4. VSCode and Git
LING 471

Learning outcomes

- Use breakpoints and debugging features in VSCode
- Open a command line interface
- Use basic CMD or Bash commands
- Describe Git at a high level
- Use basic Git commands through the command line or through VSCode
- Exit vi(m) if it shows up

Integrated development environments (IDEs)

- Source code editor + build automator + debugger
  - Plus a whole lot more! (You can even have a Pokemon to code with you)
  - Editor: enter text for program
  - Build automator: path to compiler, interpreter, etc.
  - Debugger: Inspect program state step by step
- We are using VSCode

Programming first steps (preview of next week)

- To do basic programming, you need to understand
  - Input/output (I/O)
  - Assignment (variables and types)
  - Math and logical operators (e.g., **+**, **-**, **and**)
  - Control flow: Conditionals (**if**) and repetition (**for**, **while**)
  - Functions, classes, inheritance, object properties, keywords

Debugging

- Checklist
  - Set up debugger with a running config
  - Set up entry point of program
  - Step into function
  - Step over statements
  - Inspect program state
  - Stopping, resuming
- Let’s try it in VSCode!

Guards

- In Python code, you have probably seen the `if __name__ == '__main__':` line of code
- This called an example of a **guard**
- We haven't quite covered all the parts of this line to fully understand what it is saying
- But! Basically what this is saying is, "If this script is run as a program, do this. But, if it is being imported by another script, do not do this."
- We will come back to this later

Class activity!

- Let’s build a corpus for our class!
  - Drop around 5 pages of English texts in this link:
    - https://tinyurl.com/ling471

Zipf's Law

- In natural language, word frequencies follow a predictable pattern:
  - A small number of words occur very frequently, while most words are rare.
    - r = rank of a word (1 = most frequent)
    - $f(r)$ = frequency of that word
  - The frequency of a word is **roughly inversely proportional to its rank**.

$$f(r) \propto \frac{1}{r}$$

Command line: *nix vs. Windows

- *nix:
  - “Terminal” or “console” (or some variant thereof) for command line; *bash* on Linux, *zsh* on Mac (both quite similar)
    - “ls” to list a directory
    - “cd” to **change** directory
- Windows
  - CMD and PowerShell; *batch* for CMD and *psh* for PowerShell
  - Many bash commands are aliased in PowerShell, which is convenient

Common commands for *nix and Windows

- General format: *nix / CMD/psh (some *nix works on psh)
- List directory: ls/dir
- Create directory: mkdir/mkdir
- Copy file (or directory): cp (cp -r)/copy (xcopy)
- Move (rename!) file or directory: mv/move
- Run a program (without arguments): python to run python
- Ask what the current path is: pwd/echo %cd%
- Connect to remote machine: ssh

Command line power tools

- Your terminal stores previous commands
  - Repeat a complex command by simply hitting the up-arrow
- Terminal knows which paths are available from current directory
  - Autocomplete by pressing Tab
  - When ambiguous and you keep pressing Tab
    - *nix will show possible options
    - Windows will cycle through possible options

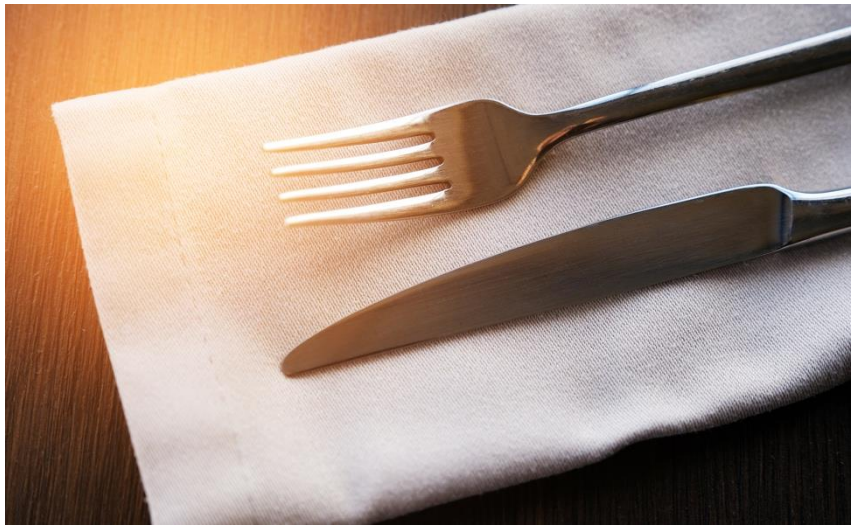
Version and source control: Git

Git logo created by Jason Long and used under CC By-3.0

- Keeps snapshots of all changes to files
  - Organized into batches with comments
- Go back to any version at any time
- Can save lives!
  - Originally for groups of people working on same project
  - Essential for soloers as well for backup and stable versions
- NB: backup requires **push**ing your changes to a remote repository

