Community Action Computing: A Data-centric CS0 Course

Ayaan M. Kazerouni  
ayaank@calpoly.edu  
California Polytechnic State University  
San Luis Obispo, California, USA

Jane Lehr  
jlehr@calpoly.edu  
California Polytechnic State University  
San Luis Obispo, California, USA

Zoë Wood  
zwood@calpoly.edu  
California Polytechnic State University  
San Luis Obispo, California, USA

ABSTRACT
A student’s sense of belonging in computing can be positively impacted when coursework can authentically be connected to real community contexts. We describe the design, materials, and preliminary evaluation of an introductory programming (CS0) course infused with a focus on societal responsibility and relevance. We take a data-centric, constructionist approach to introductory computing. Data-centricity allows us to authentically connect coursework with students’ communal and societal interests, and students’ motivation was enhanced given that they were creating and sharing artifacts as part of their coursework. Students used TypeScript to manipulate and analyze real data-sets, and created shareable websites containing statistics, data visualizations, and reflections based on the data-set of their choosing. Students chose varied topics for their assignments—they worked with data about access to CS education, climate change, and data provided by local non-profit organizations. A preliminary evaluation indicated that students who took this CS0 course attained CS-specific learning objectives equally well in the two subsequent follow-on courses as students who took alternative CS0 courses at our University. We close with instructor perspectives and reflections on lessons learned.

CCS CONCEPTS
• Social and professional topics → Model curricula; • Information systems → Web applications; • Human-centered computing → Visualization.

KEYWORDS
CS0, Socially Responsible Computing, Web development

ACM Reference Format:

1 INTRODUCTION
This work strives to address several tensions in introductory computing courses. In particular, broadening participation in computing is a pressing problem for all CS educators and there is good evidence that integrating real world contexts into introductory computing is beneficial [10, 23, 37], especially for students who have been historically under-served in computing. Additionally, the ACM Code of Ethics and Professional Conduct challenges us to prepare students to fulfill the mandate: “A computing professional should contribute to society and to human well-being, acknowledging that all people are stakeholders in computing”. Yet introductory computing requires a substantial amount of new knowledge that students must work through when learning to program prior to being able to contribute to typical computing applications for social good. With these tensions in mind, this curricular initiative presents the design and implementation of a CS0 course aiming to retain students, especially those from historically marginalized communities, by introducing them to the technical skills needed to become CS majors while working on computing applications that broadly can be considered to benefit society.

Another factor in our course design was a desire to address the needs we perceived in our community that manifested in the all too frequent emails from our local citizens and organizations asking for volunteer programming assistance. Each week our department newsletter frequently includes listings for local non-profits asking for computer science students to help with their web application. However, a new student’s ability to contribute to this type of opportunity is limited due to the complexity of web development and frequently students are only able to contribute to this kind of project later in their major.

At its heart, the goal of this curricular initiative is to broaden participation in computing. Ultimately our objective was to design a CS0 course to enable students to see how even their nascent programming skills could be applied to real-world problems, contexts, and community organizations. Our design goals are rooted in literature demonstrating the value of promoting student sense of belonging and persistence in computing by enhancing their perception of computing’s potential to benefit society.

This curricular initiative paper describes the design, implementation, and evaluation of our CS0 course as taught in its first year (2022–2023 academic year). Section 2 describes the research that drove us to create the course and Section 3 discusses our design goals. Following this, Section 4 gives an overview of the course and the context in which it was taught. Finally, we evaluate students’ attainment of CS learning objectives by studying their performance and persistence in two follow-on CS courses (Section 5), and conclude with instructors’ reflections and insight on the course (Section 6). We cannot report on effects on persistence in the major, since not enough time has passed since the course’s inception. A future research paper about these impacts will be forthcoming.

1https://www.acm.org/code-of-ethics
2 BACKGROUND AND MOTIVATION
A student’s sense of belonging [21] in an academic discipline is associated with improved academic performance, motivation, and persistence [5, 15, 20, 29, 31]. In computing, however, women are less likely than men to report a high sense of belonging [12, 31], and Black and Asian students are less likely than White students to report a high sense of belonging [29, 31]. Hispanic/Latino students have reported a lower sense of belonging in college environments than White students [26, 38]. Targeted interventions to improve sense of belonging are therefore a worthwhile endeavor.

Sense of belonging (and, as a result, persistence) can also be affected by a student’s perception of their discipline as having communal goal affordances [4, 29]. However, computing is perceived to afford lower opportunity to meet these kinds of goals than other STEM fields like the life sciences or physical sciences [4]. This incongruence can negatively impact the sense of belonging for students with stronger communal goal orientations [29], i.e., students that are more drawn to goals that further the betterment of society or their communities. Women and Hispanic/Latino, Black, and Asian students are more likely than men and White students to hold communal goal orientations [2, 4, 6, 29, 33].

