2021 ASEE ANNUAL CONFERENCE
Virtual Meeting | July 26–29, 2021 | Pacific Daylight Time
Paper ID #34711

The Rise of Program Auto-grading in Introductory CS Courses: A Case Study of zyLabs

Chelsea L. Gordon, zyBooks, A Wiley Brand

Chelsea Gordon received her PhD in Cognitive Science at University of California, Merced in 2019. Chelsea works as a research scientist for zyBooks, a Wiley company that creates and publishes interactive, web-native textbooks in STEM.

Prof. Roman Lysecky, University of Arizona; zyBooks, A Wiley Brand

Roman Lysecky is VP of Content at zyBooks, A Wiley Brand and a Professor of Electrical and Computer Engineering at the University of Arizona. He received his Ph.D. in Computer Science from the University of California, Riverside in 2005. His research focuses on embedded systems, cybersecurity, and STEM education. He has authored more than 100 research publications, received nine Best Paper Awards, is an inventor on multiple patents, and received multiple awards for Excellence at the Student Interface.

Prof. Frank Vahid, University of California, Riverside

Frank Vahid is a Professor of Computer Science and Engineering at the Univ. of California, Riverside, and co-founder and chief learning officer of zyBooks.

The rise of the program auto-grading in introductory CS courses: A case study of zyLabs

Chelsea Gordon[*], Roman Lysecky[*,†], Frank Vahid[*,°]

[*]zyBooks (www.zybooks.com)

[†]Dept. of Electrical and Computer Engineering, Univ. of Arizona

[°]Dept. of Computer Science and Engineering, Univ. of California, Riverside

Abstract

In recent years, hundreds of college courses have switched how they grade programming assignments, from grading manually and/or using batch scripts, to using commercial cloud-based auto-graders with immediate score feedback to students and the ability to debug and resubmit for a higher score. This paper provides data on the rise in usage of one of the most widely-used program auto-graders, zyLabs, as one indicator of the strong shift in college course grading to the auto-grading paradigm. The number of courses, instructors, and students using zyLabs have increased dramatically since it was first introduced, such that from 2016 to 2020, the number of courses per year grew from 284 to 2,175, the number of students per year from 24,216 to 132,121, and the number of instructors per year from 364 to 2,866. Most instructors state they previously graded programs by hand and auto-grading saved an average of 9 hours per week. The result is a substantial shift in the classroom dynamic that enables instructors and students to spend more time on quality teaching and learning.

Introduction

Nearly all college-level introductory computer science (CS) courses require students to write programming assignments, often writing one or more programs every week.

In the past, most courses graded student programs by hand. A student would develop the program on their own, and then submit that program on paper or as online files. A teacher (instructor or teaching assistant) would grade each program's runtime correctness and often the code quality, providing written feedback. Human grading's main benefit is high-quality feedback, especially regarding code-quality (style, problem-solving approach, comments). But drawbacks include extensive human resource usage, which is expensive and detracts from other high-value contributions that teachers could make, and a delay of days or weeks before students get feedback, which can hinder learning.

Today, many courses use a cloud-based auto-grader. Students submit their programs to a webpage, which in seconds gives feedback on the program's runtime correctness along with a score. Students can then resubmit to improve their score [1], [2] aided by automated feedback [3]. The benefits include reduced human resources, and immediate feedback to aid learning [4]. Drawbacks include little or no feedback on coding style [5], potential student overreliance on the auto-grader to test programs [6], and potential cheating of the auto-grader [7]. Some instructors combine manual and auto-grading, letting the auto-grader provide an initial score based on runtime correctness, and then later manually adding a score based on code quality.

To provide instructors visibility into the trend towards auto-grading, this paper provides data detailing the growth and usage of one particular auto-grader: zyLabs [8].

zyLabs since 2015

zyBooks was founded in 2012 to improve learning content in introductory CS courses. Its initial product in 2013 included a web-based textbook replacement created natively for the web, thus using less text and instead using 100+ animations and 1000+ interactive learning questions. An integrated homework system was added in 2014, consisting of short auto-graded coding challenges, where students complete a program by writing about 3-10 lines of code, or determining the output of a given program. An example graded zyLab assignment is illustrated in figures 1 and 2 below.

