Welcome.

Everyone:

- Pull the updates from the course GitHub repo:
  - `cd <46120-PiWE repo>`
  - `git pull upstream main` ← you might have “upstream2” instead

Physical students:

- Sit WHEREVER you want. 😎
- Turn off laptop volume (mute). ←IMPORTANT!
- Log into the Zoom meeting.
  - Microphone muted. Camera off.

46120: Scientific Programming for Wind Energy

Environments and communicating code

Jenni Rinker

Agenda for today.

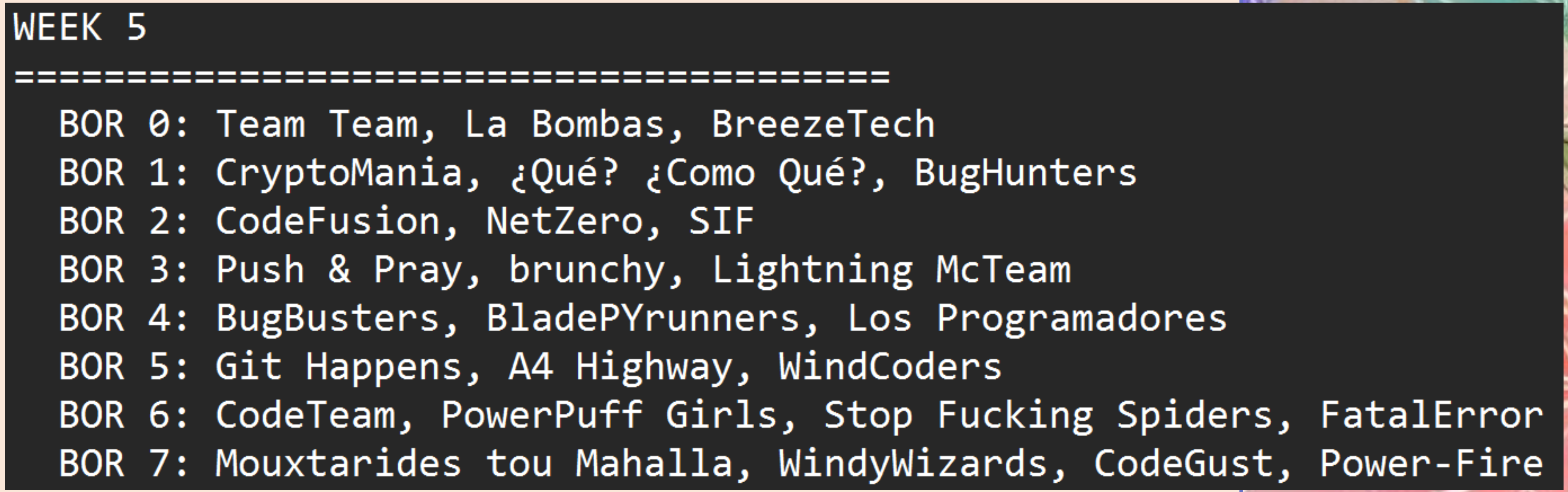
- Pull new course material ✓
- Round robin.
- Environments.
- Communicating code.
- Work on the CodeCamp project.
  - Due next Wednesday at midnight.

Round robin

Share solutions with your peers and give feedback.

Time to review and collaborate.

- 1 round of 25 minutes.
- 5 minutes: chaos.
- 20 minutes: present/discuss homework. Today's feedback focus:
    1. How "clean" do you feel the team's code is? How easy to understand?
    2. How is collaborating with git going? Any better?
- Afterwards: plenum discussion.
    - Be ready with questions!



```
WEEK 5
========================================
  BOR 0: Team Team, La Bombas, BreezeTech
  BOR 1: CryptoMania, ¿Qué? ¿Como Qué?, BugHunters
  BOR 2: CodeFusion, NetZero, SIF
  BOR 3: Push & Pray, brunchy, Lightning McTeam
  BOR 4: BugBusters, BladePYrunners, Los Programadores
  BOR 5: Git Happens, A4 Highway, WindCoders
  BOR 6: CodeTeam, PowerPuff Girls, Stop Fucking Spiders, FatalError
  BOR 7: Mouxtarides tou Mahalla, WindyWizards, CodeGust, Power-Fire
```

Notes in plenum.

- (add here)

5-minute survey, all together.

Results and analysis posted to Learn later ☺

Environments

Keep stuff from messing up other stuff.

Problems with changing package/Python versions.

- We have seen:
  - Tests that pass or fail depending on the Python/matplotlib version.
- Some other common situations:
  - Two packages require different versions of a required package.
  - A package requires a very specific order of required-package installation. You install another package, which messes everything up.
  - You have developed a package and need to test whether the dependencies get installed correctly.**
- Similar to branches in repos, we can use *environments*, which isolate specific versions of Python/packages to ensure code runs smoothly.

Environments are isolated spaces.

- Similar to branches in a repo.
- Allows you to create a special space, possibly with specific Python version, where you can install any packages you need.
- Two common ways to create environments:
  - conda -- allows different Python versions
  - Python virtual environments (venv) -- tied to a specific Python kernel
- You can switch between environments in the Anaconda Prompt and in VS Code.
- We’ll learn conda here, but you’ll see venv on the gbar cluster.
  - A comparison of conda versus venv is given in the appendices.

