Peer learning in online programming courses: pair programming labs and peer code review

Charlotte Desvages

Monday 7th June 2021

School of Mathematics, University of Edinburgh, UK

Background: Computing labs in the School of Maths

Background: Computing labs in the School of Maths

- Computing courses from Y1 to MSc, most are taught in either Python or R.
- Topics include numerical analysis, statistics, data science, machine learning, optimisation...
- Many courses run **drop-in** computer labs: students come sit in the lab and work through exercises, tutors walk around the room to check in and help.
- Typically, during a lab, pairs and triples form organically as students sit next to friends.

Moving online: Summer 2020

- *Problem:* computer labs (with shared dirty keyboards!) are not an optimal use of our teaching spaces for hybrid, socially-distanced teaching.
  $\rightarrow$ Early decision: all our computer labs will be **fully online**.
- *New problem:* keeping students engaged can be more difficult in online courses, and our students are not used to distance learning.
  $\rightarrow$ the "passive" drop-in lab is clearly not the best use of our limited contact time with the students.

Having to rethink course delivery provided an opportunity to refresh the syllabus and rethink **learning outcomes**, to emphasise good programming practice and familiarise students with standard workflows.

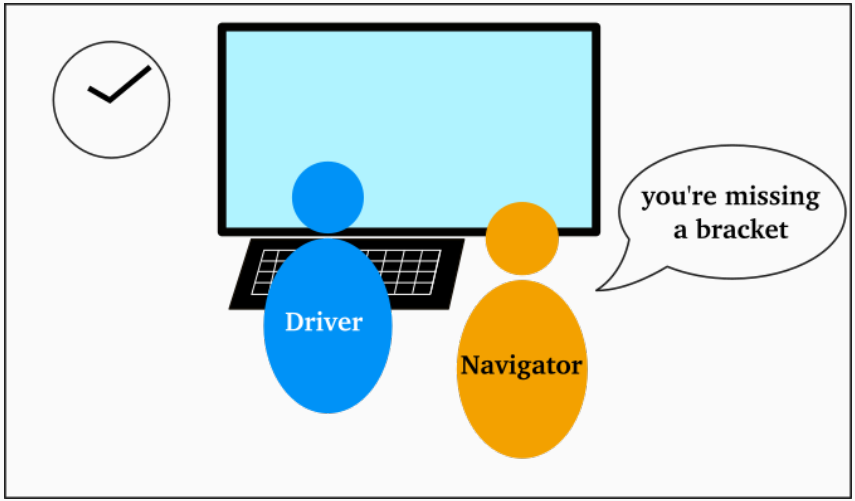
Combining collaborative learning and good practice

How can I promote peer learning and effective collaboration in an online programming class?

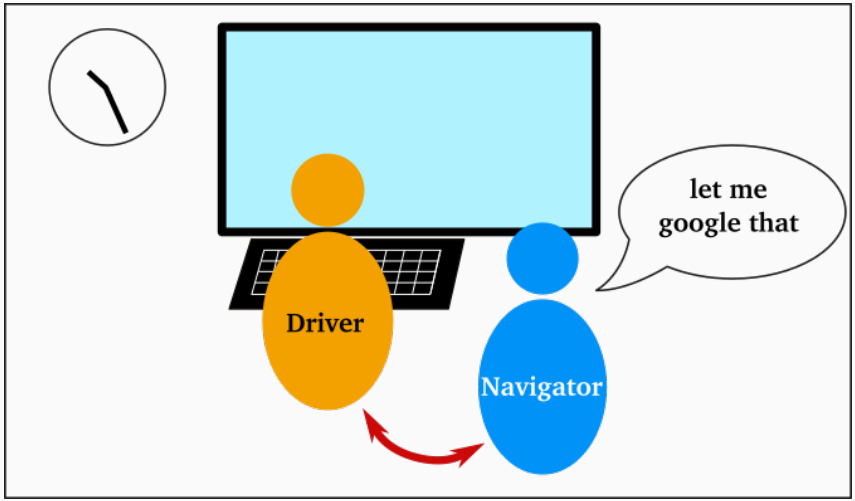
1. Synchronous:
    - **Pair programming** in problem-based labs
    - Small group discussion-based activity
    - Small group peer code review
2. Asynchronous:
    - Discussion board (Piazza)
    - **Peer-assessed code review** task

