All engineering IS software engineering in disguise.

GIT Revision Control

Provocative! All engineering involves creative input, be it text, graphics, processes , simulation .
Common to all is the ability to undo, revise, enhance partition effort and merge products the effort to create the whole engineering task/product/app/goal. Even graphical tools that don’t appear to be text, still have a revision control.

What is GIT and WHY would I use it?

- You need to Spell to Write.
- You need Arithmetic to Do Accounts,Use SpreadSheets or Calculate.
- You need Algebra to do Mathematics
- You need Revision Control to write ANY Software Source.
  - Coding never is written in ONE edit. It never works first try… If it does…
    - It evolves, gets new features … and gets bugs and breaks just before HAND-IN
  - Is usually a team effort… On many computers. As a result…
  - Many Copies of source files are created.
  - **Revision control manages file distribution , edits and merging.**
- GIT is the easiest,widely available, most professional, Open Source (free!) revision control

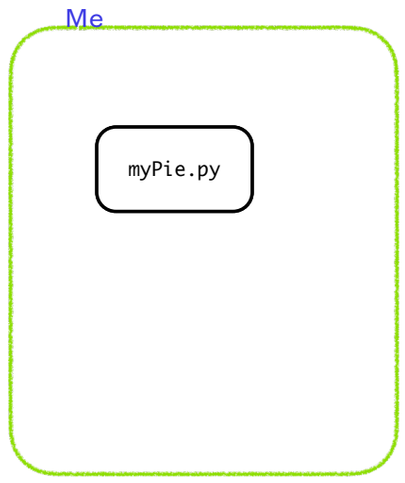
Software source can be taken loosely here. Source editing is an iterative process and when in a team development setting ,involving multiple files, handling sharing and merging individual files in a project needs a tool to help coordinate working (and broken!) Project states. In the case where the same file is being edited by two , or more people, merging the files back to one authoritative file is required. Today GIT is widely used throughout industry.

Coding

- Facebook,Instagram,~~Twitter~~ X all started with…
  - A few files written by …
  - possibly just 1 engineer.
- By the time you knew them…
  - There are zillions of files and coders! In Twitters case… Not so many Coders,
    - Bugs and Frequent REVISIONS are needed to fix or refactor or stage new features.

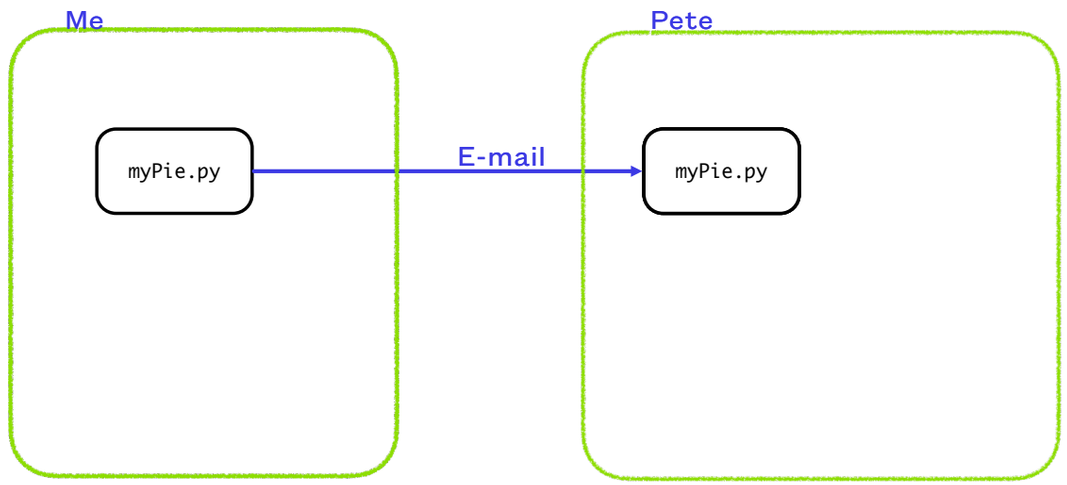
By the time you know them they are colossal file and directory structures written by “zillions” of engineers! In the case of Twitter less so now, and due to swift new features Elon had the remaining softies write, IT broke! and they quickly changed their minds on the new changes. Once a working project or widely used application is critical to all , It becomes very very important to be able to revert to the version that last worked!

But I'm not writing ~~Twitter~~X you say!