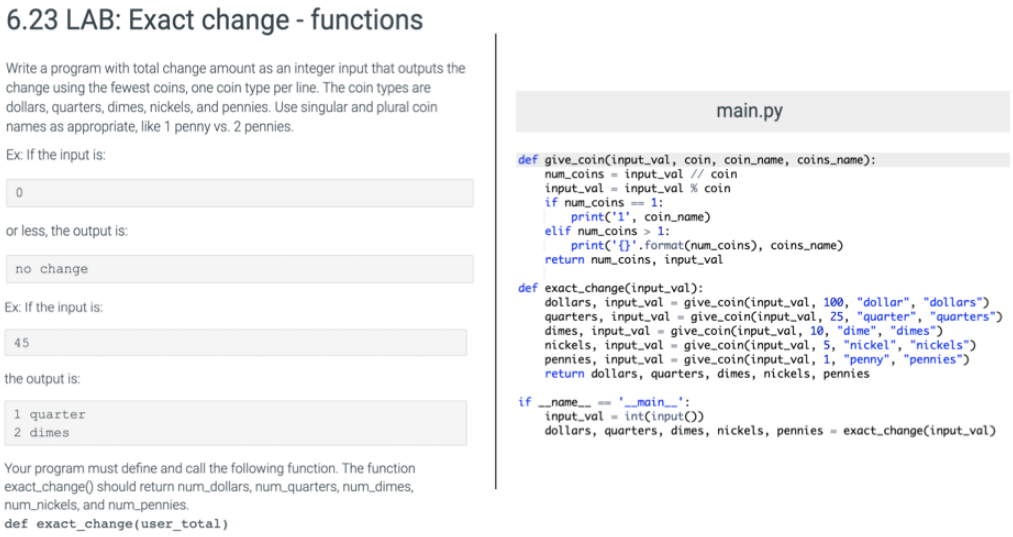

Figure 1: Example graded zyLab assignment. The assignment instructions are shown on the top left, and submitted code on the top right.

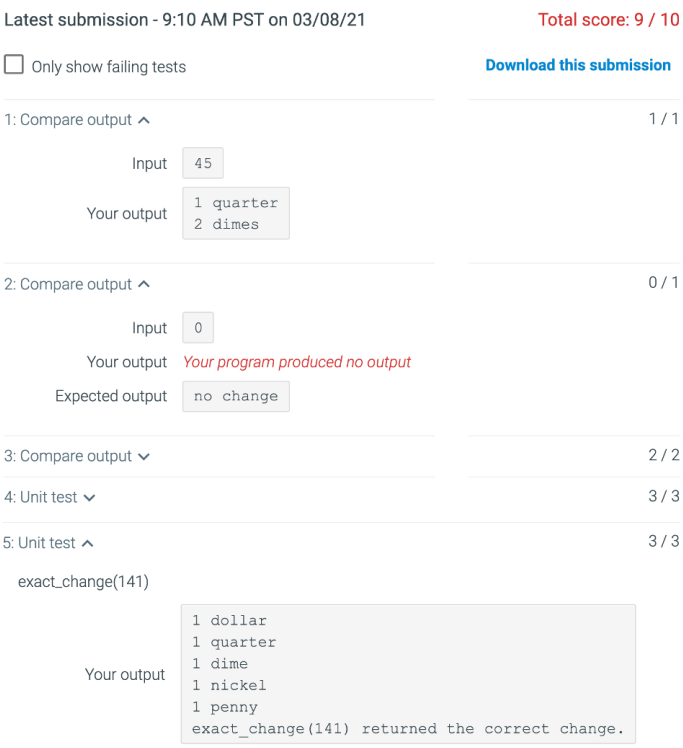

Figure 2: Test results for the example assignment. The submitted code results in a grade of 9/10, losing one point for not returning “no change” when the input is 0.

In late 2015, zyBooks released zyLabs to auto-grade the main remaining component of introductory CS courses, namely the weekly programming assignments. Instructors have the option to enable a develop mode for zyLab programming assignments. When enabled, develop mode allows students to write and run their code within the zyBook as often as they choose, before (and after) submitting the assignment for grading. Instructors who do not enable develop mode have students use other tools to develop code (IDEs -- integrated development environments), then submit that code to zyLabs for auto-grading.

Adoptions

Figure 3 shows the number of course offerings that adopted a zyBook with zyLabs enabled, per year since 2016. A course offering is a delivery of a course in a given term, such as "CS1 at Univ. of Springfield in Fall 2016". zyLabs adds an extra cost beyond a base zyBook's cost, and thus courses with zyLabs enabled almost always make use of the zyLab auto-grader. For all figures in this document, courses are included if at least 9 students were subscribed to that course.

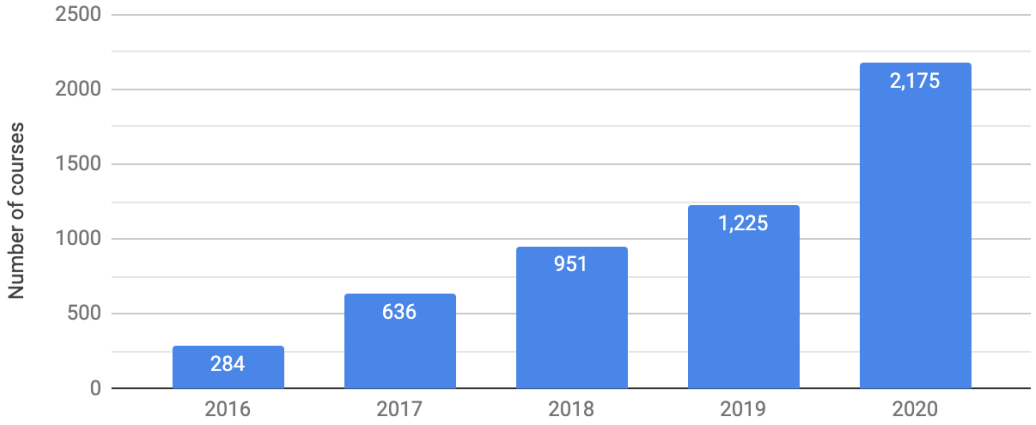

Figure 3: Course offerings adopting zyLabs per year.

