Live Coding

Question: Why do we teach programming using participatory live coding?

Objectives

- Explain the advantages and limitations of participatory live coding.
- Summarize the key dos and do nots of participatory live coding.
- Demonstrate participatory live coding.

What is Live Coding

We do not use slides in our lessons. Instead, instructors plug their laptop into the projector and work through the lesson, typing in the code, reformatting data, and talking as we go. This is called “live coding”. However, the instructor is not live coding in a vacuum. Importantly, learners are strongly encouraged to “code-along” with the instructor.
We refer to the practice of having the instructor live code and the learners code along as “participatory live coding” or, less formally, ‘code-along sessions’.

Two things to take into account

- You **may** use slides, eg. for a short introduction, showin a clarification scheme...
- You don't have to know the lesson by heart. You follow along with printouts or a tablet

Why Participatory Live Coding?

Some advantages

- Watching a program being written is more **compelling** than watching someone page through slides that present bits and pieces of the same code.
- It enables instructors to be **more responsive** to “what if?” questions. Where a slide deck is like a railway track, participatory live coding allows instructors to go off-road and follow their learners’ interests.
- **Lateral knowledge transfer**: participatory live coding facilitates the transfer of tacit knowledge – people learn more watching how instructors do things.

Some advantages (2)

- It **slows the instructor down**: if she has to type in the program as she goes along, she can only go twice as fast as her learners, rather than ten-fold faster as she could with slides.
- Learners get to **see instructors' mistakes** and how to diagnose and correct them. Novices are going to spend most of their time doing this, but it is left out of most textbooks.

Some challenges

- It requires instructors to **be able to improvise** when things go wrong or when learners have questions not directly addressed in the text of the lesson.
- It can be hard for learners to **listen and type at the same time**. This is why it is very important that instructors first explain what they are going to do, then say what they are typing as they type it, and then explain what they did again afterwards.
- It may take a bit of practice for instructors to get used to **thinking aloud while coding** in front of an audience.

> Live coding fits well into the practice-feedback model by providing learners with continuous opportunities for practice (every time they type in a line of code) and continuous feedback (their code either works or fails).

Exercise: Compare and Contrast

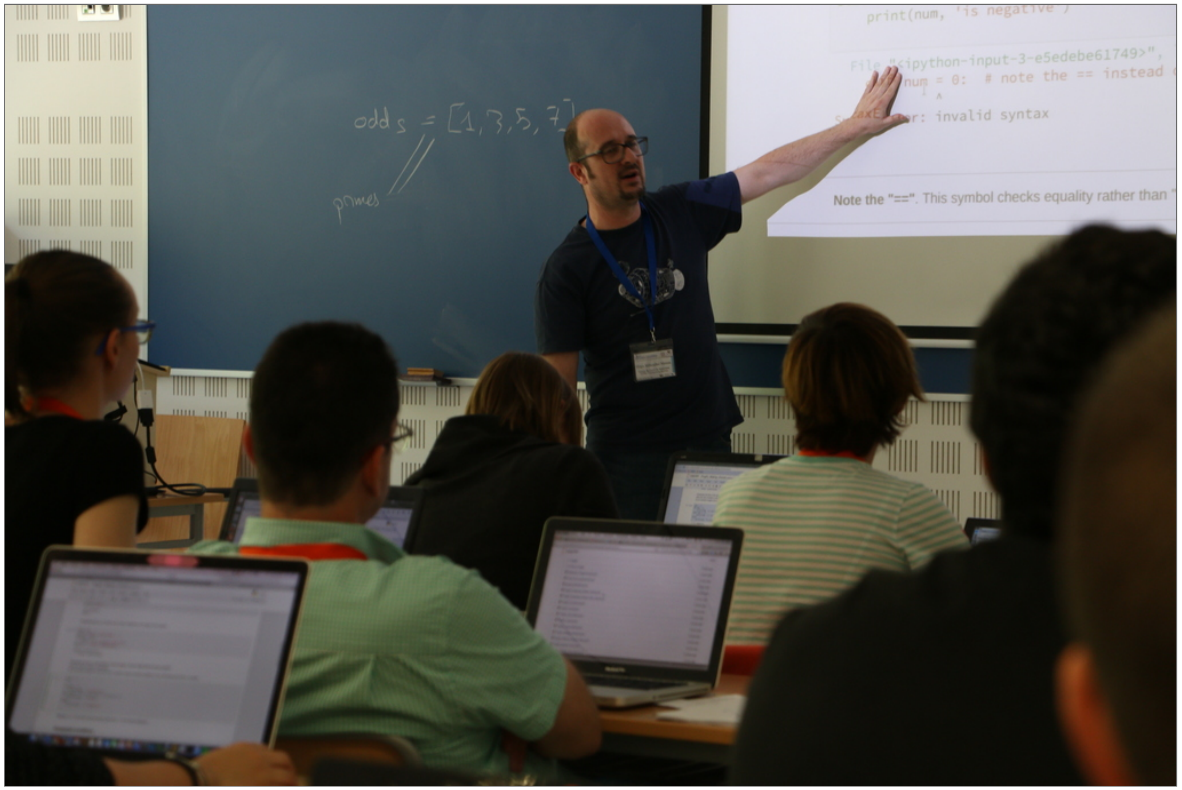
Watch this first participatory live coding demo video: **https://youtu.be/bXxBeNkKmJE** and this second demo video: **https://youtu.be/SkPmwe_WjeY** as a group and then summarize your feedback on both in the Etherpad. Use the 2x2 rubric for feedback (content and presentation + -).

