ISSN: 2473-4901
v11 n6398

Introductory Coding Interventions: Frequency of Interventions Being Implemented at Universities

Daniel Freburg
djfrebu@ilstu.edu

Todd Thomas
tethoma@ilstu.edu

Illinois State University
Normal, IL USA

Abstract

Introductory coding courses at universities have had a reputation for high failure rates that require the support of interventions to increase student success. Numerous studies have been completed analyzing the effectiveness of single interventions within introductory coding courses. Most of those studies were of a single intervention within a single university setting. These studied interventions include pair programming, tutoring, teaching assistants, professor office hours, game-based learning, live coding, competitive coding, tutorial websites, artificial intelligence, pre-introductory coding courses, and more. This study focuses on analyzing the results of a survey distributed to introductory coding course instructors to determine the frequency of interventions being applied. The authors surveyed instructors across 64 (N=64) different universities, which consist of 22 small (<5,000), 21 medium (5,000-15,000), and 21 large (>15,000) institutions. This paper presents the findings and the results of the frequency of these interventions. These findings are meant to contribute to the current literature about introductory coding courses and present information that could assist introductory coding instructors and researchers.

Keywords: Introductory Coding Courses, Introductory Coding Interventions, Introductory Coding Course DFW Rates, Coding, Computer Science Introductory Course, Artificial Intelligence

Introductory Coding Interventions: Frequency of Interventions Being Implemented at Universities

Daniel Freburg and Todd Thomas

1. INTRODUCTIONS

Introductory programming courses are notorious for having high DFW rates, including rates as high as 50% (Margulieux et al., 2020; Bennedsen & Caspersen, 2007, 2019). In the past several decades, many studies have been conducted about individual interventions being implemented at the study's location to determine if the intervention assists in lowering the failure rates of the course. Such studies included interventions such as pair-programming, tutoring, live coding, etc. Those studies indicated that each intervention increased student success.

The authors have been studying introductory coding course interventions as part of a PhD thesis and as part of job duties as an IT Learning Center Director within the School of IT at Illinois State University. The authors have read the currently available literature and studied how various interventions have been successful. The authors decided to survey introductory coding instructors across multiple universities (most responses were from US Universities) to determine the applied frequency of introductory coding interventions. The findings are meant to inform introductory coding instructors, contribute to current literature, and assist with a later research goal of determining which interventions or combinations best contribute to lowering DFW rates. Course design and institutional information are presented along with the intervention's frequencies.

2. BACKGROUND

Much research has been performed and published on the various intervention methods used in introductory coding courses. Most published research focuses on individual interventions and is often confined to a single university and study. Additionally, numerous papers about improving student success have been published. However, the authors could not locate current statistics about which interventions were being implemented in introductory coding courses at universities and colleges, and the frequency with which they are being applied. An introductory coding literature review study reported 1,666 papers from 2003 to 2017. (Luxton-Reilly, A., et al. 2018). Of those papers, none focused solely on the frequency with which interventions are being applied across multiple universities. Table 1 below shows the categories that were covered in these studies.

Groups	Papers	Subgroupings
Student	489	Learning, underrepresented groups, attitudes, behaviors, engagement, ability, experience, code reading, tracing, writing, debugging
Teaching	905	Tools, pedagogical approaches, theories of learning, infrastructure
Curriculum	258	Competencies, programming languages, paradigms
Assessment	192	Tools, approaches, feedback, academic integrity

Table 1: Introductory Coding Literature Published 2003 - 2017 (some classified into two or more groups)

In this section, we will briefly describe the interventions included in the survey that was distributed to introductory coding instructors and provide reference(s) for each intervention.

Course structure

The course structure was once considered a standard that included textbooks, lectures, assignments, quizzes, and examinations. During the past several decades, the standard course design has been examined and the aspects studied. Many studies have concluded that the design of a course contributes to student outcomes (Freeman, S. et al. 2014; Freeman, S. et al. 2011). The study includes findings that active learning course structures see an approximate 5% exam score increase and an overall 30%-50% reduction in fail rates compared to traditionally designed courses. Students in traditional course structures were 1.5 times more likely to fail than those in active learning environments. Active learning can include many learning methods or activities, some of which are asked about in the survey, such as pair-

programming, use of TAs, and competitive coding. The survey distributed to introductory coding instructors included questions about class size, office hours, course delivery methods (lab/lecture, classroom/online/hybrid), and the use of teaching assistants to give us an insight into current course structures. The related data are presented in the findings section.

Tutoring

Academic tutoring is a universally accepted practice within institutions and intuitively produces better student outcomes. A 2019 study reports a 50% reduction in failure rates of underrepresented groups with participation in peer tutoring (Made et al., 2019). Our study did not focus on how tutoring impacts students, but instead focused on how often tutoring is offered to introductory coding students.

