GitHub is the bomb

You should use it

Version Control, Bad Way

You've probably done ad-hoc version control by creating multiple versions of a file like this:

my_code.py

my_code_with_new_feature.py

my_code_I_have_no_idea_why_its_not_working.py

my_code_finally_working.py

my_code_optimized_for_speed.py

Version Control, Good Way

A better way to do version control is to use software like Git or GitHub or Git + GitHub.

Add my_code.py to a Git repository

GitHub: Drag and drop

CLI: git add my_code.py; git commit

Add a new feature to my_code.py, then commit the changes to the repository.

GitHub: Drag and drop

CLI: git add my_code.py; git commit

Advantages of using Git

1. Easily go back to any version of my_code.py that you added to the repository.
2. See the changes contained in any version of my_code.py that you added to the repository.
3. Easily find regression bugs, no not linear regression. Regression as in this old feature used to work but I added a new feature and now the old feature is broken.

Advantages of using GitHub

1. Makes it easy for a team to collaborate on code development.

2. You can cite your GitHub repository in a publication. If you archive it you can even generate a DOI for it.

3. Keeps an online copy of your most valuable work so when Italian thieves steal your laptop you don’t have to rewrite all your code.

Pitfalls

Git and GitHub do not magically know what you've done to your code.

The only versions of your code that you can go back to are the versions that you commit to the repository.

Version control tools only help when you commit your changes frequently.

GitHub for Beginners

1. Create a repository
2. Add a file to the repository
3. Modify the file locally
4. Commit the modified file to the repository
5. Look at the changes on GitHub

Collaborating with GitHub

1. Create and download a new branch
2. Make some modifications
3. Upload and commit the changed files
4. Issue a pull request on GitHub
5. Merge your branch into Master on GitHub

Git for Beginners

Create a new repository in the current directory:

```
git init
```

Copy a repository from GitHub to the current directory:

```
git clone https://github.com/user_name/repo_name
```

Add modified files to the staging area:

```
git add file1 file2 … file273
```

Commit staged modifications to the local repository:

```
git commit
```

Push local changes to a cloned repo up to GitHub:

```
git push origin
```

Git for Journeymen

Have git remind you what branch you're on, what files are staged, etc:

```
git status
```

See your edits to my_code.py since the last commit:

```
git diff my_code.py
```

View a list of changes committed to the repository:

```
git log -p
```

Switch to an older version of your code:

```
git checkout <hash #, copy hash # from git log output>
```

Switch back to the latest version of your code:

```
git checkout <branch name, probably "master">
```

Collaborating with Git

Get the latest modifications from your teammates:

git pull origin

Create and work on a new branch named new_feature:

git branch new_feature; git checkout new_feature

OR

git checkout -b new_feature

Push your new_feature branch to GitHub:

git push origin

Then issue a pull request on GitHub

Finding a regression bug

Start search and tell Git current code contains the bug:

```
git bisect start; git bisect bad
```

Find an old version of code that doesn't have the bug:

```
git log; git checkout <hash #>; git bisect good
```

Git checks out another version of the code, test it and:

```
git bisect <good or bad>
```

Continue this until Git tells you which is the first bad commit

Git for Masters

When all else fails, read the manual:

https://www.kernel.org/pub/software/scm/git/docs/user-manual.html

Git for Command Line Haters

There are GUI interfaces for Git:

https://git-scm.com/downloads/guis