



2020 SURVEY OF COMPUTER SCIENCE TEACHING

Pedagogical Practices
in the U.S.



Contributors:

Megan McHugh
Elise Deitrick, PhD
Joshua Ball

Suggested Citation:

McHugh, M., Deitrick, E., & Ball, J.
(2021). 2020 Survey of Computer Science
Teaching: Pedagogical Practices in the
U.S. Codio Inc. Retrieved from
**[www.codio.com/research/2020-
computer-science-teaching-survey](https://www.codio.com/research/2020-computer-science-teaching-survey)**

Forward

“As anyone associated with formal education knows, teaching is not just about access to a curriculum but also about having knowledge and understanding of what needs to be taught and how it should be taught—both content knowledge and pedagogical knowledge specific to that content. The phrase pedagogical content knowledge (PCK) is used to describe this crucial combination of content and teaching knowledge for teachers.”

– SHUCHI GROVER (P. XVII, 2020)

Contents

Executive Summary	2
Introduction	3
Survey Methods	3
Respondent Demographics/Context	4
Pedagogical Practices and Tools	6
Familiarity of Teaching Interventions	7
Implementation of Pedagogical Tools and Practices	9
Benefits and Challenges of Select Teaching Interventions	12
Perception of Evidence-based Pedagogy as Beneficial and Challenging	20
Conclusion	22
References	23

Executive Summary

What is really happening in Computer Science classrooms today?

Research surrounding various pedagogical practices is ample, but that does not necessarily mean educators adopt these evidence-based approaches. We surveyed over 100 computer science educators to find out why.

Our 2020 survey asked what practices computing educators were familiar with, which practices they have previously adopted and which they were considering adopting.

Specifically, we aimed to answer the following research questions:

- What evidence-based pedagogical approaches/technologies are most well-known by CS educators?
- Which evidence-based pedagogical approaches/technologies are most commonly implemented by CS educators?
- What types of benefits motivate CS educators to try new pedagogical approaches/technologies?
- What types of challenges de-motivate CS educators to try new pedagogical approaches/technologies?

By far, E-Books are the most well-known and widely used pedagogical intervention. According to our findings, educators believe digitizing classroom materials make resources more accessible to underrepresented students and promote a better understanding of course concepts.

Our findings suggest that Flipped Classroom, where teachers provide lectures out-of-class and hands-on work in class, could be widely adopted in the future. Of our respondents who have not implemented Flipped Classroom, 38% stated they are very likely to implement Flipped Classroom in the future, while the remaining 34% would like to but have some concerns.

A trend noticed throughout the survey results indicates that educators are likely to adopt a new pedagogical practice that they believe will directly improve student learning outcomes. Across the 20 pedagogical practices included in our survey, the most beneficial outcomes educators cited included:

- Improved student understanding of content
- Increased student engagement and/or interest
- Increased student participation in class

In a similar vein, we found that the biggest fear, or perceived challenge, in introducing a new pedagogical approach is that students will not like it.

This suggests that educators are less concerned with their own benefits (e.g., time savings) than student benefits (e.g., increased understanding) when choosing whether to adopt new practices.

We hope these findings will help researchers more effectively market and disseminate their evidence-based pedagogy and help administrators and educators identify promising practices to bring into their computing classrooms.

Introduction

Previous research, most notably Hovey, Barker, and Nagy (2019) and Barker, Hovey, and Gruning (2015) has looked at why Computer Science Educators adopt teaching practices. Their research, a combination of interviews and surveys, asked about more general teaching innovation. However, they noted that “there is a need to increase the use of evidence-based teaching practices” (Hovey et al, 2019).

“Despite being researchers themselves, the CS faculty we spoke to for the most part did not believe that results from educational studies were credible reasons to try out teaching practices”

(BARKER ET AL, 2015)

We build upon previous research by focusing specifically on evidence-based pedagogical approaches and technologies specific to computer science education. We examine motivations and challenges facing instructors pertaining to computer science education innovations rather than general education practices.

Our analysis maps each dimension—familiarity, implementation, benefits, and challenges—to individual pedagogical approaches/technologies to understand the uniqueness of the different interventions. We hypothesized that there are significant differences by innovation based on innovation-specific factors such as publication date, where it has been discussed, and ease of implementation that are not captured by the more generalized survey conducted by Hovey et al (2019).

Survey Methods

We emailed 5,548 computer science lecturers and professors who have taught in the US from a proprietary database a URL to the survey hosted on SurveyMonkey.

In the survey, we collected basic information about each participants’ professional context after asking the same series of multiple-choice questions along the dimensions of familiarity, implementation, benefits, and challenges for ten different well-documented, evidence-based, computer science-specific pedagogical approaches and technologies. We asked about the familiarity and implementation of an additional eleven evidence-based, computer science-specific pedagogical approaches at the end. Due to this survey design, different analyses below include different sets of practices and tools.

The survey took participants an average of 16 minutes and 7 seconds to complete, as measured by the SurveyMonkey software.

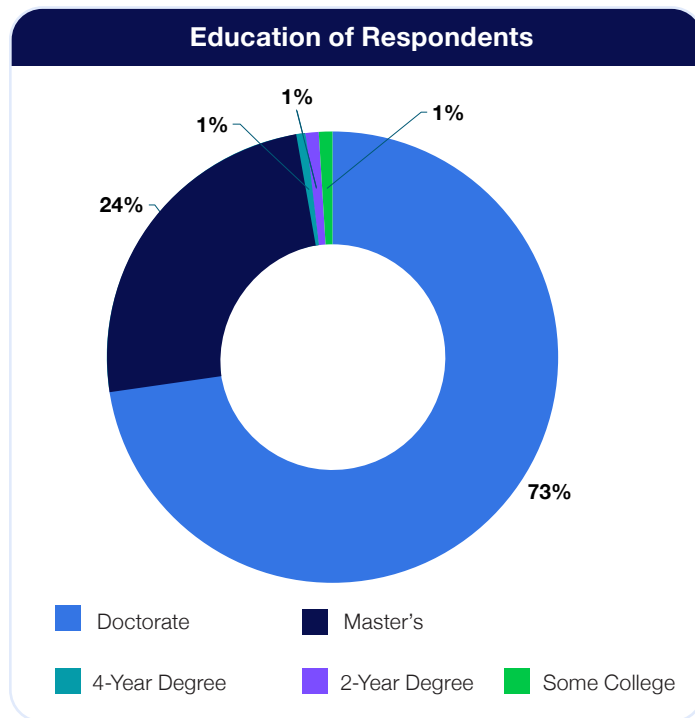
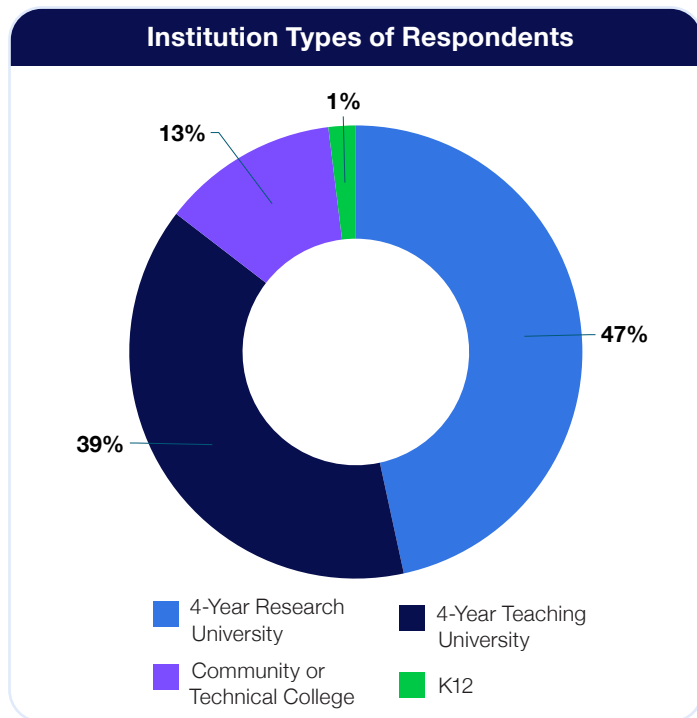
Out of the 273 CS and STEM instructors (professors and lecturers) who participated in the survey, 105 completed it. Respondents were not required to complete any of the questions, resulting in partial data for 168 participants and a full data set for 105 participants. In the following analyses, only respondents who completed the survey were included.

This way of recruiting participants has clear limitations. Given the total number of CS educators in the US, the number of respondents is small. This is compounded by recruiting participants from a database built in a proprietary, non-randomized, or representative way. It should also be noted that this data was collected during the beginning of the COVID-19 pandemic when many educators were actively changing from in-person to online instructional methods.