Game-based learning

The survey distributed to introductory coding instructors defined game-based learning as 'game characteristics and principles embedded within learning activities.' The study of game-based learning and coding began several decades ago. One of the first publications on the topic was a book titled "Children, Computers, and Powerful Ideas" (Papert, S. 1980). The article "A Meta-Analysis of the Cognitive and Motivational Effects of Serious Games" defines game-based learning and states its findings of game-based learning being more effective than conventional instruction methods (Wouters, P. et al., 2013). More recently, in a thesis titled "The Effects of Games in CS1-3", the author states that in the study, the students positively respond to game-based learning (Bayliss, J.D. 2007).

A recently published academic literature review of game-based learning in computer science reported 113 published articles between 2017 and 2021 (Videnovik, M., et al., 2023). The recent increase in published research on game-based learning and the reported student success outcomes is why the game-based learning intervention was included in the survey.

Live coding

Live coding is defined as "the process of designing and implementing a [coding] project in front of class during lecture" (Paxton, J. 2002). The article "The Effectiveness of Live-Coding to Teach Introductory Programming" concludes that live coding is as good as, if not better than, teaching with static code. The author suggests that live coding may help students prepare and complete the final project (Rubin, M.J., 2013). A recently published academic literature review article reported finding twenty academic papers, particularly on live coding as a teaching tool in the classroom (Selvaraj, A., et al., 2021). An author of this paper has also studied the success of live coding in the introductory coding classrooms, in his PhD thesis entitled "Utilizing Information Technology and Hands-on Learning Practices to Improve Student Learning Outcomes in a High Failure Rate Introductory Programming Course" (Thomas, T., 2025).

Pair-programming

Pair-programming is defined as "A software development technique in which two programmers work together at a single workstation on the same code or task. Typically, one programmer, known as the 'driver,' writes the code, while the other, the 'navigator' or 'observer,' continuously reviews each line of code as it is typed, thinking strategically about the overall direction, identifying potential errors, and suggesting improvements. The two programmers frequently switch roles" (Beck, K. 2000). The academic article "Pair-programming in Education: A literature Review" provides an in-depth review of the numerous studies conducted on pair-programming and the numerous aspects involved (Hanks, B. et. al. 2011). Because pair-programming is a widely accepted practice, it was included in the survey.

Competitive coding

Competitive coding is a mind sport where participants solve logical or mathematical problems by writing computer programs within a limited time frame and under specific constraints. The goal is to produce correct and efficient solutions that can pass a series of usually hidden test cases. The International Collegiate Programming Contest (ICPC) is recognized as being the start of competitive coding with its first contest in the 1970s (International Collegiate Programming Contest, n.d.). Competitive coding research has not been as numerous as other interventions. However, several studies have outlined the positive effects of competitive coding on student outcomes, such as "Competitive Programming in Computational Thinking and Problem-solving Education" (Yuen, K. 2023). That study states, "overall, students expressed a positive attitude toward adopting programming contests as it helped improve their problem-solving and programming skills, leading to overall employability. In a large-scale introduction programming course, classes adopting automated assessment can help to achieve a higher pass rate than those that do not." Because of the popularity of competitive coding among

college students and the research indicating it improves student outcomes; it was listed as an intervention in the survey.

Tutorial websites
A recent online search returned twenty-seven online free coding tutorial websites. Online coding tutorials, online courses, etc., have millions of users. Often, the use of resources outside of traditional teaching methods is considered cheating or a form of plagiarism in introductory coding courses. The authors were uncertain of how other higher education institutions viewed the use of online tutorial resources, but were familiar with recent research such as "A Pedagogical Analysis of Online Coding Tutorials," which studies the use and effect of students' use of online tutorials (Kim, A. S., & Ko, A. J. 2017). The use of tutorial websites was included in the survey because the authors wanted to gather information about how often popular online tutorial websites are allowed to aid students in introductory coding courses.

Artificial intelligence
The use of artificial intelligence in introductory coding courses was once considered to be a forbidden tool. However, since the recent release of online access to AI tools and AI add-ons to IDEs (Integrated Development Environments), student use of AI is increasing, and some instructors are beginning to allow some forms of AI into the classroom. Recently, much research has been conducted about using AI in the classroom and the potential benefits to students.

The use of AI in education is outlined in "Artificial intelligence in education: A review" (Chen, L. et al. 2020). The article studies the use of AI in teaching, learning, administration, and management areas of education. "Artificial Intelligence in Education: A Systematic Literature Review" reported 2,223 articles published since 1984, with more than half being published since 2016 (Wang, S. et al. 2024). The above articles indicate that AI in education has become a much-studied topic in academia, which is why it was included in the survey. Our survey included a question asking if students were allowed to use AI to complete assignments.

