CYPLAN 255

Urban Informatics and Visualization

HIT RECORD

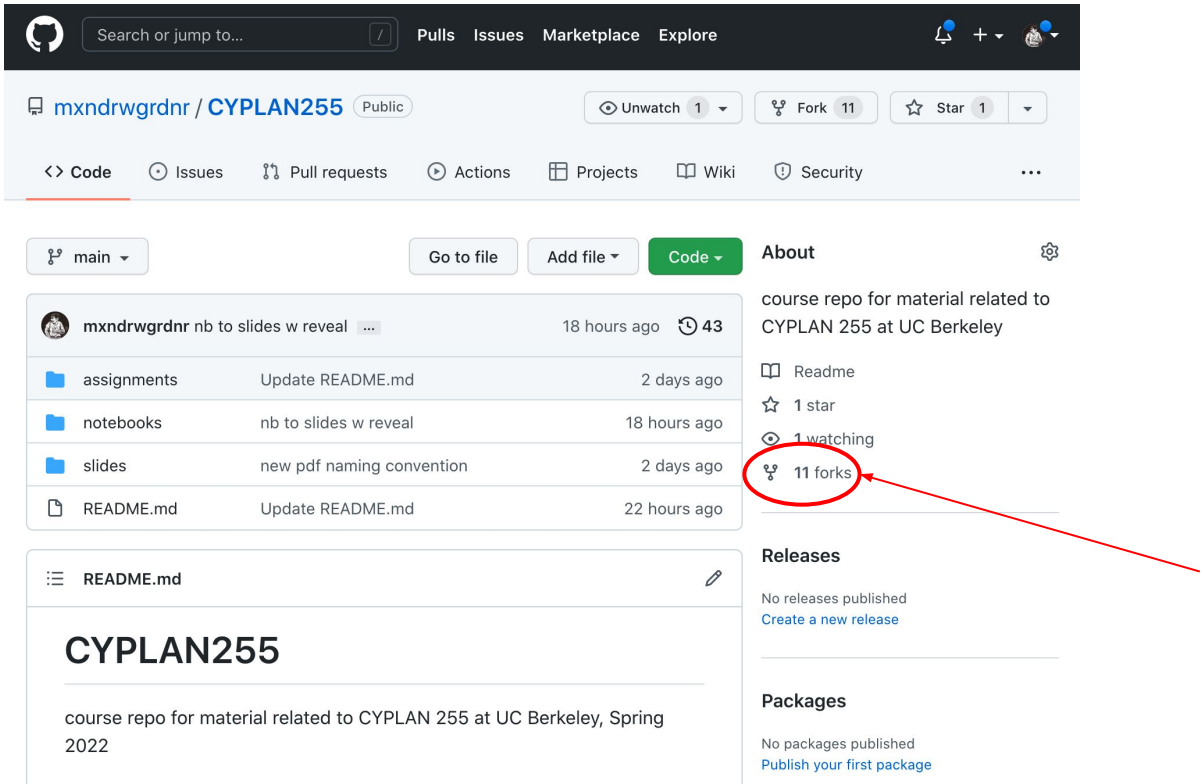
Lecture 02 – Intro to the Command-Line

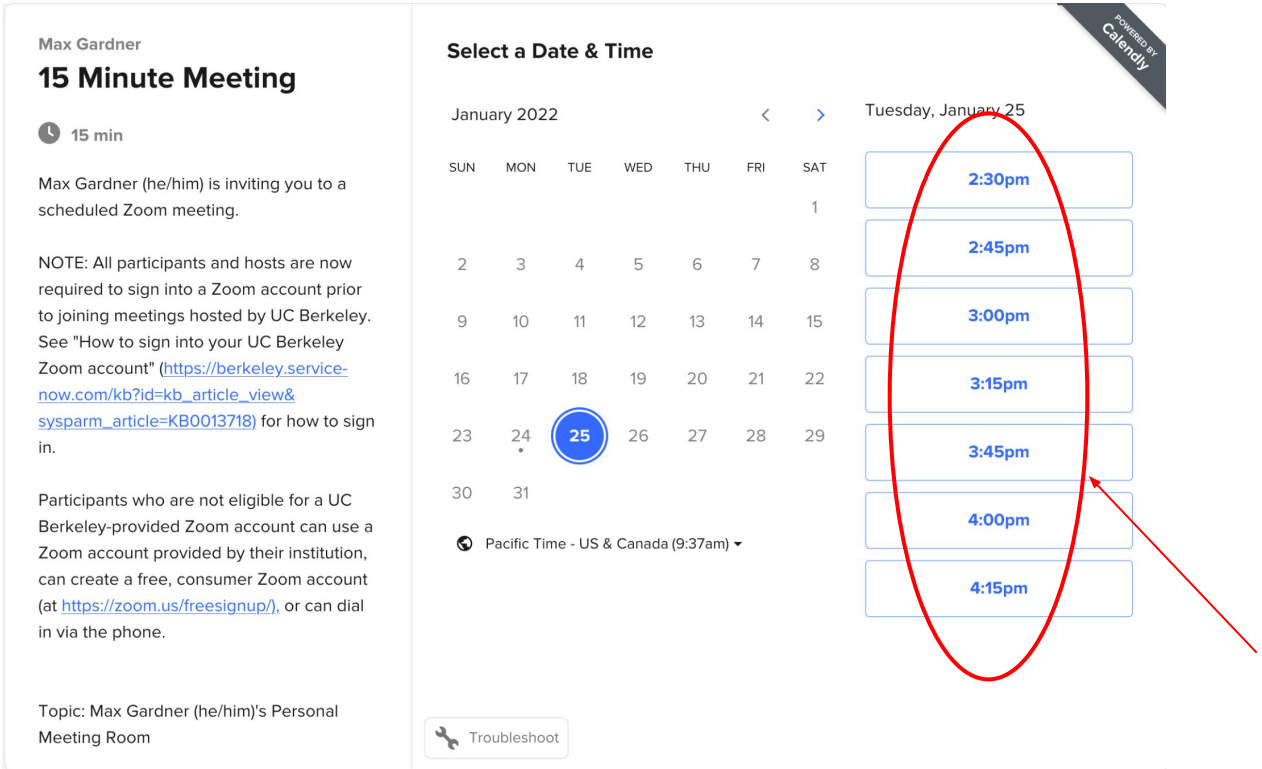
January 24, 2022

Agenda

1. Announcements
2. Command-line basics
3. Getting started with GitHub
4. For next time
5. Questions

1. Announcements


Search or jump to...
Pulls
Issues
Marketplace
Explore
mxndrwgrdnr / CYPLAN255
Public
Unwatch 1
Fork 11
Star 1
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
main
Go to file
Add file
Code
mxndrwgrdnr nb to slides w reveal
18 hours ago
43
assignments
Update README.md
2 days ago
notebooks
nb to slides w reveal
18 hours ago
slides
new pdf naming convention
2 days ago
README.md
Update README.md
22 hours ago
README.md
CYPLAN255
course repo for material related to CYPLAN 255 at UC Berkeley, Spring 2022
About
course repo for material related to CYPLAN 255 at UC Berkeley
Readme
1 star
1 watching
11 forks
Releases
No releases published
Create a new release
Packages
No packages published
Publish your first package


Max Gardner
15 Minute Meeting
15 min
Max Gardner (he/him) is inviting you to a scheduled Zoom meeting.
NOTE: All participants and hosts are now required to sign into a Zoom account prior to joining meetings hosted by UC Berkeley. See "How to sign into your UC Berkeley Zoom account" (https://berkeley.service-now.com/kb?id=kb_article_view&sysparm_article=KB0013718) for how to sign in.
Participants who are not eligible for a UC Berkeley-provided Zoom account can use a Zoom account provided by their institution, can create a free, consumer Zoom account (at https://zoom.us/freesignup/), or can dial in via the phone.
Topic: Max Gardner (he/him)'s Personal Meeting Room
Select a Date & Time
POWERED BY Calendly
January 2022
SUN MON TUE WED THU FRI SAT
1
2 3 4 5 6 7 8
9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28 29
30 31
Pacific Time - US & Canada (9:37am)
Tuesday, January 25
2:30pm
2:45pm
3:00pm
3:15pm
3:45pm
4:00pm
4:15pm
Troubleshoot

