Basic git Commands

git status

- Shows the staging status of your repo

git add -A

- I frequently use -A to add “all” files that haven’t been staged

git commit -m “commit message”

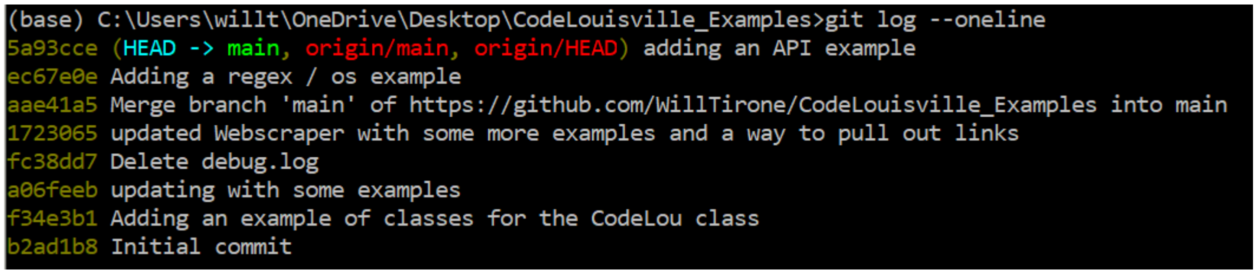
- My commit messages usually aren’t multiple lines or overly complicated so I just opt to type commit messages with the -m flag

git rm <file>

- Removes a file from the staging area

git log --oneline

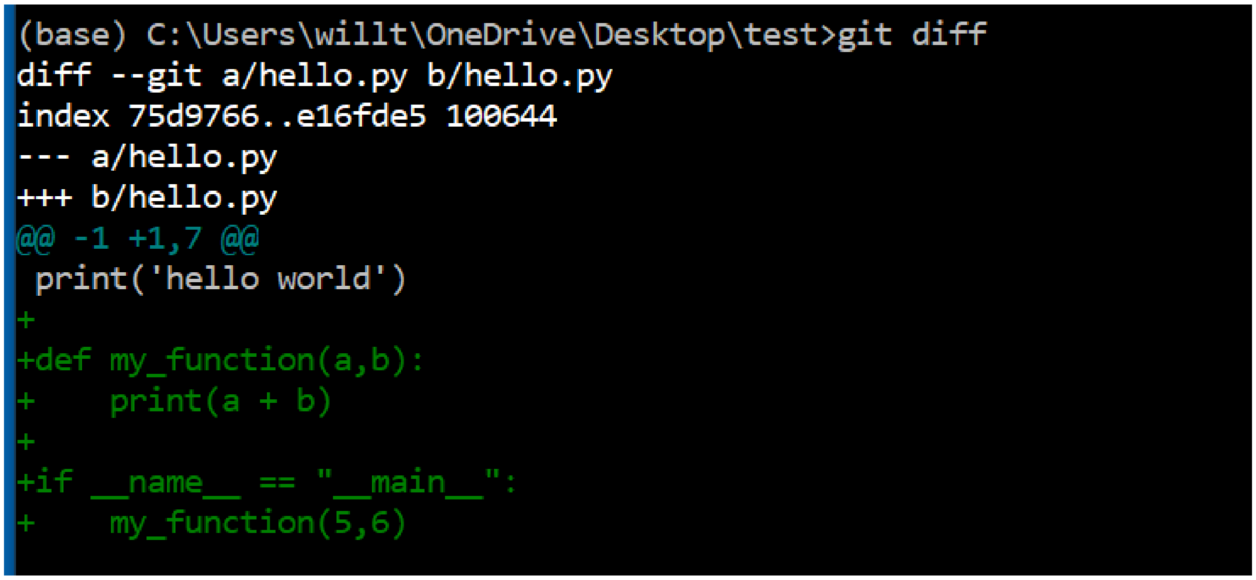
- Shows the commit history formatted compactly:



```
(base) C:\Users\willt\OneDrive\Desktop\CodeLouisville_Examples>git log --oneline
5a93cce (HEAD -> main, origin/main, origin/HEAD) adding an API example
ec67e0e Adding a regex / os example
aae41a5 Merge branch 'main' of https://github.com/WillTirone/CodeLouisville_Examples into main
1723065 updated Webscraper with some more examples and a way to pull out links
fc38dd7 Delete debug.log
a06feeb updating with some examples
f34e3b1 Adding an example of classes for the CodeLou class
b2ad1b8 Initial commit
```