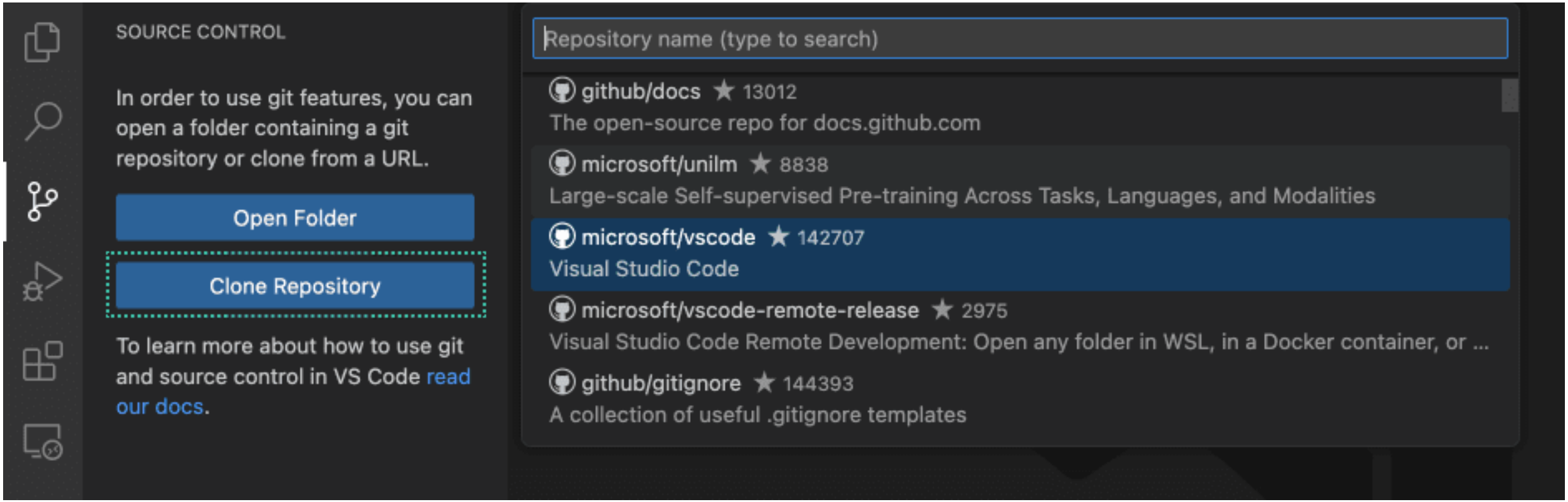
Git repo(sitorie)s

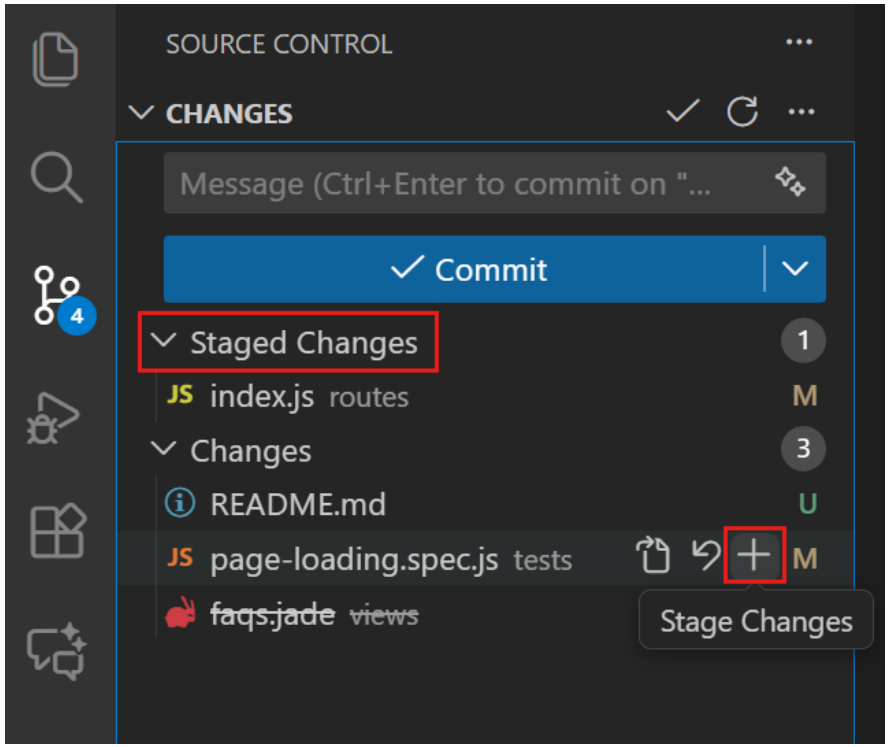
- Local repo
  - The version you have on your own local computer
  - Generally, does not need to be reconciled against other people's local copies until you push your changes
- Remote repo
  - Central location where everyone's local copies are pushed to and reconciled
  - E.g., GitHub