Respondent Demographics

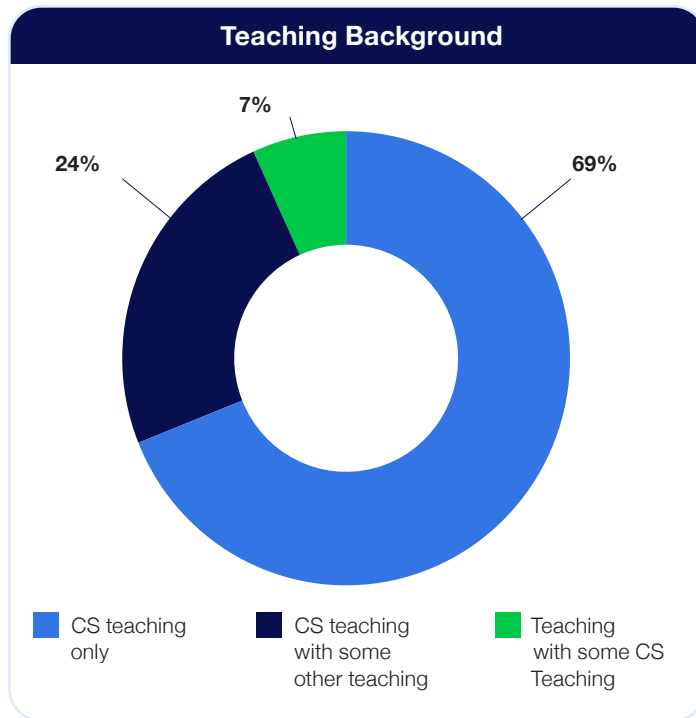
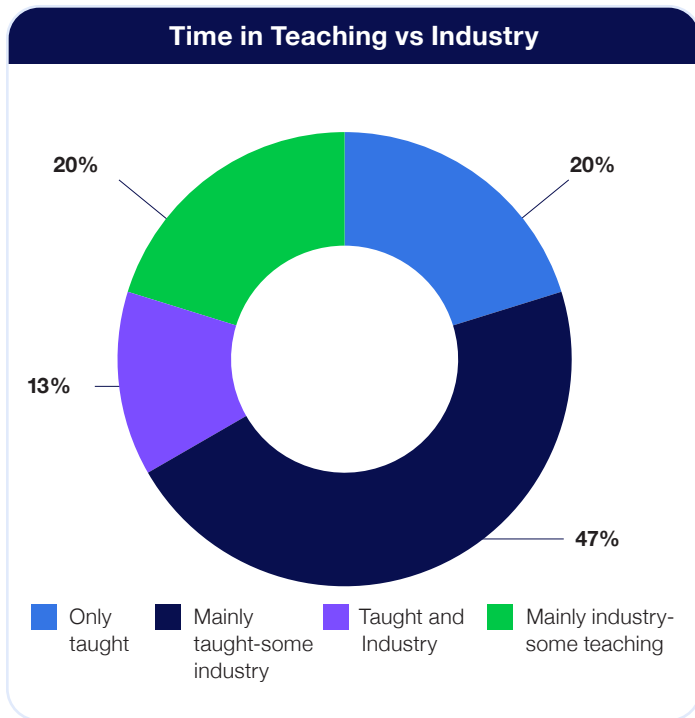
Looking at the type of institutions respondents were affiliated with at the time of the survey, 46% were at a 4-year research university, 39% at a 4-year teaching university, 12% at a community or technical college, and 2% at a K12 school.

In terms of education level, most respondents (72%) reported having a Doctorate, while 24% reported having a Master's degree. Under 3% of respondents reported having some college, a 2-year degree, or a 4-year degree.



Looking at education level by institution type, community/technical college educators were more than twice as likely to have a master's degree than a doctorate. At the same time, those at 4-year teaching or research universities were nearly five times more likely to have a doctorate than a master's degree.

	Doctorate	Master's	4-Year Degree	2-Year Degree	Some College
4-Year Research University	38	8	1	0	1
4-Year Teaching University	33	7	0	0	0
Community or Technical College	4	9	0	0	0
K12	0	1	0	1	0



Industry vs. CS Teaching

Many respondents of the survey have experience in industry as well as education. The results indicate that while 20 participants have only worked in education, the other 80 respondents have at least one year of work experience in the field. However, a majority of respondents report having more years of experience in education.

Years Teaching vs. Years Teaching CS

We found a positive correlation between years spent teaching overall and years spent teaching computer science. For upwards of 30 educators, their entire teaching career has involved computer science. This number drastically increases when you include individuals who may have spent only a year or two teaching a subject other than computing.

Pedagogical Practices and Tools

The pedagogical practices and tools we surveyed were taken from The Cambridge Handbook of Computing Education Research (Fincher & Robins, 2019). The pedagogical practices were predominantly taken from Chapter 15, Pedagogic Approaches (Falkner & Sheard, 2019), while the tools were taken from Chapter 21, Tools and Environments (Malmi, Utting, and Ko, 2019).

Activity Creation: Students create learning activities for their peers, such as a physical sorting game or instructional materials paired with an assessment

Assessment and Feedback Tools: Tools that analyze student code submission for correctness, style, syntactic structures, design, and test cases

Code Visualizers and/or Simulators: Visualizers can represent the running of actual code to reveal the notional machine or more abstract algorithms manipulating data structures through animations. Simulators ask the user to execute the steps while the system evaluates the correctness of their inputted step.

Content Creation: Students learn by creating instructional materials

E-Books: Provide instruction like a traditional textbook with technology enhancements like instant feedback on assessments, animations, and program visualizations

Games for Learning Programming: Games built to teach aspects of programming such as Rocky's Boots (logic gates), Silicon Zeroes (hardware), and Gidget (debugging). Others have a simplified language players use to manipulate the world, such as Lightbot and CodeCombat

Flipped Classroom: Classrooms structured so students cover course material (usually reading or video watching) before class and engage in related hands-on activities during class

Jigsaw: Students are put into teams where each takes on a role. The class re-arranges so that all students with the same role form groups where they discuss their assigned topic or task. The students then return to their original teams to disseminate what they learned.

Live Coding: The instructor writes code during class as part of a lecture while thinking aloud so students can hear the thought processes happening behind the programming process

Pair Programming: The practice where two students work together on the same programming task on a shared computer. One student has the role of driver and has control over the mouse and keyboard, while the other student acts as navigator and directs the driver on how to write the program. These roles are switched frequently throughout the exercise.

Parsons Problems: Problems that consist of mixed-up lines of code that students reorder into a functional code segment

Peer Assessment: When a student analyzes another student's code. The reviewer provides feedback and a grade based on their analysis, and the process is often scaffolded by prompts or rubrics designed by the instructor.

Peer Instruction: Starts with a challenging conceptual question to which students individually respond. Students then discuss the problem in small groups, then respond again.

Peer Review: When a student analyzes another student's code. The reviewer provides feedback based on their analysis, and the process is often scaffolded by prompts designed by the instructor.

POGIL: Process-oriented guided-inquiry learning is where teams of students work together through inquiry-based activities which help them uncover concepts themselves

Studio-based Learning: Students work in a shared environment where technology aids collaboration, discussion, and sharing of experiences

Tech-Assisted Collaborative Note Taking: During a lecture, students collaboratively take notes in a real-time note-taking application (e.g., google docs).

Test-First Development: A developmental approach where it is encouraged that students write tests before writing code (but not as strict as Test-Driven Development)

Test-Driven Development: A developmental approach where students write automated unit tests that fail before writing the corresponding code to pass the test. The result is many rapid iterations.

Think-Pair-Share: A form of scaffolded discussion where the teacher presents a question, students pause and think about it individually, pair up with a peer to discuss, and the class comes back together for individuals or pairs to share their thoughts or discussions

Tools that Support the Writing of Programs:

Technology that supports the writing of programs including educational programming languages, education-focused IDEs, version control systems, and static code analysis

Familiarity of Teaching Interventions

Familiarity was assessed with a Likert scale ranging from never heard about it to read about it or heard of it but could use a refresher (which we categorized as “unfamiliar”) to could describe it to a peer to Published about it (which we categorized as “familiar”).

Upwards of 75% of respondents are familiar with these four practices/tools:

1. E-Books
2. Pair Programming
3. Peer Review
4. Flipped Classroom

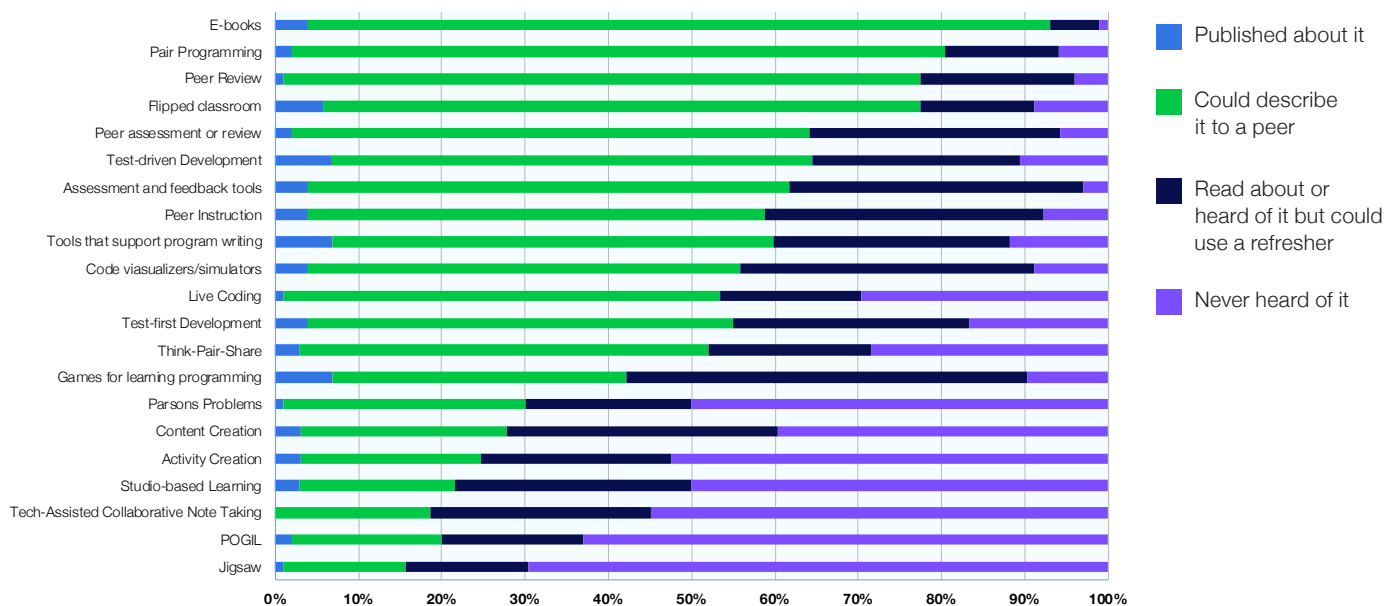
A couple of practices and tools we found surprisingly low on this list include Code Visualizers and Simulators (56% familiar) and Live Coding (54% familiar).