Pair programming as a pedagogical tool

The pair programming workflow



The pair programming workflow

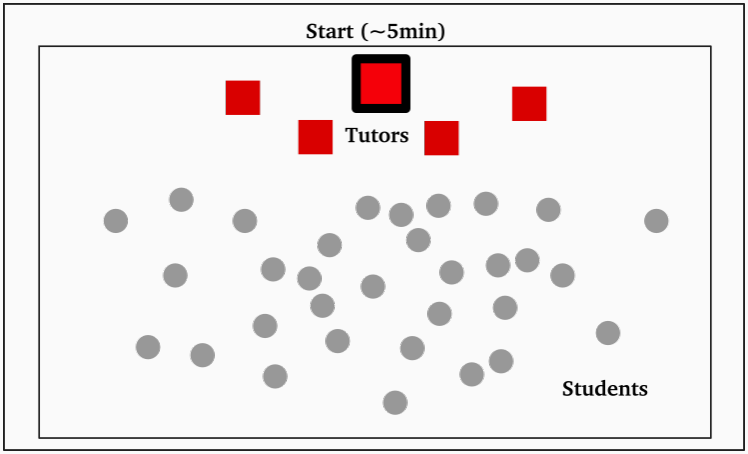


Motivations for trying pair programming

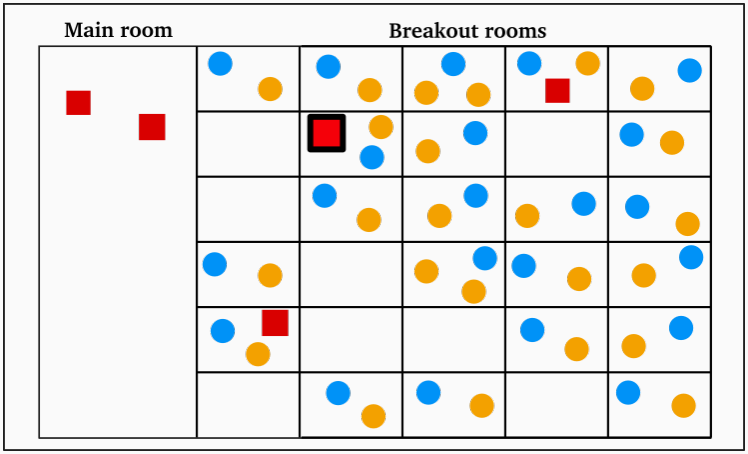
- **Small groups** (2 or 3): all students take an active role in solving the task, (have to) talk to each other.
- **Structured/scripted roles**: less time and cognitive effort spent on figuring out a way to collaborate efficiently/productively ("unscripted" collaboration can be particularly unnatural to navigate in a virtual "room").
- Tutors can't "keep an eye" on the whole room and identify who needs help: working in pairs gives everyone an **immediately accessible source of help**, minimises tutor time spent on "trivial" mistakes (syntax errors, typos...).
- Introducing standard **professional practice**.

