2023 Annual Conference & Exposition
Baltimore Convention Center, MD | June 25 - 28, 2023
The Harbor of Engineering Education for 130 Years
ASEE
Paper ID #36739

BYOP: "Bring Your Own Project": How student-driven programming projects in an introductory programming course can drive engagement and continuous learning

Dr. Udayan Das, Saint Mary's College of California

Udayan Das is a computer science professor with over a decade of experience teaching computer science.

BYOP: "Bring Your Own Project"
How student-driven programming projects in an introductory programming course can drive engagement and continuous learning

Abstract

Engaging students who are unsure about Computer Science can be a challenge. A lot of introductory programming courses cap the coursework with a standard project that isn't always appealing to students. While good instruction results in students being able to complete the required projects, the projects may not drive enough student engagement as well as a desire for continued learning beyond the end of the course. Both strong student engagement and continued learning beyond the confines of the first programming course can impact student's persistence in computer science and related majors. For students from underrepresented minorities and women the impact of a first programming course can be significant. For individuals who do not fit perceived cultural stereotypes of the tech geek or programming nerd, getting through a first programming courses successfully and finding that computing is indeed for them can be huge boost as far as program persistence, completion, and professional success. Towards supporting these goals, an open ended "Bring Your Own Project" (BYOP) for the first programming course has been adopted for several years by this author. Apart from driving instant curiousity with the BYOP moniker, we find that students are more invested in their projects. Many students remain invested in their projects even after the completion of the course, and this allows some students to follow-up and continue development on their own. Additionally, an open-ended project, with scope control being performed by the instructor, allows for an early introduction to the software design process and a consideration of ethical issues that are inherent in technology. A wide variety of projects that inevitably result from this process, also give students in class exposure to a wide range of possibilities when it comes to programming and where programming can be applied, even at their early programmer level. Although this process is intensive and requires significant instructor time and was primarily done in classes of up to 43 students, the approach described can be scaled to larger classes through trained teaching assistants and how to approach this is discussed. The value of increased engagement, continued engagement and learning after the end of the course, and, confidence boost overall makes it well worth the effort. The openness of project topics has led to student creativity and expression in class projects, including the embracing of their unique identities and exploration of more advanced materials under instructor guidance. Projects that address a gender-specific, interest-specific, or queer concern also let students (the project makers and their classmates alike) understand that computing applies in many disparate domains and there is great value to a diversity of voices in technology. This paper describes the approach, general project design outline, the ethical reflection embedded in the project, and experiences from several years of teaching (since Fall 2017). A list of student projects with brief descriptions is included so other instructors can get ideas and inspiration, as well as a discussion of how to scale this approach to larger class sizes.

Introduction

Underrepresentation and retention of minorities and women remains a critical problem in computer science and computer science adjacent fields [1]–[5]. This is a critical issue for the future of our profession which is often masked by the huge demand for undergraduate and graduate computer science courses and programs. Although going into the intricacies of the issues involved and how to address them is out of the scope of this paper (please see [1], [4]–[6] for more) I present an approach here that has shown promise towards addressing some of these issues.

Project-based learning with courses that challenge students to develop critical thinking and creativity can significantly improve learning outcomes [7]. Applying project-based learning has shown success in many introductory programming courses [8], [9] towards driving engagement, improving critical thinking and creativity, and organically supporting collaborative learning. Project-based learning also shows promise towards addressing attrition [10].

First year engineering courses that are open-ended or provide choice to students have shown that they can drive engagement and student confidence, including driving student demand for those courses [11]–[13]. Open-ended projects are not the norm in CS1 courses, but there is no reason (other than logistical) why the impact seen in first year engineering courses would not be seen in first year programming courses. This paper presents an open-ended project-based learning approach titled Bring Your Own Project (BYOP). The approach is described in detail and handouts, a course outline, and example projects are included herein.