Understanding and attending to the relationship between students’ values and interests and computing could have a positive impact on the engagement and retention of students from historically marginalized backgrounds [1, 13, 36]. Tissenbaum et al. [40] and Bart et al. [3] have advocated for the use of personally meaningful real-world contexts to help motivate students learning introductory computing, and to help them build a computational identity. For example, the incorporation of service learning to engage students has been shown to have positive academic and personal impacts [7–9]. Exposure to practical projects demonstrating how computing can positively impact society has proven to be a key factor driving women’s interest in computing [11, 22].

These studies suggest that clear signaling of computing’s ability to benefit communal goals could have positive impacts on the recruitment and retention of students from backgrounds that are historically under-represented in computing. Our course is informed by these related works. We expand further on the research that informs our design goals in Section 3.

3 DESIGN GOALS
Our intended audience for this course is first-year students who have little-to-no prior experience with programming. This includes both computing majors (Computer Science, Software Engineering, or Computer Engineering) and non-computing majors (often, since it is required for their program, Graphic Design majors). Our design goals are threefold. First, we built upon what we know about effective introductory programming education. Second, keeping in mind our over-arching goal for developing this course, we included a sustained focus on social responsibility and relevance. Finally, we prioritized the building and sharing of computational artifacts; this informed our choice of programming platform (the Web).

3.1 Introductory CS education tenets
The role of data in computing. We provided explicit instruction on data types, including compound types, and their centrality to problem solving. We included exercises for students to reason about the data definitions needed to solve programming problems [16].

Also in service of this goal, we chose to work with a statically typed programming language—TypeScript, a statically-typed superset of JavaScript—which helped emphasize to students the relationship between data and computation. Studies have shown that using a statically-typed language has a positive impact on developer productivity when the programming task involves new or unfamiliar APIs, and when fixing type-related errors [14, 17, 30].

Expressions and evaluation. We wanted students to build an abstract understanding of the program execution environment (i.e., the notional machine) they would be using. This helps the learner construct a robust mental model of how programs will execute, and improves their ability to apply this knowledge to new problems or contexts [19]. Therefore, before working with any programming, we delivered roughly 1.5 weeks of instruction and exercises about data, expressions, and evaluation, building on what the students knew from algebra. We frequently returned to this language and set of examples throughout the term.

Reading before writing. Fuller et al. [18] and Xie et al. [43] have argued that reading programs is a distinct and pre-requisite skill to writing programs. For each computing topic introduced (e.g., data types, variables, functions, control flow), we first attended to code comprehension skills through code tracing exercises before moving on to exercises that involved writing or modifying programs (e.g., programming problems or Parsons problems). We also included an emphasis on patterns for problem solving (e.g., map, filter, and reduce). These were introduced first in the imperative style (as usage patterns for the for…of loop) and then using the map, filter and reduce higher-order functions in TypeScript.

Data-centricity. Traditional introductory programming courses tend to work with programming problems based on artificial data in the form of numbers, array, or strings [39]. This can lead students—particularly non-computing majors or those without prior exposure to computing—to feel that the course is inauthentic or irrelevant to their interests. Therefore, we designed our course to be “data-centric” [28]—nearly all programming assignments involved manipulation of real data-sets. Data-sets were obtained from various sources, including the CORGIS repository [3], local non-profit organizations, and data about secondary CS enrollments in California [27]. This focus also provided a vehicle through which to engage our next goal of infusing socially relevant and responsible computing into all aspects of the course.

3.2 Social responsibility and relevance
With our audience in mind and in particular considering the importance of supporting a diversifying student population, we embraced suggestions from prior research on the importance of focusing our curriculum on the ways computing can connect with and benefit society. We chose programming assignments which used community data and focused on observations that could be made from the data, including variance based on demographics, allowing for opportunities to reflect on historical injustices, such as access to computing education based on county economics.

Our goals were for students to:
• Engage with code that has a purpose
• Work on programs and coursework that were personally or
societally meaningful by focusing on community data

This means that for the two primary offerings of this course, our coursework centered around the use of 1) educational data related to CS education access in California schools, including both regional data and student demographic data, 2) local non-profit data shared with permission, 3) various sources of communal data from CORGIS [3], and 4) proportionally-representational population data for assignments related to demands for public housing in California, including student-designed waiting list criteria such as income, family size, occupation, disability status, etc. More details appear in the course description (Section 4).

3.3 Prioritizing building and sharing

We made a number of technological choices that prioritized empowering students to design and create shareable artifacts. We are driven by tenets of constructionism [24], a constructivist learning theory that suggests that knowledge construction occurs best when the student creates tangible, shareable objects.