git diff

- Shows you what has been added and deleted in the changed file. This is extremely important!!!

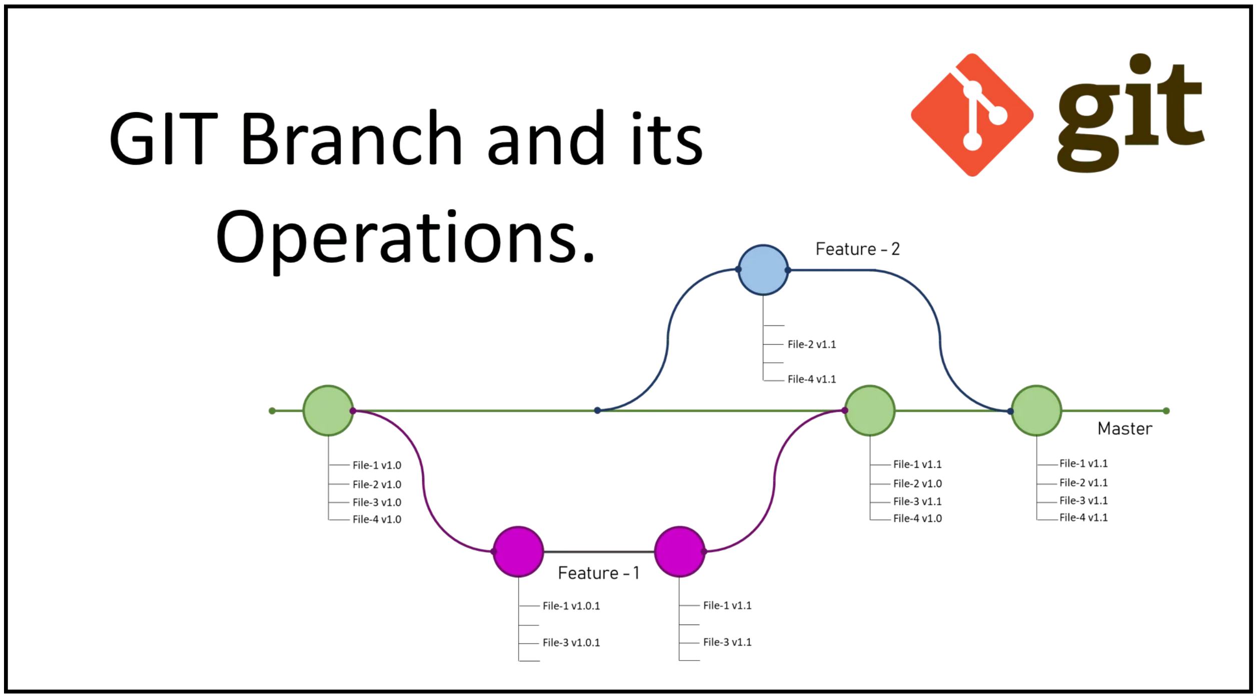


```
(base) C:\Users\willt\OneDrive\Desktop\test>git diff
diff --git a/hello.py b/hello.py
index 75d9766..e16fde5 100644
--- a/hello.py
+++ b/hello.py
@@ -1 +1,7 @@
 print('hello world')
+
+def my_function(a,b):
+    print(a + b)
+
+if __name__ == "__main__":
+    my_function(5,6)
```