Bring Your Own Project (BYOP)

In Fall 2017, I gave students in a first programming course taught in Java the option of doing a project different than the standard course project. My experience teaching a first programming course over several years had been that the standard course project is something that students turn in because they have to but not something that students are excited about. I was attempting to generate some excitement about the final project. One option available to me, and to many other instructors, is of course to look at the ACM SIGCSE's Nifty Assignments database [14], [15], and I would strongly encourage all instructors for first programming courses to do so for assignment ideas in general. Replacing a standard project with a cooler project is an idea, however, I was interested in something that could be personalized to each student. Personalization can give a student a sense of ownership of the project, and drastically reduces the chances of plagiarism. Ultimately, I decided that meeting with each student to develop an individualized project would be the right approach. At the time, I was honestly not confident that I would be able to come up with topics for each student and so the standard project option remained as more of a backup for me rather than students. As it turned out, a majority of students opted to use the individualized approach (see Table 1). Although students at first were not necessarily any more excited about a heavy grade weight assignment, towards the end of the semester it was clear that students were much more excited about the projects when making the final presentations. Additionally, there was the added benefit of students who may have partially

lagged others in the class on certain class materials who caught up in the course of accomplishing their project.

Given the success in Fall 2017, since Fall 2017, I have standardized the approach and offer this approach as the final project in a CS 1 first programming class. (Over time, I have also developed the BYOP moniker.) Table 1 summarizes the numbers of students that have been taught using this approach since Fall 2017. Students have often stayed in touch with me even after the end of class to continue working on projects and begin new projects. Once a student sees that CS can be applied to a wide variety of scenarios, hesitation in tackling newer projects diminishes.

Table 1. Courses taught with this approach. (Total = 148)

Class	**Term**	**Institution**	**n**
CS 115 OO Programming I (Java)	Fall 2017 (in person)	Illinois Institute of Technology	32 * (out of 43)
COMP 170 Intro to OO Programming (Java)	Fall 2018 (online)	Loyola University Chicago	26
COMP 170 Intro to OO Programming (Java)	Spring 2019 (online)	Loyola University Chicago	12
COMP 170 Intro to OO Programming (Java)	Fall 2019 (online)	Loyola University Chicago	13
COMP 170 Intro to OO Programming (Java)	Spring 2020 (online)	Loyola University Chicago	15
COMP 170 Intro to OO Programming (Java)	Fall 2020 (online)	Loyola University Chicago	17
COMP 170 Intro to OO Programming (Java)	Spring 2021 (online)	Loyola University Chicago	12
CS 21 Programming I (Python)	Fall 2021 (in person)	Saint Mary's College of California	21

* In the Fall 2017 section, students were given an option between a standard project and a student-driven project.
**Online here refers to sections where there were synchronous sessions (or classes) held weekly. Both class meetings and 1-on-1 meetings would take place via Zoom.

BYOP: Bring Your Own Project

If you have a problem or task in your daily life, or if you have come across an interesting problem in your studies, then you can implement a solution after approval from Prof. Das.

Our objective is to demonstrate that you have developed an ability for Structured Problem Solving and that you are able to implement a solution on a computer using the Python programming language. The key learning from this project is to be able to look at programming and software development from conception through completion. Learning how to think about a real-life problem and mapping the relevant inputs and outputs to a computing context.

Requirements:

1. User interactivity: Your program must have some user interaction.
2. File manipulation: Your program should read-from and/or read-from files.
3. Practical application: You should be able to discuss the utility of your program (entertainment viz a viz games is fine)
4. Ethical reflection: You should consider the end user of the app and how they are affected. Depending on your project there might be other stakeholders to consider.

Deliverables:

You are required to turn in your source code, as well as relevant files (inputs and outputs), and a project report. Your report must discuss the problem under consideration in detail and lay out how you approached putting together a solution. It is recommended that you proceed as follows:

1. Discuss the problem.
2. Present solution approaches considered, and related research, if any.
3. Flowchart of the solution you selected.
4. Implementation of your flowchart, any issues you face with implementation and how you addressed those issues.
5. Discuss the testing process.
6. Discuss the results, your conclusions, and **future development**, if any.
7. Present any ethical considerations that may have arisen and how you dealt with them. Ex: if you are using sensitive data, then how did you anonymize it or protect it. Note that all projects have ethical considerations, you must think carefully. (Refer to the ethical reflection guide.)

We will be using the 1-on-1 sessions to discuss the project in more detail. There will be a final presentation in Class during the final week.

Grading: Implementation: 35; Report: 60 (Algorithm Description: 10; research and implementation description, testing, and conclusion: 50; ethical reflection: 15)

The Project

Students in the course are instructed on the general goals of the assignment. The main intent of the project is for students to be able to demonstrate that they can apply skills learnt towards the solving of a practical problem.