Figure 4 shows the number of students subscribed to those zyBook course offerings each year, thus representing the number of students to whom the zyLab auto-grader was available and likely used.

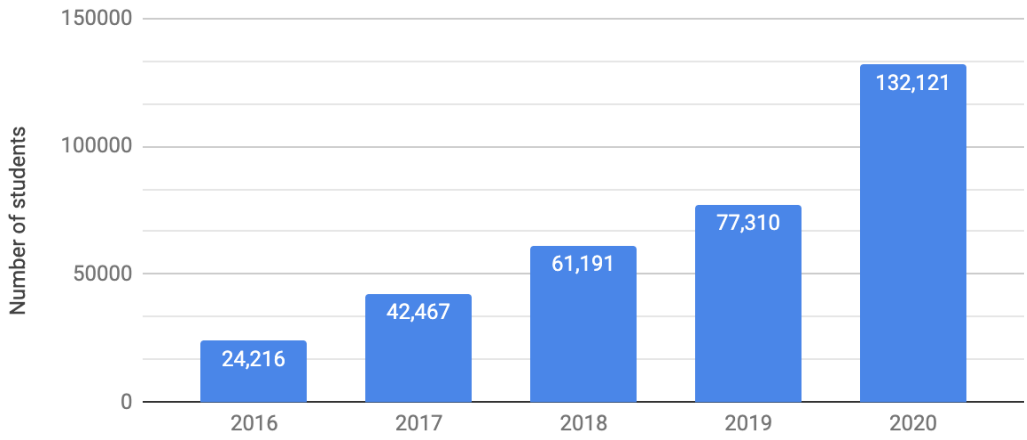

Figure 4: Student zyBook subscriptions with zyLabs enabled per year.

Figure 5 shows the number of instructors teaching classes using a zyBook with zyLabs enabled, per year. As multiple instructors might teach the same course, this number is larger than the number of courses shown above.

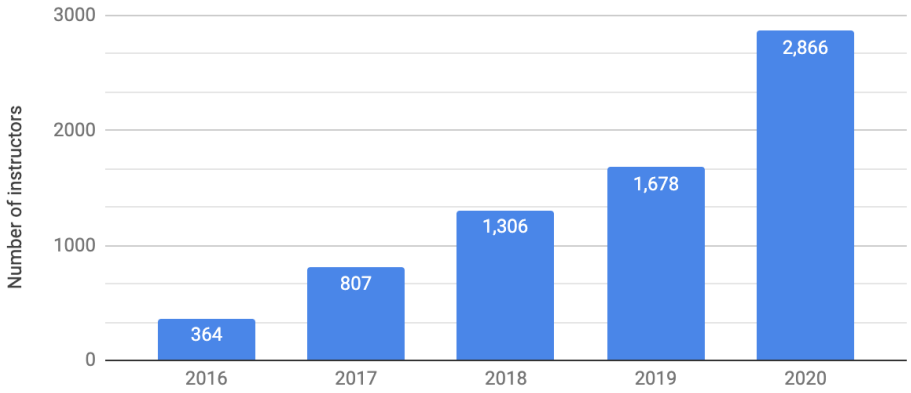

Figure 5: Instructors teaching courses using a zyBook with zyLabs enabled, per year.

Figure 6 shows the proportion of zyBooks in C, C++, Java, and Python courses that have enabled zyLabs, per year. The proportion has been increasing, such that now substantially more zyBooks have zyLabs enabled.

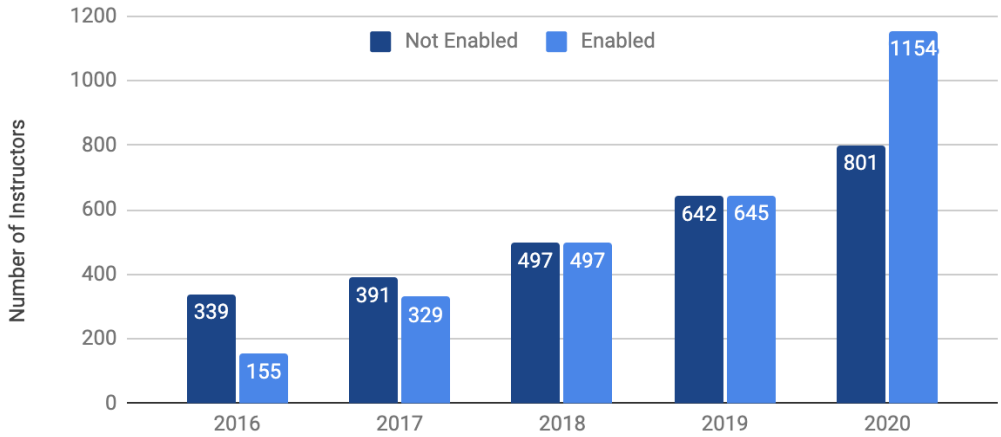

Figure 6: Instructors teaching C, C++, Java, Python courses using a zyBook with zyLabs enabled and not enabled, per year.

