How Pairing by Code Similarity Influences Discussions in Peer Learning

Shiyu Xu
University of Michigan
Ann Arbor, Michigan, USA
shiyuxu@umich.edu

Ashley Zhang
University of Michigan
Ann Arbor, Michigan, USA
gezh@umich.edu

Steve Oney
University of Michigan
Ann Arbor, Michigan, USA
soney@umich.edu

ABSTRACT

Peer learning, as a form of collaborative learning, has been widely used in programming courses as a means of promoting active learning and enhancing students' programming skills. However, it is challenging for instructors to group students effectively so that they can have fruitful conversations. We conducted a study with 15 students from an introductory programming course to investigate whether and how grouping students with similar or different solutions affects the discussions that take place within groups. The findings indicate that pairing students by the similarity of their code might influence students' learning and coding skills. Specifically, students who were paired with people that had different solutions had, on average, more engaging conversations and were more likely to write more diverse solutions in the future. The results also highlight the need for tools to facilitate the pairing process in programming courses in order to optimize the learning outcomes for students.

KEYWORDS

peer learning, programming education

ACM Reference Format:
Shiyu Xu, Ashley Zhang, and Steve Oney. 2023. How Pairing by Code Similarity Influences Discussions in Peer Learning. In *Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems (CHI EA '23), April 23–28, 2023, Hamburg, Germany*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3544549.3585837

1 INTRODUCTION

The number of students who chose to take a computer science class or major in computer science has been dramatically increasing throughout the years, which has led to a growth in enrollment for introductory programming classes. However, programming skills can be difficult to learn, especially for novice programmers. Getting familiar with different kinds of IDEs and environments, writing syntactically correctly code, understanding the logic of code, and debugging code are all essential skills for a developer, but all these can be quite challenging for beginners. What makes it more challenging is that teachers and tutors sometime struggle to help individual students due to large enrollments. Therefore, instructors often use peer learning, a form of collaborative learning where students provide mutual support and feedback by working together or discussing code in groups. Studies have shown that peer programming can be beneficial for students by improving their ability to read and write code [17]. By discussing different methods of solving the problem and reading other's solutions, students can expose themselves to different thought processes and enhance their coding skills.

Despite these benefits, little prior research has focused on how to effectively group students in order to maximize students' learning. Große's study [5] demonstrated that grouping students with different solutions has been effective in the field of mathematics, but there has been limited research on the effectiveness of grouping individuals in peer programming in the field of computer science. This is the primary motivation for conducting our study. In this paper, we conducted a study to explore whether and how grouping people with similar or different solutions affects their subsequent discussions in peer learning. We used OverCode [4], a tool that clusters code segments by their similarity, to help us group participants. We grouped the participants into two main groups—one group with a similar solution to their partners, and one with a different solution to their partners. And by observing their discussion and their code, we explored how grouping students by the similarity of their code in peer programming affects their learning of programming skills. We found that on average, pairing students with different solutions improved the quality of their discussion, it also helped students for having a better understanding of solving coding questions in different methods. In our study, students described benefiting from viewing alternative approaches. The results also highlighted the need for tools to facilitate the pairing process in programming courses in order to optimize the learning outcomes for students.

2 RELATED WORK

2.1 Computer science education and Peer programming

Prior work has recognized the need to improve computer science education. Based on literature reviews, students are having difficulties in understanding the coding language, writing code using the correct syntax, and establishing strategic knowledge such as debugging the program [13]. For instance, Piwek et al. indicated that introductory Python students find it difficult to understand and effectively use different IDEs, as well as understanding Python-specific topics [10].

Due to large enrollments in introductory programming courses [15], there are limited interactions between the lecturer and the students while novice programmers need more support. Instructors take a

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
CHI EA '23, April 23–28, 2023, Hamburg, Germany
© 2023 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9422-2/23/04.
https://doi.org/10.1145/3544549.3585837

variety of approaches such as using a flipped classroom to encourage interactive activities and collaboration inside the classroom [1]. Peer learning is a widely used learning technique happening in labs or discussions for effective programming learning, and previous research has shown that peer learning, as a form of collaborative study, enhanced students' abilities to learn computer science [9]. Sitthiworachart et al. designed a peer coding assessment by dividing students into small groups where they review each other's code [16]. This study indicated that by reading and understanding other's codes, students improved their proficiency in understanding the code's syntax and problem-solving skills [16]. Our research further explores how group work and viewing other's solutions affect learning efficiency in introductory programmers.