Getting copies of repos

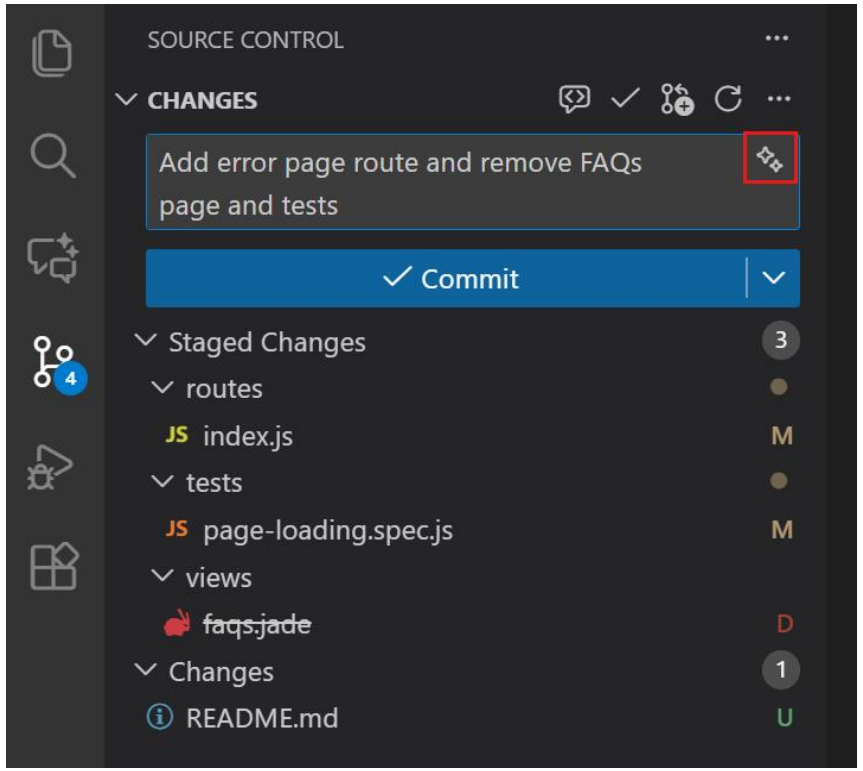
- Clone
  - Using **git clone URL** command (and replace URL with the URL of the git repo) or equivalent in PyCharm
  - Gives you a local copy of entire repo from GitHub
- Fork
  - Creates a clone on GitHub that is linked to the original repo
  - Often used when you want to contribute code to a project (e.g., an algorithm)


SOURCE CONTROL
In order to use git features, you can open a folder containing a git repository or clone from a URL.
Open Folder
Clone Repository
To learn more about how to use git and source control in VS Code read our docs.
Repository name (type to search)
github/docs ★ 13012
The open-source repo for docs.github.com
microsoft/unilm ★ 8838
Large-scale Self-supervised Pre-training Across Tasks, Languages, and Modalities
microsoft/vscode ★ 142707
Visual Studio Code
microsoft/vscode-remote-release ★ 2975
Visual Studio Code Remote Development: Open any folder in WSL, in a Docker container, or ...
github/gitignore ★ 144393
A collection of useful .gitignore templates