In the videos, the bash shell `for` loop is taught, and it is assumed learners are familiar with how to use a variable, the `head` command and the content of the `basilisk.dat unicorn.dat` files.

This exercise and discussion should take about 15 minutes.

Top Ten Tips for Participatory Live Coding in a Workshop

1. Stand up and move around the room if possible



- makes the experience more interactive
- use the board for clarifications, higlight things, visually emphasize
- use a microphone if possible

Top Ten Tips for Participatory Live Coding in a Workshop

1. Stand up and move around the room if possible
2. **Go slowly**

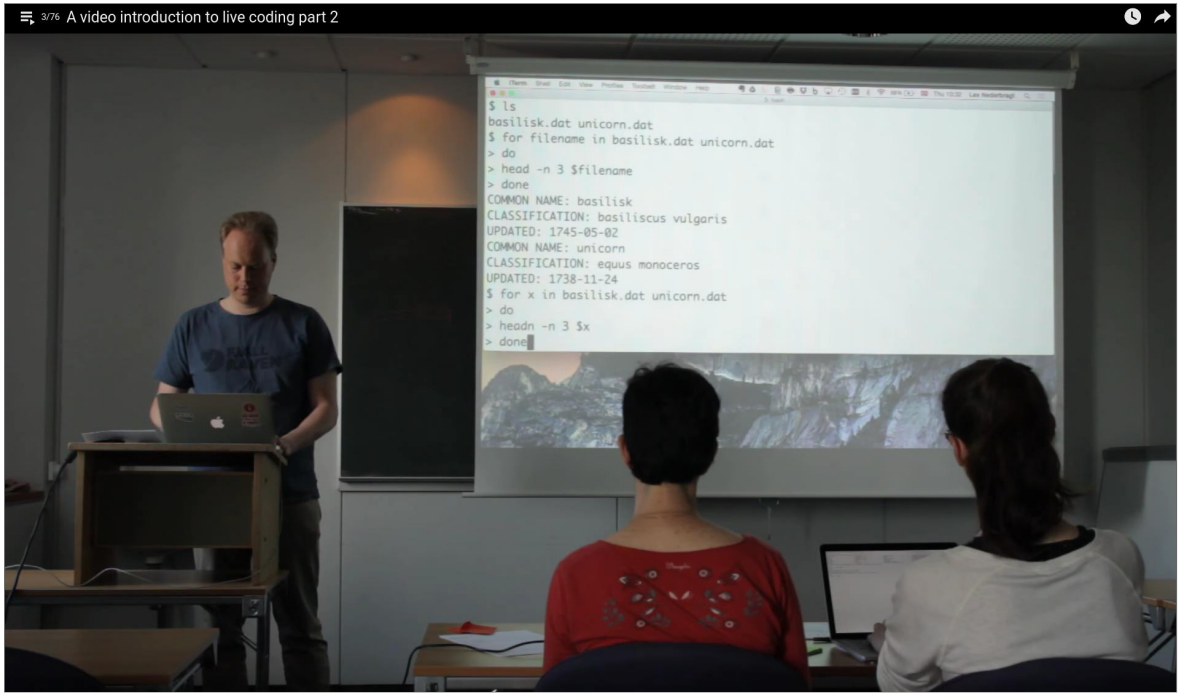
- Everything you do: every command you write, word of code you type, menu you click... whatever, say out loud what you are doing while doing it.