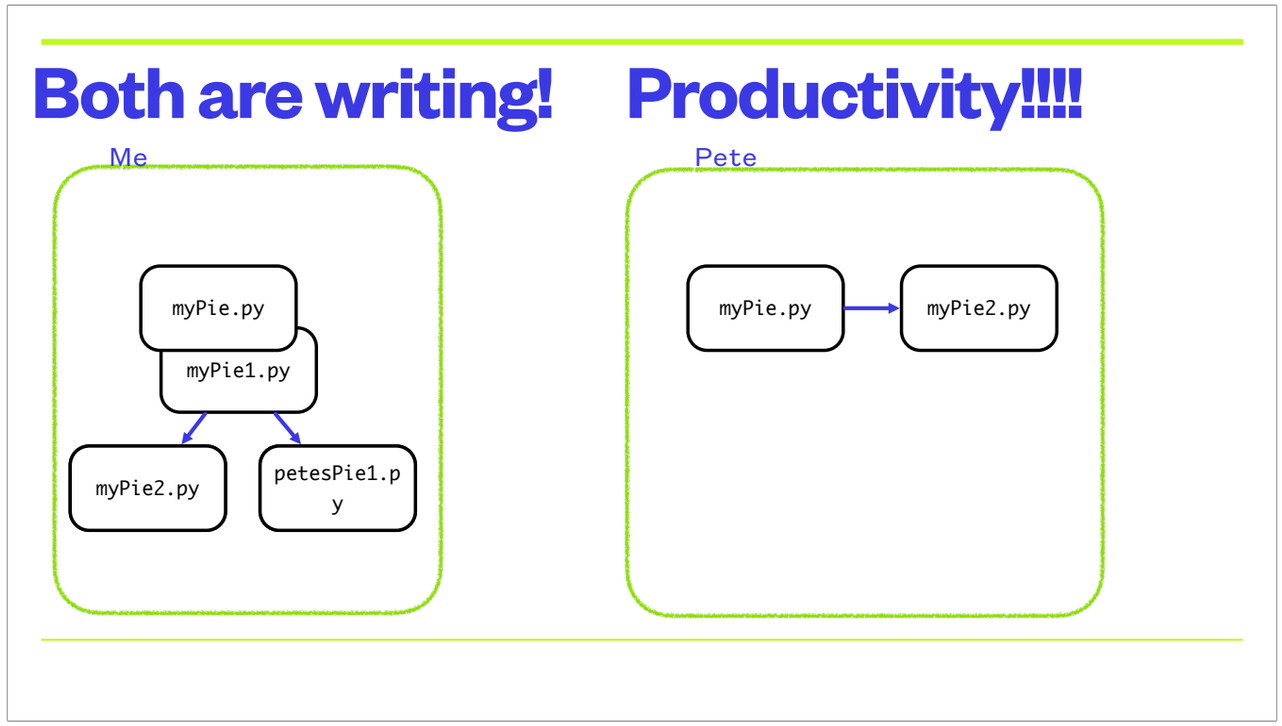


You start with one file myPie.py. add some 50 lines it soon works completely…

All coding starts simple!

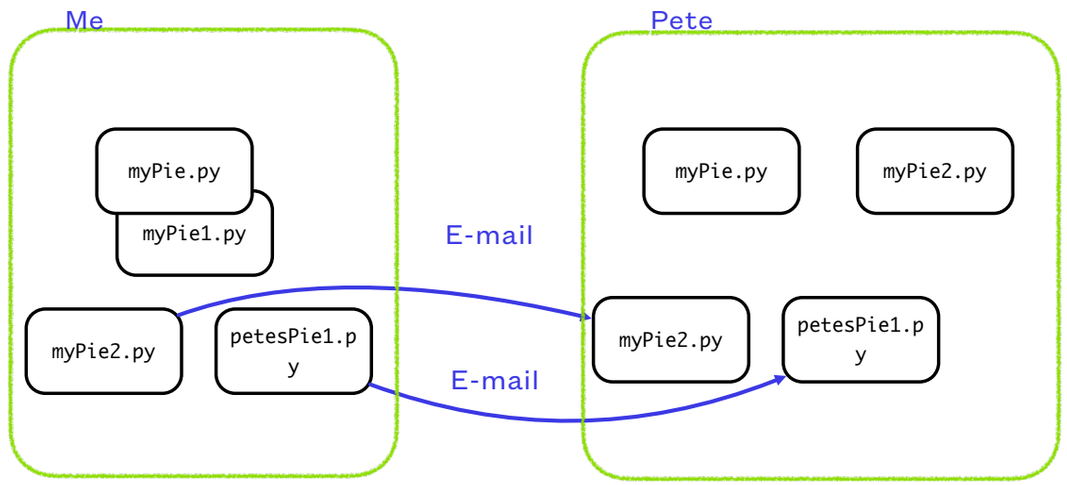


You start with one file myPie.py. add some 50 lines it soon works completely… you email it to your project team buddy Pete. Why not!? Email is so easy to attach that one file!



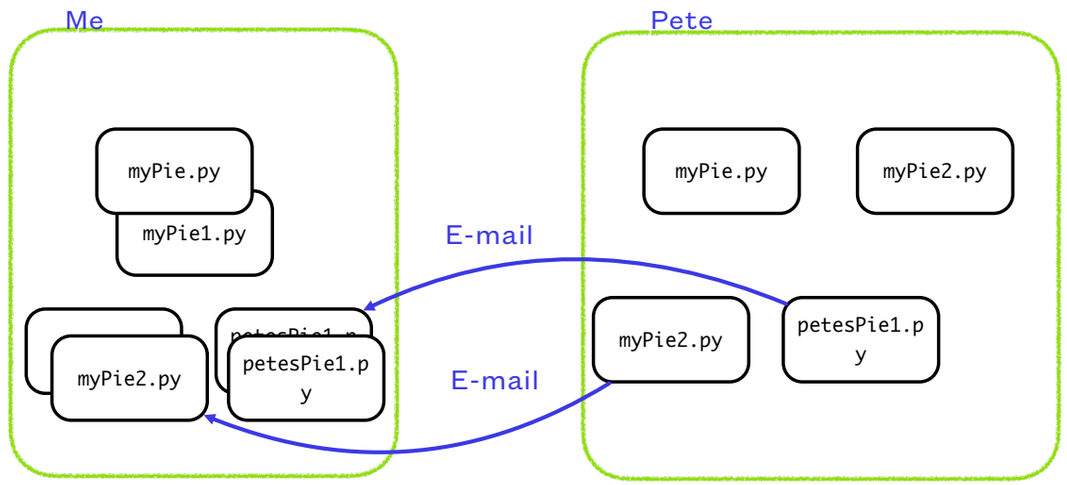
Pete saves your attached file, and fixes a bug you missed and ands some proposed features you might not want so puts them in a renamed version of your file. He emails both back.You try to add Pete’s source and add features of your own.

Pete needs my new stuff! Swoosh!