Usage

Figure 7 below shows the average number of zyLab programming assignments used in a course offering. We consider a zyLab as being "used" if at least 5 students submitted programs for grading. We also calculated the average number of zyLabs programming assignments used per student, and the numbers were the same as those used in a course offering.

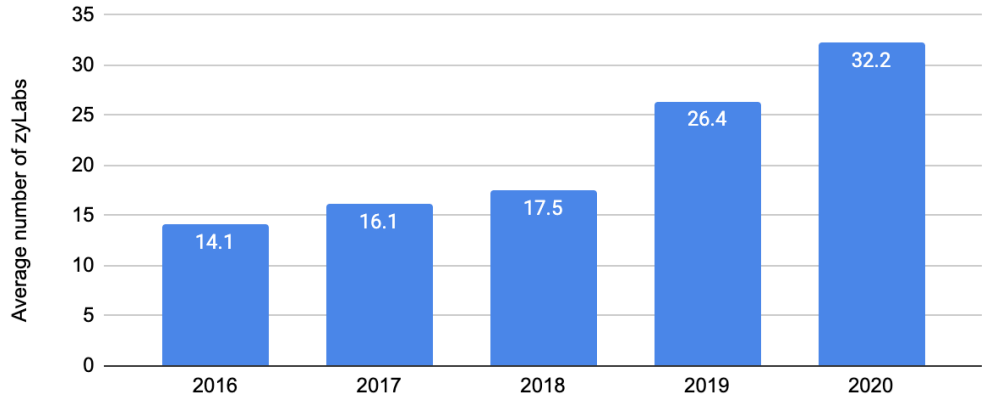

Figure 7: Average number of zyLab programming assignments used in a course offering. “Used” means that at least 5 students submitted programs for grading.

Figure 8 shows the distribution of the number of zyLab programming assignments that were used by instructors. Not only is the number of instructors using labs increasing each year, but the total number of labs that instructors use is also rising substantially. When zyBooks introduced zyBooks Maintained Labs (ZMLs) in 2019, there was a substantial increase in the number of instructors using 20+ labs, and even those using 100+ labs in a course.

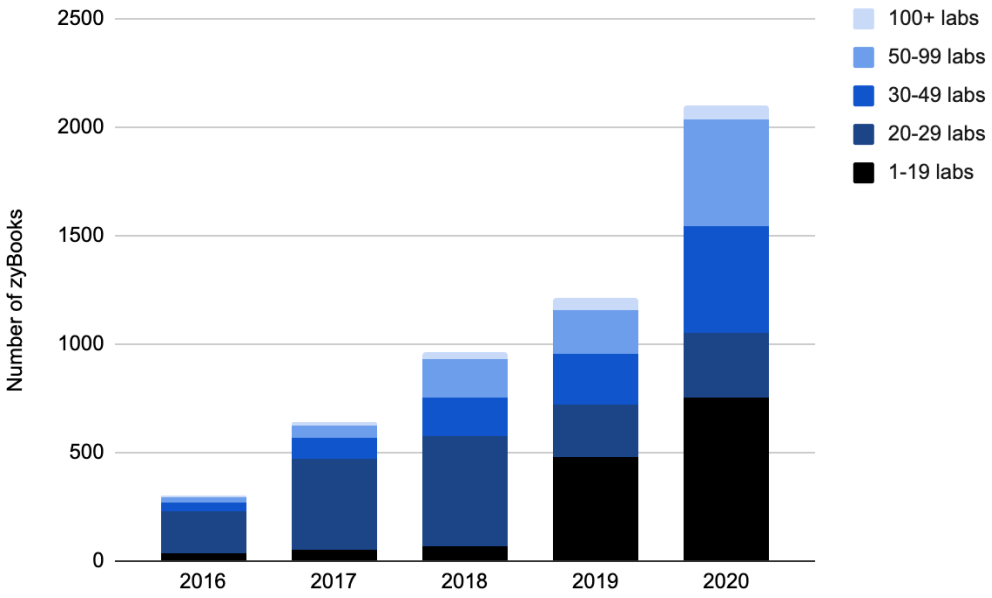

Figure 8: Distribution of approximate number of zyLabs used by instructors each year.

Figure 9 shows the number of submissions graded by the auto-grader per year. The figure shows that not only are the number of zyLab programming assignments per course increasing, but the usage of those assignments is increasing as well. 2020 saw a dramatic 2.4x increase in usage.