2. Command-line Basics

Terminology

- **Command-line interface (CLI)**
  - Generic term which describes how the user sends commands to a computer program
  - As opposed to a *graphical* user interface (GUI), i.e. point-and-click
- **Command-line interpreter** or **“shell”**
  - Refers to a specific program which handles the interface:
    - DOS
    - cmd.exe
    - Powershell

    } Windows

    - **Bash** (“Bourne Again Shell”)
    - Zsh (“Z shell”)
    - Ksh (“KornShell”)

    } Unix

[1]

Options for emulating a Bash shell on Windows

- Cygwin
- Git Bash via Git for Windows
  - Git-specific commands
- Windows Subsystem for Linux
  - Native Bash support on Windows (requires Windows 10+)
- Terminal emulator via PyCharm
  - All-in-one “IDE” solution (Bash/Python/version control)
- Terminal on https://datahub.berkeley.edu
  - Actual Bash shell (hosted remotely)

Basic shell commands and programs (LIVE DEMO)

- `whoami` print username
- `pwd` print working (current) directory
- `ls` list files in working directory
- `cat <filename>` concatenate (print) contents of a file all at once
- `head <filename>` print the first *N* lines of a file
- `less <filename>` scroll through a file one line at a time
- `man <cmd>` show the manual/documentation for a command
- `top` show information about processes
- `echo <arg>` print arguments (strings, env vars, etc.)

Navigating the file system (LIVE DEMO)

- `cd <path>` change directory
- `touch <filename>` make a file
- `rm <filename>` remove a file
- `mkdir <dirname>` make a directory
- `cp <old> <new>` copy a file or directory to a new file or directory
- `mv <old> <new>` move a file or directory to a new file or directory

Navigating the file system - special characters (LIVE DEMO)

- `.` or `./` current directory
- `..` up one directory
- `~` or `~/` home directory
- `/` root directory
- `*` wildcard

Editing text files (in a terminal)

- **Pico / Nano**
- **Vi / Vim**
- Emacs

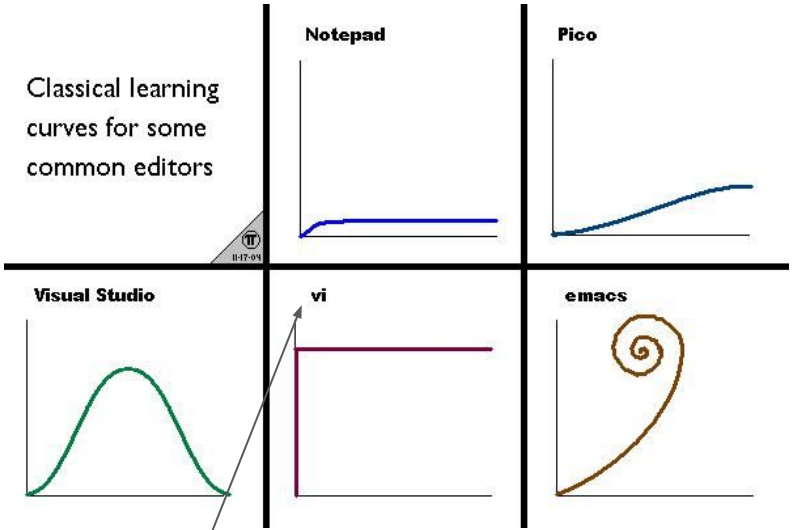
Command-line

- Notepad(++)
- **Sublime Text**

GUIs

- PyCharm
- VSCode

IDEs



[4]