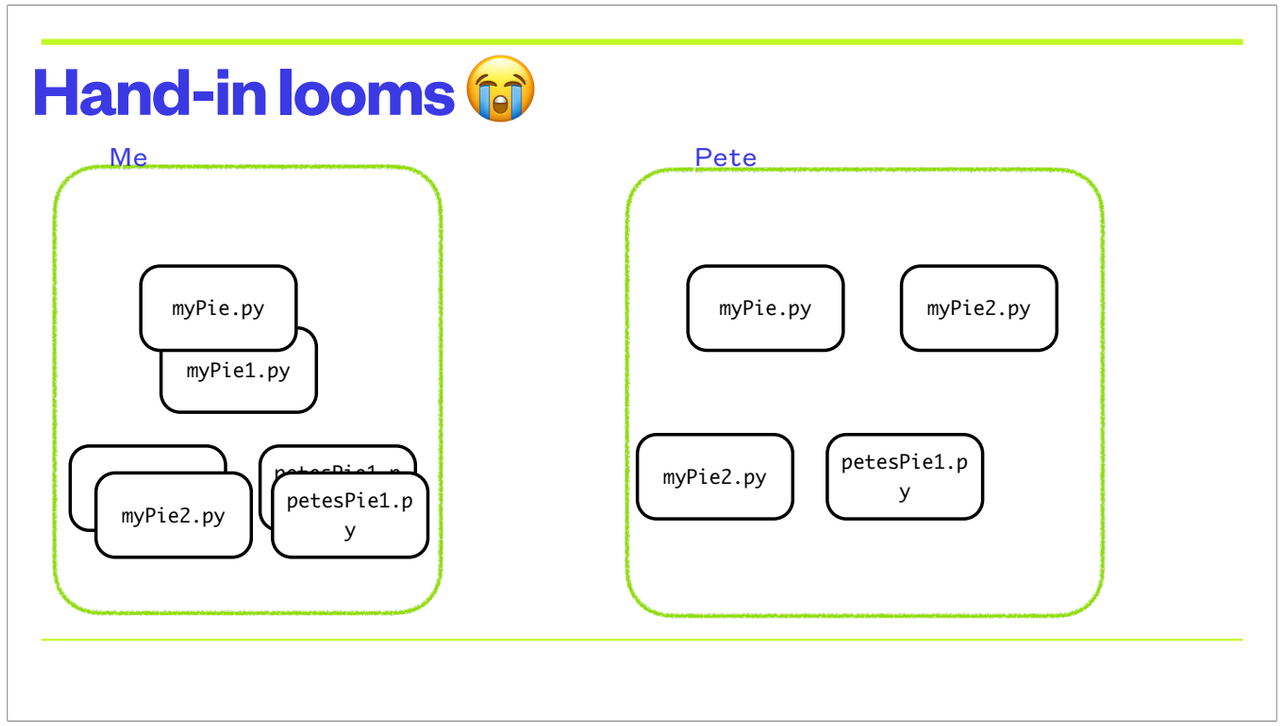


You e-mail Pete your new files!

I need Pete's stuff . Swoosh!



Pete e-mails you just 10mins before hand-in!



Now! I can’t see Pete’s directory (green box) and he can’t see mine.

Whose/What files do you hand in?

- Hand-in is in 10 mins!
- You could spend hours opening all files to see which one.
- If you do do this linux:Mac sdiff can compare files and show what the differences are.
- But don’t! This is the dark side! The force will not be with you.

Which files work and which do you submit. Hopefully one of the files HAS a comprehensive test suite so you KNOW it works.

A tool to manage all the source files , assets and directories in a project.
It can be a simple a 1 file.
It can be only you that edits it.
It can be only on your computer.

BUT!
It can be any number of files.
It can be any number of team members.
It can be running or being developed and debugged on any number of computers.

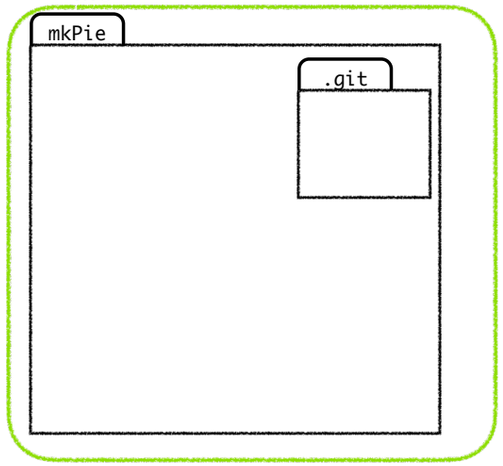
How does GIT help?

- The simplest way GIT helps is to get STUFF!
  - Most useful libraries you need for Arduino/Python/C/C++ are on the Web.
  - They usually come with many files and directories
  - They often change over time … bug fixes and enhancements
  - Note the suffix .git
  - To just download to your computer…
- C:> `git clone https://github.com/micropython/micropython.git`

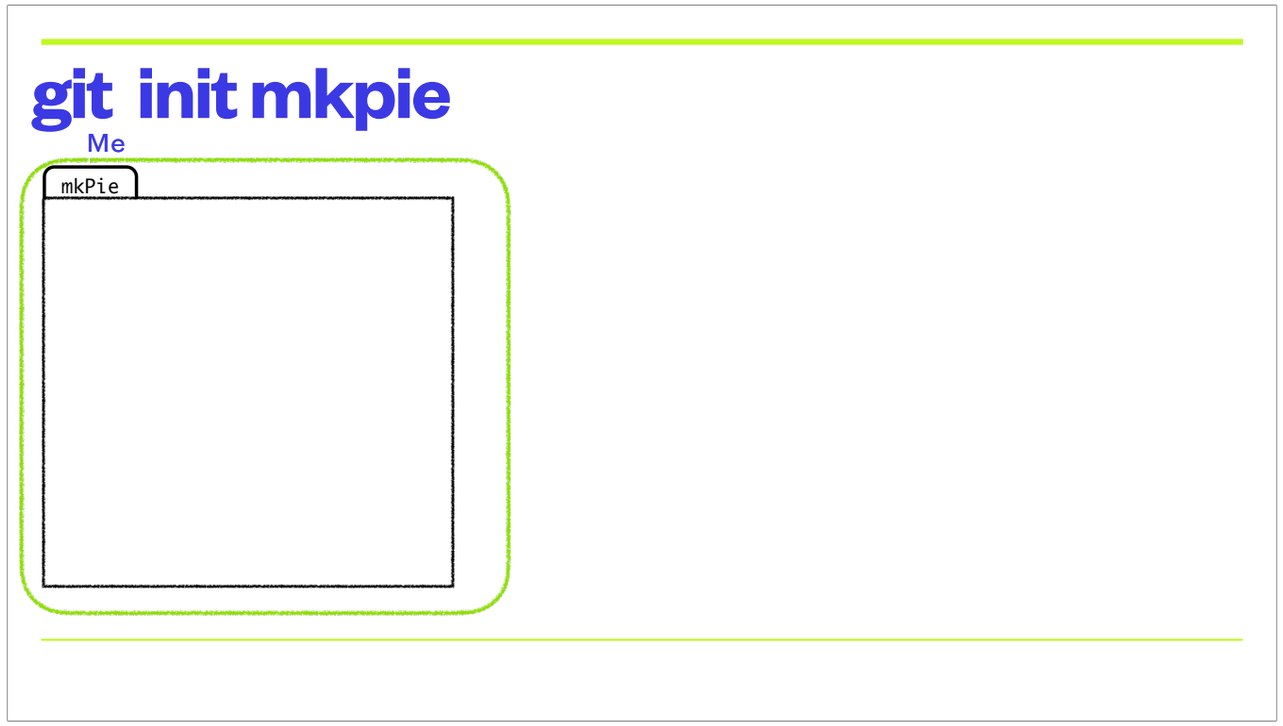
The simplest introduction to git is to ‘grab’ a copy of a library/code from the web.
Git works with repositories (repo’).
A repo is a directory or directory hierarchy that contains source files and resources.
The whole caboodle is managed as one entity.