A handout similar to the one shown on the previous page is used. As can be seen these instructions are fairly open ended as far as project choices with a set of minimum requirements that must be met. The choice of what programming constructs to use is left to the student. Typically, most projects involve the use of key elements covered in the course such as branching, loops, and functions. Objects are introduced towards the end of the course but students are not expected to master objects until the end of the next course which is a Data Structures and Algorithms course. However, some students to end up using objects. Other students use objects implicitly through the use of Graphical User Interface (GUI) functionality or other library functionality.

A project example sheet similar to the one included in the next section is also shared with students. It is important that students get a sense of what projects are possible and that there is a wide variety of potential projects that they could pursue.

I meet with students twice. The first meeting is to finalize the project topic. This meeting is critical since this is the instructor's best opportunity to manage the scope of the project and make sure that the project that the student can complete in time for the final presentation. A later meeting is used to review the project design and assess the student's progress towards completion. During the final presentations I usually provide feedback on next steps based on a future work slide.

Table 2 summarizes the different milestones mapped to a 15-week semester. The expectation is that there is about 6 weeks total for the project, with a possible 4 weeks of actual development. In practice, the actual development time varies from student to student and varies from 1 week to 4 weeks. 2 weeks should be enough time for a student who has been making satisfactory progress through the course and if the project is scoped correctly; however, all instructors should bear in mind that the end of a semester is a stressful time for students with many heavyweight assignments coming due, some grace goes a long way. The timeline is open to an instructor's preference, however, the steps as shown are strongly suggested at least for the first go around. Instructors can make changes as they see fit in the future.

In a small number of cases, I have allowed students to work in pairs. One of the recommendations for future work is to allow pairs or groups of students.

An outline of what is part of the ethical reflection is included below. This reflection uses the Who/What/How/Why/Where methodology of asking questions which is reinforced in later coursework in a Tech ethics course.

Table 2. Suggested timeline for BYOP.

Week	Milestone	Goals	Comments
Week 6/7	Project announced	Get students thinking about project topic	Project can be announced before specific topics have been covered in class (ex: file I/O)
Week 8/9	Finalize Project topic meeting	Finalize project topic	**Important:** Clarify that some elements of a project are things students will learn to do over time and encourage students to continue working on projects beyond
Week 12/13	Project progress meeting	Assess project progress and design	Suggest changes if necessary. Watch for scope creep. Can scale back in exceptional circumstances. (TA escalate if needed.)
Week ?	Support session (Optional)	As needed	To address any issues. (These are particularly suited to a TA or peer tutor.)
Finals week	Project Presentation Project submission	Demo project Submit source code and report	The project presentation which demos the project can be before the final project submission

Ethical Reflection

For you project, consider each of the following and write in brief regarding each of these points of consideration:

- Who are the stakeholders for your project? (Note: it could be you, otherwise there is at least the user and the programmer.)
- What are the concerns of the stakeholders?
- How are the stakeholders and stakeholder groups impacted?
- Where in your project design or implementation are there opportunities to address concerns and potential harms?
- Why is it important to consider the ethics of what you are building?

Project Examples

The next page shows examples of projects completed by students in various sections since Fall 2017 with brief descriptions. Apart from these projects there are other projects that are similar to each other and which I categorize as below:

- Recommenders: Ex: Tea recommender, Boardgame recommender, Book recommender. Knitting and gem design recommenders. In each case there was a database of recommendations saved typically in a CSV file (though there was at least one occasion with an SQL DB) from which a selection was made based on results from a user questionnaire. The last two also required the incorporation of images and visual elements.
- Various games: Ex: Hangman, Minesweeper, Snake game, other word games, etc.
- Library like utilities: Virtual bookshelf (book list w genre, number of pages etc.), Music library (info only, artist, genre, instruments), game collection, etc.
- Various office automations: Reformatting files: student was seeking to automate the task of reformatting large spreadsheets that was a routine part of their job. Transacting with SQL databases handling customer or retail information. Analyzing video files for a security company based on metadata. Often office automation projects have proprietary components and care should be taken to enforce confidentiality. In turn, the ethical reflection helps to strengthen students understanding of confidentiality and intellectual property. Working on a project is understandably more time consuming as dummy data and data obfuscation may be needed.
- Directories of resources: LGBTQ resource centers. Health centers. Incorporating geographic information potentially sets up API concept teaching opportunities.