Given the popularity of PythonTutor, which had over 200k users and was used by “instructors in a dozen universities such as UC Berkeley, MIT, the University of Washington, and the University of Waterloo” as of 2013 (Guo, 2013), and which has grown to claim “[o]ver ten million people” on their main page, we would have expected to see higher familiarity.

Similarly, with Mark Guzdial’s SIGCSE 2019 keynote (Guzdial, 2019) seen by hundreds of CS educators demonstrating live coding, we expected higher familiarity. In the same keynote, Parsons problems are listed on slide 64—but they have even less familiarity (30%) than live coding.

1. E-Books
2. Pair Programming
3. Peer Review
4. Flipped Classroom
5. Peer Assessment
6. Test-Driven Development
7. Assessment and Feedback Tools
8. Peer Instruction
9. Tools that Support Program Writing
10. Code Visualizers and Simulators
11. Live Coding
12. Test-First Development
13. Think-Pair-Share
14. Games for Learning Programming
15. Parsons Problems
16. Content Creation
17. Activity Creation
18. Studio-Based Learning
19. Tech-Assisted Collaborative Note Taking
20. POGIL
21. Jigsaw

Familiarity with Teaching Interventions



We also considered what pedagogical practices and tools the respondents were not just familiar with but were experts in. The pedagogical practices and tools most frequently published on by our respondents include:

1. Test-driven Development
2. Tools that Support Program Writing
3. Games for Learning Programming

Each intervention has been researched by seven of the CS educators surveyed.

Interestingly, despite games for learning programming being published about by our respondents at a higher rate than 17 of the other practices and tools, less than 50% of respondents could describe it to their peers.

1. **Test-Driven Development (7)**
2. **Tools that Support Program Writing (7)**
3. **Games for Learning Programming (7)**
4. **Flipped Classroom (6)**
5. **E-Books (4)**
6. **Assessment and Feedback Tools (4)**
7. **Peer Instruction (4)**
8. **Code Visualizers and Simulators (4)**
9. **Test-First Development (4)**
10. **Think-Pair-Share (3)**
11. **Content Creation (3)**
12. **Activity Creation (3)**
13. **Studio-based Learning (3)**
14. **Pair Programming (2)**
15. **Peer Assessment or Review (2)**
16. **POGIL (2)**
17. **Peer Review (1)**
18. **Live Coding (1)**
19. **Parsons Problems (1)**
20. **Jigsaw (1)**

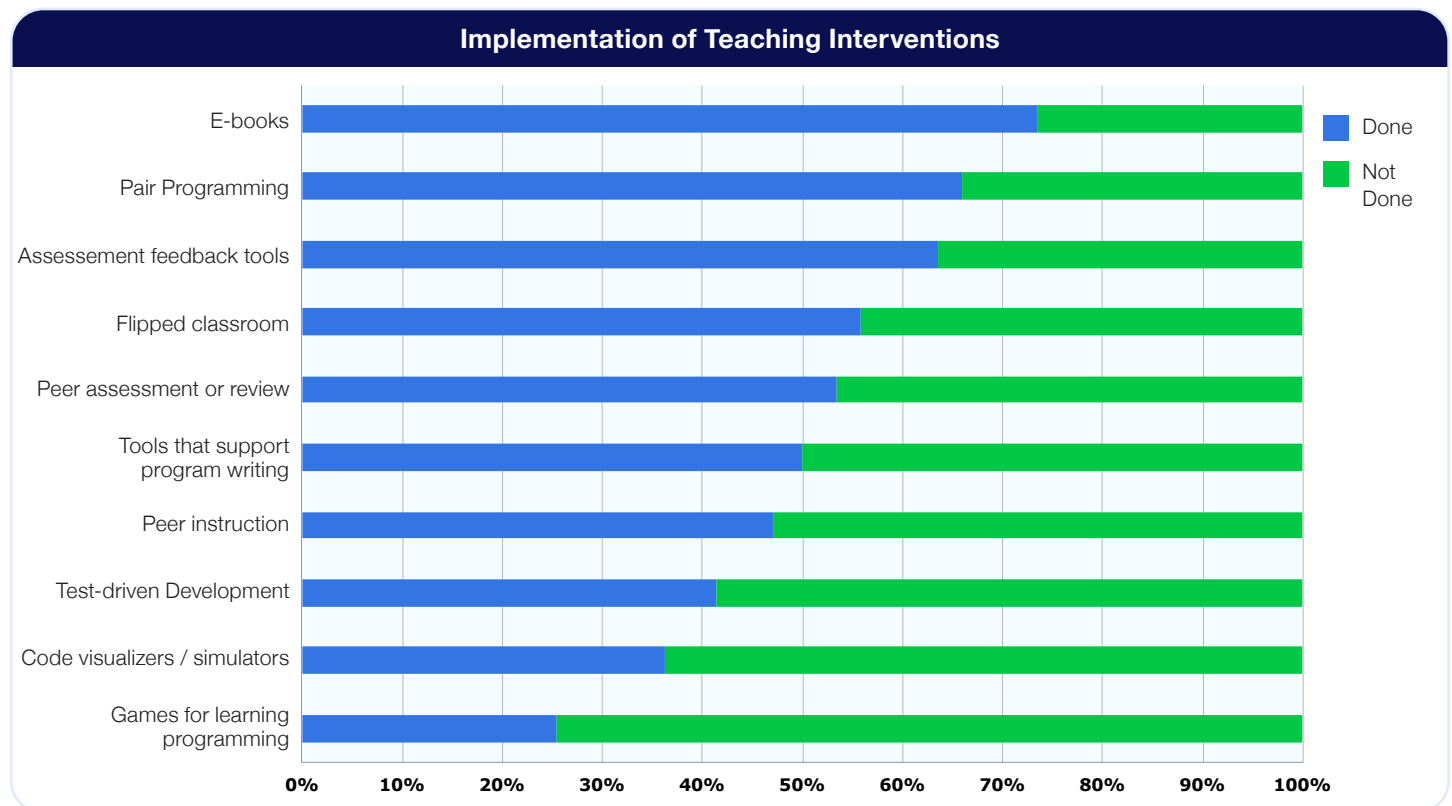
Implementation of Pedagogical Tools and Practices

Many of the most implemented pedagogical practices align with the familiarity rates of these practices. Each of the four practices well known by 75% of the respondents is also used by upwards of 50% of respondents:

- E-Books (1st in familiarity, 1st in implementation)
- Pair Programming (2nd in familiarity, 2nd in implementation)
- Peer Review (3rd in familiarity, 5th in implementation)
- Flipped Classroom (4th in familiarity, 4th in implementation)

The most widely known and widely implemented pedagogical tool is E-Books. Upwards of 70% of respondents use this tool in their classroom.

One teaching intervention, Assessment and Feedback Tools ranked 7th in familiarity yet is the 3rd most commonly used by educators. Only 61% of 102 respondents reported being familiar enough with Assessment and Feedback Tools well enough that they could describe them to a peer. However, 63% of 102 respondents asked if they currently use these tools indicated that they do. The familiarity to implementation rates of Assessments and Feedback Tools are more closely aligned than any other pedagogical intervention. This could be due to institutional adoption of Learning Management Systems (LMSs) and similar tools, which frequently offer assessment and feedback tools as features.



Which pedagogical practices are the most up and coming in CS education?