Top Ten Tips for Participatory Live Coding in a Workshop

1. Stand up and move around the room if possible
2. Go slowly
3. **Mirror your learner's environment**

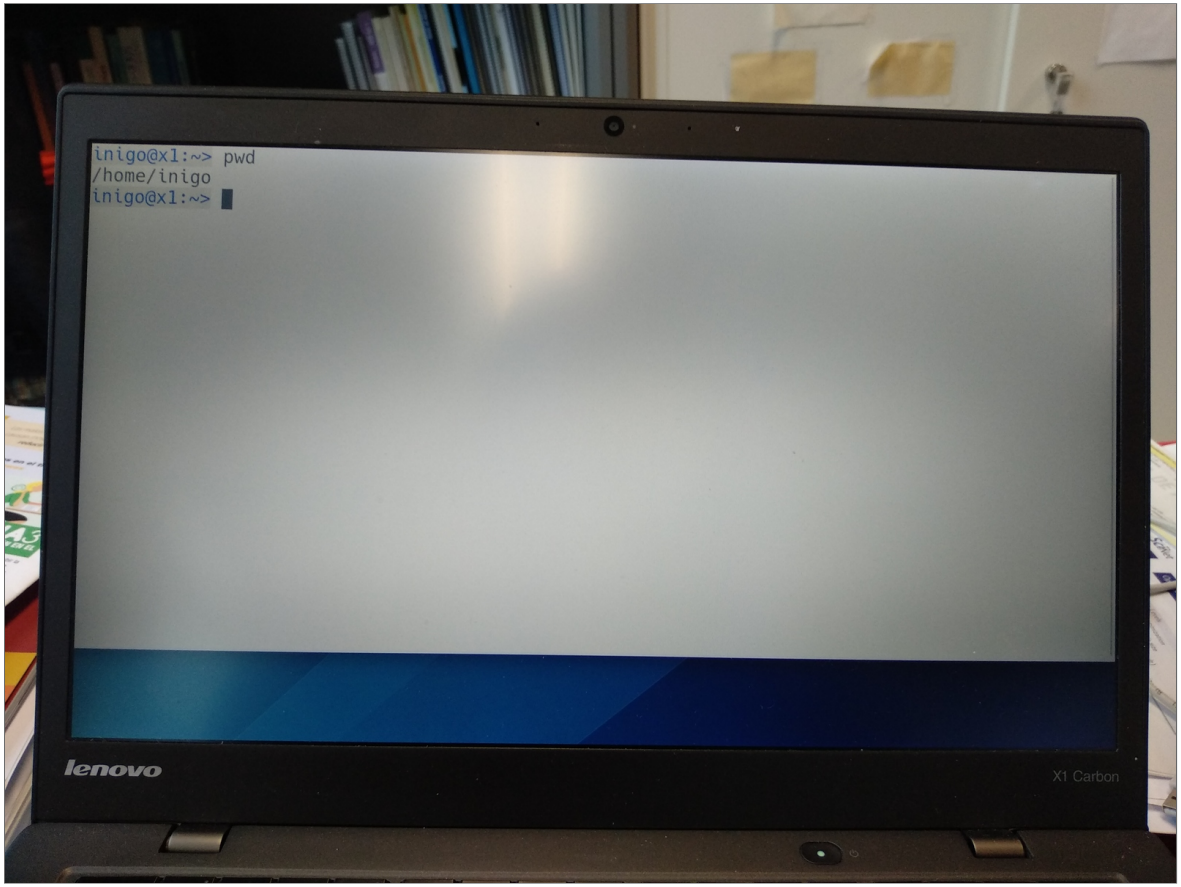
- Try to create an environment that is as similar as possible to what your learners have in order to reduce cognitive load: same terminal prompt (tell them), no shorcuts, no aliases... even a clean user