git revert <sha1>

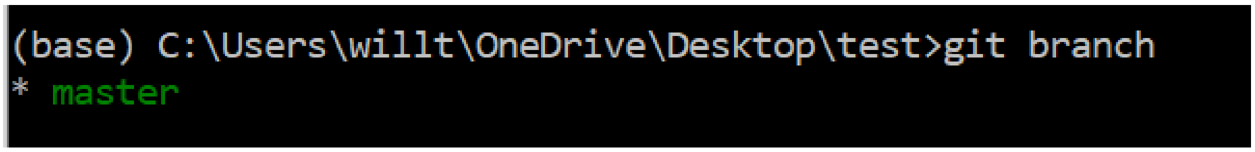
- This will revert your repo to a previous commit if you make a mistake or realize you broke the code. You want to copy the sha1 string of the commit you actually want to remove. This is similar to git reset, but git revert will actually make a new commit.

Branching



git branch

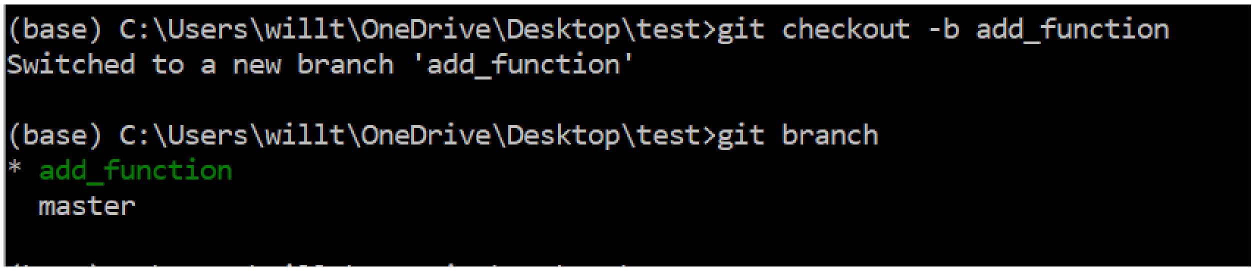
- With no optional arguments this will just tell you what branches you have. If you haven't done anything, you will just have a master branch. The asterisk and color green indicate which branch you're currently on.



```
(base) C:\Users\willt\OneDrive\Desktop\test>git branch
* master
```

git checkout -b branch_name

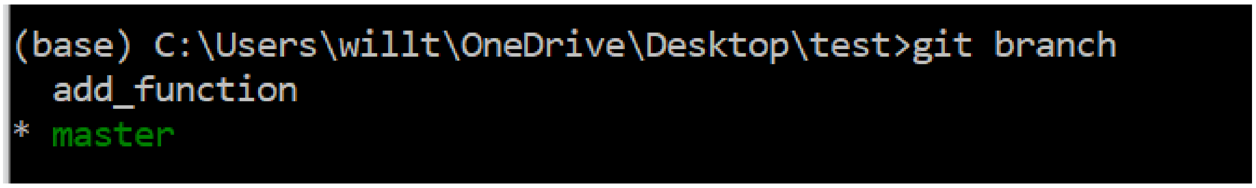
- I usually use checkout -b which creates a branch and then switches you to it, for convenience. You can also do this with two different git commands. You'll see that the asterisk and color have changed to my feature branch.