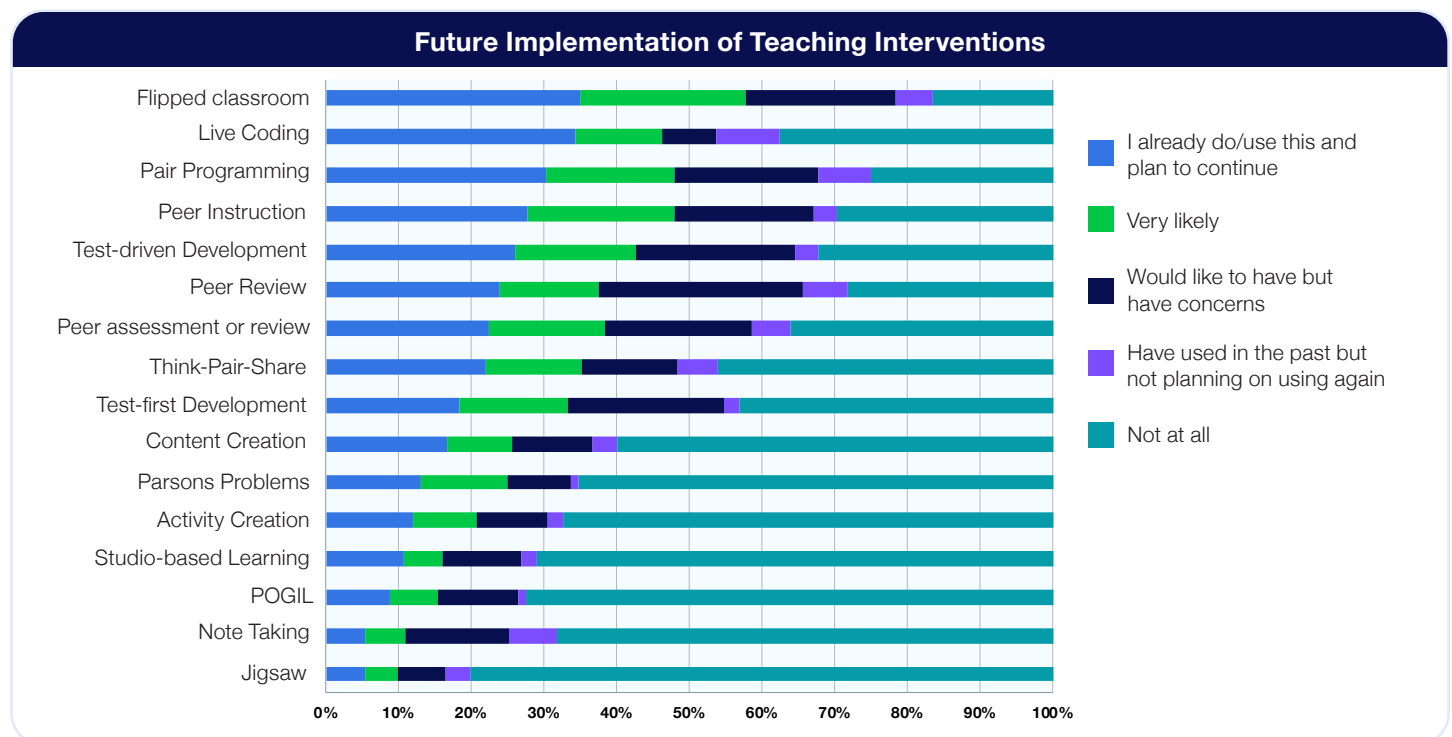
In the top spot, Flipped Classroom. Not only is Flipped Classroom the 4th most well-known and implemented practice, but it's also the method that most educators cite they are "very likely" to use in the future. Additionally, the skepticism surrounding Flipped Classroom is by far the lowest among the surveyed pedagogical interventions. Of all the respondents who have never tried Flipped Classroom, only 28% have no intention of doing so in the future.

What other practices are educators eager to try that they have not already?

- Pair Programming
- Peer Instruction
- Test-Driven Development
- Peer Assessment

Three of these interventions, Pair Programming, Peer Instruction, and Peer Assessment, allow students to simultaneously build their technical and soft skills. This list validates the idea that developing social skills among computing learners is becoming a cornerstone of the classroom.

Previous research has shown that social skills such as written and oral communication, project management, and teamwork are commonly lacking in graduating CS students but expected by employers (Radermacher & Walia, 2013). Similarly, Radermacher and Walia found testing on their frequently identified deficiency list, so these up-and-coming pedagogies directly address industry feedback. Their potential upswing may be a reaction to industry requirements as opposed to the evidence of their effectiveness produced by researchers.



What practices have educators implemented but are not proving to be sustainable?

When we asked educators about their plans to use various practices and tools in the future, one of the categories was “Have used in the past but not planning on using again.” This implies that there is some reason (e.g., challenges implementing, feedback from students) that made their experimentation with the practice or tool flop.

At the top of the list are:

- Live Coding
- Pair Programming
- Peer Review
- Tech-Assisted Collaborative Note

Considering this list in the context of the COVID-19 pandemic, which these results were collected during, we have to consider that some educators struggled to translate these practices to remote learning. Notably, Live Coding, Pair Programming, and Tech-Assisted Collaborative Note-Taking are generally synchronous activities.

- **Live Coding (8)**
- **Pair Programming (7)**
- **Peer Review (6)**
- **Tech-Assisted Collaborative Note (6)**
- **Flipped Classroom (5)**
- **Peer Assessment (5)**
- **Think-Pair-Share (5)**
- **Peer Instruction (3)**
- **Test-Driven Development (3)**
- **Content Creation (3)**
- **Jigsaw (3)**
- **Test-First Development (2)**
- **Activity Creation (2)**
- **Studio-based Learning (2)**
- **Parsons Problems (1)**
- **POGIL (1)**

Benefits and Challenges of Select Teaching Interventions

Implementing a new pedagogical practice in the classroom takes an investment of time, research, departmental support, money for resources, and possibly professional development. Although educators or students may benefit from particular practices, instructors have an acute awareness of the challenges that come with each teaching method. This section explores the benefits and challenges of a few interventions on a practice-by-practice basis and what may indicate future use.

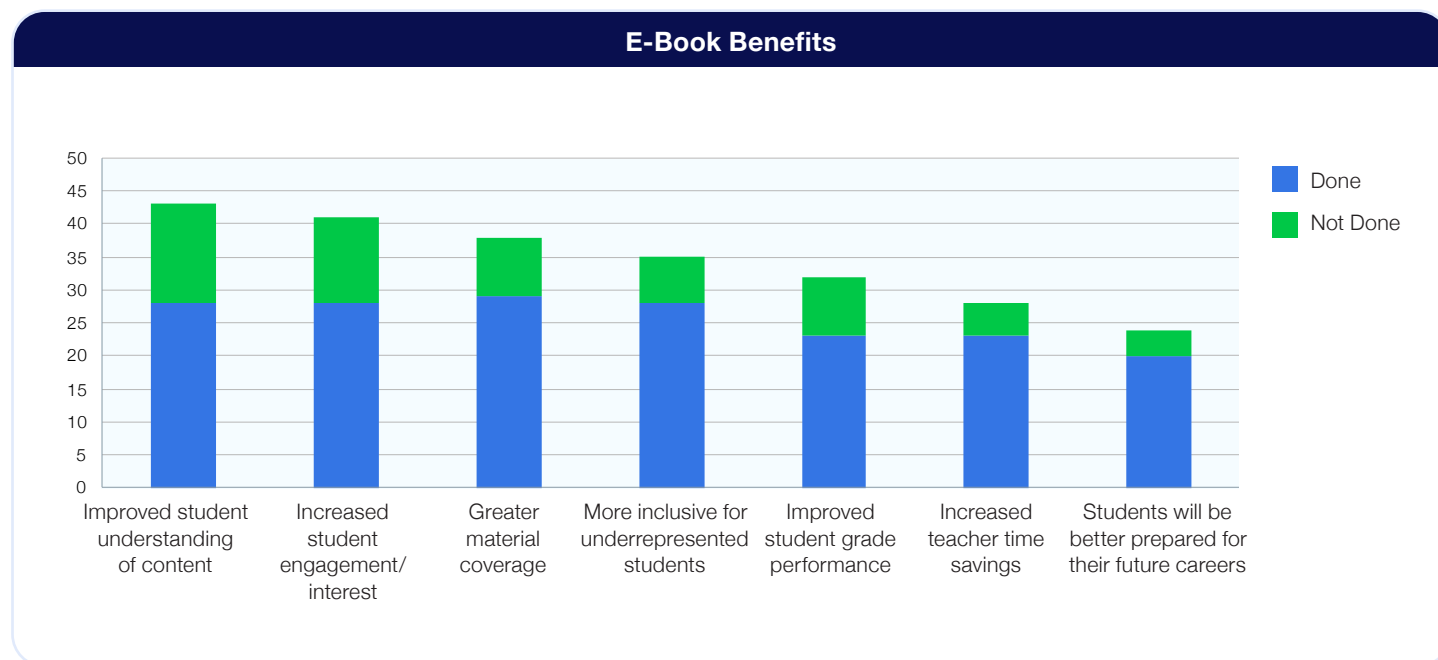
Most Implemented: E-Books

Educators use E-Books to offer course textbooks in a digital format, often accessible from desktop, tablets, or mobile devices. Roughly 93% of educators are familiar with E-Books, and upwards of 70% of survey respondents have implemented E-Books