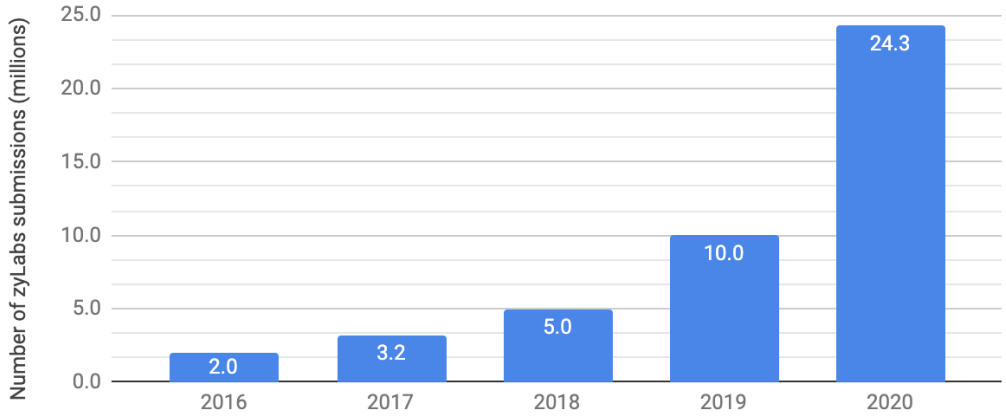

Figure 9: Total number of auto-graded zyLab submissions per year.

Figure 10 shows the distribution of the number of non-comment lines of code in the instructor solution to auto-graded assignments in one term, Fall 2020. The auto-grader has been used for assignments that range from 1 line of code to 780 lines of code in the instructor's solution, with a median of 24 lines of code. (Note: Some solutions might include template code provided to the student). One can see that the auto-grader can be used for small to large programs ("large" in the context of CS classes), with over 1,000 programming assignments being for programs with 100+ line solutions. Figure 11 zooms into the larger programming assignments, some of which have more than 400 lines of code.

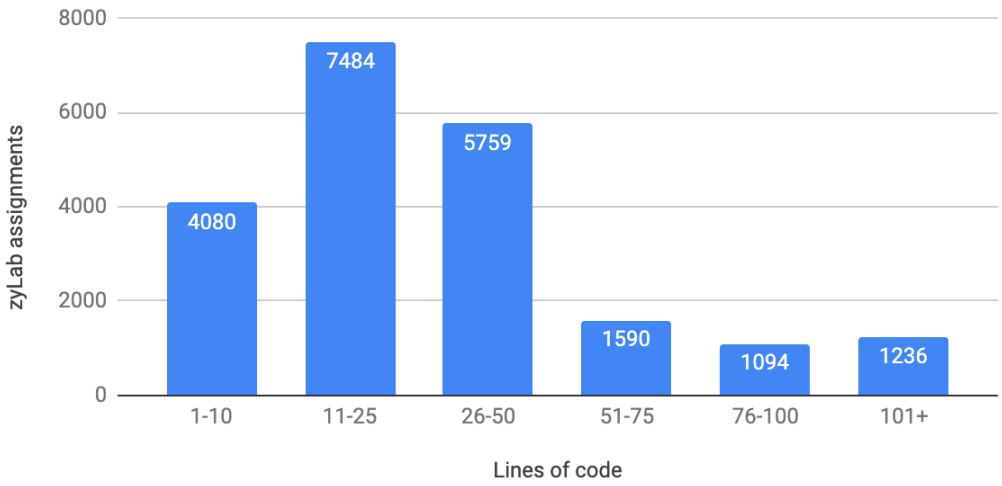

Figure 10: Distribution of instructor solution lengths for auto-graded assignments in Fall 2020.

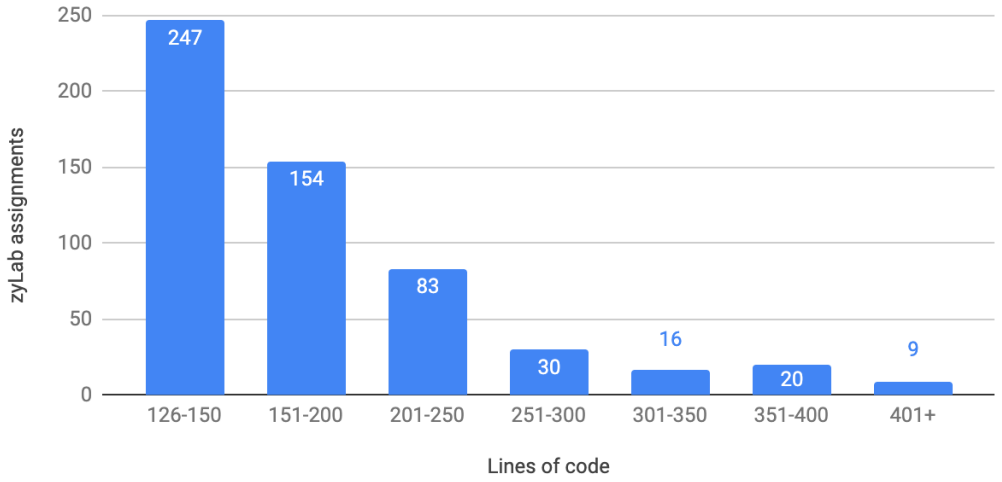

Figure 11: Distribution of instructor solution lengths for large auto-graded assignments in Fall 2020.