Course alterations
The survey asked if the course had been examined and altered to improve student pass rates. The survey included the traditional methods: curving grades, content complexity reduced, dropping the lowest assignment/quiz/exam scores, change in course delivery, reduced class size, and others. The article "The Impact of Grading on A Curve: A Simulation Analysis" presents arguments for and against using a curved grading scale and data from a simulation (Kulick, G., & Wright, R., 2008). We included it in the survey because of the wide use of curving grades.

Pre-introductory course
Universities sometimes create a pre-introductory programming course to prepare computer science and information technology students without coding skills to be prepared for the introductory coding course. This concept is detailed with information about its success as an intervention. "CS 0.5: A Better Approach to Introductory Computer Science for Majors" (Sloan, R. H., & Troy, P. 2008). The intervention was listed in the survey, and the frequency is listed in the findings section.

3. METHODOLOGY

The survey was created using Qualtrics, an online survey tool. The questions were vetted by faculty knowledgeable about this field and an authority on qualitative interview techniques. The survey consisted of multiple-choice, multiple-answer, and fill-in-the-blank questions. This survey was distributed to ISCAP (Information Systems and Computing Academic Professionals) 2024 conference attendees. The survey was also distributed online via the ACM (Association of Computing Machines) listservs of the special interest groups of IT education and CS education (SIGITED, SIGCSED), which produced most of the responses. A link to the survey was also posted online to the MWAIS (Midwest Association of Information Systems) member forum posts section. The total number of responses received was 70. Of the 70 responses, 64 were 100% completed. Six survey response results were disregarded due to insufficient responses to the intervention frequency questions. Sixty-four results were analyzed with SPSS and Excel using frequency distribution. The comments provided were analyzed and coded into themes.

4. FINDINGS

Survey Demographics
The survey results include 64 (N=64) universities, which consist of 22 small (<5,000), 21 medium (5,000-15,000), and 21 large (>15,000) institutions. Public institutions provided 36 (56%) of the results, while private institutions accounted for 28 (44%). Most reporting institutions grant a bachelor's degree (95%), while a small percentage (5%) only grant associate's degrees.

ISSN: 2473-4901
v11 n6398

The top degree path offered was Bachelor of Science, Computer Science (53%). A more detailed demographic breakdown can be found in Appendix A.

Course Design
The reported introductory coding class sizes were 15-40 (75%), >80 (14%), 40-80 (6%), and <15 (5%). The survey respondents were asked to report on the course design/delivery methods and asked to select all that apply. The results were as follows: lecture/lab (88%), lecture (31%), in-person (72%), online (23%), and hybrid (16%).

The coding languages offered included Python, Java, C++, JavaScript, C#, C++, HTML, Scheme, and C. The breakdown of which programming language is being taught can be seen in Figure 1 below. The other category was comprised of Scheme, C, and MATLAB.

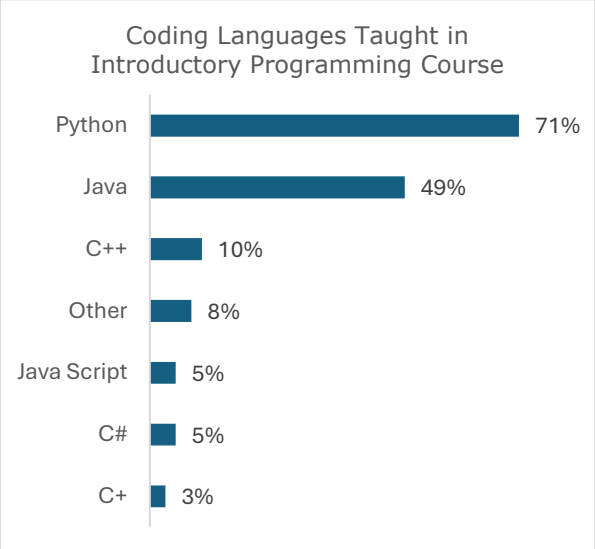

Figure 1: Coding Languages Taught in Introductory Programming Courses

Teaching Assistants (TAs) were used by 61% of respondents. The use of TA's was reported in the following ways: TAs in labs (45%), TAs as tutors outside of lab time (42%), TAs in lecture (13%), TA usage as depicted below:

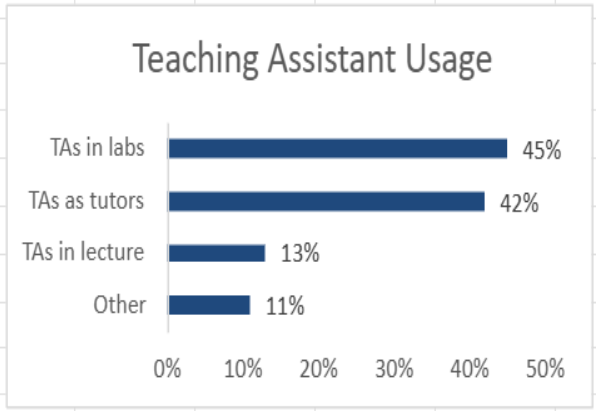

Figure 2: Teaching Assistant Uses

Interventions
The primary findings of this study, the frequency of interventions being implemented, can be seen in figure 3.

