my git reference card

Routine

Task	Command	Notes
Start new repo	`git init`	- just do this in the new directory - don't nest repos!
Check on things	`git status`	tells you what's up
	`git ls-files`	shows files currently being tracked
Add stuff	`git add <file>`	instead of `<file>`, you can do: - period to add *everything* - subfolder name to add everything in that folder
Update stuff	same as adding	can use `git commit -a` to commit all changes to already-tracked files
Delete stuff	`git rm <file>`	need to do this even if file deleted using "normal" means
Move stuff	`git mv <file> <newlocation>`	same as `git add <newloc>` and `git rm`
Rename stuff	same as moving!	
Commit changes	`git commit`	opens editor to commit message
Commit with message	`git commit -m "message here"`	adds message without opening editor

Oops!

Task	Command	Notes
Recover deleted file (unstaged)	`git checkout -- <file>`	this works before the change is committed – if `git status` tells you a file was deleted and you want it back!
Take changes (adds or rms) out of staging	`git reset HEAD <file>`	this takes changes from the "staging" area, but does not undo the changes!
Messed up recent `commit`	#ERROR	do this *after* doing some more adds/rms (or just with a new message); updates previous commit with new stuff see here for some details

Dealing with history

Task	Command	Notes
Peek at history	`git log`	hit “q” to get back to regular prompt
	`git log --pretty=oneline`	more condensed view
	`git log -pretty=format:"%h %ad %s"`	REALLY HANDY view! Use `%ar` for “relative” date, or `-date=short` for just dates, etc.
	`git log --since=<time>`	lots of options for `<time>` - `"yesterday"`, `"1 week ago"`, `1.week` or `"1.week"`, `"2013-01-30"`, `"10 minutes ago"`, `"last Tuesday"`, etc.! see here for some more details
History of just one file	`git log <file>`	only shows commits that affected `<file>`
“Rewind” temporarily	`git checkout <hash>`	the `<hash>` is (at least) the first several characters of the long SHA1 hash code “address” for a particular commit you want to rewind to NOTE: must first have a clean (committed) working branch (e.g., “master”)
Go back to the current state	`git checkout master`	After you’re done messing around with the “rewind”