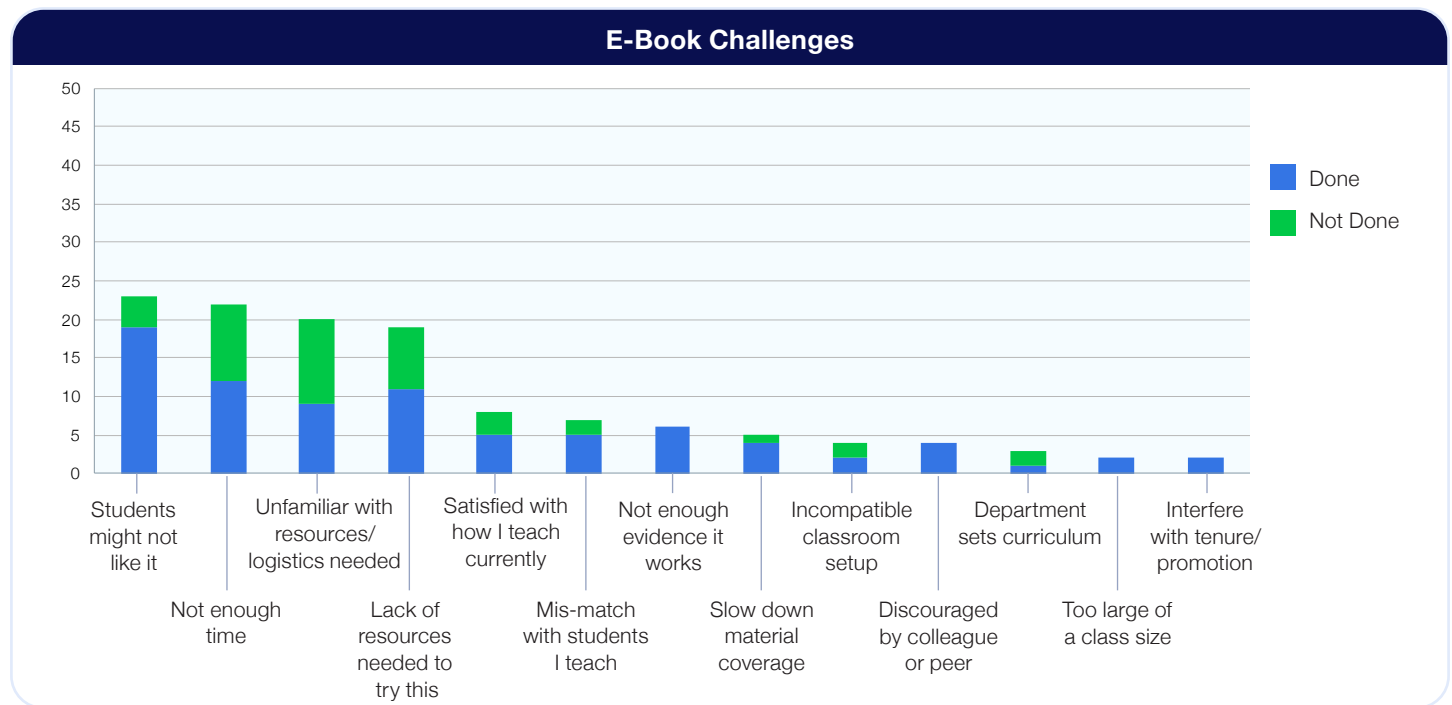
in their courses. This makes E-Books the most widely used pedagogical intervention among our survey respondents.

Previous research found that E-Books allow students to read faster than traditional print textbooks and that student performance on assessments increases (Patel & Morreale, 2014).

The results of this survey suggest that a reason E-Books are a popular intervention is educators' perception of the learning benefits students experience. Educators who both use and do not use E-Books likely believe that not only are these materials more accessible to underrepresented students, but they also help improve student learning outcomes and increase student grade performance.



Overall, educators seem less concerned about the challenges that come with implementing E-Books when compared with other pedagogical interventions.



Interestingly, the order of perceived challenges varies slightly between those who have adopted E-Books and those who've not.

Here is a brief breakdown of order based on history of implementation:

Top Concerns of Educators Who Have Adopted E-Books	Top Concerns of Educators Who Have Not Adopted E-Books
<ul style="list-style-type: none"> • Students may not respond well to or even like E-Books • Not enough time • Lack of access to resources needed to try this • Unfamiliar with resources/logistics needed 	<ul style="list-style-type: none"> • As an educator, they are unfamiliar with E-Book resources and the logistics needed to implement the materials • Not enough time • Lack of access to resources needed to try this

Pair Programming

48% of educators use or plan to use pair programming as a pedagogical practice. Pair programming involves two students working together on a single computer to work through programming together. Ideally, each student will get the opportunity to control the workstation with the guidance of their partner.

Previous research finds that students who typically score low in CS1 & CS2 courses perform better with the intervention of pair programming (Radermacher & Walia, 2011). The results of our survey indicate that educators find additional benefits associated with Pair Programming. Educators believe that it allows students to be more engaged in their work and gain more substantial interest in programming. This echoes what we see in existing research, which “strongly showed that pair programming in the classroom, despite some shortcomings, results in significant improvements in student performance, as well as improving recruitment into, and retention in, computer-science related majors” (Jacobson & Schaefer, 2008, p.93).

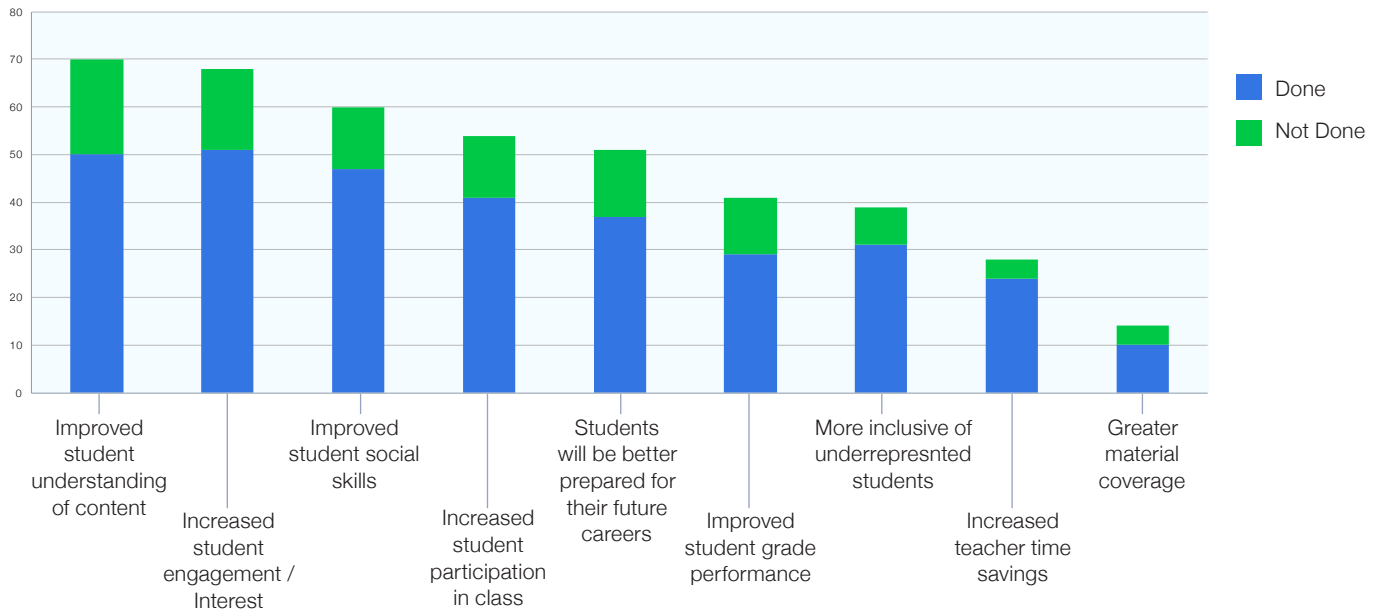
They also believe it has an additional benefit of allowing students to be social while learning and developing their

soft, collaborative skills. This translates well into educator’s belief that Pair Programming prepares students for a future career in the computing field. Often, lack of soft skills is cited to be a problem in the professional skills gap (Radermacher & Walia, 2013).

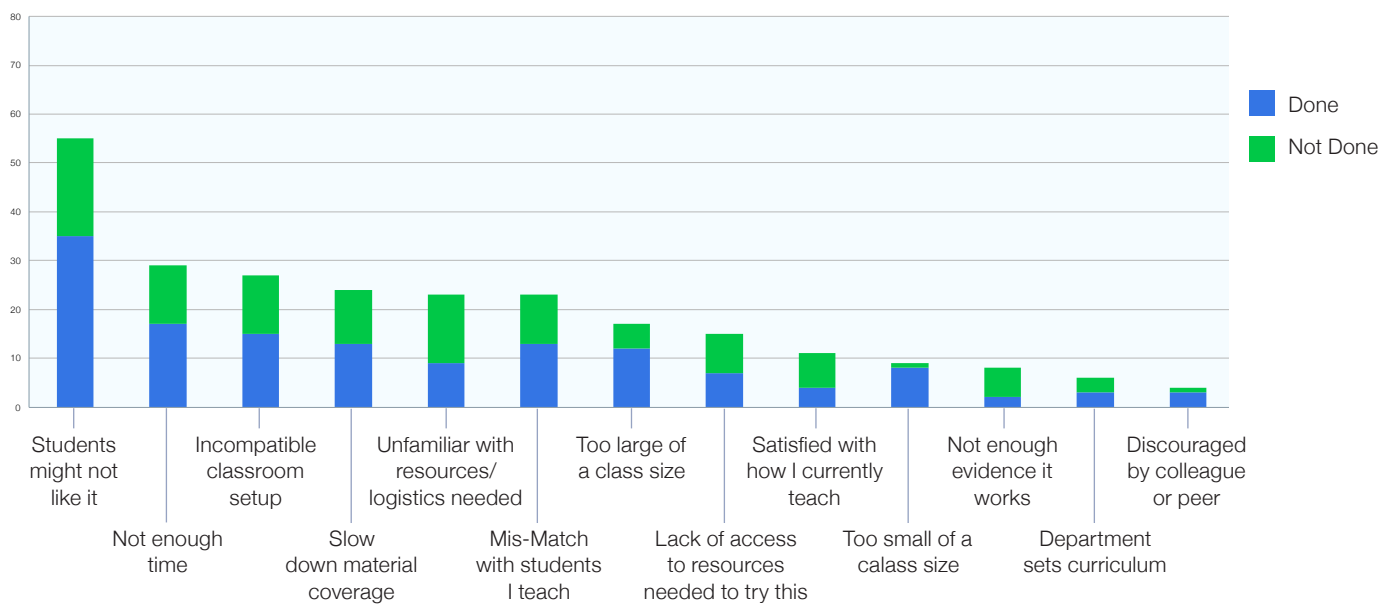
Of course, educators have concerns about using the intervention as well. The most considerable concern was that students might not like it. The next four concerns, with far fewer instructors concerned, were logistical: not enough time, incompatible classroom setup, slow down material coverage, and unfamiliar with resources/logistics needed. The last concern before a second noticeable drop off in instructor count, mismatch with students I teach, is an interesting one given the above discussion of how pair programming could help address the technical skills gap between academia and industry.