The only git common you need is in blue.

git init mkpie

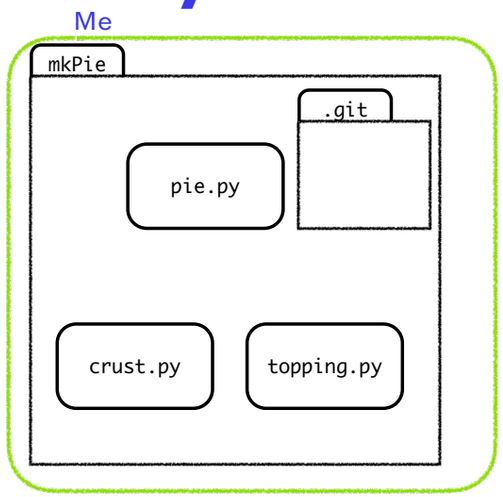


Git repos are both local and remote. Local when you are creating or using the file.
Remote when you need to share or backup the source files.
You can create a git repository with one simple command
It can work with OTHER git repo’s … You have to be a Software Guru tho!
It can and should ignore binary files you generate as well as others.
NOT ALL FILES ARE INCLUDED IN YOUR REVISION.
You opt for WHAT files are revision controlled…
Messy engineering “what ifs” don’t need shared



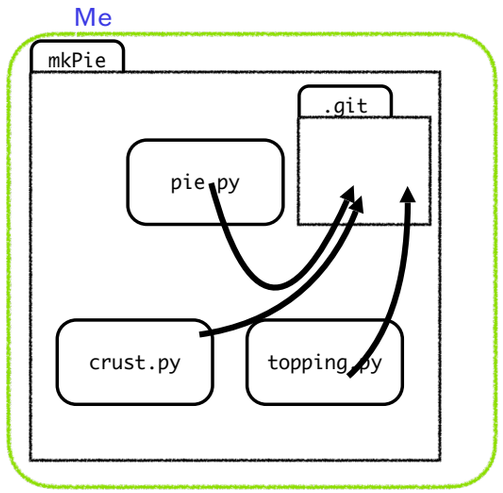
.git is a directory you can’t even see. It’s hidden.
Git works as a command line interface cli or Tortoise or Atlassian SourceTree.
And others.
Its often part of good editors.

Plan your basic source files



As Jon will/has told you … You plan what files might be needed and WHO is doing what
In linux…
touch pie.py, crust.py, topping.py
Creates all three named empty files (0 bytes)

add files and commit to your repo’

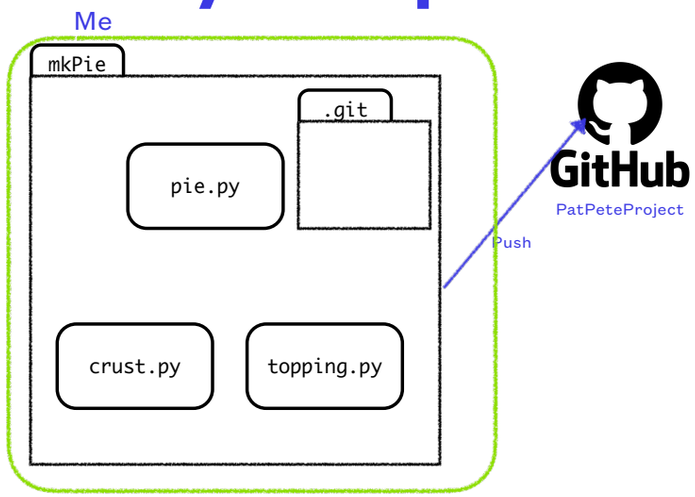



```
commit 74d371f5a35c33a0809b66c58449fa18d19224af
Author: Hugh Wallace <VitalSparkPlug@gmail.com>
Date:   Sun Feb 12 10:15:28 2023 -0700

    Fixed CRC: p1 of crc needed to be inverted
```

Once your skeleton files are created… you add files and commit ALL the files to your repo’
The commit gets a URI, Date, Author and **most importantly** a
modification string. Git log will list all the commits Its all stored in the .git
A warning to even competent hackers!!! You know who you are ! DO NOT even think about editing anything in the .git directory… It hidden for a very very good reason.
Every commit has a log entry that has a very big hex number. This is a URI (Uniform Resource Identifier)
Your more familiar with URL (Uniform Resource Locator) A unique address on the internet.

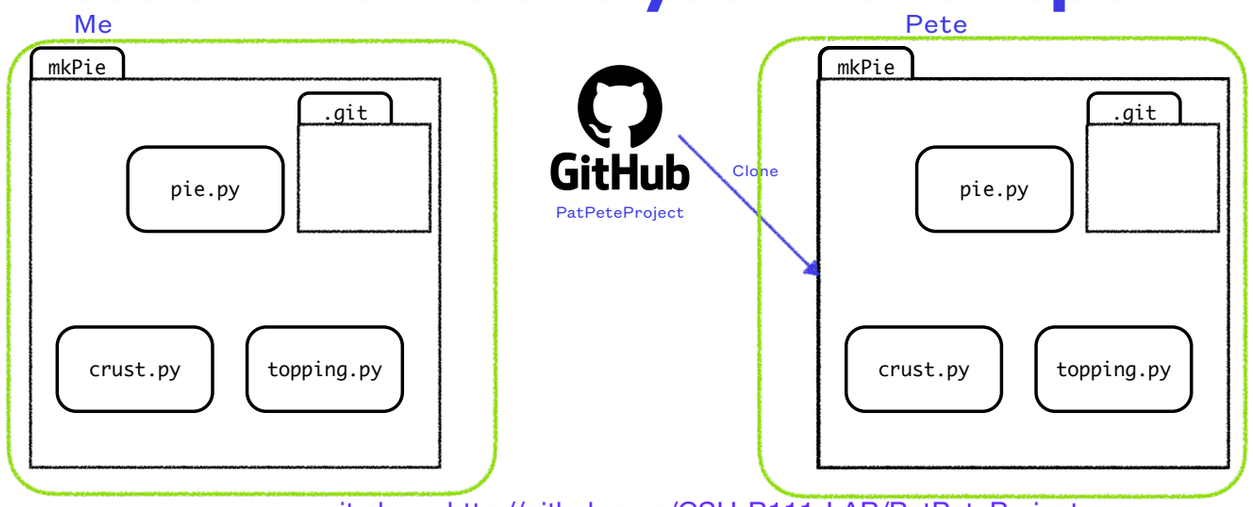
This URI names a unique state which can be returned to or compared to.