4. **Use screen wisely**



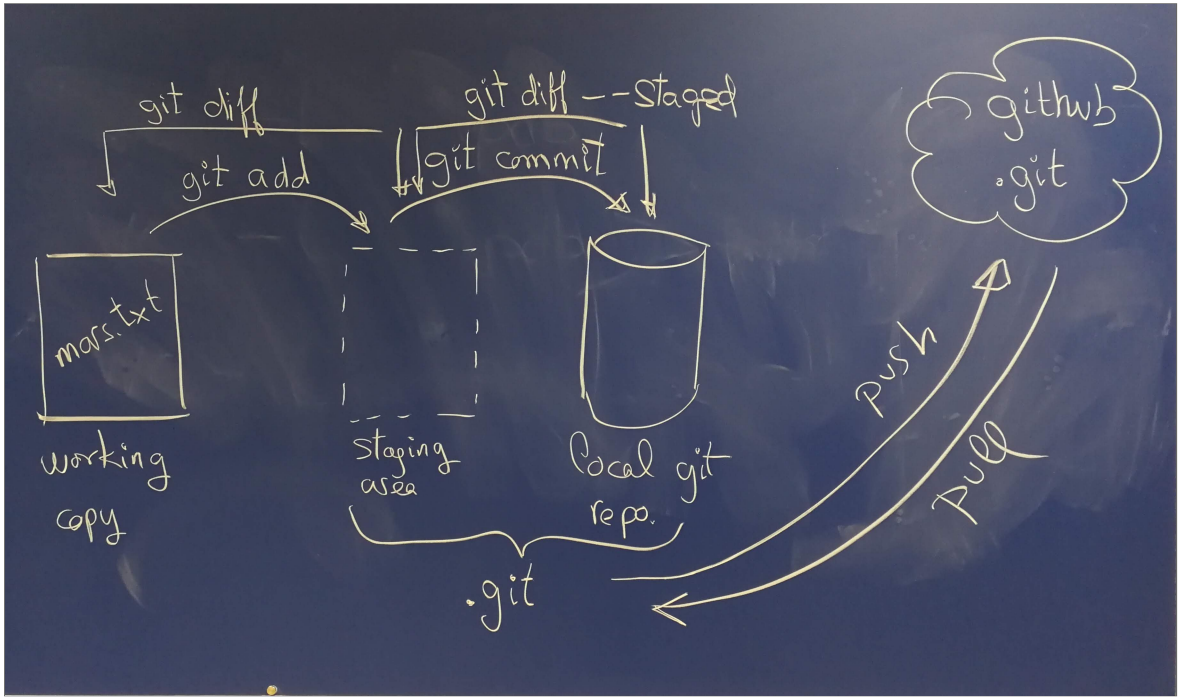
- Use a big and clear font
- Maximize the window
- Black font on white background
- Be careful with the bottom of the screen

4. **Use screen wisely...** try hard...



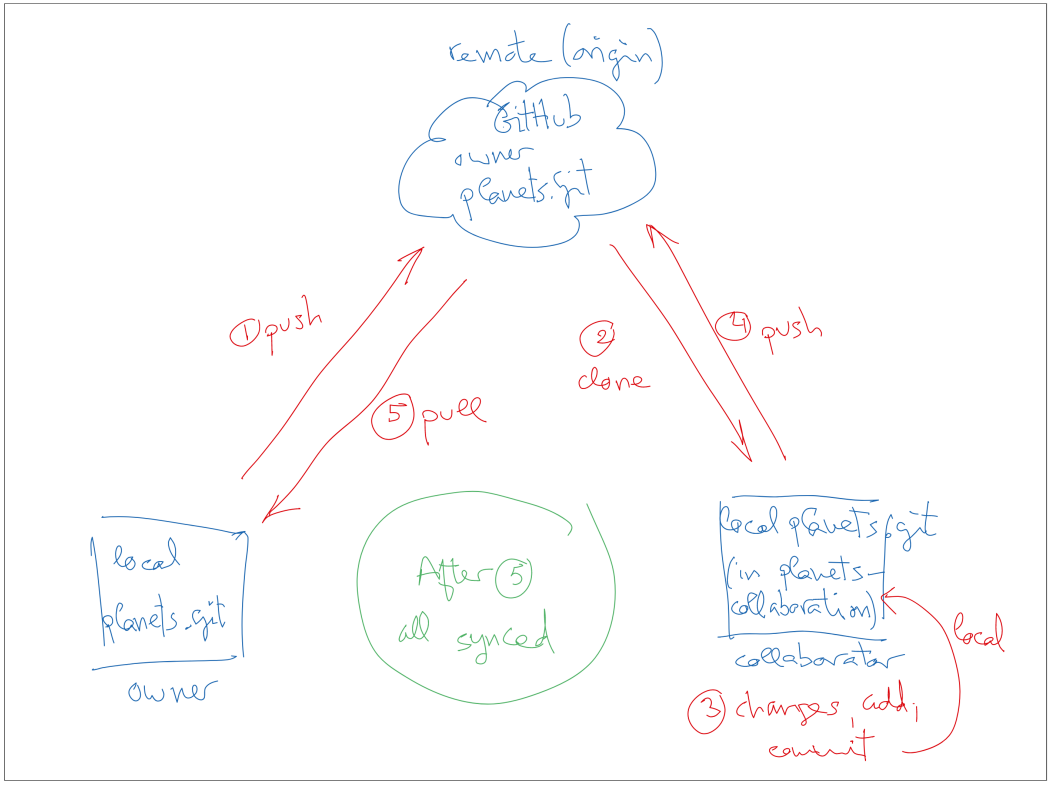
- With the terminal, remove every border
- When using jupyter notebooks, go full screen.
- Use virtual desktops with eg different colour for the terminal and the web browser, even with notifications when changing.

