Assignment 6 Working Methods - Initial Reflection Report

Helena Rasche

2022-02-07

Given the previously sub-optimal Python lesson which relied on students individually completing assignments, a task with which they struggled and felt quite isolated, here we discuss the replacement of individual exercises with pair programming. This improved the lesson significantly, with the additional improvement of teaching an industry-relevant skill; pair programming is often used in the industry to decrease defects and improve the onboarding process.

Pair Programming

Pair programming is a software development methodology requiring two programmers to work together to solve problems and implement code.

> One, the driver, writes code while the other, the observer or navigator, reviews each line of code as it is typed in. The two programmers switch roles frequently [Williams]

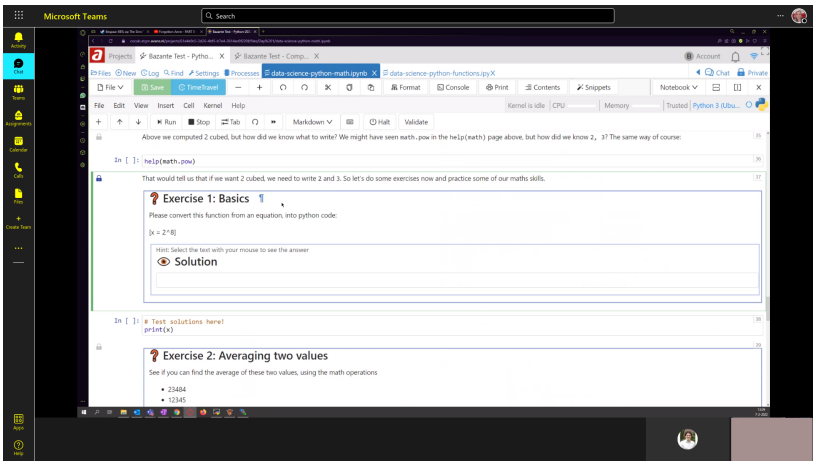

Figure 1: Screenshot of a teams call in which two students sit, one actively working on writing code, the other leading the discussion. One student's screen is showing CoCalc, a notebook platform where students can type code and immediately evaluate it's correctness. This platform allows us to write rich documentation, provide links to external resources, and write exercise problems which students can then easily work on individually or together.

Here we adapt this to the teaching environment via use of breakout rooms with two students, and screen sharing permitting the same situation of a "driver" and a "navigator". This should reduce cognitive load and frustration amongst students and allow them to rely on each other more for technical discussions, rather than seeing the teacher as the primary source of information [Williams and Upchurch, 2001].

Positives

This methodology should hopefully bring significant improvements for students as it has been empirically shown to be an effective learning

methodology for students [Mendes et al., 2005, 2006]. Specifically this technique has also been shown to be extremely beneficial for women in computer science and gives them better chances for success in future programming endeavours [Werner et al., 2004], potentially by changing the concept of programming from a solo activity to a collaborative activity as it often exists in practice. Given that in business and academia, programming often involves peer code review to ensure high quality results, this makes it an ideal fit for our educational practices and a very positive experimental change to make to classes.

Pitfalls

However there are some notable pair programming pitfalls in a teaching environment, such as the student with more coding experience doing all of the work by themselves [Cliburn, 2003, Williams et al., 2002, McDowell et al., 2003], or students failing to swap roles leading to a similar result. These issues can be mitigated through a number of methods including student self-reporting their contributions or lecturers taking a supervisory role [Williams et al., 2002]. Mentz et al. [2008] additionally reports success with the integration of the concepts of "Cooperative Learning" and provides guidelines for mitigating potential failure modes. These can be addressed easily via teacher intervention and were done so during the course, limited only by the availability of teacher and/or TOAs moving between groups.

Feedback

Student feedback was extremely positive, averaging 4.35 for all measures. The students overall liked the use of pair programming in their learning environment and found that it invited them to actively participate in the lesson.

Question	Min	Max	Mean	Std.Dev.	Median	Mode
The teaching method(s) used in the lesson fit the content of the lesson.	4	5	4.41	0.49	4	4
I liked the teaching method(s) used in the lesson.	3	5	4.33	0.62	4	4
The teacher's instruction was clear to me.	4	5	4.33	0.47	4	4
The used teaching methods invited me to active participation.	4	5	4.41	0.49	4	4
The teaching methods used fit my way of learning.	3	5	4.33	0.62	4	4

Question	Min	Max	Mean	Std.Dev.	Median	Mode
I liked the teacher's approach in this lesson better than in other lessons.	3	5	3.91	0.49	4	4
I thought it was a nice lesson.	4	5	4.75	0.43	5	5

They additionally provided comments on the above, filtered for comments[1] specifically concerning the aspect of pair programming:

- "The breakout rooms are very nice. It is a nice way to work with the students and get some interaction and get to work with the theory. For me the theory stick better."
- "It was a lot of information for one day but working on the excercises together helped a lot"
- "I think working in pairs in breakoutrooms really helped in solving some of the questions and getting some insight"
- "I think it was nice to work on it together in breakout rooms, [...]"
- "The alternation between explanations and making assignments was good."
- "Good mix between excersises and theory"

[1] The full response set is attached at the end.