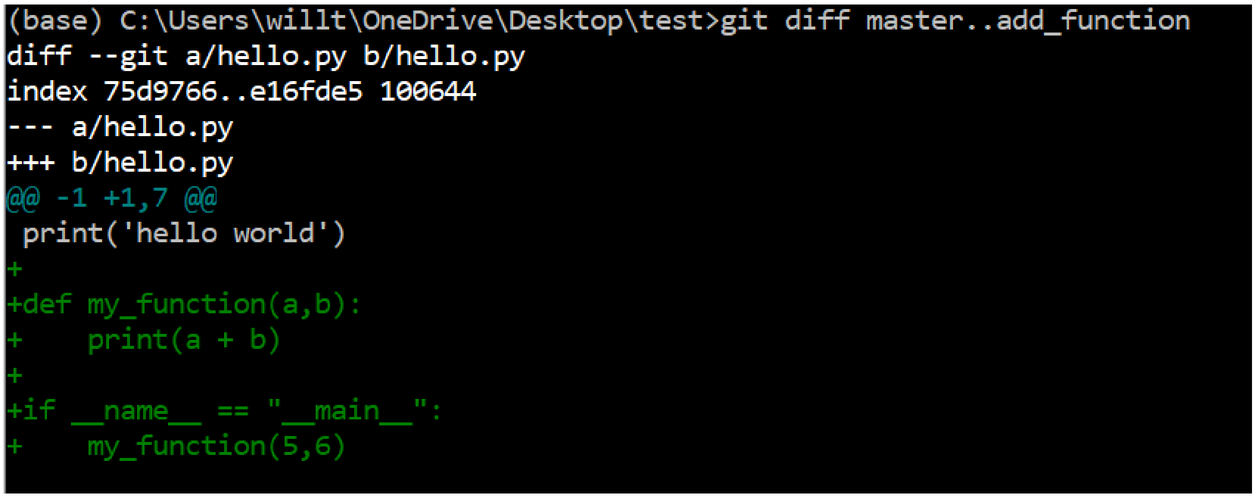

```
(base) C:\Users\willt\OneDrive\Desktop\test>git checkout -b add_function
Switched to a new branch 'add_function'

(base) C:\Users\willt\OneDrive\Desktop\test>git branch
* add_function
  master
```

git diff master..feature_branch

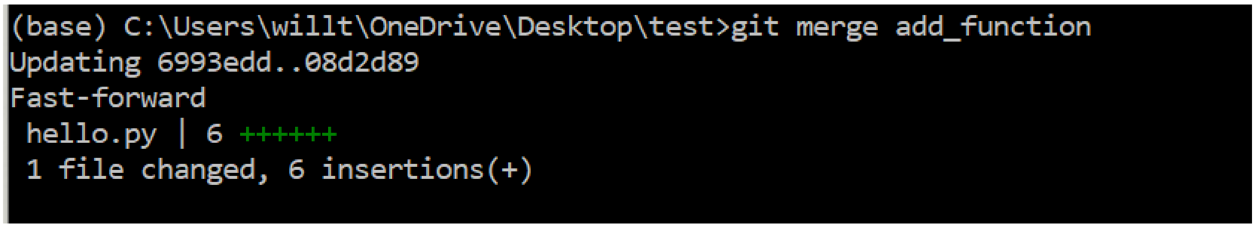
- Also VERY important. You do this once you have committed all your changes on the feature branch and have checked out the master branch again. Notice I'm now back on the master branch. This will tell me what the differences between the two branches are.


```
(base) C:\Users\willt\OneDrive\Desktop\test>git branch
  add_function
* master
```

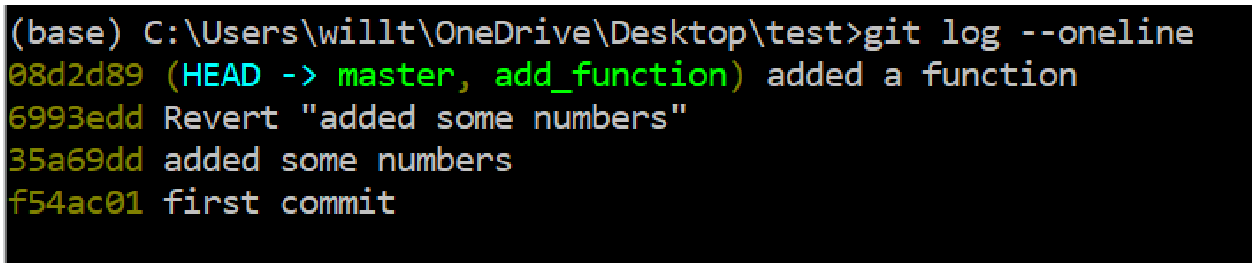

```
(base) C:\Users\willt\OneDrive\Desktop\test>git diff master..add_function
diff --git a/hello.py b/hello.py
index 75d9766..e16fde5 100644
--- a/hello.py
+++ b/hello.py
@@ -1 +1,7 @@
 print('hello world')
+
+def my_function(a,b):
+    print(a + b)
+
+if __name__ == "__main__":
+    my_function(5,6)
```