As can be seen there is a huge variety of projects that have emerged over the course of several years of teaching using BYOP. Many BYOP projects are such that they can be showcased beyond the classroom and an academic or professional context. Unlike many traditional course projects, these are ones that a student can also show to friends and family outside of the CS or CS-adjacent disciplines. This can be a further driver of engagement. As noted on the next page, some students also continue working on their projects after the end of the class, supporting continuous learning. Doing a non-standard project may also provide students with the confidence to tackle projects they have not encountered before which can also encourage further learning.

There are ample opportunities for instructor learning here as well. Consider for example, the project that attempted to estimate the runway direction. This is something that I did not know about, although in hindsight it made perfect sense. Another example was the tailoring project that gave me insight into how some base measurements (waist, chest, etc.) are used for templates for clothing. In many other situations, I end up learning something in more detail (puppy vaccination schedule, weight ratings; what is Magic the Gathering and how is it played). So even though this is more work on the part of the instructor, in my experience, the benefits far outweigh this added work.

Project Examples (a sheet like this is provided to students for inspiration)

- **DNA Codon Sequence Optimization**
  https://github.com/jmattick/CodonOptimization; http://codewithjess.com/

- **Boxr:** an app for keeping track of your moves; what item is in what box. Project began in Java and was later followed up as an Android app in a Mobile Development course.

- **Microwave Popcorn shutoff estimator (based on audio feed):** Although actually shutting off microwave was out of scope, the student used an audio recording to determine at what point the shutoff signal should be generated. Student was encouraged to try pursuing this project in Python and did so even though course was taught in Java.

- **Puppy growth tracker and vaccination schedule tracker:** Student had recently gotten a puppy and wanted to use programming to make a tracker. Later followed up as an Android app in Mobile Development course.

- **Gardener's pal:** seed starting calendar and guide.

- **Pregnancy tracker:** Student was pregnant and wanted to automate the tracking of dates and medical visits and medical care information.

- **Soccer league score tracker:** Scoring for a soccer league that a student played in.

- **Sports scheduling:** Scheduling dates and locations with a group of people.

- **Grocer's Dilemma**: making efficient aisle routes based on your shopping list

- **Magic the Gathering deck generator**: self-explanatory.

- **Calculating runway direction near airport:** Student lived near an airport and wanted to be able to know when to schedule BBQs. Apparently, runway direction is dependent on wind direction. Weather API was used to get the wind direction info [JSON] for Midway airport and runway direction estimated accordingly.

- **Tailoring measurements generator:** based on style, cut, and base sizes such as shoulder and waist for a Theater company.

- **Hot desk utilization evaluator:** Student worked for an office space company for which evaluating desk usage utilization was important.

- **Recursive org chart:** Student delved into more advanced Data Structures study.

- **Basketball analytics:** for a team for which student was assistant coach.

Scaling to larger courses and 2-year programs

Depending on the willingness of the instructor, this type of project can be administered in a class of up to 45-50 students. In classes larger than 30 students, the instructor may have to hand out the project sooner to make sure there is enough time for student meetings. (In my opinion, it would be unreasonable to expect a per-course adjunct to undertake this work without significant support in the form of a TA or other departmental resources.)

For large courses with 100+ students which are typical of large state universities, an instructor must lean on a team of TAs for course administration. To have a successful BYOP experience TAs must be effectively trained in managing scope and the willingness to escalate to the instructor if they are unsure of how to proceed. Graduate student TAs, which may not be typical of a first programming course, would be especially helpful in this context. Their practical experience and programming knowledge can be useful in the overall defining of the project, scoping, as well as progress assessment process.

In a departmental context, training a team of students to become project mentors can be beneficial to both the students in the first programming course and for the project mentors themselves. In theory, the team of project mentors need not necessarily be the regular TAs for the course, they can be mentors that students connect with for the project meetings, though note that TAs would be more familiar to the students and that can impact the degree of comfort that students have. The degree of comfort in turn will have a significant impact on how open students are with ideas and the success of the final projects.

Since one of the main drivers of choosing this approach is increased engagement, it is essential that we keep the needs of minority, non-cis-male, women, and others in mind. Students from underrepresented groups feel more supported especially when they interact with TAs who are like them.[16] Ensuring that TAs and TA teams are appropriately trained in Diversity, Equity, and Inclusion practices would greatly contribute to the success of this approach.