Survey data

To understand how the zyLabs auto-grader is changing the classroom, zyBooks surveyed instructors who used zyLabs during the Fall 2020 semester. A self-selecting sample of 116 instructors using zyLabs responded.

Figure 12 shows the number of instructors who used various methods of grading assignments prior to switching to zyLabs auto-grader. The large majority (79%) report grading assignments by hand, which can be very time-consuming.

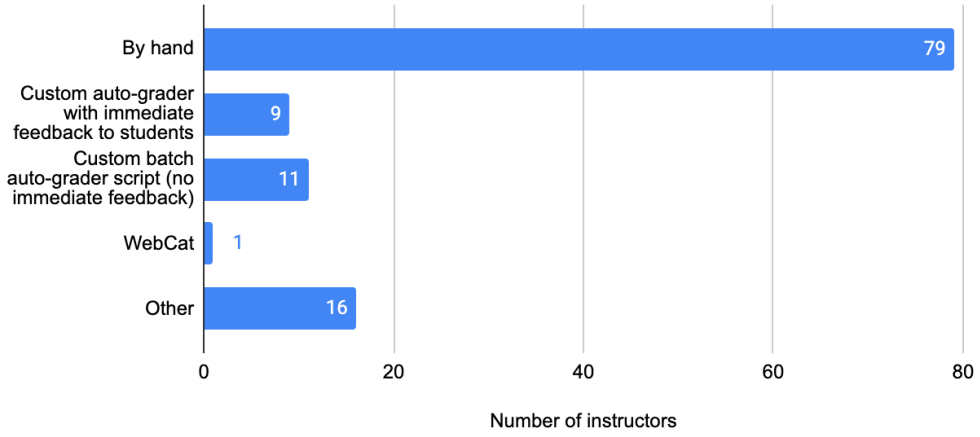

Figure 12: Number of instructors using various methods of grading prior to switching to zyLabs

Figure 13 shows the reported number of grading minutes saved *per student* each week after switching to zyLabs. Nearly all instructors reported spending less time grading; only three instructors (not shown) reported spending *more* time, ranging from 1 minute to 6 minutes more per student per week.

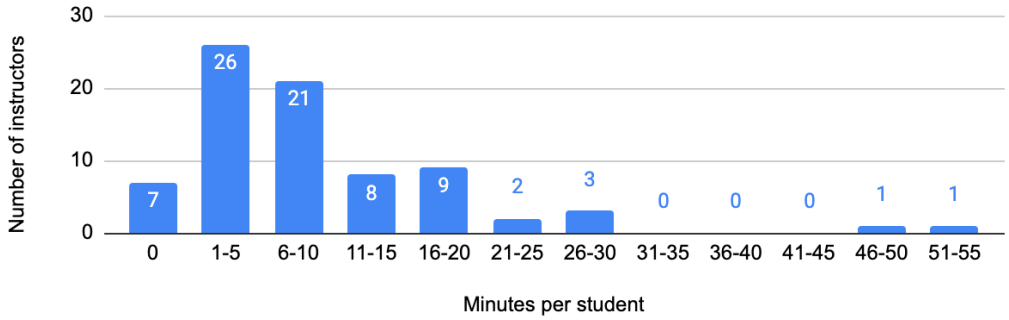

Figure 13: Reported number of grading minutes that instructors saved per student each week after switching to zyLabs.

Figure 14 shows the *total* number of hours that instructors saved per week on grading. Nearly half (48.7%) of reporting instructors said that they saved at least 5 hours per week using the zyLabs auto-grader, and over a quarter (26.3%) said that they saved 9 or more hours per week using zyLabs. The median reported grading time saved per week was 4.3 hours, and the mean was 9 hours saved per week.

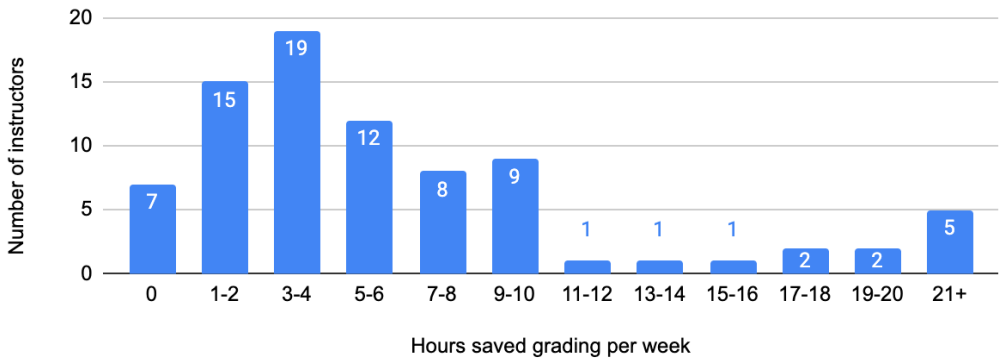

Figure 14: Reported number of grading hours saved each week after switching to zyLabs.