Distributed/remote pair programming

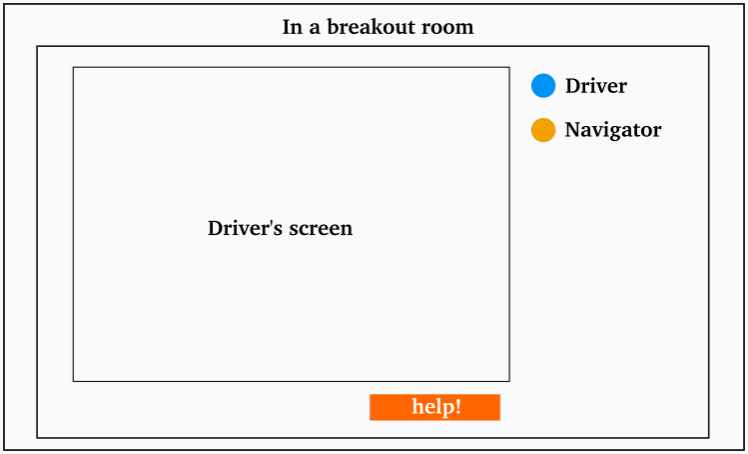
Session organisation



Session organisation



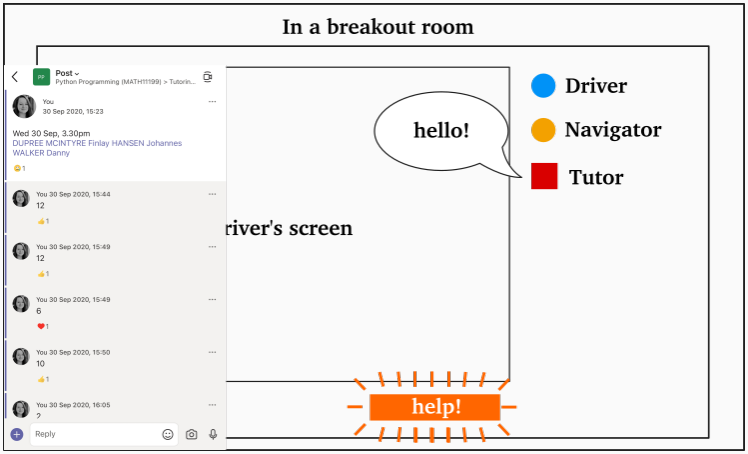
Inside a breakout room



Inside a breakout room

1. Driver **forks** a template GitHub repo containing the task; navigator joins the forked repo (easily set up with GitHub Classroom).
2. Driver **clones** the repo, shares their screen, and starts working on the task.
3. Navigator reads instructions, actively observes, talks to the driver, checks documentation...
4. They **switch roles** after $\sim$15 minutes. Old driver commits and pushes their changes, new driver pulls, shares their screen, and continues.

Getting help



Peer-assessed code review

Motivations for peer code review

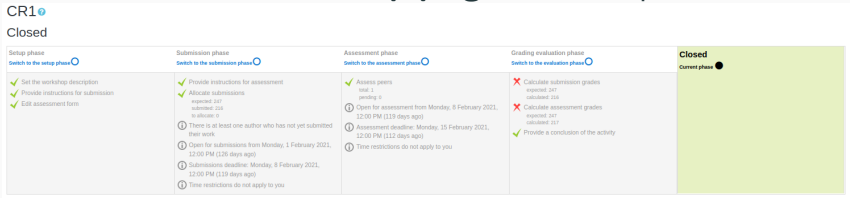
- **Peer learning** by seeing how others have approached the same task.
- Building **good habits**: students are asked to assess robust testing, commenting and style, code structure...
- Helping to build a sense of **community**: students don't have as much opportunity to work and learn together.
- Introducing **standard professional practice**.
- Better understanding of the assessment process.

Planning

- Code review tasks were alternating with quizzes as weekly homework (10% total grade).
- Set over 2 weeks: submit your solution by the end of the first week, peer-assess 3 submissions by the end of the second week.
- In code review weeks, the **synchronous lab** was a small group scaffolding + peer review activity:
  - Peer review only was not particularly successful. Students would be finished in quite a short time, and the submissions they had to actually assess were not those from their discussion group.
  - Starting the activity with a scaffolding task was helpful: e.g. a debugging task to practice understanding error traces and troubleshooting, a discussion task on useful commenting and code style. As seen with pair programming, **structure helps!**

Setup using Moodle Workshop

- I used the **Moodle Workshop plugin**, built for peer-assessed tasks.



- The grade is calculated as 50% peer-assessed, and 50% for *how close* your assessment was to the consensus between the 3 reviewers.
- The task description included the problem specification, instructions for submission format, and marking scheme to be used.