Given the feedback it can be concluded that this was a successful intervention and practice that should be continued throughout the rest of the period. Additionally given that there are two sections of this course, one taught by a teacher not employing this method, it will provide an opportunity to further confirm or refute the success of the experiment.

Do and Don'ts

This method is potentially not for every teacher, some teachers may want more direct control over student learning but I've found it works well for me personally, and after reflecting on the past weeks of implementing this method. Additionally some Don'ts were unfortunately discovered through personal experience[2].

[2] Student feedback noted that "I found it difficult to do the assignments with you 1:1, because I don't understand the assignments immediately and sometimes can feel like I am to slow" which makes me thankful I collected the feedback anonymously, they might not have supplied it otherwise.

Do	Don't
Switch pairs often! Ideally every exercise someone else codes, someone drives.	But don't get hung up on this, screensharing can be slow and annoying to switch especially for short problems.
Pair students randomly, so they get experience working with different personalities and skill levels.	Don't pair with a student if there's an odd number / leftovers, a teacher's skill level is too different.
Allow them some privacy to work on the problems without feeling observed or pressured.	Don't micromanage if they're doing something wrong, guide them to the answer.

Do	Don't
'Walk' around to check in near the end, after they've made some progress.	Don't be aggressive in correcting small mistakes, let them self correct first before guiding.

Reflection

In this lesson a significant overhaul was made to the material to function better as an online lesson. Further an intervention of pair programming was applied via breakout rooms and pairing students off in duos. I decided in the first breakout room to give students some privacy and let them discuss together, while in the second round of breakout rooms I "walked around" and visited the various breakout rooms to see how students were getting on[3]. The breakout rooms went successfully and students shared their results at the end. This is a process I wish to automate in the future, sharing results back, in order to give students some anonymity while I discuss common issues I saw during their exercise. Here I could only comment on mistakes based on their final result (if I did not observe) or the process if I was present in the breakout room to see the specific error occur.

[3] This proved to be quite useful, I could see students making mistakes I would not have noticed otherwise, or if I'd only seen the final result.

Looking Back

Overall this went quite well, I saw students interacting well and switching off between roles but could not confirm the uniformity of switching each time. I think this was a very successful intervention based on my observations as a teacher and the reporting of the students.

Awareness

Based on the research included here, I think I need to further improve my observation of the students, either transparently via the programming environment we're using, or directly via my presence in the breakout rooms. One of these options scales better, but it might leave students without a good source of advice when they encounter difficulties in the exercises. Whether this builds skills to overcome small challenges, or leaves them struggling when they need assistance is yet to be seen.

Alternatives

Instead of pairing them in duos, I could try pairing them in threes but this might lead to one person sitting back and not participating, whereas the pair situation forces them to participate but doesn't risk them feeling isolated. Next time I should continue my experiments of whether or not to visit them in their individual breakout rooms during the exercises to examine what effects that has on students, and additionally survey them on their preferences to find a common solution that achieves both my goals of a more supportive learning environment

while avoiding disrupting them or having them feel a stressful scenario where they pressured to defend a topic they do not know well.

Trials

Given the results of Werner et al. [2004], women normally drop out of computer science based activities at a very high rate, a rate which is significantly decreased with pair-programming implemented. So, I'll be carefully watching the outcomes of my course compared to my colleague's to see how the women in my class feel compared to theirs, and if the difference is significant to help them implement pair programming in their class. However I need to be careful that I'm not blinded by the success of this methodology on a global scale, and make sure I am adapting to students who struggle with this model and prefer working on their own, perhaps to be determined via further student surveying.

References

Daniel C Cliburn. Experiences with pair programming at a small college. *Journal of Computing Sciences in Colleges*, 19(1):20–29, 2003.

Charlie McDowell, Brian Hanks, and Linda Werner. Experimenting with pair programming in the classroom. In *Proceedings of the 8th annual conference on Innovation and technology in computer science education*, pages 60–64, 2003.

Emilia Mendes, Lubna Basil Al-Fakhri, and Andrew Luxton-Reilly. Investigating pair-programming in a 2nd-year software development and design computer science course. In *Proceedings of the 10th annual SIGCSE conference on Innovation and technology in computer science education*, pages 296–300, 2005.

Emilia Mendes, Lubna Al-Fakhri, and Andrew Luxton-Reilly. A replicated experiment of pair-programming in a 2nd-year software development and design computer science course. *ACM SIGCSE Bulletin*, 38(3):108–112, 2006.