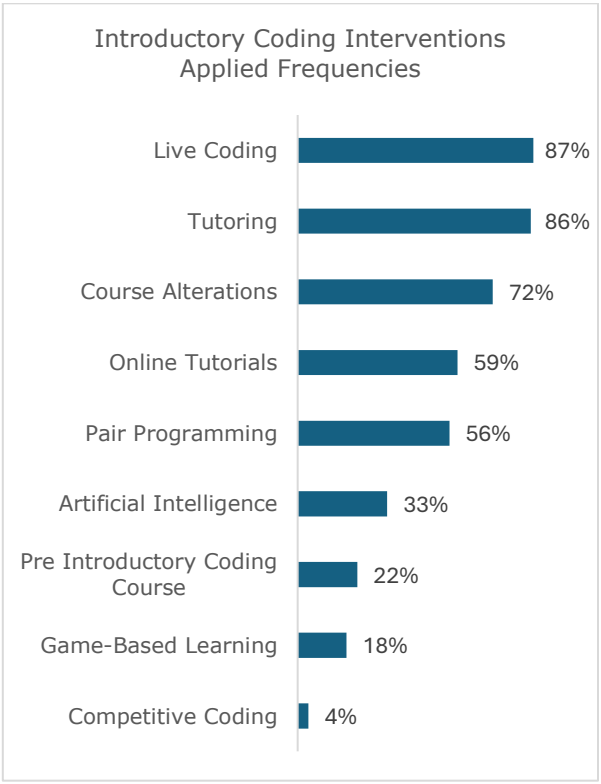

Figure 3: Introductory Coding Interventions Applied Frequencies

The primary interventions that are being utilized are Live Coding (87%), Tutoring (86%), Course Alterations (72%), Online Tutorials (59%), and Pair Programming (56%).

The vast majority of institutions offer some type of tutoring (86%), with in-person (84%) and one-

ISSN: 2473-4901
v11 n6398

on-one (51%) being the most reported versions. Online (31%) and group-based (38%) are the next highest reported groups. Tutors were also used to assist with debugging programs (33%) and test prep/test review (20%). Only 9% of universities offered no form of tutoring. A few respondents (5%) reported being unsure if tutoring was offered.

Respondents were asked if course alterations were implemented to improve student learning outcomes. Of the 72% that responded yes, the most commonly used strategies were changing course delivery (42%) and dropping the lowest scores on homework, labs, or exams (33%). The full results can be seen in Figure 4 below.

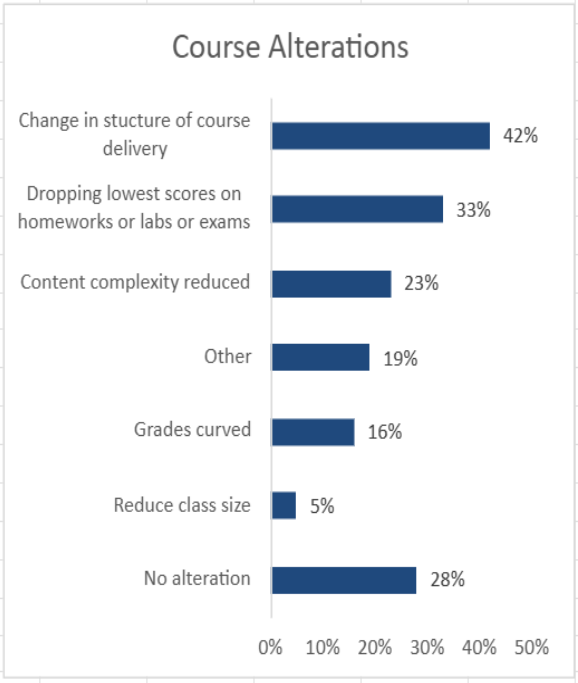

Figure 4: Introductory Coding Course Alterations

Online tutorials were reported to be allowed by students by 59% of introductory coding instructors. The least reported introductory coding interventions reported were use of AI to complete assignments (33%), use of a pre-introductory coding course (22%), game-based learning (18%), and competitive coding (4%).

Complete survey results are listed in Appendix A.

Qualitative Feedback

The completed surveys included valuable comments left by the respondents that share their lived experience with implementing these interventions. In this section, we will summarize and group these comments by theme. Please see Appendix B for a list of all comments.