2.2 Impact of viewing different solutions during collaborative learning

Study shows that pair programming built an environment that encourage students to identify alternative solutions for the problem with their partners [8]. In the pair programming tasks, the students are expected to design, code, test, and assist others to complete the task assigned, which has a more positive influence on learning [8]. Solving a problem with different approaches are also called "multiple solution methods". Multiple solution methods have been widely encouraged and discussed in teaching mathematics. Rittle's study shows that learners with little prior knowledge gain flexibility by comparing different approaches [14]. Große's study showed that approaches with similar representation but different strongly from a mathematical point of view are extremely helpful [5]. Nevertheless, when provided different types of solutions, students with individual learning styles might not be good at self-evaluating themselves, which could lead to a shallow understanding of the knowledge, or so-called "illusions of understanding" [5]. Zeller indicated that exposing students to different codes, especially proving students with a targeted solution that differs from their answer can encourage students' self-teaching and inspiration [18]. In our study of collaborative learning, we attempt to leverage the power of multiple solution methods but mitigate the 'illusion'.

We aim to delegate the responsibility of evaluation to each participant, when everyone elaborates their solution to others and exchanges solutions, learners' comprehension has been enhanced. The impact of viewing different solutions has been studied in other subjects, but rarely in computer science. We did the study to explore the benefits of viewing diverse solutions in collaborative learning in computer science.

2.3 Code clustering tools

One difficulty that introductory coding courses face is large enrollments. Sahami et al. indicated that enrollment for computer science courses has dramatically increased since 2007 and due to the size of the class, it is difficult for teachers to give individual feedback for in-class exercises and homework assignments [15].

Code clustering can help instructors make sense of large numbers of code samples. In code cluster, large amounts of code is grouped using clustering algorithms for similarity detection, so code that is more similar to each other will be grouped together. Instructors can then find larger meaningful patterns by analyzing these clusters [12]. Code clustering tools are essential to computer science education and have been commonly used to support grading [3], generate feedback for coding assignments [6], visualize massive assignment solutions [11], and detect code plagiarism [2]. Helminen et al. created Jype, a web-based visualization tool that used code clustering to provide immediate feedback and visualize errors in code through line-by-line execution [7]. Another example of a code clustering tool is OverCode where Glassman et al. designed OverCode for analyzing and clustering similar results in students' coding solutions [4]. OverCode puts similar solutions into piles with a visualization interface so teachers can provide feedback at scale for various solutions. However, there are also same limitations on the current code clustering tools [4]. For example, OverCode only targets solutions that pass the autograder, so the coding piles will exclude code with errors [4], while novice programmers create lots of syntax errors and logical errors [13]. Our study uses clustering tools to cluster in-class exercises so it can help us group students by the similarity of their solutions, helping us group the participants in order to explore the effectiveness of exposing and discussing different solutions in peer programming.

3 METHOD

3.1 Study setup

We conducted a one-hour study with novice learners from introductory python courses on campus after the study was approved by the IRB. Due to the ongoing pandemic and the prevalence of online programming courses and programs, we conducted the study virtually through Zoom. Audio and video were recorded during the section, and students used Google Colab for writing solutions during this study. The study involves four main steps with the research aim to see whether and how reading and understanding of multiple solutions in a collaborative learning environment benefits students' coding skill and their discussion.

Step 1:
First, the participants were asked to use Python programming language to pick the longest word that ends with 'e' from an array of words within a 10 minute time frame. Participants are free to use any resources such as Stack Overflow. The level of difficulty for the Python question was intentionally aligned with the level of their Introductory Python courses.

Step 2:
The participants were assigned to a partner according to their solution. The participant was either paired with a partner who has a solution similar to the participants' or different from the participants'. There were two focused groups. One group consisted of participants who have similar solutions to their partners, and the other group of participants who have different solutions compared to their partners. To facilitate this step and reduce bias, the study coordinators first used Overcode [4] to generate initial clusters and then assigned students to groups manually. The study had four different solution groups with two-person per group and three similar solution groups with one group of three people and two groups of two people. While the study coordinators worked on grouping, participants were asked to fill out a survey with questions about their background in programming skills and the approach they took in order to solve the coding problem.