E. Mentz, J. L. van der Walt, and L. Goosen. The effect of incorporating cooperative learning principles in pair programming for student teachers. *Computer Science Education*, 18(4):247–260, dec 2008. DOI: 10.1080/08993400802461396. URL https://doi.org/10.1080/08993400802461396.

Linda L Werner, Brian Hanks, and Charlie McDowell. Pair-programming helps female computer science students. *Journal on Educational Resources in Computing (JERIC)*, 4(1):4–es, 2004.

L. Williams. Integrating pair programming into a software development process. In *Proceedings 14th Conference on Software Engineering Education and Training. 'In search of a software engineering profession' (Cat. No.PR01059)*. IEEE Comput. Soc. DOI:

10.1109/csee.2001.913816. URL https://doi.org/10.1109/csee.2001.913816.

Laurie Williams and Richard L. Upchurch. In support of student pair-programming. *ACM SIGCSE Bulletin*, 33(1):327–331, mar 2001. DOI: 10.1145/366413.364614. URL https://doi.org/10.1145/366413.364614.

Laurie Williams, Eric Wiebe, Kai Yang, Miriam Ferzli, and Carol Miller. In support of pair programming in the introductory computer science course. *Computer Science Education*, 12(3):197–212, sep 2002. DOI: 10.1076/csed.12.3.197.8618. URL https://doi.org/10.1076/csed.12.3.197.8618.

avans hogeschool

LESSON PREPARATION FORM

Lecturer: Helena Rasche

Date:

Group: ATGM/BML

Number of students: 10

Classroom:

Subject/lesson: Python Programming

Starting situation:
What do the students already know about the subject and what can they already do? How do they feel about it? Have they already gained work experience?
Describe the composition of the group. When and where does the lesson take place? And similar.

They generally know nothing about coding or programming in general. According to recent data, they probably do not even understand the concept of files and folders very well which is integral to software development and bioinformatics coding. They might understand something of automation (many students have phone apps that help them automate) but I think increasingly they are unaware of the entire field. If they do know about it it's probably an intimidating and/or scary subject for them as it essentially requires learning an "obscure" language and learning to write this. However they have had Galaxy courses which will have exposed them to the concept of workflows which can be a useful idea for them to transfer.

The composition of the group is Year 2 students who have experience with bioinformatics concepts but not with the command line or code yet. The lessons take place in computer rooms (or virtually on teams).

Objective/lesson objective:
Describe the objective(s) of the lesson according to the 3C model, taking account of the taxonomy level according to Bloom.

Write simple mathematics statements to learn what is and is not valid python and develop the ability to classify correct and incorrect statements
Execute intentionally incorrect code to study the debugging process and develop issue resolution skills.
Understand the structure of a "function" in order to be able to construct their own functions and predict which functions will not work.

Educational resources:
Which learning materials do you use during your lesson? (book, smartboard, whiteboard, paper, etc.)

- Lesson documentation will be online, there are documents which cover the entire lesson and further information in tip boxes if students wish to re-review the lessons' content and further their learning.
- During the lesson students will have a computer available
- An online system called CoCalc which lets them read a copy of the lesson materials
- AND directly try out and experiment with the lesson content we're teaching them.

Assessment procedure:
Provide a description of the final assessment of the unit of study and at which of Bloom's taxonomy levels the assessment will be carried out.

Formative assignments using the CoCalc system, we will distribute a homework assignment to them where they do the most basic task as it is the first week:
- Some skeleton of the output will be provided for them, asking them to complete a missing statement demonstrating their ability to predict how the code works and respond with the missing component in order to achieve the desired output (APPLY)

Every piece of code they will write will be evaluated with a number of sample inputs, so they can check their work (to an extent) and make sure it works ok. We will additionally have secret "teacher-only" tests which will ensure that even if the student decided to only handle the cases described, that they've properly considered all aspects of the problem and any potential exceptions they might encounter.

Deviations from Normal:
Indicate clearly in the form: a. which teaching method you use (how?) b. why you specifically choose this teaching method (accountability) c. what your role is during the lesson and what activities the students perform (how?) .

