pets)
$ git commit -m "added new pet"
[new_pets 24f78b5] added new pet
 1 file changed, 1 insertion(+)

Caleb Kronstad@Caleb MINGW64 /c/Development/Classes/ITSC-3155/github_practice (new_pets)
$ git checkout main
Switched to branch 'main'

Caleb Kronstad@Caleb MINGW64 /c/Development/Classes/ITSC-3155/github_practice (main)
$
```

Question 5 (2pts)

We are happy with the change we made to the pets.txt file. Now, we want to merge this change with the main branch.

[5-1] **Make sure you are on the master branch**. While on the master branch, type the command git merge new_pets -m "Added new pet" **(2pts)**

> Your merge strategy may be different depending on the changes you made to get here. Take a look at this resource. to get familiar with different types of merging.

```
(base) sandrawiktor@Lotus github_practice % git merge new_pets
Merge made by the 'recursive' strategy.
 pets.txt | 3 ++-
 1 file changed, 2 insertions(+), 1 deletion(-)
```

> This merge was nice and easy, but they won't always be. Take a look at this resource. in the "Merge Conflict" section to learn more.

Provide screenshots of each step in this part for credit!

```
Caleb Kronstad@Caleb MINGW64 /c/Development/Classes/ITSC-3155/github_practice (main)
$ git merge new_pets -m "Added new pet"
Updating 59698e0..24f78b5
Fast-forward (no commit created; -m option ignored)
 pets.txt | 1 +
 1 file changed, 1 insertion(+)
```

Question 6 (BONUS)

Did you add your own pet's name to the file? Embed a picture of your pet below for 1 point extra credit on this assignment. :