Our results indicate that Pair Programming will continue to be used in the future of computer science education. Of those who do not currently use pair programming exercises or assignments, 53% of them would like to implement the intervention in the future.

Pair Programming Benefits



Pair Programming Challenges



Assessment and Feedback Tools

Assessment and feedback tools allow instructors to auto-grade assessments and projects or provide students with automated feedback based on their responses. There's no denying that these tools play an important role in giving educators more time and the ability to teach more students, but what do educators have to say?

Besides saving instructors' time, educators believe that assessment and feedback tools primarily benefit students. A peek at the first four benefits include:

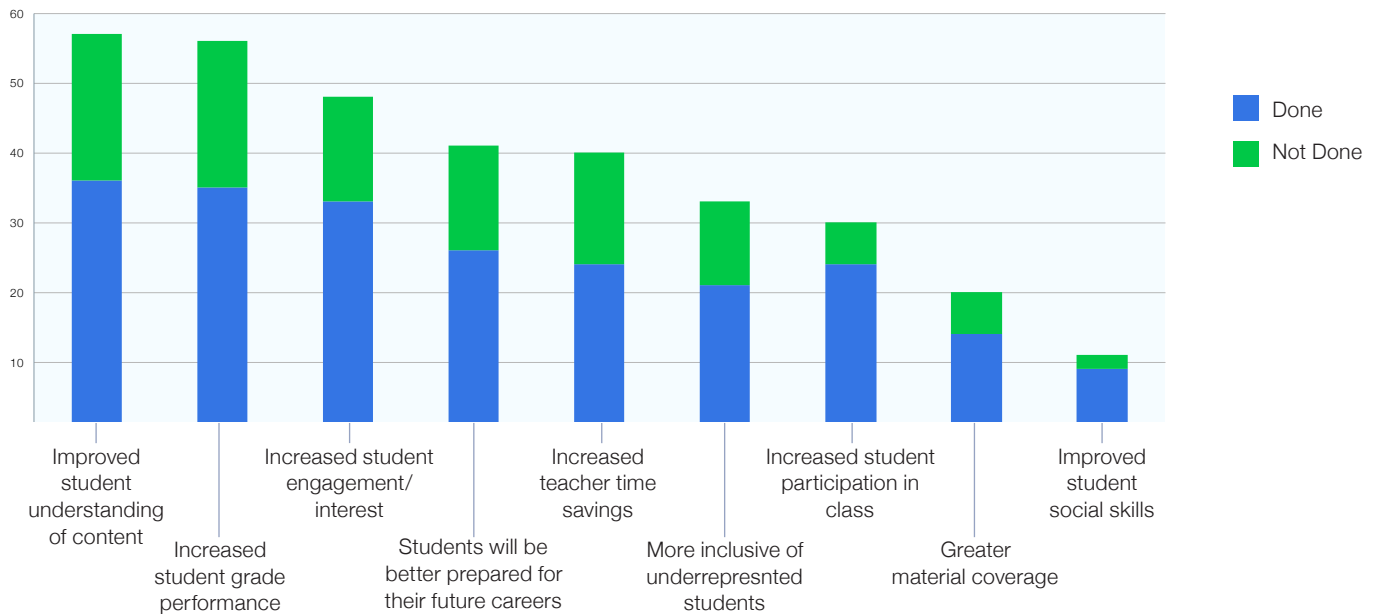
1. Students gain a better understanding of course content
2. Improvement in student grade performance
3. More engagement and interest from students
4. Students will be better prepared for their future careers

Understandably, the 5th benefit is increased teacher time savings. However, it is notable that educators primarily recognize that students benefit from the instant feedback in assessment and feedback tools.

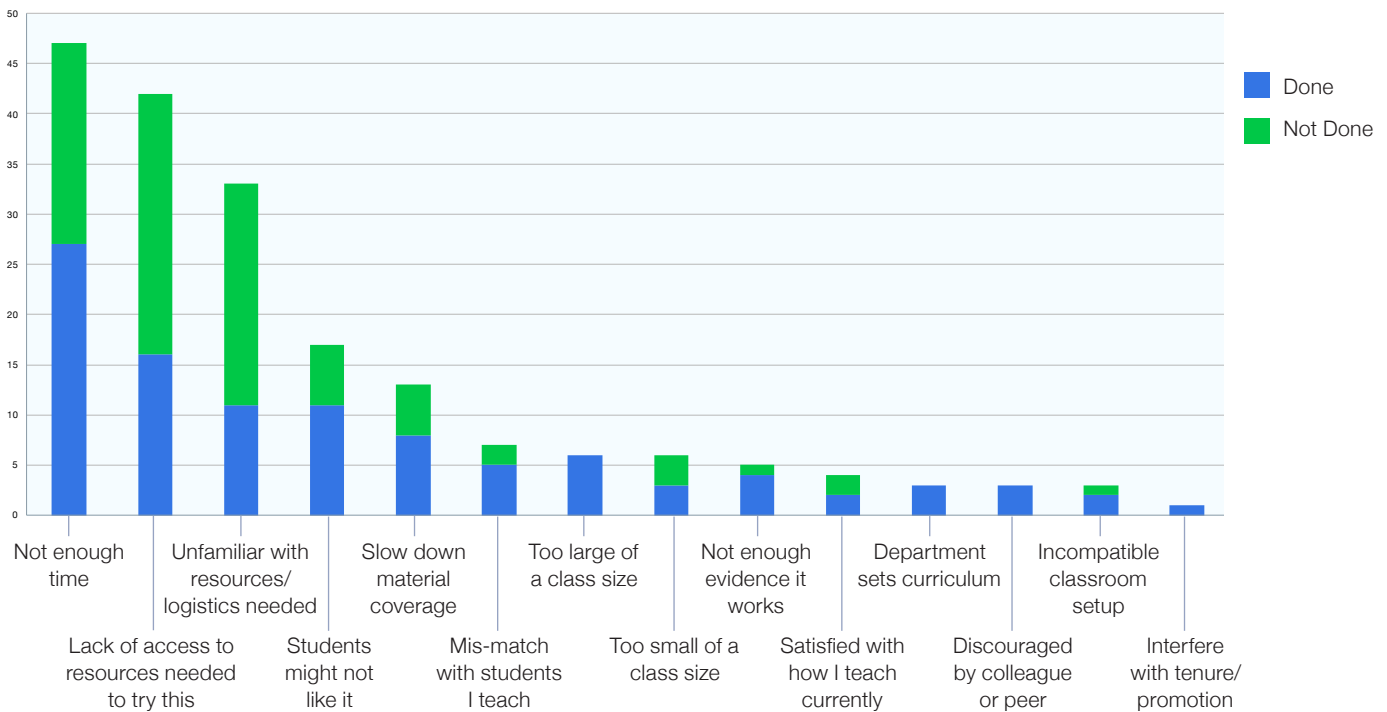
Based on the survey responses, it appears that the most significant barrier instructors have to overcome before implementing assessment and feedback tools is time. This could be due to the time to learn the tool and create automated graders as the next two concerns are about lack of access to resources to try this and unfamiliar with resources/logistics needed.

Aside from that, less than 20% of educators have concerns about the tools' actual impact on their course or students. This could be the reason why nearly $\frac{2}{3}$ of instructors already use these tools in some capacity.