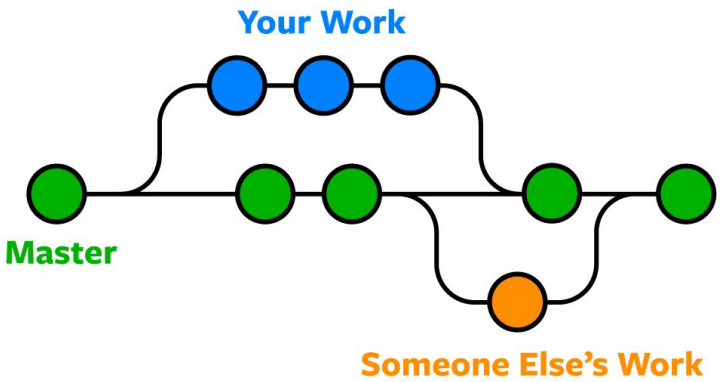
Frequently the system default

3. Git + GitHub

https://docs.github.com/en/get-started/getting-started-with-git

Git

- Git is a tool for *distributed version control*
  - Track changes
  - Log history of changes
  - Merge changes and histories from multiple contributors



[2]

GitHub

- GitHub is a website/service for hosting Git-based projects
- Stores a **remote** copy of a project along with commit history
- Tracks **branches** and **forks** of the main copy (repository)
- Provides a user interface for communicating with collaborators, tracking bugs, viewing commit history, and executing specific Git operations

[3]

Terminology

- **repository** – a folder of files constituting a project
- **commit** (*noun/verb*) – a set of changes entered into the tracking system
- **push** (*verb*) – uploading commits from your local work space to GitHub
- **branch** (*noun*) – a separate working copy of the files, accumulating changes that will be merged into the main version later on
- **pull request** (*noun*) – a request, made through GitHub, for the person in charge of the main branch to merge in changes from another branch
- **fork** (*noun/verb*) – a third-party copy of a public repository, only loosely connected to the original
- **remote** (*noun/adjective*) – the copy of your project stored on GitHub

Configuring Git (LIVE DEMO)

- `git config --global user.name "Mona Lisa"`
- `git config --global user.email "mlisa@berkeley.edu"`
- `git config --list`

Typical Git Workflow on local

1. Do some work (make local changes)
2. Make sure you're on the right **branch**
   - `git branch` or `git checkout <branch name>`
3. Tell Git which changes you want to **commit**
   - `git add <filename>`
4. **Commit** your changes and describe them
   - `git commit -m "this is my first git commit"`
5. **Push** your local changes to remote
   - `git push`

Forking the class GitHub repo (LIVE DEMO)

- Breakout Rooms Part I
  1. Open a browser and go to https://github.com/mxndrwgrdnr/CYPLAN255
  2. Create your own fork
  3. Clone your fork
- 5 MINUTE BREAK: Instructor will create Assignment 0
- Breakout Rooms Part II
  1. Sync your fork
  2. `git pull` the changes to you local (cloned) copy
  3. `git checkout` the “assignments” branch
  4. Create/copy a new file WITH A UNIQUE NAME and make some changes to it
  5. Add, commit, and push your changes (see previous slide)
  6. Open a pull request

4. For next time

For next time (“homework”)

- Complete Assignment 0
- Complete any of the readings you have not completed

5. Questions?

Image attribution

[1] https://elessar20.files.wordpress.com/2008/05/korn1.jpg
[2] https://www.nobledesktop.com/blog/what-is-git-and-why-should-you-use-it
[3] https://github.com/logos
[4] https://missing.csail.mit.edu/2019/files/editor-learning-curves.jpg

Bonus Material

GitHub Authentication with SSH

1. `ssh-keygen -t ed25519 -C "your_email@example.com"`
2. “Enter file in which to save the key (/home/jovyan/.ssh/id_ed25519):”
   - `<return>`
3. “Enter passphrase (empty for no passphrase):”
   - `<return>`
4. “Enter passphrase again:”
   - `<return>`
5. Add your ssh key to your GitHub account
   - https://docs.github.com/en/authentication/connecting-to-github-with-ssh/adding-a-new-ssh-key-to-your-github-account

Screen

Screen is an incredibly useful tool for doing work at a unix terminal. It allows you to leave a program running in the background. This allows you to do something else in the same terminal window, or close the window altogether, and then come back to it whenever you like. The basic usage is as follows:

- `screen -S <my-screen>` Open a screen named “my-screen”
- `screen <ctrl + a> <d>` “Detach” from the screen
- `screen -r my-screen` Resume your screen
- `<ctrl + d>` Terminate the screen