Step 3:
After placing all participants into groups, the participants took turns explaining their solutions to their partners during a 15-minute discussion. While they are explaining their solutions, we asked the participants to share their solution screens and record the session so we can observe their interactions later.
Step 4:
The participants were then asked to solve a similar Python coding question that involves picking the smallest even number from a numbered array, and we asked our participants to solve the problem in as multiple ways as possible if applicable. We designed these two coding questions, in which students can solve the question in a similar approach as the question in step 1. The two Python questions both require students to extract a specific word or number from a given array and can be solved using a common concept such as a 'for loop'. Then they finished the study by filling out a second questionnaire. The second questionnaire was a post-study survey that collected information on how the participants felt about solving the two questions, the methods and approaches they tried, and the nature of their discussion. Additionally, the questionnaire included some general questions that asked participants to rate their partner's level of activity during the discussion and their perceptions of their partner's skill level. The study was designed in this way for us to compare if grouping students by the similarity of answers (similarity, in this case, is by human judgment) has effects on the efficiency of collaborated learning, and whether reading and understanding multiple solutions is beneficial for beginner programmers.

3.2 Recruitment Process

We reached out to instructors who teach introductory Python courses on campus. These instructors then sent the study information to their current students to guarantee that we are finding participants that fit our study design. All participants are 18 years of age or older, and they signed the consent form prior to the beginning of the study.

3.3 Recruited Participants' background

This case study included a total of 15 participants, consisting of 13 females and 2 males. Among the participants, 9 were between the ages of 18 and 24, 4 were between 25 and 30, and 2 did not disclose their age. were selected under the condition that they finished at least one introductory Python course, or are currently enrolled in an introductory Python course on campus. Eleven participants have less than 6 months of experience in Python coding and have taken at least one introductory Python course.

4 OBSERVATIONS

Peer Discussions:

When explaining their concept to their partners, the first thing students usually do is to go through the code line by line for explaining their process. As they are walking through their lines of code, they also mention what variable they created, what functions and methods they used or intended to use during their explanation. For example, most students explain the concept following a similar style to "I first built a dictionary with words ending with e. Then with a for loop to get the length of that item. And then I use the max function to find the max value."

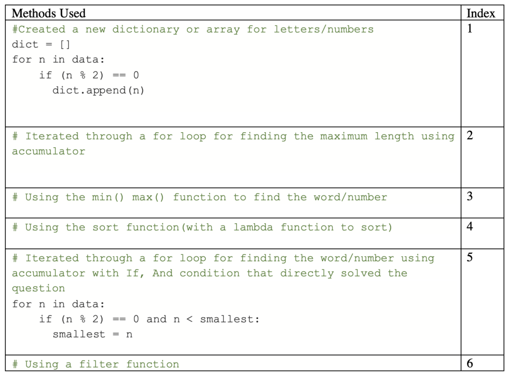
During their explanation, eleven participants assumed that their partner understood them while four of participants asked their partner whether they understood/explained the concept clearly.

After students finished explaining, some students will share their thought processes and how to improve the answer. Students will generally talk about how different strategies might be better in solving the question. They will also answer their partner's question if there is a question raised.

During the peer discussion, students also helped their partners in clarifying the coding question's instruction or misunderstanding a concept. We have three groups of students who had a partner that ignored the requirement of "finding words that end with e" and instead they only found the longest word. They either discovered that themselves or their peers pointed out their misunderstanding of the question.

Problem Solving Results:

For the first question, ten participants successfully solved the question without any syntax or logical errors, three participants solved the question partially by only finding the longest words rather than finding the longest words that end with 'e', and two participants failed to solve the question. For the second question, thirteen participants were able to solve the question and seven participants were able to solve the question in multiple ways. Figure 1 and Figure 2 are detailed results of how participants solved their questions.