5. **Use illustrations... in person**



- Drawings / schemes
- commands used
- clarifications

5. **...or online**


remote (origin)
GitHub
owner
planets.git
1 push
2 clone
4 push
5 pull
local
planets.git
owner
After 5
all synced
local planets.git
(in planets-
collaboration)
collaborator
3 changes, add, commit
local

Top Ten Tips for Participatory Live Coding in a Workshop

1. Stand up and move around the room if possible
2. Go slowly
3. Mirror your learner's environment
4. Use your screen wisely
5. Use illustrations
6. **Turn off notifications**

- Both from the phone and the laptop

Top Ten Tips for Participatory Live Coding in a Workshop

1. Stand up and move around the room if possible
2. Go slowly
3. Mirror your learner's environment
4. Use your screen wisely
5. Use illustrations
6. Turn off notifications
7. **Stick to the lesson material**

- Lessons are *very* well designed and all follow a story... follow it!
- Use notes, tablet, laptop and don't be shy to use it
- If something can go wrong, it will go wrong

Top Ten Tips for Participatory Live Coding in a Workshop

1. Stand up and move around the room if possible
2. Go slowly
3. Mirror your learner's environment
4. Use your screen wisely
5. Use illustrations
6. Turn off notifications
7. Stick to the lesson material
8. **Leave no learner behind**

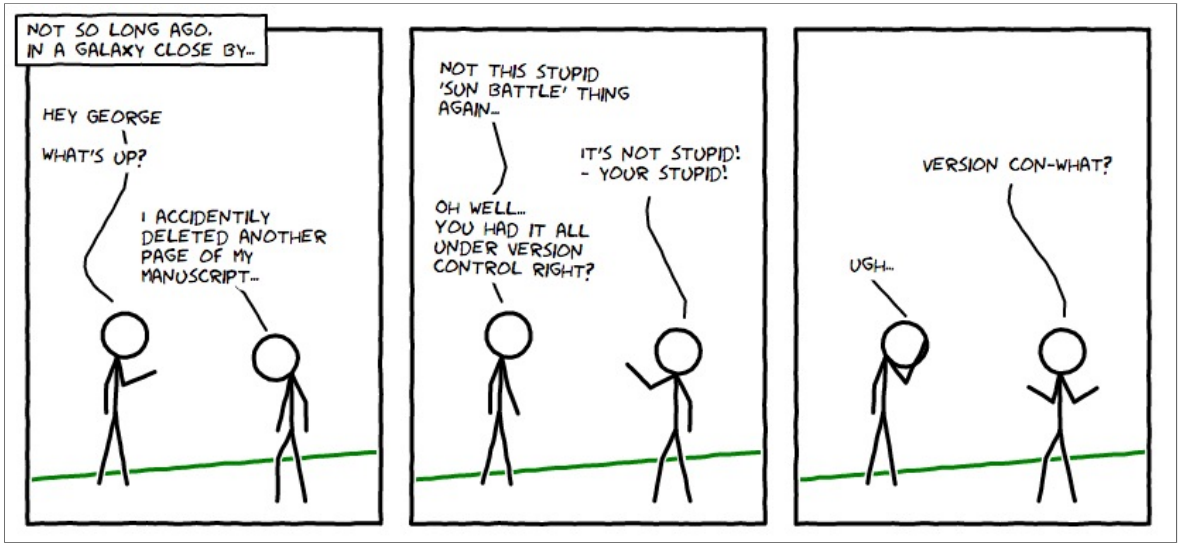
- Use the sticky notes for feedback on too slow - too fast
- Better some learners bored for a while that one lost forever