Course Design Feedback

Multiple respondents reported offering more than one type of introductory programming course, offering different languages and complexities. This method allows students to actively choose the amount of lower-level introductory programming that is needed before moving up to the intermediate level. Two commentators mentioned that they allow students to bring the course text with them into the exams, which helps students with syntax issues. One commentator mentioned they use a parachute option for failing students, a bridge course for those who were on the bubble and allow the resubmission of previously failed material.

Flipped Classroom

The strategy of using a flipped classroom was mentioned multiple times. This is a strategy where most of the class time is spent on lab activities, and a much smaller portion of time is spent in a traditional lecture. All respondents who used this strategy also used pair programming. The main difficulty reported with this strategy is that not all students are good partners. One commented that a flipped classroom was beneficial, but not a magical solution.

POGIL (Process Oriented Guided Inquiry Learning)

A student-centered instructional approach that has students work in small groups and use a learning model that emphasizes exploration, concept invention, and application. (should probably include site for this) Three respondents reported using this teaching method with good success.

Intervention Complications

The main struggles that were most reported with all of these interventions were that students often do not work well in a group setting. Another repeated complaint was that some students struggle with an active learning style and are more accustomed to a passive role with direct instruction. Another complication that was mentioned multiple times is that it's difficult to get all professors to do the exact same thing. When the same class is taught by many different professors, getting everyone to teach with the exact same methodology can be difficult. And one comment on the use of AI in introductory coding courses seems to be a trend the authors are also currently witnessing:

ISSN: 2473-4901
v11 n6398

> "Because of AI, we are now getting students with straight A's in the intro sequence, that are completely unable to write/trace/modify/explain even the smallest program in their later classes. It is a disaster, they just get passed through."

Self-reported DFW rates

The self-reported DFW rates of the survey respondents are listed below. These are not included as major findings because the authors did not evaluate the courses, course content, grading methods, and the DFW rate methodology of the respondents. DFW rates will be examined in a further study.

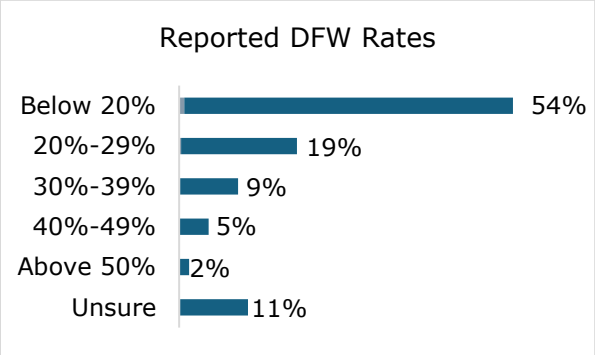

Table 5: DFW rates

5. CONCLUSION

We can conclude from the findings that the average introductory coding course is of size 15 – 40 and is taught in an in-person lecture/lab design using either Python or JAVA. More than half of the courses are supported by teaching assistants, and professors provide an average of 1-4 office hours per week. Nearly all courses have some form of intervention to assist students with being successful in the course. More than 50% of the respondents reported utilizing the interventions of live-coding, tutoring, course alterations, online tutorials, and pair programming. We can also conclude that while competitive coding and game-based learning are well-documented interventions that increase student success, the interventions are being implemented less frequently than anticipated. The allowed use of AI by students to complete assignments was reported to be allowed by one-third of respondents. We expect that with the increased academic research about AI in education, this statistic could easily change within the coming years.

6. FURTHER STUDY

A follow-up study is planned to look deeper into the correlation between which interventions have the biggest direct impact on student learning outcomes and DFW rates. This follow-up study will try to find the correlation between student success rates and which interventions and combination of interventions are most closely correlated to their success.

7. REFERENCES

Bayliss, J. D. (2007, February). The effects of games in CS1-3. In Microsoft Academic Days Conference on Game Development in Computer Science Education (pp. 59-63).

Beck, K. (2000). Extreme programming explained: embrace change. Addison Wesley professional.

Bennedsen, J., & Caspersen, M. E. (2007). Failure rates in introductory programming. ACM SIGCSE Bulletin, 39(2), 32-36. https://doi.org/10.1145/1272848.1272879

Bennedsen, J., & Caspersen, M. E. (2019). Failure rates in introductory programming: 12 years later. ACM Inroads, 10(2), 30-36. https://doi.org/10.1145/3324888

Chen, L., Chen, P., & Lin, Z. (2020). Artificial intelligence in education: A review. IEEE Access, 8, 75264-75278. https://doi.org/10.1109/ACCESS.2020.2988510

Freeman, S., Eddy, S. L., McDonough, M., Smith, M. K., Okoroafor, N., Jordt, H., & Wenderoth, M. P. (2014). Active learning increases student performance in science, engineering, and mathematics. Proceedings of the national academy of sciences, 111(23), 8410-8415. https://doi.org/10.1073/pnas.1319030111