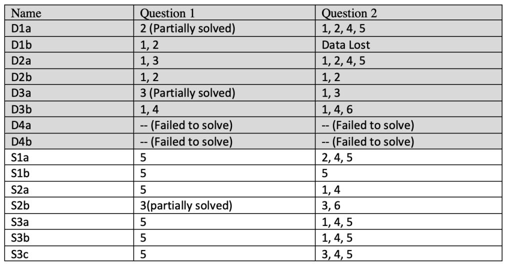
Methods Used	Index
`#Created a new dictionary or array for letters/numbers` `dict = []` `for n in data:` `    if (n % 2) == 0` `      dict.append(n)`	1
`# Iterated through a for loop for finding the maximum length using accumulator`	2
`# Using the min() max() function to find the word/number`	3
`# Using the sort function(with a lambda function to sort)`	4
`# Iterated through a for loop for finding the word/number using accumulator with If, And condition that directly solved the question` `for n in data:` `    if (n % 2) == 0 and n < smallest:` `      smallest = n`	5
`# Using a filter function`	6

Figure 1: The left column is summarized methods/descriptions of code that appeared in the participant's answer. In the right column are the indexes that were given to the methods/descriptions that will be used in Figure 2.

5 FINDINGS

5.1 Students' thinking process is not fully presented in their code. Pair programming helped them express their knowledge.

When we are reviewing the recorded video discussion and the questionnaires where participants were explaining their thought



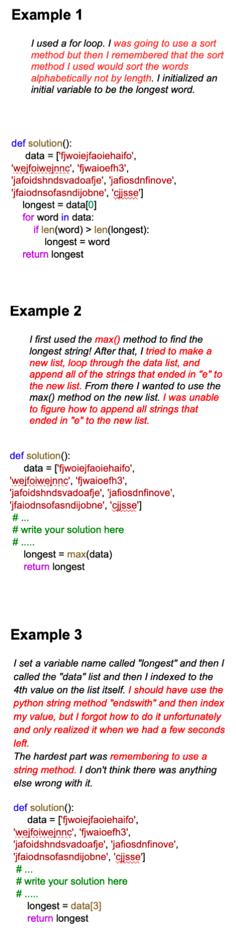
Name	Question 1	Question 2
D1a	2 (Partially solved)	1, 2, 4, 5
D1b	1, 2	Data Lost
D2a	1, 3	1, 2, 4, 5
D2b	1, 2	1, 2
D3a	3 (Partially solved)	1, 3
D3b	1, 4	1, 4, 6
D4a	-- (Failed to solve)	-- (Failed to solve)
D4b	-- (Failed to solve)	-- (Failed to solve)
S1a	5	2, 4, 5
S1b	5	5
S2a	5	1, 4
S2b	3(partially solved)	3, 6
S3a	5	1, 4, 5
S3b	5	1, 4, 5
S3c	5	3, 4, 5

Figure 2: The first column is the participants. The middle column represented question 1 and the right-most column represented question 2. The indexes in the middle and the right-most column represent what specific method the participants used in order to solve question 1 and question 2. The participant's name with grey background represented participant's from different solution groups and the white background represented participants from similar solution groups.

processes, we found that students have a deeper understanding of the problem than what was been reflected by their code. For example, during the discussion, the participant explained her thought process in solving the problem to her partner (Figure 3, Example 2), which demonstrated her understanding of for loop, append function, and max function. While on her answer sheet there was only one line of code using the max function. We think that without collaborative learning and discussion among peers or tutors, it's very hard for students to accurately express their knowledge.

5.2 Participants tend to provide more insights on how to solve the question after discussion with partners who have different solutions

Both groups has similar ratings on how active their discussion was. According to the questionnaire, the average rating on how actively the participant was involved in the discussion (rated by participants themselves) for different groups' active levels was 4.75 out of 5, and for the similar-solution groups, it was 4.571 out of 5 ($\sigma = 0.535$). And for how actively their partner was involved in the discussion was 4.75 out of 5 for the different-solution groups, and 4.571 out of 5 ($\sigma = 0.535$) for the similar-solution groups. These numbers indicated both groups' participants perceive their conversations as engaging. However, more details were provided by students who paired with different solution partners when answering the questionnaire question "What did you learn from your partner's solution?" and "What specific aspects of the discussion do you find useful". Participants who paired with different solution partners provided more details on the specific function and variables used such as the sort method, dictionaries, and lambda function. They thoroughly described how different their solutions are compared with their partners. In contrast, the participants in a similar solution group answered those questions vaguely without listing details on how to approach the question.