Top Ten Tips for Participatory Live Coding in a Workshop

1. Stand up and move around the room if possible
2. Go slowly
3. Mirror your learner's environment
4. Use your screen wisely
5. Use illustrations
6. Turn off notifications
7. Stick to the lesson material
8. Leave no learner behind
9. **Embrace mistakes**

- You will make mistakes
- As you became an expert in the lessons you mistake rate will lower a lot so... prepare mistakes beforehand on purpose!

10. **Have fun!!**


NOT SO LONG AGO. IN A GALAXY CLOSE BY...
HEY GEORGE
WHAT'S UP?
I ACCIDENTILY DELETED ANOTHER PAGE OF MY MANUSCRIPT...
NOT THIS STUPID 'SUN BATTLE' THING AGAIN...
IT'S NOT STUPID! - YOUR STUPID!
OH WELL... YOU HAD IT ALL UNDER VERSION CONTROL RIGHT?
VERSION CON-WHAT?
UGH...

10. **Have more fun!!**

Top Ten Tips for Participatory Live Coding in a Workshop

1. Stand up and move around the room if possible
2. Go slowly
3. Mirror your learner's environment
4. Use your screen wisely
5. Use illustrations
6. Turn off notifications
7. Stick to the lesson material
8. Leave no learner behind
9. Embrace mistakes
10. Have fun!

Exercise: Practice Training

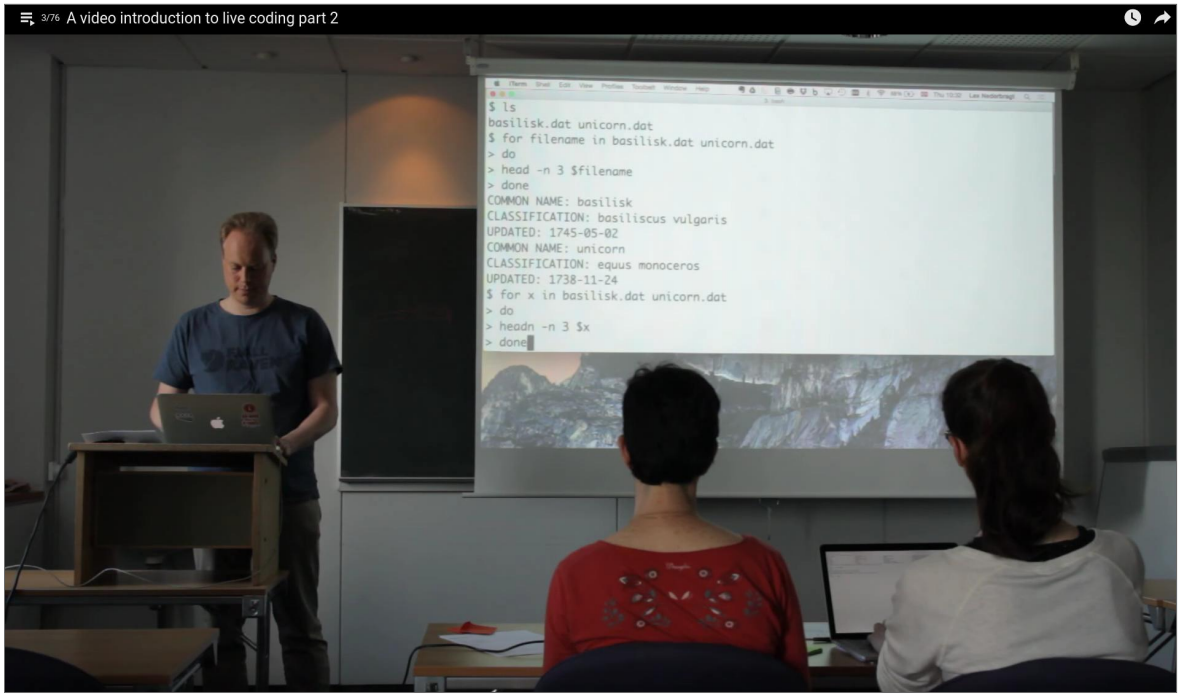
1. Split into groups of three.
2. Assign roles, which will rotate: presenter, timekeeper, note-taker.
3. Have each group member teach 3 minutes of your chosen lesson episode using live coding. For this exercise, your peers will not “code-along.” Before you begin, briefly describe what you will be teaching and what has been learned previously.
4. After each person finishes, each group member should share feedback (starting with themselves) using the 2x2 rubric for feedback (content and presentation + -). The timekeeper should keep feedback discussion to about 1 minute per person; this may leave some time at the end for general discussion. The note-taker should record feedback in the Etherpad.
5. Trade off roles.