Freeman, S., Haak, D., & Wenderoth, M. P. (2011). Increased course structure improves performance in introductory biology. CBE-Life Sciences Education, 10(2), 175-186. https://doi.org/10.1187/cbe.10-08-0105

Hanks, B., Fitzgerald, S., McCauley, R., Murphy, L., & Zander, C. (2011). Pair programming in education: A literature review. Computer Science Education, 21(2), 135-173. https://doi.org/10.1080/08993408.2011.579808

International Collegiate Programming Contest. (n.d.). The ICPC International Collegiate

ISSN: 2473-4901
v11 n6398

Programming Contest. Retrieved from https://icpc.global/

Kim, A. S., & Ko, A. J. (2017, March). A pedagogical analysis of online coding tutorials. In Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education (pp. 321-326). https://doi.org/10.1145/3017680.3017728

Kulick, G., & Wright, R. (2008). The Impact of Grading on the Curve: A Simulation Analysis. International Journal for the Scholarship of Teaching and Learning, 2(2), n2. https://doi.org/10.20429/ijsotl.2008.02020

Luxton-Reilly, A., Simon, Albluwi, I., Becker, B. A., Giannakos, M., Kumar, A. N., ... & Szabo, C. (2018, July). Introductory programming: a systematic literature review. In Proceedings companion of the 23rd annual ACM conference on innovation and technology in computer science education (pp. 55-106). https://doi.org/10.1145/3293881.3295779

Made, A. F., Hasan, A., Burgess, S., Tuttle, D., & Soetaert, N. (2019). The effect of peer tutoring in reducing achievement gaps: A success story. Journal of Computing Sciences in Colleges, 35(1), 57-65.

Margulieux, L.E., Morrision, B.B. and Decker (2020) 'Reducing withdrawal and failure rates in introductory programming with subgoal labeled worked examples', International Journal of STEM Education, 7(1). doi:10.1186/s40594-020-00222-7 https://doi.org/10.1186/s40594-020-00222-7

Papert, S. (1980). Children, computers, and powerful ideas (Vol. 10, pp. 978-3). Eugene, OR, USA: Harvester.

Paxton, J. (2002). Live programming as a lecture technique. Journal of Computing Sciences in Colleges, 18(2), 51-56.

Rubin, M. J. (2013, March). The effectiveness of live-coding to teach introductory programming. In Proceeding of the 44th ACM technical symposium on Computer science education (pp. 651-656). https://doi.org/10.1145/2445196.2445388

Selvaraj, A., Zhang, E., Porter, L., & Soosai Raj, A. G. (2021, June). Live coding: A review of the literature. In Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1 (pp. 164-170). https://doi.org/10.1145/3430665.3456382

Sloan, R. H., & Troy, P. (2008). CS 0.5: a better approach to introductory computer science for majors. ACM SIGCSE Bulletin, 40(1), 271-275. https://doi.org/10.1145/1352322.1352230

Thomas, T. (2025). Utilizing Information Technology and Hands-on Learning Practices to Improve Student Learning Outcomes in a High Failure Rate Introductory Programming Course. https://ir.library.illinoisstate.edu/etd/2136/

Videnovik, M., Vold, T., Kiønig, L., Madevska Bogdanova, A., & Trajkovik, V. (2023). Game-based learning in computer science education: a scoping literature review. International Journal of STEM Education, 10(1), 54. https://doi.org/10.1186/s40594-023-00447-2

Wang, S., Wang, F., Zhu, Z., Wang, J., Tran, T., & Du, Z. (2024). Artificial intelligence in education: A systematic literature review. Expert Systems with Applications, 252, 124167. https://doi.org/10.1016/j.eswa.2024.124167

Wouters, P., Van Nimwegen, C., Van Oostendorp, H., & Van Der Spek, E. D. (2013). A meta-analysis of the cognitive and motivational effects of serious games. Journal of educational psychology, 105(2), 249. https://doi.org/10.1037/a0031311

Yuen, K. K., Liu, D. Y., & Leong, H. V. (2023). Competitive programming in computational thinking and problem solving education. Computer Applications in Engineering Education, 31(4), 850-866. https://doi.org/10.1002/cae.22610

ISSN: 2473-4901
v11 n6398

APPENDIX A
Survey Answers

Q01 - Please select the category below that best describes the total enrollment, both undergraduate and graduate, at your institution during the Fall 2024 semester.

Size	%	Count
< 5000	34%	22
5,000 - 15,000	33%	21
> 15,000	33%	21

Q02 - Does your institution primarily grant associate degrees or bachelor's degrees?

Degree	%	Count
Associates	5%	3
Bachelors	95%	61

Q03 - Is your institution public or private?

Public / Private Institution	%	Count
Public	56%	36
Private	44%	28

Q04 - Please select the degree program for which you will be answering the remainder of these questions.