Push your repo' to the cloud



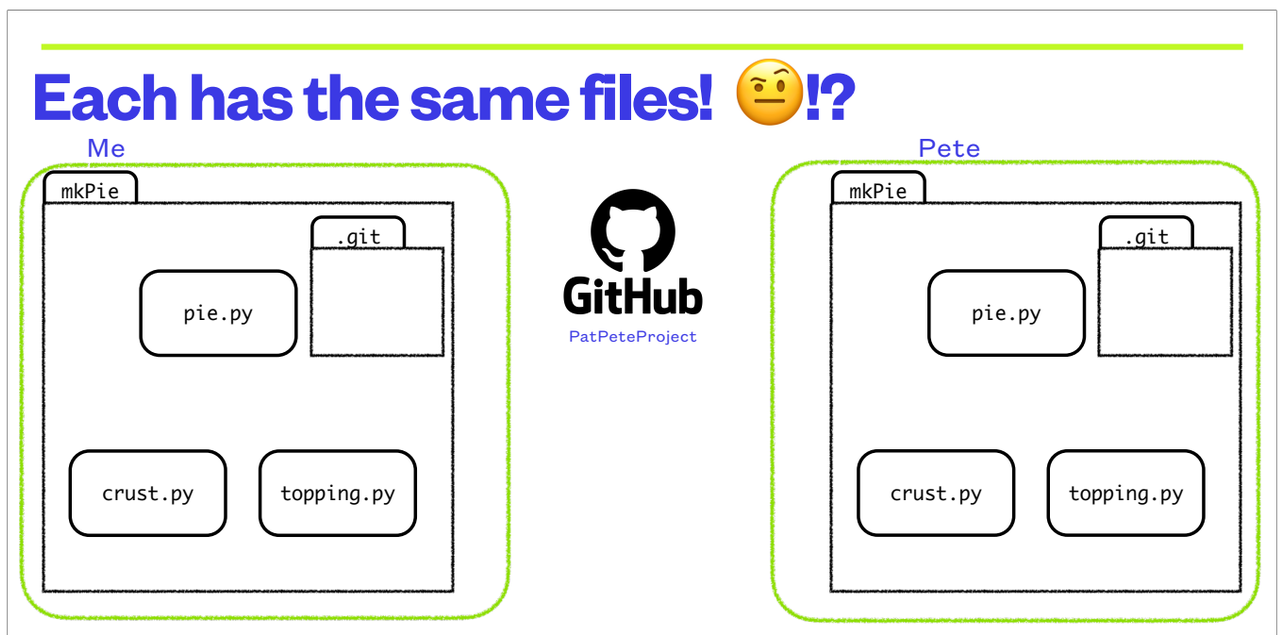
There is a git repository (repo') that is somewhere in the cloud or you employers network.

Pete can now clone your cloud repo



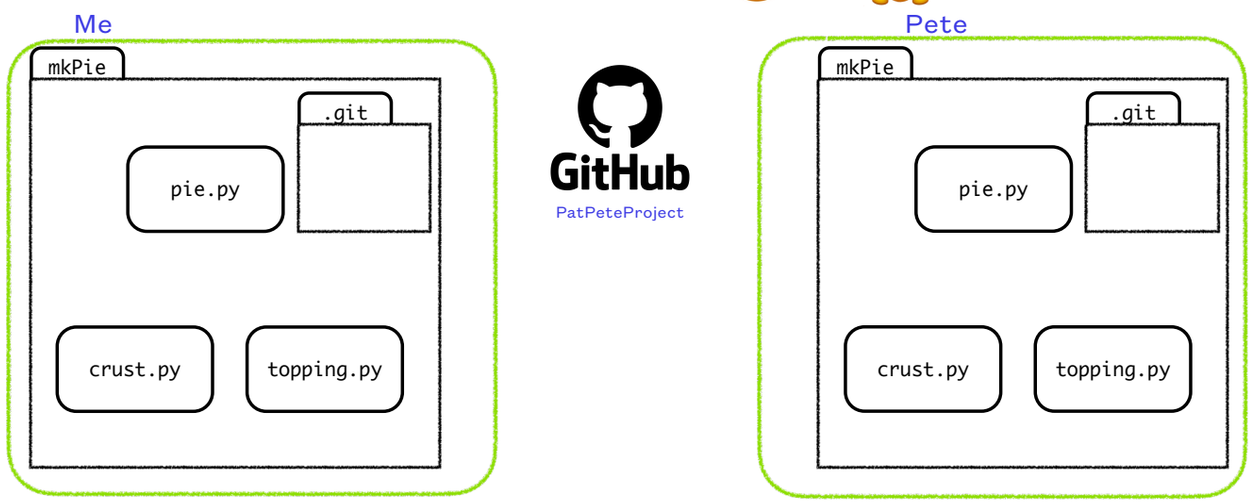
git clone http://github.com/CSU-B111-LAB/PatPeteProject

Pete uses
Git clone http://github.com/B111Lab/PatPeteProject



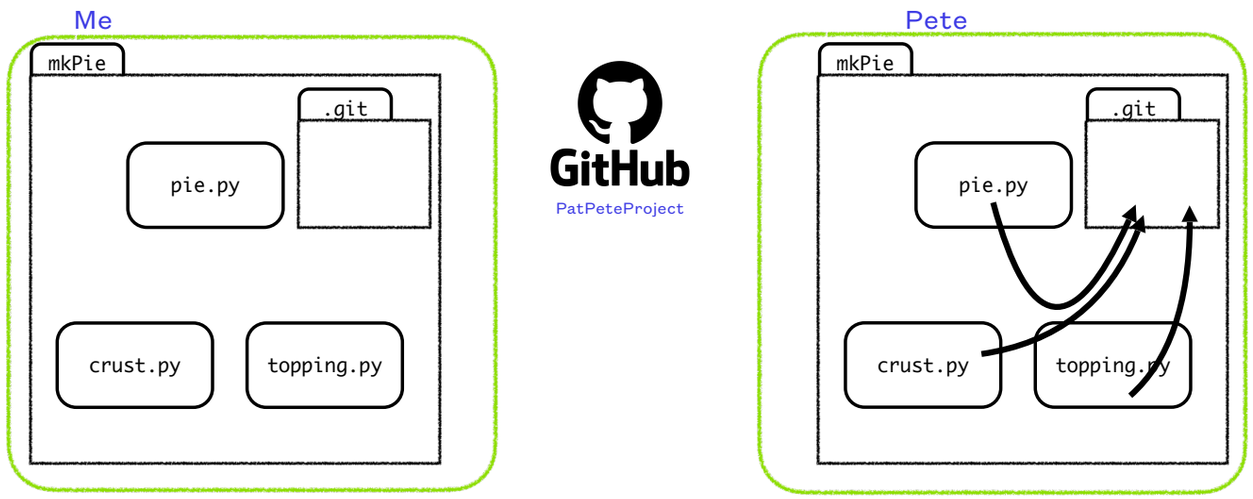
Ahh! Each has the same files I hear you say.