Assessment and Feedback Tool Benefits



Assessment and Feedback Tool Challenges

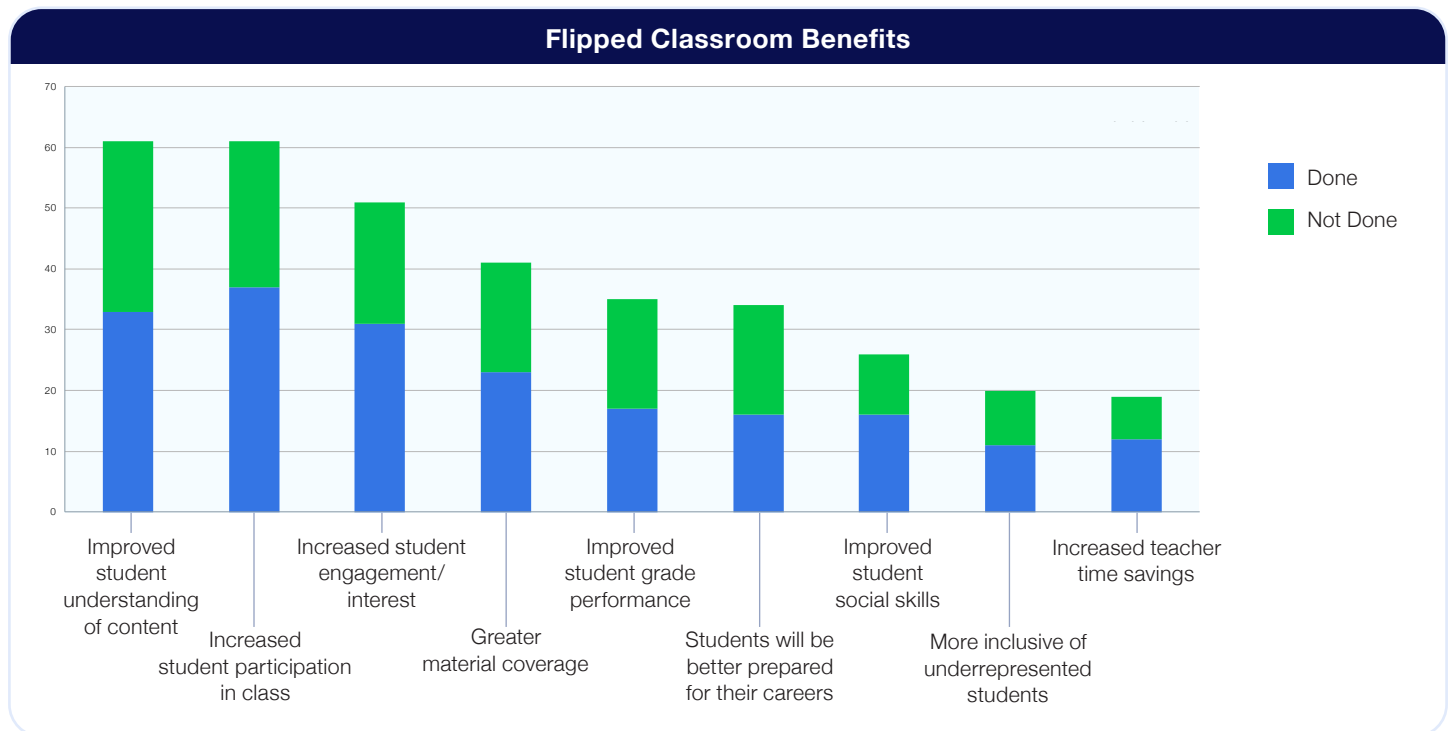


Flipped Classroom

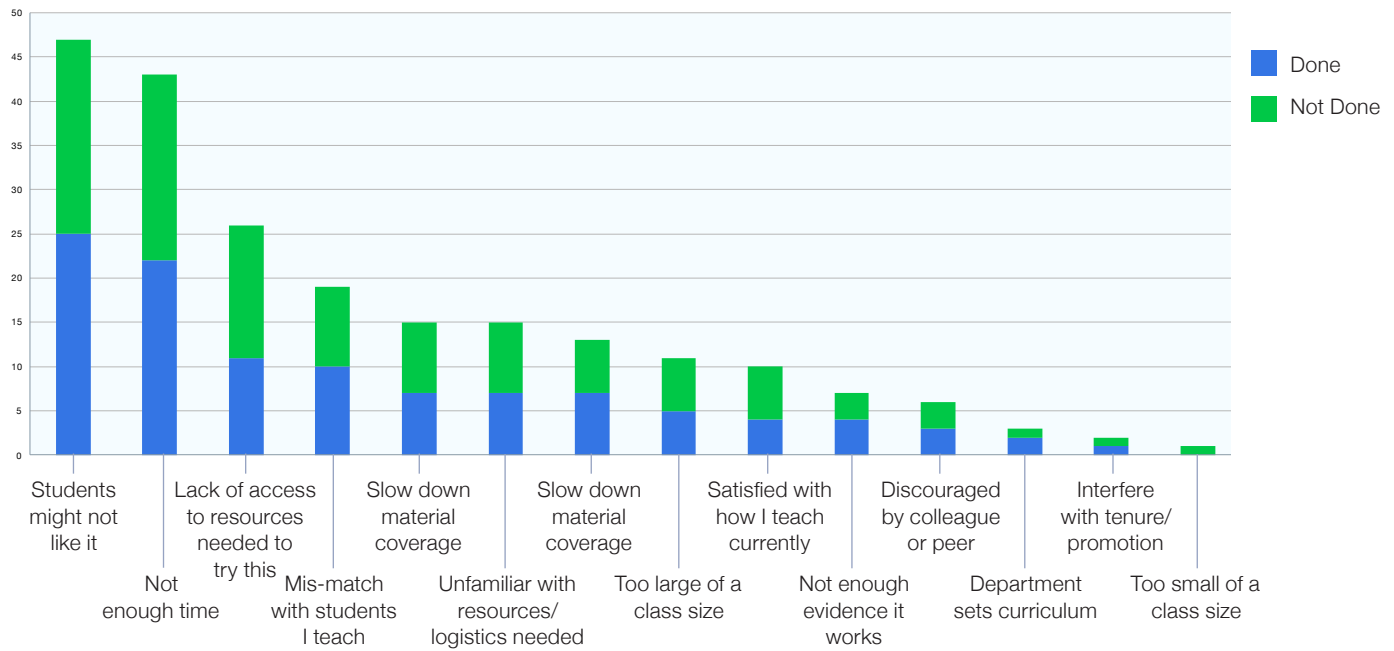
As a pedagogical intervention, flipped classroom breaks the mold of traditional classroom learning and lectures. It is based on the idea that students learn course material independently, and class time is used flexibly to engage with the concepts on a deeper level.

Educators believe that there are primarily two compelling benefits of using flipped classroom: students' understanding of course concepts improves, and students are more likely to participate in class than in a traditional lecture-style environment. This is also the one intervention with a significant number of educators citing that Flipped Classroom learning allows them to cover more material within a semester.

Despite the benefits of improved understanding by students, many educators fear that their students won't like the learning model. This concern is followed closely by the challenge of time it would take to implement this pedagogical practice. It would require instructors to prepare and provide most course material outside of class and possibly alter their existing homework assignments to better fit class time.



Flipped Classroom Challenges



Perception of Evidence-based Pedagogy as Beneficial and Challenging

To consider the overall perception of evidence-based pedagogy, we look at how the various pedagogical approaches and tools we asked about stack up against each other.

Benefits

When it comes to the most beneficial practices and tools, all of which have upwards of 25% of educators strongly believing that they are beneficial, there are very few commonalities between them. This list includes:

- Peer Instruction
- Assessment and Feedback Tools
- Tools that Support Program Writing
- E-Books

Peer Instruction and Assessment and Feedback Tools are methods instructors use to gauge students' understanding and improve learning behavior. Tools that Support Program Writing provide students with hands-on learning experiences, while E-Books focus on the instruction of course material in a traditional sense.

As the graph on Page 21 indicates, over 80% of educators found all pedagogical interventions beneficial except one: games for learning programming. This could be due to the demographics of our survey respondents, primarily being higher education instructors.

Beneficial Pedagogical Interventions

1. Peer Instruction
2. Assessment and feedback tools
3. E-Books

4. Test-driven Development
5. Tools that support program writing
6. Code visualizers/simulators
7. Peer assessment or review
8. Pair Programming
9. Flipped classroom
10. Games for learning programming

Challenges

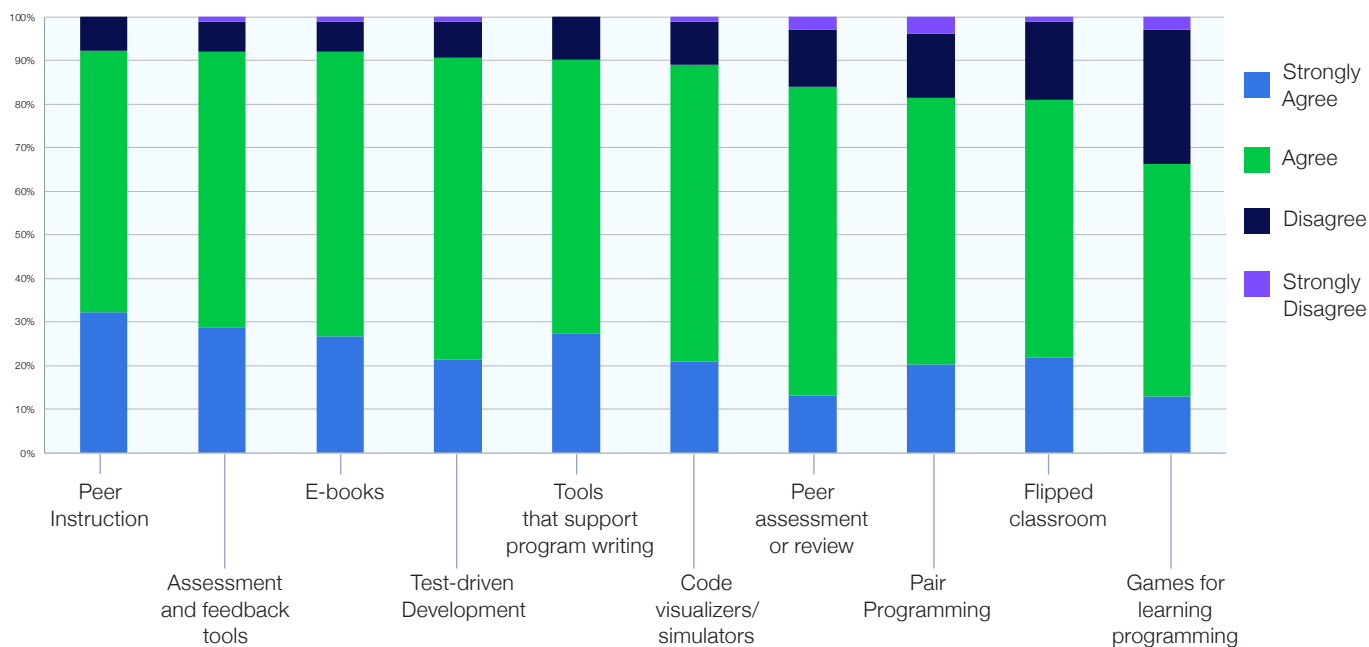
E-Books, the most widely adopted pedagogical intervention, also ranks as the least difficult to implement. The other two interventions that ranked higher in benefits than in challenges are Tools that support program writing and Peer Instruction.