Example 1

I used a for loop. I was going to use a sort method but then I remembered that the sort method I used would sort the words alphabetically not by length. I initialized an initial variable to be the longest word.

```
def solution():
    data = ['fjwoiejfaoiehaifo',
'wejfoiwejnnc', 'fjwaioefh3',
'jafoidshndsvadoafje', 'jafiosdnfinove',
'jfaiodnsofasndijobne', 'cjjsse']
   longest = data[0]
   for word in data:
      if len(word) > len(longest):
         longest = word
   return longest
```

Example 2

I first used the max() method to find the longest string! After that, I tried to make a new list, loop through the data list, and append all of the strings that ended in "e" to the new list. From there I wanted to use the max() method on the new list. I was unable to figure how to append all strings that ended in "e" to the new list.

```
def solution():
    data = ['fjwoiejfaoiehaifo',
'wejfoiwejnnc', 'fjwaioefh3',
'jafoidshndsvadoafje', 'jafiosdnfinove',
'jfaiodnsofasndijobne', 'cjjsse']
 # ...
 # write your solution here
 # .....
    longest = max(data)
    return longest
```

Example 3

I set a variable name called "longest" and then I called the "data" list and then I indexed to the 4th value on the list itself. I should have use the python string method "endswith" and then index my value, but I forgot how to do it unfortunately and only realized it when we had a few seconds left.
The hardest part was remembering to use a string method. I don't think there was anything else wrong with it.

```
def solution():
    data = ['fjwoiejfaoiehaifo',
'wejfoiwejnnc', 'fjwaioefh3',
'jafoidshndsvadoafje', 'jafiosdnfinove',
'jfaiodnsofasndijobne', 'cjjsse']
 # ...
 # write your solution here
 # .....
    longest = data[3]
    return longest
```

Figure 3: Examples of the participant's thought process expressed during the discussion compared to their submitted codes.

5.3 Difficulties in learning code for novice programmers

Our participants listed some difficulties they encountered during their studying in python in their questionnaire including but not limited to, understanding the logical thinking of the codes, finding relevant resources over the internet, debugging, and testing their code, and finding tutors when they need help. This relates to what Qian et al found on difficulties that novice programmers are experiencing [13].

5.4 The different answer paired group have slightly longer discussions with their partner compared to the similar group

Discussions among groups with different answers tend to discuss more on how they can approach the question with different strategies, how they can solve the questions differently with better efficiency, and what functions they can use to achieve this, while similar groups' participants tend to only walk through their solutions with their partner with fewer conversations on how to approach the question. However, whether the participants have the same level of experience in programming and whether both participants read the questions correctly/solved the question are some factors that can influence this result.

5.5 Students express that pairing is very important for pair programming

During the discussion, one participant mentioned that "it is strange. it's nice to learn with someone else and look at different perspectives. but if you have a bad partner, it can mess up your whole experience". The importance of finding a partner with the same level of skills was also mentioned as an important factor of peer programming so that both partners have the opportunity to code rather than one partner do the majority of work. In the questionnaire, we asked the participants to rate their partner's programming ability as well as their own ability, and some students gave their partners high scores and themselves low scores vice versa. However, although there is a gap in between their coding skill scores, improvements in coding skills were seen in both participants. We cannot draw the conclusion that pairing two different coding skill level students is bad, but it can be something to take into consideration for future work when considering how to pair students in peer programming.

6 LIMITATIONS

This research, however, is subject to several limitations. The primary limitation to the generalization of these results is the sample size used in this study is rather small. It was hard to represent all beginner programmers with a sample size of 15 people. The second limitation is that we conducted this study virtually through Zoom in a lab setting instead of a classroom setting. Participants might be more comfortable in a classroom setting where peer programming typically happens. This may have an impact on the participant's discussion section thus influencing the study results. A further limitation of this study is that it only focused on the use of Python as the coding language. Future studies could be conducted with individuals who are enrolled in courses involving other programming languages to determine the generalizability of the findings.