Each has the same files! 😐!? 😱?



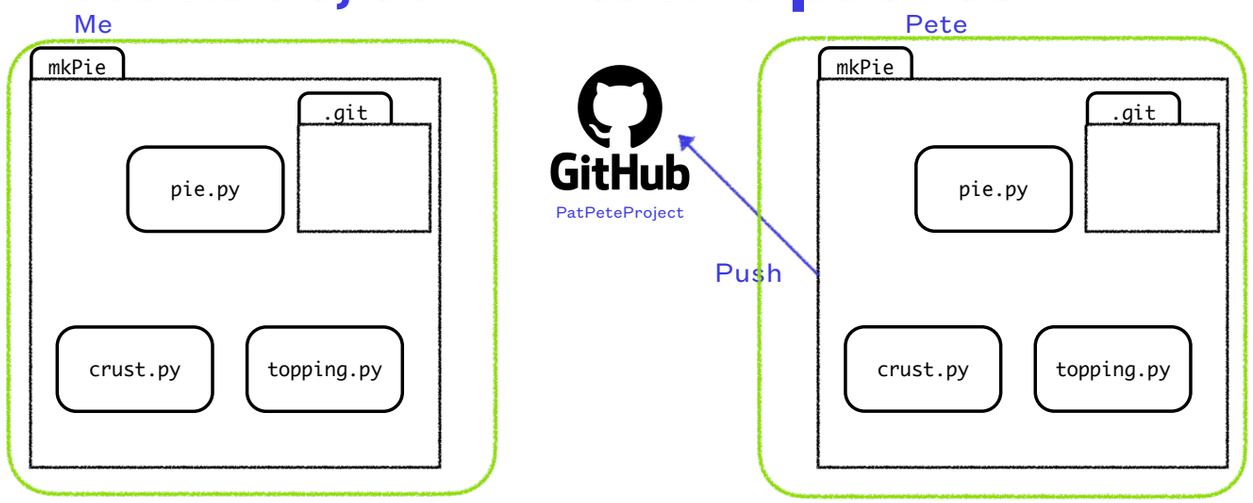
But some files are for Pete to work on.
He does NOT rename files to “SAVE” state.
Simply create , modify , debug code in ANY file ! NO Renaming.
You can refactor to add or remove files too.

Each has the same files!

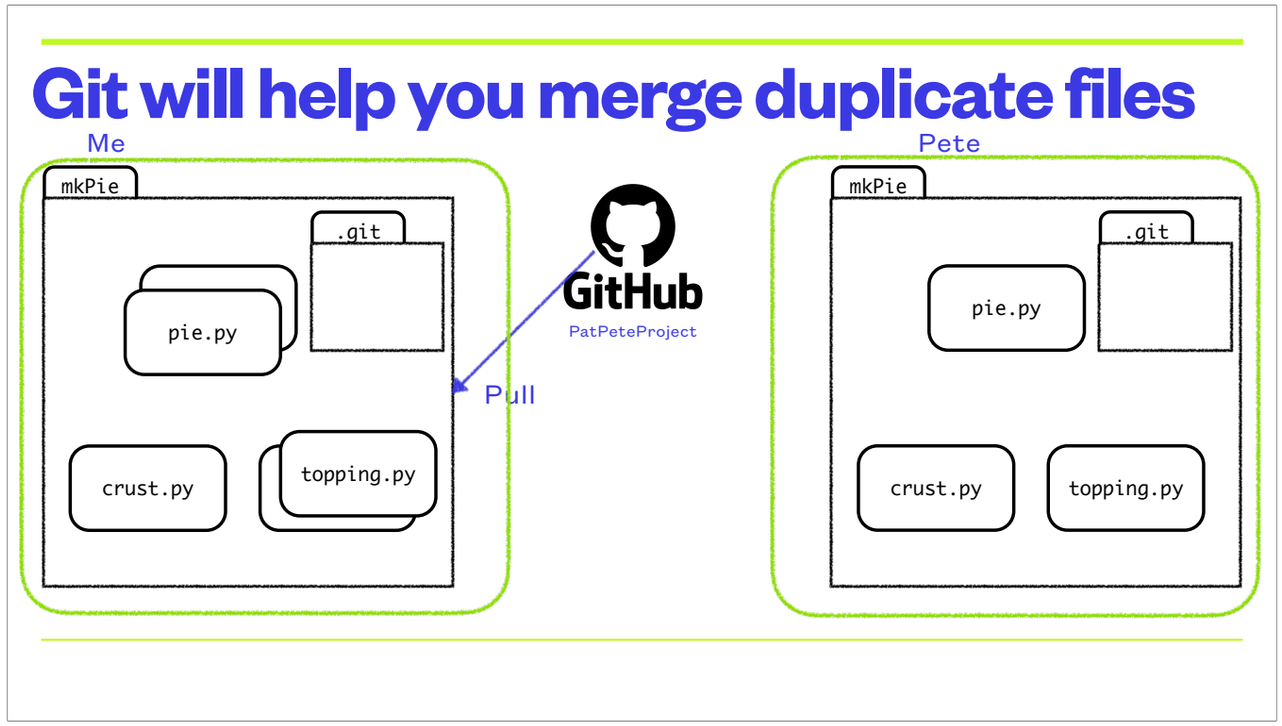


When Pete is happy with his modifications to the files.
He stages wanted files first… in case he made some temporary files NOT needed in the project
He the commits to his local repo be careful to add a modification string, saying what he has done .
Merging will deal with this soon.