Degree Program	%	#
Associate of Science, Computer Science	5%	3
Associate of Science, Information Systems	3%	2
Bachelor of Science, Computer Science	53%	34
Bachelor of Science, Information Systems	13%	8
Bachelor of Business Administration, Information Systems	8%	5
Bachelor of Arts, Computer Science	11%	7
Bachelor of Science, Mathematics	3%	2
Other	5%	3

Other:
- Bachelor of Engineering, Computer Science and Engineering
- Bachelor of Cybersecurity Analytics and Operations

Course structure

Q05 - Please select the average number of students per section enrolled in introductory programming courses at your institution (please answer specific to IS/CS disciplines).

Course Size	%	Count
Less than 15	5%	3
15 - 40	75%	48
41 - 80	6%	4
more than 80	14%	9

Q06 - Please select the coding languages taught in the introductory programming courses

Code Language	%	Count
Java	49%	31
Python	71%	45
C++	10%	6
C+	3%	2
C#	5%	3
Java Script	5%	3
Other	8%	5

Other:
- C
- racket/scheme
- Scheme
- MATLAB
- HTML, CSS

Q07 - Please select the course delivery / teaching methods (select all that apply).

Course Delivery Method	%	Count
Lecture	31%	20
Lecture / Lab	88%	56
Online	23%	15
In-person	72%	46
Hybrid (In-person & online)	16%	10

Q08 - How many office hours per week do professors offer for introductory programming course students?

# of Office Hours	%	Count
0	2%	1
1-4	64%	41

ISSN: 2473-4901
v11 n6398

5+	33%	21
Unsure	2%	1

Q09 - Are teaching assistants used to support the introductory programming course, and if so, how?

TA usage	%	Count
No	39%	25
TAs in labs	45%	29
TAs in lecture	13%	8
TAs as tutors	42%	27
Other	11%	7

Other:

- We provide peer supplemented instructions twice a week.
- We have PALS (Peer Academic Leaders) that attend at least one class a week
- TAs as graders
- Graders and cross-section tutors
- Sometimes we hire students as TA's for labs, but it's not a normal practice
- We have lots of adjuncts teaching instead of TAs
- TAs as tutors and in labs as available, but not always

Tutoring

Q11 - Does your institution offer tutoring for introductory programming course students?

Tutoring	%	Count
Yes	86%	55
No	9%	6
Unsure	5%	3

Q12 - Please select the choices that describe the available tutoring (select all that apply).

Types of Tutoring	%	Count
In-person	84%	54
Online	31%	20
Personal (one-on-one)	53%	34
Group	38%	24
Debugging	33%	21
Test prep / Test review	20%	13
Intelligent Tutoring Systems	2%	1

Game-based learning

Q13 - In what proportion of your institution's introductory programming class is game-based learning implemented as a teaching method and / or teaching intervention? (where game characteristics and principles are embedded within learning activities)

Game-based learning	%	Count
No sections	72%	46
Some sections	16%	10
All sections	2%	1
Unsure	11%	7

Live coding

Q14 - In what proportion of your institution's introductory programming class is live coding used as a teaching method and / or teaching intervention? (teaching method where a computer program is written from scratch in real-time in front of a class)

Live coding	%	Count
No sections	3%	2
Some sections	34%	22
All sections	53%	34
Unsure	9%	6

Pair programming

Q15 - In what proportion of your institution's introductory programming class is pair programming a part of homework / programs / labs? (students are paired to complete homework / programs / labs with one student coding and the other supporting and the pairs switch positions)

Pair programming	%	Count
No sections	36%	23
Some sections	34%	22
All sections	22%	14
Unsure	8%	5

Competitive coding

Q17 - In what proportion of your institution's introductory programming class is competitive coding used as a teaching intervention? (programmers competing against each other to solve programming questions in a limited amount of time - popular websites HackerRank, LeetCode, Code Chef)

Competitive Coding	%	Count
No sections	91%	58
Some sections	5%	3
Unsure	5%	3

Online tutorials
Q18 - In what proportion of your institution's introductory programming class are students formally encouraged to use tutorial websites? (W3Schools, GeeksForGeeks, StackOverFlow, etc..)

Online Tutorials	%	Count
No sections	30%	19
Some sections	36%	23
All sections	23%	15
Unsure	11%	7

Artificial Intelligence
Q19 - In what proportion of your institution's introductory programming class are students allowed to use Artificial Intelligence to complete assignments?