7 FUTURE WORK

7.1 Need of better clustering tools

We previously planned to use OverCode for clustering similar answers for coding question one in order to pair the participants with their partners. However, when we run the coding results through OverCode, we found that OverCode put every participant's answer in different piles, generating 14 piles in total, so we had to group the participants by our judgment on similarity. Our study shows that OverCode cannot cluster when errors are presented in the code segment, and it's difficult to cluster code with the same logic. Further study should try to find or develop a clustering method that can categorize code segments by their logic and method use without excluding code with errors, this will improve the efficiency when a massive number of students needs to be paired.

7.2 Design of Study

We think that another round of study with more participants should be done in order to analyze how similarity of solution impact peer programming. For our study design, multiple factors influence the study result including participants' coding skills, the personality of the participants (which influence the quality of the discussion). With a larger sample size, it will be easier to obtain a clear result. A way for students to show their thought processes will also be essential to this study since some students' thought processes were not reflected in their codes. Also, a system that clearly defines standards of similar or non-similar codes will be helpful when grouping participants. Moreover, further investigation could be to examine how various techniques for warming up during pair programming can facilitate the initiation of fruitful discussions.

7.3 Quality of Discussion

Another important topic that rises during our study is how to define a "high quality" discussion. Right now we usually judge by whether two participants are actively engaged in the conversation, and how long their discussion lasted. Other factors like whether questions were raised and solved between the participants, the time frame that they spend on the actual coding questions versus other topics like prior experiences can also be considered when judging the discussion's quality. Standard metrics for evaluating the quality of discussions may be integrated, such as turn-taking patterns, level of conceptual understanding, and other indicators of effective peer learning, such as perceived utility, comprehensibility, and degree of reflection. With a clear system in rating discussion quality, it will be easier to compare study results and obtain non-biased conclusions.

7.4 Generalizability

The study was conducted with a limited number of participants, which may limit the generalizability of the findings to real-world programming courses. Future investigations could be conducted in actual classroom settings, where larger student groups can be observed, and the tool can be tested in a more realistic environment. Moreover, the current study solely focused on exercises related to selecting an element from a list in Python programming. It remains uncertain how the outcomes would apply to other programming languages and concepts. For example, students' discussions on fundamental syntax may differ greatly from those in object-oriented programming, where the definition of a distinct approach is also challenging. Conducting further research in real-world classrooms that employ alternative programming languages and exercises will allow for a more comprehensive understanding of the applicability of our findings to general programming courses.

8 CONCLUSION

In this paper, we presented a study around how pairing students by the similarity of their solution affects peer programming. This study was conducted with 15 participants that accomplished two coding problems with one discussion with a paired partner in between. From the results of the coding questions and observations from the discussion, we found that pairing by the similarity of code will influence students' learning and coding skills. Different solution groups had more engaging conversations and descriptive solutions. Other questions like better tools for code clustering, how to reflect students' thought processes in exercise or assignments were also raised. This study opens up various topics and studies that can be done in the future.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under DUE 1915515.

REFERENCES

[1] Saleh Alhazbi and Osama Halabi. 2018. Flipping Introductory Programming Class: Potentials, Challenges, and Research Gaps. In *Proceedings of the 10th International Conference on Education Technology and Computers* (Tokyo, Japan) *(ICETC '18)*. Association for Computing Machinery, New York, NY, USA, 27–32. https://doi.org/10.1145/3290511.3290552

[2] Sébastien Combéfis and Arnaud Schils. 2016. Automatic Programming Error Class Identification with Code Plagiarism-Based Clustering. In *Proceedings of the 2nd International Code Hunt Workshop on Educational Software Engineering* (Seattle, WA, USA) *(CHESE 2016)*. Association for Computing Machinery, New York, NY, USA, 1–6. https://doi.org/10.1145/2993270.2993271

[3] Lei Gao, Bo Wan, Cheng Fang, Yangyang Li, and Chen Chen. 2019. Automatic Clustering of Different Solutions to Programming Assignments in Computing Education. In *Proceedings of the ACM Conference on Global Computing Education* (Chengdu,Sichuan, China) *(CompEd '19)*. Association for Computing Machinery, New York, NY, USA, 164–170. https://doi.org/10.1145/3300115.3309515

[4] Elena L. Glassman, Jeremy Scott, Rishabh Singh, Philip J. Guo, and Robert C. Miller. 2015. OverCode: Visualizing Variation in Student Solutions to Programming Problems at Scale. *ACM Trans. Comput.-Hum. Interact.* 22, 2, Article 7 (mar 2015), 35 pages. https://doi.org/10.1145/2699751

[5] Cornelia S Große and Alexander Renkl. 2006. Effects of multiple solution methods in mathematics learning. *Learning and Instruction* 16, 2 (2006), 122–138.