Here we will try Pair Programming in Duos. Pair-programming is often used in corporate programming environments in order to help new employees become more famliiar with the codebase. One person is designate as the "driver" write the code, the other as the "observer" reviews each line of code as it is typed and they discuss the code as they go. They frequently switch roles. This often results in fewer defects in the code as a result of two people discussing and interacting (Cockburn & Williams, 2000). The teacher's role is to introduce the activity, split them into pairs, and to facilitate their discussions and co-working.

avans hogeschool

Schedule (how long?)	Content (what?)	Teaching and learning activities/work forms (how?)		Justify: how will this be used to reach the objective?
		Teacher	Student	
Introduction/start: Around 20 min.	Here we will have a very short introduction to what the python language is, and how it looks and how code is structured. More importantly we'll discuss why automation is important for them.	Presentation on python Discussion of how python lets you automate things.	Introductions Mentimeter / Poll ("What sort of automation do you think of in the context of technology" / "what would you like to automate in your life")	The topic is quite large and students need an introductory period where they get to hear and see a bit of the topic before diving on in a hands in manner. Hopefully this step builds the foundation for furthering their knowledge during the rest of the lesson and provides the necessary base of Remember/Understand
~10 minutes	We'll introduce students to the CoCalc platform with a series of slides which introduce the basic mechanics of moving around the interface.	Walkthrough/demo	Students log in and access a notebook guiding them through the process	Same as above, separate skill but same requirement of basic understanding/remembering required for subsequent portions of the lesson
Math ~20 min.	- Math (* / + -) in python, math.sqrt, math.pow - Translate some known math functions (e.g. euclidean distance, root algorithm) into python	Live-coding (Discuss code and execute cells one-by-one)	Code-along for first portion, executing cells and listening. **This is followed by break out room where students work in DUOs** (pair programming) and are given a function to translate into python code	Here students gain practice running code and then begin learning to Apply a human description of a function to the transformations necessary for Python.
10 min	Q&A and discussion, check in with feelings	Discussion	Answering questions, posing their own, integrating knowledge.	Check in with student feelings to ensure their cognitive load is not overwhelming.
Math ~20 min.	- Strings (add / format) - Translate function into python (exercise)	Live-coding	Code-along for first portion, followed by break out room where students are given a function to translate into python code	Here students gain practice running code and then begin learning to Apply a human description of a function to the transformations necessary for Python.
~15 min			Break	
Functions! ~20 Min	- What is a function (conceptually) - What do they look like	Lecturing	Obtaining the knowledge	Same as above, but separate skill. The student takes the knowledge from the live-coding portion where they watched the teacher write some code (and did so along with the teacher)
Breakout & Discussion 20 min	- Fill in the missing part of a function (exercise) - Discussion of the results	Evaluating right and wrong answers, making sure everyone got it right.	Integrating it.	and then apply this in small exercises which are shared with the class.
Functions! ~20 Min	- Write a new function that does a sepcific computation, building on previous portion (exercise)	Live-coding	Students will code-along with the teacher before moving into breakout rooms for the second exercise.	Same as above.
Conclusion: ~15 Min	- Recap of Math + Strings - Recap of Functions - Q & A - "How did you Feel"	Presentation Questions Open Discussion	Students will answer pop quiz type questions during the recap presentations	Here we test the lower level Remember skills before the homework assignments will test their higher taxonomgy levels. Here we just want to make sure they internalised the contents of the lesson before they go home. Use MS Forms to have students construct true/false questions and everyone answers them, and discuss the responses.

Supplementary Data

ID	The teaching method (s) used in the lesson fit the content of the lesson.	I liked the teaching method (s) used in the lesson.
2	5	5
3	5	4
4	4	4
5	4	5
6	5	4
7	4	5
8	5	3
9	4	5
10	4	4
11	4	4
12	5	5
13	4	4

The teacher's instruction was clear to me.	The used teaching methods invited me to active participation.	The teaching methods used fit my way of learning.
4	4	4
4	5	4
5	5	5
4	5	4
4	4	5
5	5	5
4	4	4
5	5	5
4	4	4
4	4	4
4	4	5
5	4	3

I liked the teacher's approach in this lesson better than in other lessons.	I thought it was a nice lesson.	Remark/Suggestions on any of the above points
4	5	Helena does explain it clear and very well, albeit sometimes a bit too quick
4	5	
4	5	
4	5	
4	5	you're doing a great job. The breakout rooms are very nice. It is a nice way to work with the students and get some interaction and get to work with the theory. For me the theory stick better.
3	4	great lesson, it was a lot of information for one day but working on the excercises together helped a lot. bugs happen even to the best so no complains :)
4	5	I think working in pairs in breakoutrooms really helped in solving some of the questions and getting some insight
4	5	I think it was nice to work on it together in breakout rooms, sometimes it was bit to fast to follow every steps, but still manageable. Also I really like how Helena approaches us and reacts when we ask a question.
4	4	The explanation was clear. All the topics were clearly covered. At times, it was perhaps a little too fast, which made it difficult to follow. The alternation between explanations and making assignments was good.
4	4	I liked the lesson, it was all still quite new so it was a lot of information for today. It was nice that there was some interaction so I could follow along with you to understand the process more. The explanation was clear and easy to follow.
5	5	Nice lesson, with a nice explanation about the subjects. Good mix between excersises and theory lessons/CoCalc needs to be tweeked a little bit, to make it more userfriendly and stable
3	5	I enjoy your way of teaching. You seem patient, and you make difficult subjects (like coding) sound very understandable.