Assessment and Feedback Tools are considered one of the most beneficial tools we explored throughout this research. It is also regarded as the most difficult to implement.

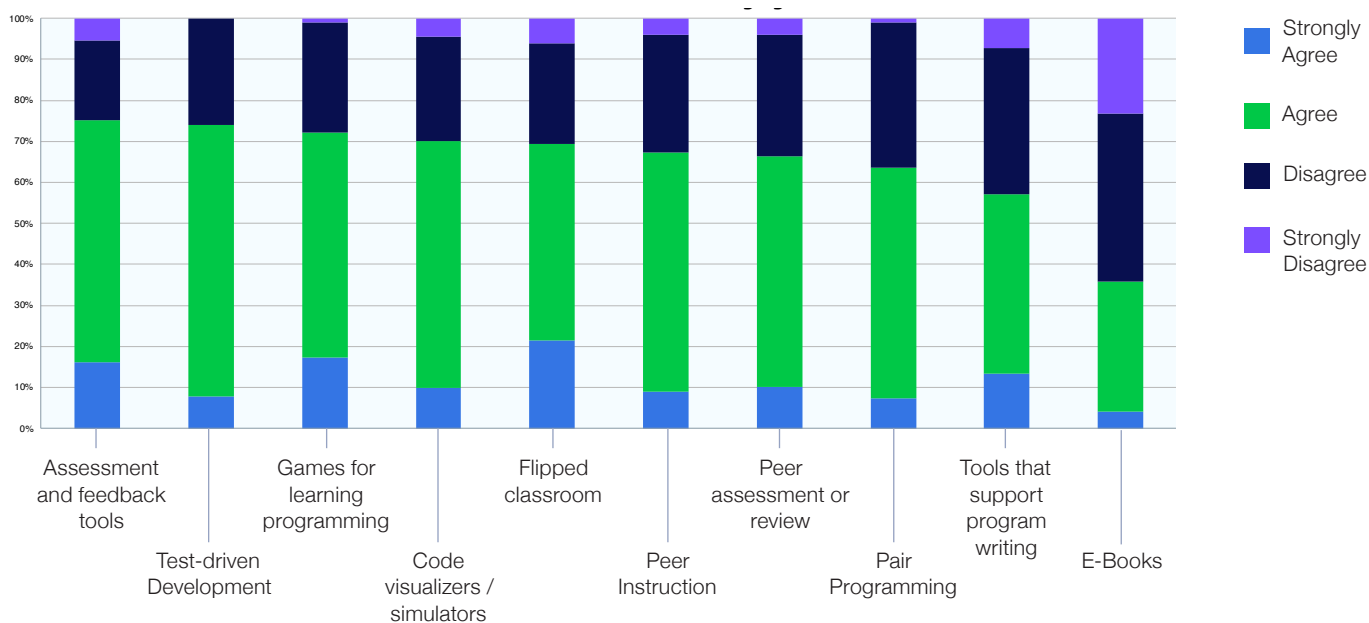
Most Challenging Pedagogical Interventions to Implement

1. Assessment and feedback tools
2. Test-driven Development
3. Games for learning programming
4. Code visualizers/simulators
5. Flipped classroom
6. Peer Instruction
7. Peer assessment or review
8. Pair Programming
9. Tools that support program writing
10. E-books

Is This Intervention Beneficial?



Is This Intervention Challenging?



Conclusion

Computer Science educators are not necessarily developers—with over 20% never having worked in industry and over 60% who primarily have taught. This could explain why assessment and feedback tools—which often require large amounts of development—were ranked as the most challenging intervention to implement.

Based on the results of this survey, it is apparent that educators adopt practices that benefit students. Consistently, educators ranked various elements of student learning outcomes as the most beneficial outcome for adopting a new learning practice. The most important benefit found across the pedagogical interventions was improved student understanding.

Rankings of benefits

1. Improved student understanding of content
2. Increased student engagement/interest
3. Increased student participation in class
4. Improved student grade performance
5. Students will be better prepared for their future careers
6. More inclusive of underrepresented students
7. Improved student social skills
8. Increased teacher time savings
9. Greater material coverage
8. Too large of a class size
9. Satisfied with how I teach currently
10. Not enough evidence it works
11. Too small of a class size
12. Discouraged by colleague or peer
13. Department sets curriculum
14. Interfere with tenure/promotion

Potentially more notably than how beneficial educators found these interventions (since 9/10 were rated as beneficial by more than 80% of respondents), educators varied more widely on which interventions were challenging. Their top concern was once again student-centered (i.e., students might not like it), followed by more logistical challenges.

Rankings of Challenges

1. Students might not like it
2. Not enough time
3. Unfamiliar with resources/logistics needed
4. Lack of access to resources needed to try this
5. Slow down material coverage
6. Mis-match with students I teach
7. Incompatible classroom setup

These challenges also seem to carry more weight than the benefits. Peer instruction, ranked by the most instructors to be beneficial, also ranked as about average in terms of how challenging instructors believed it would be to implement—which could explain it being found in less than half of CS classrooms despite the rich research behind it.

We hope these findings will help researchers more effectively market and disseminate their evidence-based pedagogy by focusing on the benefits to students—and possibly, more importantly, how to mitigate the challenges of implementing the intervention.

Similarly, we hope this helps administrators and educators identify promising evidence-based practices to bring into their computing classrooms and illustrates that while educators are motivated to do right by their students, they need support from principles, chairs, and deans. Administrators can offer educators support by providing time and access to resources and training opportunities to keep up with the ever-evolving field of computer science education.

References

- Barker, L., Hovey, C. L., & Gruning, J. (2015, February). What influences CS faculty to adopt teaching practices?. *In Proceedings of the 46th ACM Technical Symposium on Computer Science Education* (pp. 604-609).
- Falkner, K., & Sheard, J. (2019). Pedagogic approaches. *The Cambridge Handbook of Computing Education Research*, 445-480.
- Fincher, S. A., & Robins, A. V. (Eds.). (2019). *The Cambridge handbook of computing education research*. Cambridge University Press.
- Grover, S. (2020). *Computer Science in K-12: An A-To-Z Handbook on Teaching Programming*. Edfinity.
- Guo, P. J. (2013, March). Online python tutor: embeddable web-based program visualization for cs education. *In Proceeding of the 44th ACM technical symposium on Computer science education* (pp. 579-584).
- Guzdial, M. (2019, February). Computing education as a foundation for 21st century literacy. *In Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (pp. 502-503).
- Hovey, C. L., Barker, L., & Nagy, V. (2019, February). Survey Results on Why CS Faculty Adopt New Teaching Practices. *In Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (pp. 483-489).
- Jacobson, N., & Schaefer, S. K. (2008). Pair programming in CS1. *ACM SIGCSE Bulletin*, 40(2), 93-96.
- Luxton-Reilly, A. (2009). A systematic review of tools that support peer assessment. *Computer Science Education*, 19(4), 209-232. <https://doi.org/10.1080/08993400903384844>
- Malmi, L., Utting, I., & Ko, A. J. (2019). Tools and environments. *The Cambridge Handbook of Computing Education Research*, 639-662.
- Patel, H., & Morreale, P. (2014). Education and learning: electronic books or traditional printed books? *Journal of Computing Sciences in Colleges*, 29, 21-28.
- Radermacher, A., & Walia, G. (2011). Investigating the effective implementation of pair programming. *In Proceeding of the 42nd ACM technical symposium on Computer science education - SIGCSE '11*.
- Radermacher, A., & Walia, G. (2013, March). Gaps between industry expectations and the abilities of graduates. *In Proceeding of the 44th ACM technical symposium on Computer science education* (pp. 525-530).

Codio makes it easy to adopt the pedagogical tools and practices mentioned in this report.

Our fully-editable interactive course resources are enhanced with auto-graded assessments for immediate student feedback, along with tools like code visualizers to maximize student engagement.

Evaluate and edit any of our course resources with a free instructor account. Simply click any of the resources on the right to get started.




CS: Introduction in Python

Evaluate >



CS: Introduction in Java

Evaluate >



Engineering Software as a Service

Evaluate >



CS: Introduction in C++

Evaluate >

Looking for something else?

Check out our entire catalog of editable, interactive course resources!

[Browse Catalog >](#)