How to create/activate a PiWE environment.

Create/set up environment

1. Open an Anaconda Prompt and cd into your team repo.
2. Create a new environment with the name “piwe” and Python 3.11:

   ```
   conda create -n piwe python=3.11 -y
   ```
3. Activate the environment:

   ```
   conda activate piwe
   ```
4. Install the packages we need:

   ```
   pip install numpy matplotlib scipy
   pandas pytest
   ```

Activate environment

In Anaconda Prompt

1. Open an Anaconda Prompt.
2. Activate the environment:

   ```
   conda activate piwe
   ```
3. Run pytest:

   ```
   pytest test_week4.py
   ```

In VS Code

1. Bottom bar, hold mouse over options until you find one that is a path to `python.exe`.
2. Click that, then select piwe from the drop-down menu.

How to use environments in PiWE.

- For CodeCamp project, not required.
  - You are welcome to use it if you want your tests to pass.

- For final project, will be essential.
  - You will want to test installing your package in a clean environment, so you know that everything is installed correctly.

Questions?

Communicating code

Part of programming is thinking programmatically.

- Three pillars of Computational Literacy [1]:
  - Cognitive CL: Breaking down a problem into steps.
  - Material CL: Turning those steps into code.
  - Social CL: Communicating/collaborating on code.
- Thinking programmatically is useful for both writing code and for communicating code.
- Three main tools we'll present:
  - Pseudocode
  - Comment prototyping
  - Code diagrams

Pseudocode.

- As the name suggests, **pseudocode** is fake code.
    - Useful for sketching out logic with pen and paper.

- No formal syntax, so you can develop your own style.

- A personal example combining 6 10-minute files to 1 1-hour dataframe.

```
define path to data directory

get list of files          initialize empty dataframe

for each file:
    load the 10-minute data
    update column names
    assign to master dataframe
reset time index to start from zero
```

Comment prototyping.

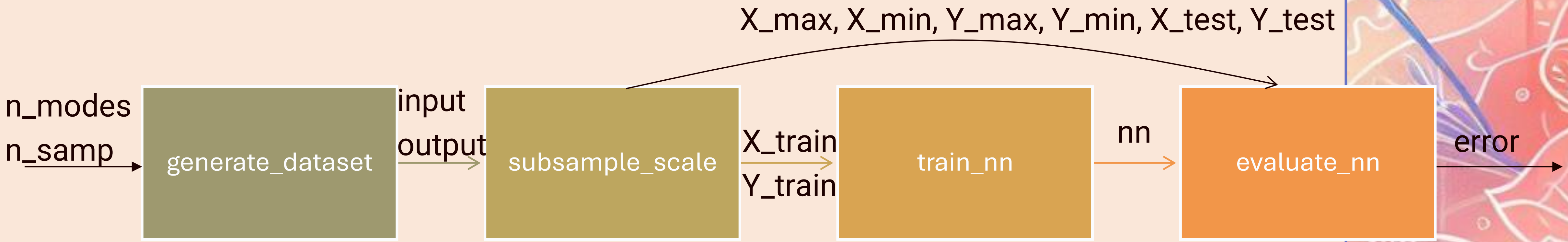
- This is a Jenni^{TM} thing.
  - It's not on the internet.
- Helps if you're staring at a blank .py file and have programmer's block.
- Simple concept: prototype your code by writing the comments first.
  - Nice thing: comes right from pseudocode.

```
# define path to data directory

# get list of files in directory

# initialize empty dataframe

# loop over files

        # load the 10-minute data

        # update column names

        # assign to master df

# reset time index to start from 0
```

Code diagrams.

- For larger code, you may want to communicate/discuss/review the **architecture**.
    - I.e., how the code is structured, what are the inputs/outputs of your functions, etc.
- There isn't really a formalized way, so there is flexibility.
    - Although class diagrams [4] are good for classes.
- Could of course use your black-box diagrams.
    - Simple example with a neural-net training:

Potential to combine with pseudocode



In short.

- There are many ways you can communicate about your code.
- For CodeCamp, your `README.md` file must have a quick-start guide and explanation of how the code works.
    - Target audience is a fellow student who has freshly cloned the repo. Assume that they have not taken this class but have the same Python/terminal skills as you.
- The sky is the limit!

Questions?

Homework for this week

Turbie here we cooooommmmeeeee!

Remember, you're expected to work ***about 6 hours outside of class****. Schedule accordingly.*

Homework.

- **Short summary**: finish the CodeCamp project.
  - Updated description/requirements in week06 subfolder on GitHub, including more info on peer feedback next week.
- We'll open BORs in a minute. Enter room corresponding to your Team ID (on team excel sheet).
- **To get help during class**: Post in Slack / #debugging if you want a TA to enter your BOR or come find your group.

Any questions?

References.

1. DiSessa, Andrea A. *Changing minds: Computers, learning, and literacy*. Mit Press, 2000.