As noted in previous sections, being open to a student's specific interest can be implicitly supportive of a student's background and identity by validating CS as a tool to be applied towards whatever interest the student may have. This can also be a bit of an art, since some students may prefer to only come up with ideas that they think will work for a programming project but not others. Learning about the student, asking about their hobbies and interests can help the instructor or TA learn more about the student and help in the topic generation process.

Most students are good at generating ideas that can guide discussion, a small number of prompts for those that are unable to come up with any ideas should be prepared. Prompts should focus on student's interest and hobbies, pet peeves that they might want to address using a program and should encourage students to think beyond what they think they might be capable of.

Relative lack of experience may be the biggest challenge for a TA team. Formalized training and clear escalation chains is essential. On the other hand, TA's lack of experience can be a benefit

because they bring a wealth of experience and openness to ideas to buttress against the jadedness of some of us faculty.

BYOP can be invaluable in a community college and technical college context. Community college and technical college graduates typically start jobs with less training than bachelor's degree holders on average. The capstone experience can also be significantly different. BYOP represents the opportunity to add to the student's portfolio of projects. Smaller class sizes however must be balanced against heavy teaching loads for faculty. Developing project mentorship that enhances both the BYOP students and more advanced students experience may be one approach. Another approach may be to partner with 4-year colleges and universities. The value of the learning experience is considerable for the project mentors and the exposure to connections with the 4-year program participants can be useful in developing pathways for the BYOP students. In partnership programs care should be taken such that mentors are aware of the needs of community college students and adequately trained on Diversity, Equity, and Inclusion practices.

Discussion and conclusion

Even though students are not sure what to expect when the project is first announced, over the course of the semester students are usually excited about the final projects. It takes time for students to realize the value of this style of learning but usually by the second meeting students start to feel more interest and ownership regarding their projects. BYOP allows students to showcase their learning, both to their peers as well as people such as family and friends who may be outside the CS field.

BYOP also works at the level of changing the learning objects that are seen within a CS program [17], and could have an impact on the overall culture of a program. Research suggests that assignments and learning methods that can function as "equalizers" can have great value particularly when it comes to addressing the gender gap[3].

BYOP naturally brings out the creativity of individuals and showcases how creative the field is and how creative it can be. Early, somewhat informal, exposure to the design process and algorithm development process also shows students that the field is a lot more than simply programming and that programming simply allows practitioners to implement their solutions and designs. In turn, that can open students up to ideas of other roles to play within CS than simply "programmer." This can also be an opportunity to talk about the wide variety of non-tech industry jobs in which CS and CS-adjacent graduates work, significantly broadening the student's worldview as far as places to work.

BYOP can support a wide diversity of students by empowering them to create projects that interest them rather than something prescribed by us as instructors with our somewhat limited perspective. Cultural cues and stereotypes have been identified as demotivators for students of different backgrounds, especially women[17], [18]. While testing students for the fundamental skills they are expected to have at the end of the first programming course, BYOP can let them

express themselves and their various identities. Consider the example of the pregnancy tracker, it is unlikely that such a project would naturally emerge but for the fact that there was a pregnant student who wanted to apply their skills to addressing their own tracking needs. In the case of the tailoring project the individual who was of LGBTQIA background wanted to develop the application for their partner. A regular project would not have allowed them to make this subtle reference to their identity. Other examples are those of a knitting styles directory and gem design outlines. In larger courses many other interesting reflections of student's identity and creative expression would emerge. Towards our goal of supporting students from various backgrounds and embracing diversity, signaling early that learning communities can benefit from the wealth of experience and perspectives that various students can bring is vital. BYOP is one among various other approaches which when used in conjunction can help us diversify and improve our field.

The early emphasis on ethical considerations which is followed up throughout the curriculum at Saint Mary's College of California and Loyola University Chicago is another key element of establishing a culture that cares about the impact of our work on larger society. While the ethical considerations in individual may be limited, the training is in undergoing the process of reflection and understanding the ethical framework which students are encouraged to keep on applying beyond the end of the course.

The appeal of continuing a project beyond the completion of a course is a dream for me as an instructor. To a student it also communicates that project work is not meant to be a one-and-done but that the work can be going and we can continue to improve solutions over time, particularly in a learning environment enhancing projects as students learn more. I have referenced earlier the learning that I receive from the course in other domains which certainly makes BYOP a unique experience for me as an instructor.