[6] Sumit Gulwani, Ivan Radiček, and Florian Zuleger. 2018. Automated Clustering and Program Repair for Introductory Programming Assignments. In *Proceedings of the 39th ACM SIGPLAN Conference on Programming Language Design and Implementation* (Philadelphia, PA, USA) *(PLDI 2018)*. Association for Computing Machinery, New York, NY, USA, 465–480. https://doi.org/10.1145/3192366.3192387

[7] Juha Helminen and Lauri Malmi. 2010. Jype - a Program Visualization and Programming Exercise Tool for Python. In *Proceedings of the 5th International Symposium on Software Visualization* (Salt Lake City, Utah, USA) *(SOFTVIS '10)*. Association for Computing Machinery, New York, NY, USA, 153–162. https://doi.org/10.1145/1879211.1879234

[8] Tie Hui Hui and Irfan Naufal Umar. 2011. Pair Programming and LSs in Computing Education: Its Impact on Students' Performances. *Online Submission* (2011).

[9] Lei Liu and Cindy E. Hmelo-Silver. 2007. Computer-Supported Collaborative Learning and Conceptual Change. In *Proceedings of the 8th Iternational Conference on Computer Supported Collaborative Learning* (New Brunswick, New Jersey, USA) *(CSCL'07)*. International Society of the Learning Sciences, 454–463.

[10] Paul Piwek and Simon Savage. 2020. *Challenges with Learning to Program and Problem Solve: An Analysis of Student Online Discussions.* Association for Computing Machinery, New York, NY, USA, 494–499. https://doi.org/10.1145/3328778.3366838

[11] Kris Powers, Paul Gross, Steve Cooper, Myles McNally, Kenneth J. Goldman, Viera Proulx, and Martin Carlisle. 2006. Tools for Teaching Introductory Programming: What Works?. In *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education* (Houston, Texas, USA) *(SIGCSE '06)*. Association for Computing Machinery, New York, NY, USA, 560–561. https://doi.org/10.1145/1121341.1121514

[12] Real Python. 2021. K-means clustering in Python: A practical guide. https://realpython.com/k-means-clustering-python/

[13] Yizhou Qian and James Lehman. 2017. Students' Misconceptions and Other Difficulties in Introductory Programming: A Literature Review. *ACM Trans. Comput. Educ.* 18, 1, Article 1 (oct 2017), 24 pages. https://doi.org/10.1145/3077618

[14] Bethany Rittle-Johnson and Jon R Star. 2009. Compared with what? The effects of different comparisons on conceptual knowledge and procedural flexibility for equation solving. *Journal of Educational Psychology* 101, 3 (2009), 529.

[15] Mehran Sahami and Chris Piech. 2016. As CS Enrollments Grow, Are We Attracting Weaker Students?. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (Memphis, Tennessee, USA) *(SIGCSE '16)*. Association for Computing Machinery, New York, NY, USA, 54–59. https://doi.org/10.1145/2839509.2844621

[16] Jirarat Sitthiworachart and Mike Joy. 2004. Effective Peer Assessment for Learning Computer Programming. In *Proceedings of the 9th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education* (Leeds, United Kingdom) *(ITiCSE '04)*. Association for Computing Machinery, New York, NY, USA, 122–126. https://doi.org/10.1145/1007996.1008030

[17] Andreas Zeller. 2000. Making Students Read and Review Code. *SIGCSE Bull.* 32, 3 (jul 2000), 89–92. https://doi.org/10.1145/353519.343090

[18] Andreas Zeller. 2000. Making Students Read and Review Code. In *Proceedings of the 5th Annual SIGCSE/SIGCUE ITiCSEconference on Innovation and Technology in Computer Science Education* (Helsinki, Finland) *(ITiCSE '00)*. Association for Computing Machinery, New York, NY, USA, 89–92. https://doi.org/10.1145/343048.343090