git merge branch_name

- Once we have determined the changes look good, you can just merge your changes back into the main branch.
- Notice that these now point at the same commit when you run git log



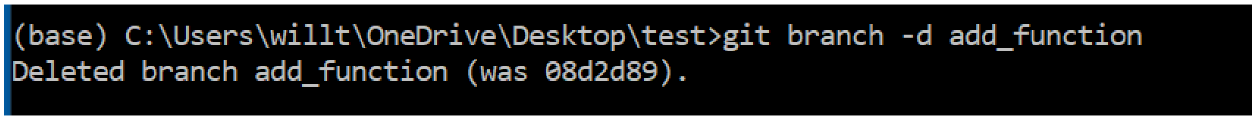
```
(base) C:\Users\willt\OneDrive\Desktop\test>git merge add_function
Updating 6993edd..08d2d89
Fast-forward
 hello.py | 6 ++++++
 1 file changed, 6 insertions(+)
```



```
(base) C:\Users\willt\OneDrive\Desktop\test>git log --oneline
08d2d89 (HEAD -> master, add_function) added a function
6993edd Revert "added some numbers"
35a69dd added some numbers
f54ac01 first commit
```

git branch -d branch_name

- You can now safely delete the feature branch. Note that it won't let you do this if you have unmerged changes. If you decide the changes aren't going to work and want to force delete it, use -D



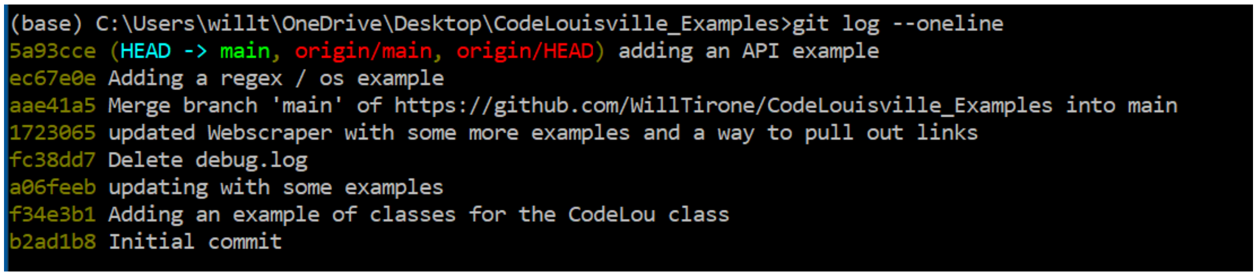
```
(base) C:\Users\willt\OneDrive\Desktop\test>git branch -d add_function
Deleted branch add_function (was 08d2d89).
```

GitHub

git clone <url>

- Clones your repo to the current working directory

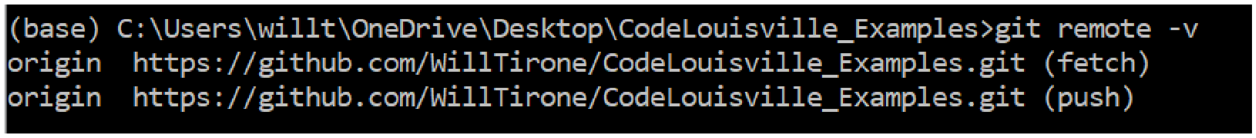
I cloned this repo below; notice that it also has the current status of the origin (the repo that I cloned it from).



```
(base) C:\Users\willt\OneDrive\Desktop\CodeLouisville_Examples>git log --oneline
5a93cce (HEAD -> main, origin/main, origin/HEAD) adding an API example
ec67e0e Adding a regex / os example
aae41a5 Merge branch 'main' of https://github.com/WillTirone/CodeLouisville_Examples into main
1723065 updated Webscraper with some more examples and a way to pull out links
fc38dd7 Delete debug.log
a06feeb updating with some examples
f34e3b1 Adding an example of classes for the CodeLou class
b2ad1b8 Initial commit
```