BYOP encourages collaboration because the work products are different so students can become co-participants in learning. This goes well beyond the co-participation by students working together in a group. Students often share their knowledge and learning towards implementing their solutions without any risk of cheating or plagiarism. Understanding that in the completion of project connecting with others is a natural process of professional life is a vital early lesson.

Many BYOP projects require extra learning beyond what is covered in class (ex: SQL, understanding APIs, more file structures). The extra learning happens organically as a way to support a student's project and this is another early signal to the student practitioner that this type of during-the-project-learning is common in computing. This early lesson is invaluable to CS and tech learners for not only the rest of their schooling but the entirety of their professional life. The final presentations not only expose students to the varieties of situations in which computing can be applied but also the varieties of approaches towards solving a particular problem at hand.

The end product of individualized projects takes away the motive to copy and may in turn inform student behavior in the future. The student essentially enters into an agreement with the instructor as to what is required, there is no need to look to other students and their projects and

code for “better” grades. Generally speaking, I am a believer that high quality assessments dramatically reduce incidences of plagiarism and cheating. BYOP is an example of an assignment and assessment where cheating or plagiarism simply doesn’t make sense. I would encourage all our fellow faculty to consider revisiting the assessment process as a means of addressing academic integrity, along with communicating clearly during courses what the implications of cheating and plagiarism are on the cheating process, thus that ultimately the student is the loser.

BYOP has been successfully implemented in both in person and online courses (see Table 1) and there is no element of the methodology that requires being in person to achieve quality results. I do not have significant experience in online courses that do not have regular weekly “class” sessions so I cannot comment intelligently on how to implement BYOP in an asynchronous format. The connection between instructor and student is vital for the success of this approach.

For future work, the first thing to consider is group projects. Both pair programming[19], [20] and team-based learning[4] have been shown to impact student positive learning and can go a long way in addressing gaps in prior preparation. Although I use group activities throughout the first programming course, I have not incorporated it extensively in the BYOP assignment. Historically, this has been due to the personalization component. However, I think that there can be effective ways to extend the project to 2 or more people, particularly when class sizes are very large. Being careful about who is teamed up with who is critical in this context, and there can be the additional work of ensuring that all participants are contributing equally. It is also important to ensure that each participant is interested in the project topic. I have also not (yet) removed a final exam in the course, however, I think there is a case to be made for replacing the final exam altogether with BYOP.[21]

Though BYOP requires more investment by the instructor the benefits to this approach are considerable:

- Increased engagement
- Continued engagement beyond the first programming course
- Students strengthen their learning in material that they need to accomplish their project, thus “catching up” in a sense
- Students learn material not directly covered in class
- Underrepresented students can feel supported because projects they are introduced are validated as being worth doing
- Continuous learning beyond the course through continued development and follow-up in future classes
- Students can add a unique project to their portfolio
- Drastically reduces the chances of plagiarism and cheating
- Students learn about the wide variety of CS applications by being exposed to their peer projects
- Students gain the confidence to continue to stay in CS and become future professionals

- Students are introduced to thinking about a complete project design in their first course
- Students are introduced to thinking about all stakeholders and the impact of a solution on each stakeholder group
- Introduces students early to the perception that CS is a creative field
- Introduces them to self-directed learning and its value in CS
- Students learn to manage a project and manage a project timeline
- Reinforces that programming is a tool that allows practitioners to implement solutions and designs and is far from the end all and be all of CS
- Makes collaboration to learn from peers natural impacting overall learning

When students have more agency over the project, they are empowered to become owners of their learning process.

References

[1] S. B. Jenkins, "The Experiences of African American Male Computer Science Majors in Two Year Colleges," University of South Florida, 2019.

[2] L. J. Sax, H. B. Zimmerman, J. M. Blaney, B. Toven-Lindsey, and K. J. Lehman, "DIVERSIFYING UNDERGRADUATE COMPUTER SCIENCE: THE ROLE OF DEPARTMENT CHAIRS IN PROMOTING GENDER AND RACIAL DIVERSITY," *J. Women Minor. Sci. Eng.*, vol. 23, no. 2, pp. 101–119, 2017, doi: 10.1615/JWomenMinorScienEng.2017017221.

[3] R. Benbow and E. Vivyan, "Gender and Belonging in Undergraduate Computer Science: A Comparative Case Study of Student Experiences in Gateway Courses," University of Wisconsin, 2016.