Figure 15 shows the change in hours per week that *students* spent on programming assignments after their course switched to using zyLabs. About half (53%) of instructors reported that students spend about the same amount of time working on programming assignments. About a third (35%) of instructors reported that students have spent more time on programming assignments, with 16% indicating their students spend 3+ additional hours per week on such assignments.

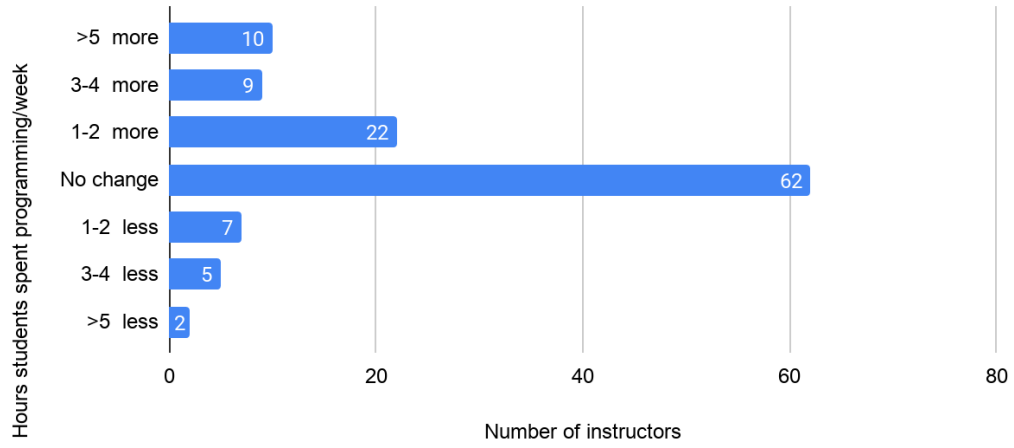

Figure 15: Reported change in time that students spent on programming assignments each week after switching to zyLabs

Discussion

Steep rise

The data above shows a steep rise in courses using the zyLabs auto-grader, growing from zero to over 2,000 course offerings in just a few years, used by over 130,000 students in 2020. In most cases, courses switched to zyLabs not from another auto-grader but rather from manual grading. The auto-grader is used for small and large

programs. This growth suggests substantial changes in the nature of programming courses:

- Previously, students would not get feedback for days or even weeks. But now students get immediate feedback, knowing what score they have earned, and can correct their code to earn a better score. This tight feedback loop can yield better learning, and also reduce disappointment due to a lower-than-expected grade with no opportunity for correction.
- Instructors are saving large amounts of time grading, averaging 9 hours per week, with some reporting 20+ hours saved. These reductions are quite large, with the average representing nearly 25% of a 40-hour work week. The large reduction frees instructors to spend time doing higher value teaching, including holding help sessions, answering questions, creating new activities, analyzing student code for common errors for class discussion, detecting and meeting with struggling / at-risk students, implementing class research experiments, or doing higher-value grading based on style or problem-solving approach. An instructor responding in the survey put it this way: "[zyLabs] has freed up a lot of my time so I can spend more time working with students." Instructors can also handle more students or teach more classes, and/or departments can assign fewer teaching assistants for the same number of students.

We note that part of the extra steep rise in 2020 in particular was due to the COVID-19 pandemic, where most courses were suddenly being taught online. Many instructors quickly adopted zyBooks that year, stating the increased need for better quality learning outside of class.

These changes illustrate a future with a different classroom dynamic. The role of instructors shifts away from grading and more towards educating. This shift may also shape the way that students see instructors, seeing them less as their testers/assessors, and more as their aids/educators.

Potential issues

Immediate feedback focuses on correctness rather than style or approach. As such, auto-graders might allow poor coding style/approach. This issue can be addressed by instructors complementing auto-grading with manual grading of at least some programs, which requires far less time than fully manual grading.

Thinking of how to test one's own code is an important part of creating good programs. Auto-grading may reduce students' focus on testing their own code, with some students over-relying on the auto-grader. This issue can be addressed by

future techniques that require students to create their own test cases before submitting for auto-grading, and even auto-grading the quality of those test cases. It is partially addressed today by the zyLab instructor-configurable options to meter submissions by requiring a minimum separation time between submissions (5 minutes is common but as high as 60 minutes has been observed) and/or by limiting the total number of allowed submissions.