Pete adds, commits and pushes



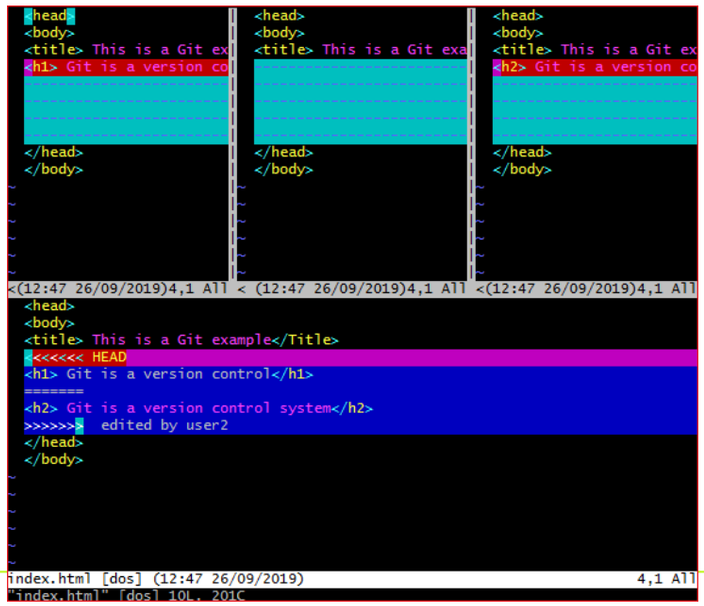
Pete does what you did.
Commits with a modification string and then
Pushes the changes to the cloud



You now pull from the cloud… where the code is safe and documented what has changed.
How do you know that the repo changes? There is a git command that tells you that there is a newer head.
Pete could also txt you …. “Just committed the changes I made… see you after spring break :-) “

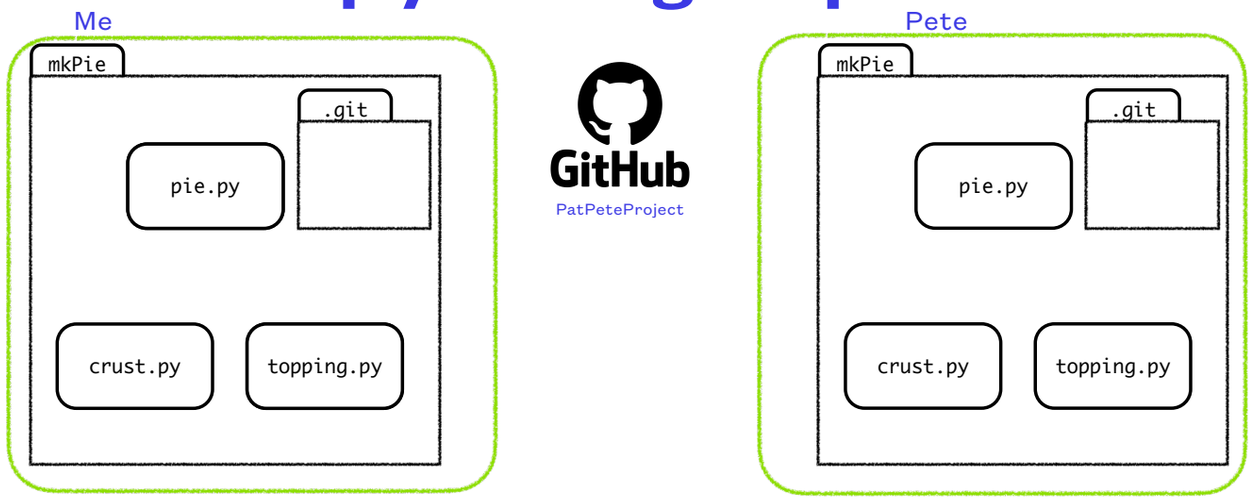
Git merge

- git merge tool
- Simple cli version
- Many slick GUI merge tools are available



When you pull. If Pete only added to files you didn’t change then they will update with Pete’s stuff.
If he edited a file that you edited then merging is the process of deciding if the edits clash or are harmonious.

Git will help you merge duplicate files

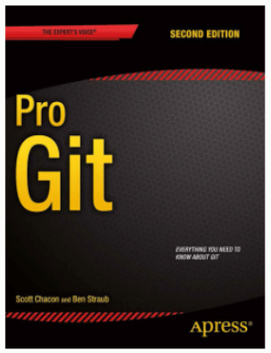


Merging happens ….
You NOW have the same files again.
BUT Pete will **not have** what you have edited after you last push.!!!!!!!!
For that I need to Commit the push to the Cloud GIT Repo’ and Pete needs to pull possibly merging like I just did but accepting ALL my changes … If he trusts I did it correctly ;-)

The End

https://git-scm.com/book/en/v2

- Free book!
- If you love it … Buy paper one!
- First 8 chapters only (page 108)
  - It has (800 pages!)



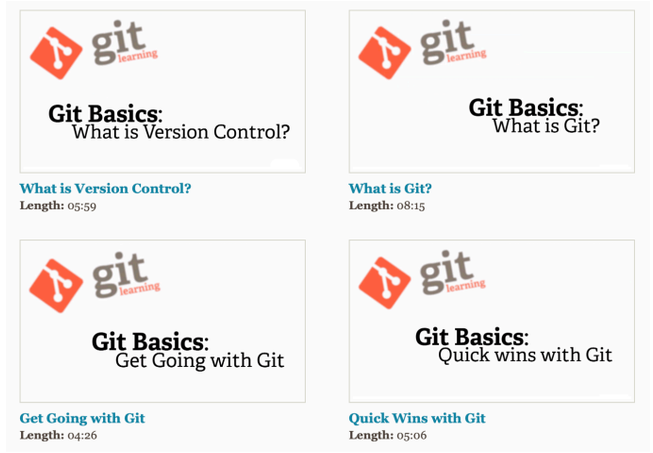
2nd Edition (2014)

Download Ebook

pdf epub mobi

If you are a Computer science or Computer Engineering . YOU should read the whole thing and be very super guru level knowledgable. Add it to your resume…
Git at high levels is very powerful … but can get messed up due to not quite understanding it! Those engineers who can wield git at Gandalfs levels are revered in ANY org.

Videos for the Dyslexic ... Me!