[4] C. Stephenson *et al.*, *Retention in Computer Science Undergraduate Programs in the U.S.: Data Challenges and Promising Interventions*. New York, NY, USA: ACM, 2018. doi: 10.1145/3406772.

[5] A. O'Connor, "Women's Persistence in Computer Science:".

[6] R. M. Powell, "Improving the persistence of first-year undergraduate women in computer science," in *Proceedings of the 39th SIGCSE technical symposium on Computer science education*, Portland OR USA, Mar. 2008, pp. 518–522. doi: 10.1145/1352135.1352308.

[7] L.-J. Shannon, "Developing Project Based Learning, Integrated Courses from Two Different Colleges at an Institution of Higher Education: An Overview of the Processes, Challenges, and Lessons Learned," 2016.

[8] M. Frydenberg and K. Mentzer, "From Engagement to Empowerment: Project-Based Learning in Python Coding Courses," 2021.

[9] F. Ling and S. Gong, "Research on Project-Based Learning Python Programming Course," *Front. Comput. Intell. Syst.*, vol. 1, no. 2, pp. 79–82, Oct. 2022, doi: 10.54097/fcis.v1i2.1884.

[10] O. Solarte Pabón and L. Machuca Villegas, “Fostering Motivation and Improving Student Performance in an introductory programming course: An Integrated Teaching Approach,” *Rev. EIA*, vol. 16, no. 31, pp. 65–76, Jan. 2019, doi: 10.24050/reia.v16i31.1230.

[11] K. Dunnigan, J. Bringardner, and G. Georgi, “From Pre-Defined to Open-Ended Projects: Evaluating First-Year Ability to Innovate and Problem Solve,” in *2019 ASEE Annual Conference & Exposition Proceedings*, Tampa, Florida, Jun. 2019, p. 32866. doi: 10.18260/1-2--32866.

[12] T. Shepard, “Implementing First-Year Design Projects with the Power of Choice,” in *2013 ASEE Annual Conference & Exposition Proceedings*, Atlanta, Georgia, Jun. 2013, p. 23.708.1-23.708.14. doi: 10.18260/1-2--19722.

[13] L. Meadows, R. Fowler, and E. Hildinger, “Empowering Students with Choice in the First Year,” in *2012 ASEE Annual Conference & Exposition Proceedings*, San Antonio, Texas, Jun. 2012, p. 25.524.1-25.524.19. doi: 10.18260/1-2--21282.

[14] “ACM SIGCSE.” https://sigcse.org/ (accessed Feb. 13, 2023).

[15] “Nifty Assignments.” http://nifty.stanford.edu/ (accessed Feb. 13, 2023).

[16] R. Varma, “Making computer science minority-friendly,” *Commun. ACM*, vol. 49, no. 2, pp. 129–134, Feb. 2006, doi: 10.1145/1113034.1113041.

[17] S. Cheryan, V. C. Plaut, P. G. Davies, and C. M. Steele, “Ambient belonging: How stereotypical cues impact gender participation in computer science.,” *J. Pers. Soc. Psychol.*, vol. 97, no. 6, pp. 1045–1060, 2009, doi: 10.1037/a0016239.

[18] S. Cheryan, A. Master, and A. N. Meltzoff, “Cultural stereotypes as gatekeepers: increasing girls’ interest in computer science and engineering by diversifying stereotypes,” *Front. Psychol.*, vol. 6, 2015, Accessed: Feb. 13, 2023. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fpsyg.2015.00049

[19] C. Mcdowell, L. Werner, H. E. Bullock, and J. Fernald, “The impact of pair programming on student performance, perception and persistence,” in *25th International Conference on Software Engineering, 2003. Proceedings.*, Portland, OR, USA, 2003, pp. 602–607. doi: 10.1109/ICSE.2003.1201243.

[20] J. Brougham, S. Freeman, and B. Jaeger, “Pair Programming: More Learning And Less Anxiety In A First Programming Course,” in *2003 Annual Conference Proceedings*, Nashville, Tennessee, Jun. 2003, p. 8.912.1-8.912.9. doi: 10.18260/1-2--11728.

[21] R. Whalen, S. Freeman, and B. Jaeger, “Get With The Program: Integrated Project Instead Of A Comprehensive Final Exam In A First Programming Course,” in *2005 Annual Conference Proceedings*, Portland, Oregon, Jun. 2005, p. 10.663.1-10.663.17. doi: 10.18260/1-2--14179.