git remote -v

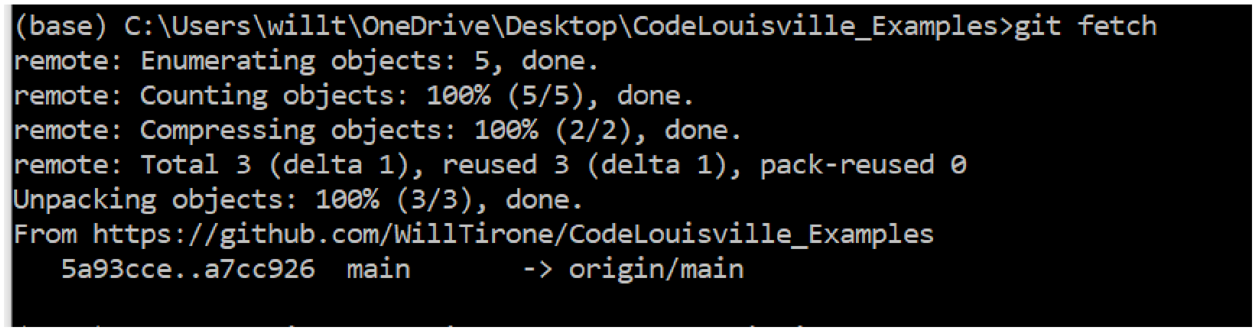
- The v stands for "verbose" and gives you the location of the origin



```
(base) C:\Users\willt\OneDrive\Desktop\CodeLouisville_Examples>git remote -v
origin  https://github.com/WillTirone/CodeLouisville_Examples.git (fetch)
origin  https://github.com/WillTirone/CodeLouisville_Examples.git (push)
```

git fetch

- Because of the previous command, I know that if I type git fetch it will just pull anything from the address given



```
(base) C:\Users\willt\OneDrive\Desktop\CodeLouisville_Examples>git fetch
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 3 (delta 1), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/WillTirone/CodeLouisville_Examples
   5a93cce..a7cc926  main       -> origin/main
```

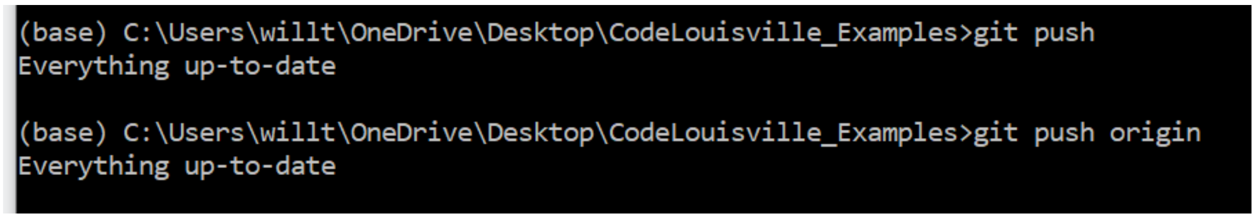
git pull

- This will actually make changes to your local repo whereas git fetch will not. You will see people use both in different contexts.

https://www.git-tower.com/learn/git/faq/difference-between-git-fetch-git-pull/

git push

- This will push any changes you've made on your local repo. You can also include the name of the remote to be sure.



```
(base) C:\Users\willt\OneDrive\Desktop\CodeLouisville_Examples>git push
Everything up-to-date

(base) C:\Users\willt\OneDrive\Desktop\CodeLouisville_Examples>git push origin
Everything up-to-date
```