Setup using Moodle Workshop

Moderation:

- As instructor, I could override grades, add my own assessment, or change weights of different assessments.
- I checked a sample of random submissions/assessments each week.
- I set up a form which students could use to request further instructor moderation.

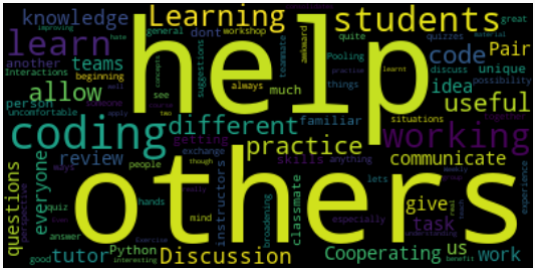
How did it go?

Cohorts

Pair programming and peer-assessed code reviews deployed in two courses:

- Semester 1: **Python Programming** (MSc, 280 students). Introductory Python course, focused on programming skills, overviewing some applications in applied maths and data science.
- Semester 2: **Computing and Numerics** (Y2, 250 students). Introductory Python course, but learning outcomes focused on computational mathematics and numerical methods.

Student feedback: the positives



"What do you consider to be the most useful aspects of the workshops?"

- Most of the responses point at cooperation, discussion, peer learning as positive aspects.
- Other positive comments: the workshop tasks are interesting, I get to apply the skills I've learned, I can get good help from tutors.

Student feedback: the positives

Some representative comments heard/read from students:

- "This is the most 'social' course I have" / "This workshop is my main social interaction for the week"
- "I didn't interact much with tutors since we often solved coding issues by ourselves. This is a good feature of the workshop."
- "[Workshops] allow you to have interaction with other students and help from tutors at the same time."
- "It was good to meet people and learn to pair program."
- "Peer (sic) programming is great. I'm not sure if this was a new addition to the course or not, but I hope it stays."
- "Working with others is also a real benefit as it allows you to learn from others, or teach others which further consolidates your own learning."

Student feedback: the negatives

- Most of the negative feedback on **pair programming labs** mentions the lack of time – 1 hour is too short.
- Some friction with technical issues, complicated workflow with git/GitHub/IDE for beginners, particularly at the start of the semester. This improved as the students got used to their tools.
- Some issues with **incompatible pairs**.
- Most of the negative feedback on **code reviews** was what I expected: students don't feel comfortable being assessed for credit by another student.

Student feedback: the negatives

Some ad-hoc comments heard/read from students:

- "I didn't like the format (work in pairs) so I decided not to go to many of them."
- "There are a lot of tasks for the short time of the workshops usually. Also pair programming online is sometimes just watching another person googling something, so I'm not really sure if Pair Programming is the way to go for the workshops."
- "I found [the workshops] useful when I found a good group of people to work with."
- "Too much work not enough time. I would have benefited from a two hour workshop instead."

Incompatible pairs

As a "non-specialist" cohort, levels of programming experience (and even general computer literacy) vary immensely between the students coming into the course. Dysfunctional pairs were typically:

- very experienced + complete beginner (a frustrating experience for both),
- two complete beginners (got stuck very easily).

Zoom update in Autumn 2020 allowed attendees to choose their own breakout rooms. Issues with mismatched pairs and lack of confidence from beginners were greatly reduced by:

- letting students choose their pair (so they could work with a friend),
- letting students work in triples instead of pairs, with a rotating driver and two navigators.

Lessons from lockdown: what's next?

Lessons from lockdown: what's next?

The fast move to hybrid/online teaching over the summer gave us the **momentum** to make changes and try new things.

Overall, pair programming was a success in fostering productive **cooperation** and **peer-learning** in an online setting, *for students who attended*. The main issues were around time and pairing compatibility.

What can we take with us as we progressively come back to campus?

Back to school

- "Tutorials" in our Maths courses are ran as collaborative **workshops**, students work together in small groups to solve exercises.

- How do we make collaborative computer workshops successful, online or in a classroom?