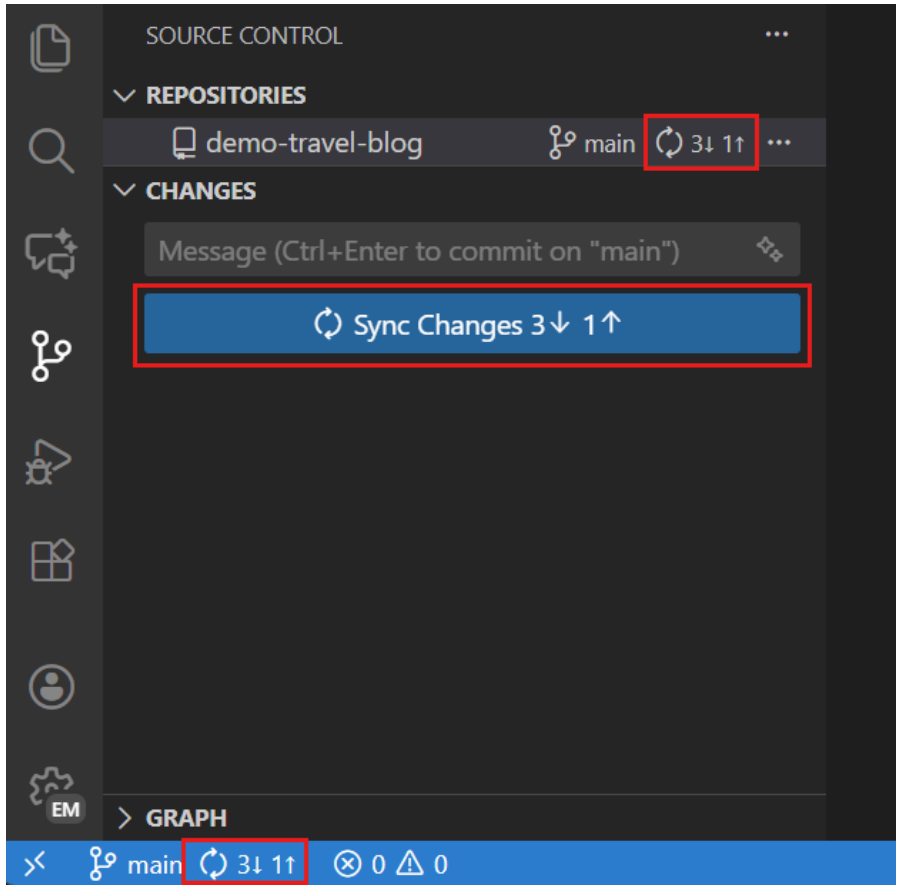
Staging changes

- Tell Git which changes to include in next "version"
- Don't have to include every changed file at once
- Use **git add** command or VSCode equivalent



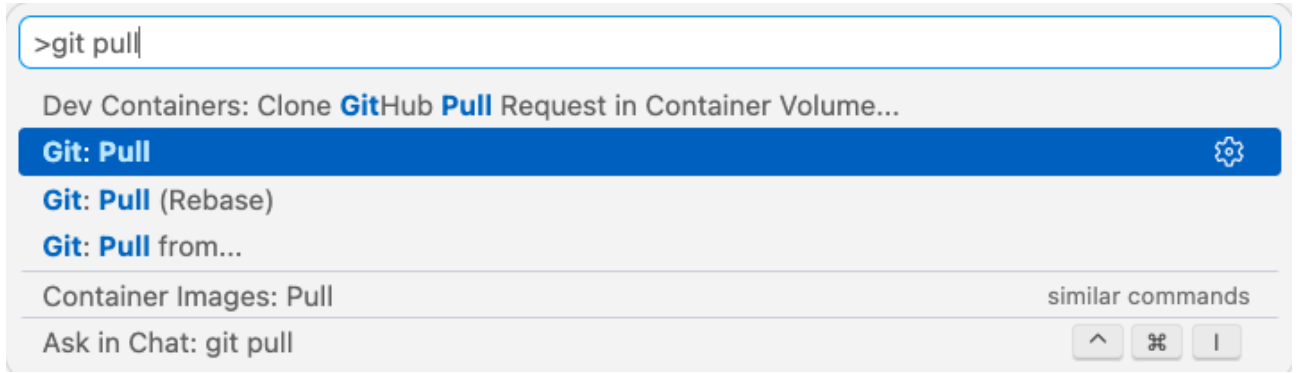
Committing changes

- Tell Git to make a record of your changes
- Use **git commit -m "SUMMARY"** (and fill in the summary)
- Or VSCode equivalent



Pushing changes

- Push your committed local changes to remote repo
- Often, **git push origin main** or VSCode equivalent
  - VSCode defaults to 'Sync Changes', which is a pull and push
  - This is for small, simple projects
  - Larger projects usually don't push to main
- Pushes affect **other people** working on the repo!



Pulling changes

- Asks remote (like GitHub) what new changes were committed and downloads those
- Use **git pull** or VSCode equivalent
- Pulling affects **your** files!

Merge conflicts (aka “headaches”)

- Sometimes, your changes and another person’s (sometimes, past you) conflict with each other
- Git tries its best to resolve these situations but sometimes needs you to mediate
- You have to manually tell Git which changes to keep and which ones to discard, usually using a text editor
- Some advice from Atlassian (which makes a GitHub competitor called BitBucket)

A tip about merge conflicts and vi(m)

- Many Git installations default to handling merge conflicts with a command line program called vim (sometimes vi on older computers)
- This program is useful, but it has a moderately steep learning curve and you probably don't want to use it
- Even exiting the program can be difficult if you don't know how…
  - Just type `:q` and press enter ;)

In-class exercise

- Learn to use the terminal
  - Find out where you are
  - Create and move into a folder
  - Create and rename a file
  - Run a Python program
- Bonus challenges
  - Learn to use Git
    - Clone the ling471 directory at https://github.com/siyuliang/ling471
    - Debug ling471/demos/4_09/debug_demo.py
    - Push to your own GitHub account
  - Connect to Patas and try some terminal commands there too