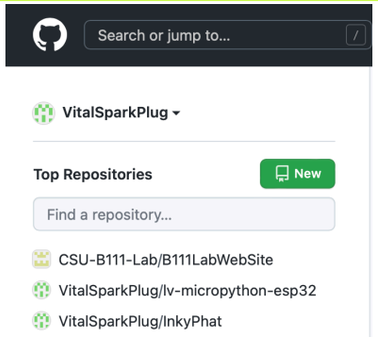


Nice concise videos…
And there is YouTube!

GitHub.com

- Get an account
- You can make it private
- I set up a CSU-B111-LAB team site where …
- Only team members can have share private repo's

- The also have virtual machines you can use for free!



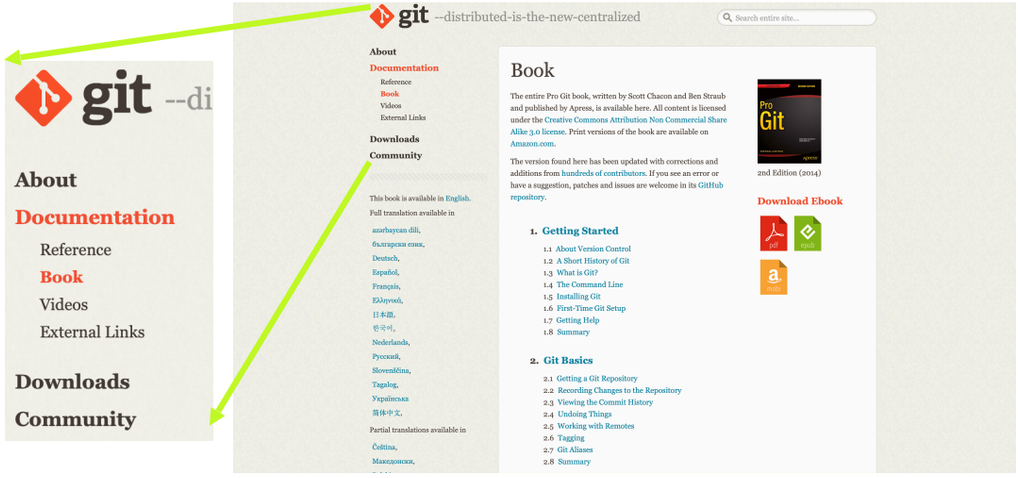
CSU-B111-Lab/b111labteam

CSU-B111-Lab/reactinfmirror

CSU-B111-Lab/neopixelclockers

CSU-B111-Lab/abc

https://git-scm.com/book/en/v2



Git CLI (Mac,Linux,PowerShell)

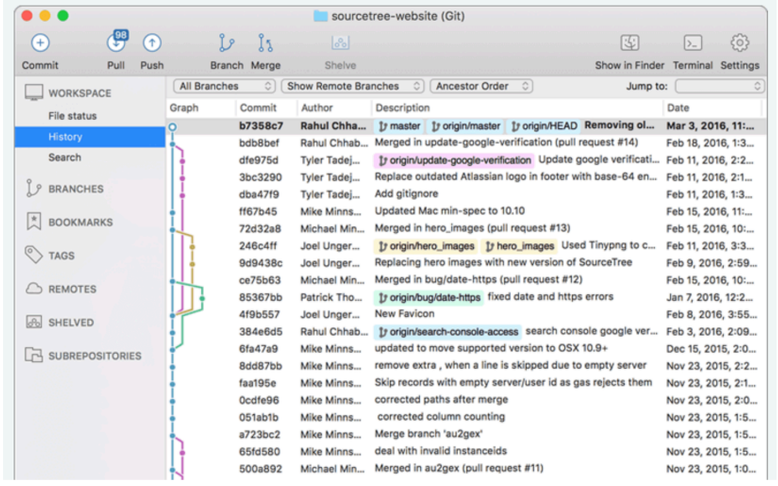
C:> git help

```
These are common Git commands used in various situations:

start a working area (see also: git help tutorial)
   clone     Clone a repository into a new directory
   init      Create an empty Git repository or reinitialize an existing one

work on the current change (see also: git help everyday)
   add       Add file contents to the index
   mv        Move or rename a file, a directory, or a symlink
   restore   Restore working tree files
   rm        Remove files from the working tree and from the index

examine the history and state (see also: git help revisions)
   bisect    Use binary search to find the commit that introduced a bug
   diff      Show changes between commits, commit and working tree, etc
   grep      Print lines matching a pattern
   log       Show commit logs
   show      Show various types of objects
   status    Show the working tree status

grow, mark and tweak your common history
   branch    List, create, or delete branches
   commit    Record changes to the repository
   merge     Join two or more development histories together
   rebase    Reapply commits on top of another base tip
   reset     Reset current HEAD to the specified state
   switch    Switch branches
   tag       Create, list, delete or verify a tag object signed with GPG

collaborate (see also: git help workflows)
   fetch     Download objects and refs from another repository
   pull      Fetch from and integrate with another repository or a local branch
   push      Update remote refs along with associated objects
```

git log

```
commit 70ad8c9f2573bbd35d788c5e6a2782686e751fee (HEAD -> main)
Author: Hugh Wallace <VitalSparkPlug@gmail.com>
Date:   Sun Feb 12 10:36:16 2023 -0700

    LinDecoder1.py: reads  WebastoLin19k2 output and validates idCRC and CRC. parse1.py reads LinDecoder1.py output and static d
SI location and green and red

commit 74d371f5a35c33a0809b66c58449fa18d19224af
Author: Hugh Wallace <VitalSparkPlug@gmail.com>
Date:   Sun Feb 12 10:15:28 2023 -0700

    Fixed CRC: p1 of crc needed to be inverted

commit 6c72bef3c4da5d49e8996db7f9cce2e890c8608d
Author: Hugh Wallace <VitalSparkPlug@gmail.com>
Date:   Sun Feb 12 10:10:36 2023 -0700

    Combined RxBreak and Rx s/mc to remove irq enable of Rx that took too long. Output now catches Sync (0x55). CRC still wrong

commit 0720a8233e0e92988187ea9c51654fa4a92daf65
Author: Hugh Wallace <VitalSparkPlug@gmail.com>
Date:   Sun Feb 12 10:06:20 2023 -0700

    RxBreak s/mc and Rx s/mc work but the latency of the RxBreak irq to enable Rx s/mc was 2.5 bits (too long) [Broken]
```

SourceTree (Mac)



TortoiseGit (Win)