Artificial Intelligence	%	Count
No sections	52%	33
Some sections	28%	18
All sections	6%	4
Unsure	14%	9

Course Alterations
Q20 - Has the introductory programming course been evaluated and / or updated to improve student pass rates in the following ways? (select all that apply) - Selected Choice

Course Alterations	%	Count
No	28%	18
Grades curved	16%	10
Content complexity reduced	23%	15
Dropping lowest scores on homework or labs or exams	33%	21
Change in structure of course delivery	42%	27
Reduce class size	5%	3
other	19%	12

Other:
- restructure curriculum to be spread out over more terms, slowing pace
- Some faculty drop the lowest HW or Quiz, but not exams.
- The course is always being updated, but the improvements are not motivated by pass rates.
- Change in syllabus structure, scaffolding, grading scheme (not curving)
- Grading for Equity approaches, in particular mastery grading and revisions
- Our classes are no larger than 20.
- Mastery grading, Flipped classroom
- Differential enrollment based on background
- Specializing training for instructors to encourage growth mindset
- Adjusting course topic breadth to allow more time on difficult topics
- Use of online texts with interactive content - zyBooks

Pre-Introductory coding course
Q16 - Does your university offer a Pre-Introductory Coding Course? (a prep course for students without programming experience designed to prepare students to be successful in Introductory Programming courses)

Pre-Introductory coding course	%	Count
Yes	22%	14
No	75%	47
Unsure	3%	2

Self-reported DFW rates
Q22 - What is your institution's current failure rate (letter grades of D, F, or withdraw) for introductory programming courses?

DFW rate	%	Count
Below 20%	53%	34
20%-29%	19%	12
30%-39%	9%	6
40%-49%	5%	3
Above 50%	2%	1
Unsure	13%	8

ISSN: 2473-4901
v11 n6398

APPENDIX B
Comments

We added a two hour lab to a flipped class. Students spend six hour in class per week. It's beneficial, but not a magic solution
Teaching functions first with Python has been successful in CS1. I personally use POGIL predominantly in my CS1. Many students appreciate the guided knowledge building of POGIL and the grade distribution has sifted where the lowest grade is a C (I attribute this to the actively engaging with the material). Some students struggle with active learning because they are being used to being taught using direct instruction (and assuming a passive role during class meetings) or they don't like to work in groups.
We offer six different introductory coding points with different complexity and languages that all feed into our intermediate course and with students allowed to take more than one introductory course with credit towards the major to allow students to select how much introductory preparation they need before moving to the intermediate level.
I use POGIL in about a third of my classes. Changes in course structure are mainly that we changed to Python in the first semester course, but retained Java in the second semester course. It is difficult to get all professors to do the same thing, except for dropping some grades on some assignments.
We are teaching computer logic before programming now. Started as of Fall 2024, still evaluating. Initial results look promising.
Approaches: Mastery grading, Flipped Classroom. Mastery grading makes students much less stressed. Flipped classroom (most of class time spent on lab-like activies in pairs) seems to have enhanced learning. Difficulties: Mastery grading involves much more grading time (and, sometimes, problem-writing time) from the instructor. Difficulties: Not all students are good partners in pair programming. Some students end up encountering microaggressions from their partners. We've addressed this issue by spending a day in class discussing how to be a good partner, how to deal with differences in apparent knowledge, and having the students set ground rules. Other: Many years ago, I heard Maria Klawe say that "theming" introductory classes has been one of the more successful approaches to broadening participation in computing.
low attendance

Hard to say, especially post Covid. Student abilities seem to have been going down, but maybe that's all over, or maybe our school is attracting worse students. Also, because of staffing, our CS1 sections are currently taught by Math PhD's, not CS PhD's..
No issues in implementing any of the interventions but arguably the interventions themselves have not led to success. Other interventions we've implemented: a "parachute" option for failing students (CS1 to CS0.5 at week 5), a "bridge" course for those that were on the bubble (C-) who wish to move on to CS2; limited resubmission of previously failed material
research faculty are so focused on research they don't seem to value much the development of full-time instructors.
Students have rebelled with the Pair programming so now used more informally but still a communal activity which is helpful
Letting students take the course text into final examination, so they know we won't be asking simple syntax questions
Our intro programming course is designed for students with no prior programming experience. It is taught "mandatory credit/non" (essentially pass/fail, with an additional "level" of a recommendation to continue immediately in cs2).
Because of AI, we are now getting students with straight A's in the intro sequence, that are completely unable to write/trace/modify/explain even the smallest program in their later classes. It is a disaster. They just get passed through
Helping students to debug their programs interactively.
We have separate streams in the first programming courses for students with programming experience and novices
Varying levels of effectiveness. Concerns of reducing the rigor of the course.
Students have more trouble adjusting to real work because the classroom has become such a successful control environment
Drop lowest grade intervention was implemented to offset late homework policy where 10% of the grade is deducted for each day late.
Getting student buy in
Biggest issue is motivating faculty
Student delay asking for the help they need until I ask an advisor to intervene (unless the student is my advisee) to determine whether

ISSN: 2473-4901
v11 n6398

the student has other obstacles I am unaware of that is preventing them from seeking the help they need.
Instructors can request that students in need receive weekly one-on-one tutoring from a fellow student.