Future directions

Commercial program auto-graders are relatively young. Looking forward, some improvements may include:

- Providing automated hints, especially for common mistakes. When surveyed, students' most common request in programming classes is often for "more help when stuck". In fact, some "tutorial" labs could be designed in a tutoring style, allowing students to try to develop alone, but then incrementally providing parts of a solution upon request.
- Logging a student's develop and submission runs, such that instructors could give credit not just for the final program but also for the effort along the way. zyLabs already provides "effort signatures" and lets instructors see the code for all develop and submission runs of each student. But more logging and more compact representations may be possible ahead.
- Detecting similar submissions, not only across a class, but across terms, or with solutions on the internet. zyLabs already provides a built-in similarity checker across a class.
- Auto-generating problems so that each student gets a unique problem, and/or so that students can get more practice.
- Using auto-graders not just for weekly programs but also for exams. zyLabs is already used in hundreds of courses for exams, and in the future may provide even more support.

Conclusions

Several commercial cloud-based auto-graders have been introduced in recent years. Our survey showed that auto-graders are saving instructors substantial time, averaging 9 hours saved per week, while also providing students with immediate feedback. The data presented in this case study for one of the most popular such auto-graders, zyLabs, suggests a rapid and somewhat dramatic shift in programming courses from manual grading to auto-grading. These insights may help instructors or departments considering whether to switch to using auto-graders. The prevalence of instructors shifting to auto-grading suggests the need for extensive new research on how to best use auto-graders in courses, especially to overcome stated opposition to auto-grading [9]. For example, recent research shows

the benefits of using weekly many-small-programs versus one-large-program in CS1 courses [10, 11], which is made possible by auto-graders. Other research directions include best practices for pair-programming with auto-graders, using auto-graders for different kinds of tasks (weekly programs, quizzes, in-class activities, etc.), requiring students to create test cases, auto-grading for code style and comments, techniques to encourage early starts or to decrease cheating, new experiences using auto-graders' built-in similarity checkers to reduce cheating [12], and much more.

References

[1] M. Sherman, S. Bassil, D. Lipman, N. Tuck, and F. Martin, "Impact of auto-grading on an introductory computing course," *Journal of Computing Sciences in Colleges*, vol. 28, no. 6, pp. 69-75, Jun 2013.

[2] R. Pettit, J. Homer, R. Gee, S. Mengel, and A. Starbuck. "An Empirical Study of Iterative Improvement in Programming Assignments." in *Proceedings of the 46th ACM Technical Symposium on Computer Science Education, SIGCSE*, pp. 410-415, Feb 24 2015.

[3] G. Haldeman, A. Tjang, M. Babeş-Vroman, S. Bartos, J. Shah, D. Yucht, and T.D. Nguyen, "Providing meaningful feedback for autograding of programming assignments," in *Proceedings of the 49th ACM Technical Symposium on Computer Science Education, SIGCSE*, pp. 278-283, Feb 21 2018.

[4] H. Keuning, J. Jeuring, and B. Heeren. "Towards a Systematic Review of Automated Feedback Generation for Programming Exercises," in *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education, ITiCSE '16*, pp. 41-46, Jul 2016.

[5] J. Moghadam, R.R. Choudhury, H. Yin, and A. Fox, "AutoStyle: Toward Coding Style Feedback at Scale," in *Proceedings of the Second (2015) ACM Conference on Learning @ Scale*, pp. 261-266, Mar 14, 2015.

[6] T. Daradoumis, J.M. Puig, M. Arguedas, and L.C. Liñan, "Analyzing students' perceptions to improve the design of an automated assessment tool in online distributed programming," *Computers & Education*, vol. 128, pp. 159-170, Jan 2019.

[7] I. Albluwi, "Plagiarism in Programming Assessments: A Systematic Review," *ACM Transactions on Computing Education, TOCE,* vol. 20, no. 1, pp. 1-28, Dec 2019

[8] The zyLabs program auto-grader. https://www.zybooks.com/catalog/zylabs-programming (accessed March 2021).

[9] T. Beaubouef and J. Mason, “Why the high attrition rate for computer science students: some thoughts and observations,” *ACM SIGCSE Bulletin*, vol. 27, no. 2, pp. 103-106, Jun 2005.

[10] J.M. Allen, F. Vahid, A. Edgcomb, K. Downey, and K. Miller, “An Analysis of Using Many Small Programs in CS1,” in *Proceedings of the 50th ACM Technical Symposium on Computer Science Education, SIGCSE*, pp. 585-591, Feb 22, 2019.

[11] J.M. Allen, F. Vahid, K. Downey, and A. Edgcomb, “Weekly Programs in a CS1 Class: Experiences with Auto-graded Many-small Programs (MSP),” in *2018 ASEE Annual Conference & Exposition*, Jun 23, 2018.

[12] K.W. Bowyer and L.O. Hall, “Experience using MOSS to detect cheating on programming assignments,” in *FIE'99 Frontiers in Education. 29th Annual Frontiers in Education Conference. Designing the Future of Science and Engineering Education. Conference Proceedings*, vol. 3, pp. 13B3-18, 1999.