25 min

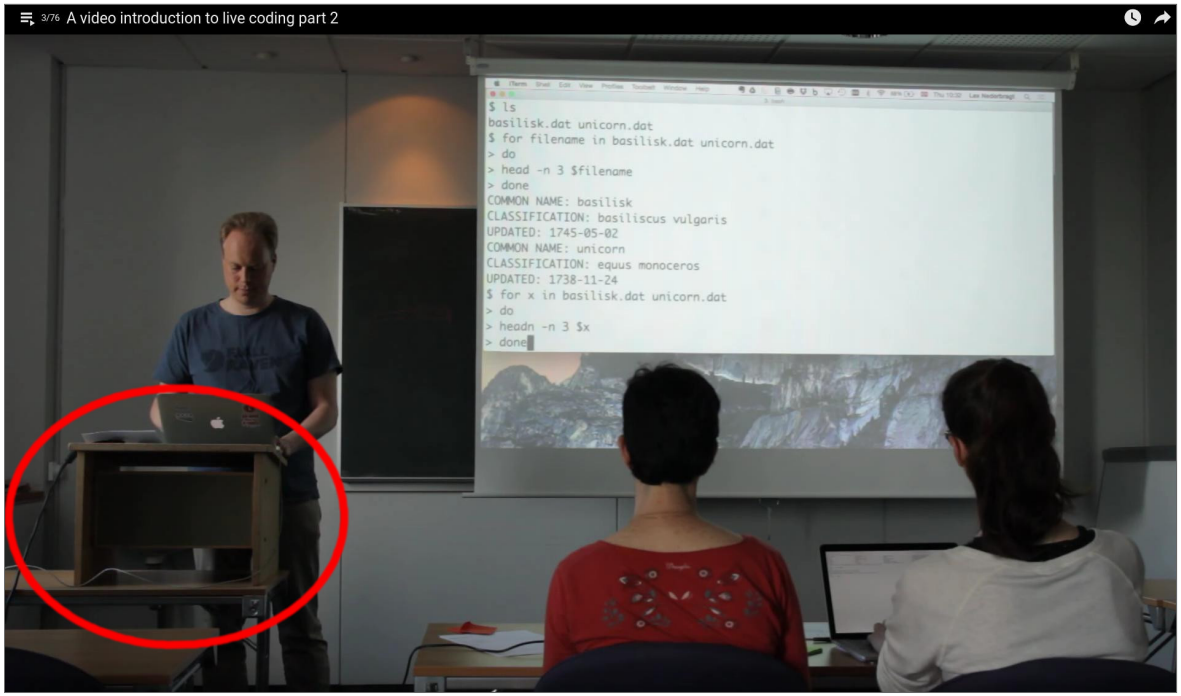
Key Points

- Live coding forces the instructor to slow down.
- Coding-along gives learners continuous practice and feedback.
- Mistakes made during participatory live coding are valuable learning opportunities.

Learn from others!



Learn from others!



Learn from others!

Exercise: Practice Training - Reoun Two (if time allows)

1. Return to your groups and repeat the previous live coding exercise, re-teaching the same content as before. This time, the presenter should incorporate changes based on feedback received, and everyone should try to ‘level up’ their feedback using the rubric for teaching demos.
2. When you are finished, add some thoughts on this process to the Etherpad: What did you change? Did it work better or worse with the change? How might you do it if you were to teach it again?

This exercise should take about 10 minutes for rubric discussion, 25 minutes for teaching, and 10 minutes for de-brief.

Thanks for your attention!