We culminated the course with a final project in which students created shareable websites in which their TypeScript programs would run. The Web is a ubiquitous computing platform (students see and interact with web applications in their daily lives) and there is clear potential for web applications to be shared publicly. Our choice of TypeScript as the course’s programming language was partially driven by this goal.

The Web and our goal of data-centricity (Section 3.1) worked together particularly felicitously. Students used Vega-lite [35] to create web-based data visualizations based on the data-sets being used in each assignment. Vega-lite provides a declarative JSON syntax that is simple enough for these students to use (low floor), but powerful enough to where one can create remarkably expressive visualizations (high ceiling).

The programming aspects of the course began with students creating standalone Vega-lite figures using the Vega online editor.2 This was a useful vehicle through which students could engage with the affordances of different types of data (quantitative, ordinal, temporal, and nominal). By the end of the course, when their TypeScript skills were sufficiently strong, the students were able to manipulate, analyze, and prepare data-sets using TypeScript, create figures using Vega-lite, and embed them in publishable webpages.

Of course, all of this comes with the clear challenge that web development typically requires a complex development environment. For example, students were writing TypeScript code that needed to be compiled to JavaScript before it could be executed or included in an HTML page. Properly setting up all of these pieces included in an HTML page. Properly setting up all of these pieces needed to be compiled to JavaScript before it could be executed or

3In the Java object-oriented sense. Here, object refers to a dictionary of key-value pairs, and interface refers to the TypeScript construct for specifying the “shape” of an object.
Table 1: An overview of the 10-week course, as taught to computing majors in the Fall 2022 term (Winter term included the alternative assignments as listed in the paper). ★ denotes a societally or personally (to the student) meaningful context.

<table>
<thead>
<tr>
<th>Week</th>
<th>Topic</th>
<th>Major assessments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ACM Code of Ethics Introduction to data</td>
<td>Read the ACM Code of Ethics and respond to a reflection prompt ★ Identify the types of data used in a figure or problem (quantitative, nominal, or ordinal) Use Vega-lite to visualize data provided by the local cat shelter and housing data obtained from CORGIS [3] ★</td>
</tr>
<tr>
<td>2</td>
<td>HTML and CSS fundamentals Expressions and evaluation</td>
<td>Create a styled webpage with self-help materials for first-time college students ★ Evaluate the given compound numerical expressions</td>
</tr>
<tr>
<td>3</td>
<td>Statements and expressions in TypeScript Variables and data types Arrays</td>
<td>Declare and initialize variables with types for the given (string, number, or boolean) expressions Given arrays containing data about K-12 CS offerings in counties in California, compute statistics and answer questions about which counties can offer the least or most CS courses ★</td>
</tr>
<tr>
<td>4</td>
<td>Functions and control flow</td>
<td>Code tracing exercises Write functions to answer parameterized questions about CS education access using the data-set from the previous assignment ★</td>
</tr>
<tr>
<td>5</td>
<td>Loops and loop patterns (imperative map, filter, and reduce)</td>
<td>The Rainfall problem</td>
</tr>
<tr>
<td>6</td>
<td>Compound data (objects and interfaces) 4</td>
<td>Given a richer data-set about CS education enrollments in California, declare an interface to represent individual records ★ Write functions to answer questions about girls’ enrollments in CS courses in secondary school ★</td>
</tr>
<tr>
<td>7</td>
<td>Functions as values</td>
<td>Code tracing exercises Use the in-built higher-order functions map, filter, and reduce to answer questions about the data-set from the previous assignment ★</td>
</tr>
<tr>
<td>8</td>
<td>TypeScript in a webpage</td>
<td>Given a still richer data-set about CS education in California—now including data about race—use Vega-lite to create figures and embed them in a website; respond to reflection prompts about your figures and analysis ★</td>
</tr>
<tr>
<td>9</td>
<td>Review</td>
<td>No new assessments</td>
</tr>
<tr>
<td>10</td>
<td>Final project (in groups)</td>
<td>In consultation with the instructor, choose a data-set and use what you have learned so far (Vega-lite, HTML, CSS, TypeScript) to create a website containing your insights and reflections on your chosen topic ★ Present your report to the rest of the class ★</td>
</tr>
</tbody>
</table>

Following this, we discussed data types and their affordances, supporting the discussion with figures created using Vega-lite. For instance, using data provided by the local cat shelter, students created scatter plots to study the age at which male and female cats arrived at the shelter (working with nominal and temporal data). Then, after a brief introduction to HTML, CSS, and the Web, students created 2–3-page websites containing tips for secondary-level students preparing to apply to and move to post-secondary school (“college” in the US).

The first TypeScript content was introduced in Week 3. By this time, we had established a shared vocabulary and understanding for “data types”, “abstractions”, “statements”, and “expressions” to which we returned throughout the quarter. For the first TypeScript assignment, students were given data about the percentage of secondary schools in each California county that offer any CS courses, and were asked to compute statistics about CS education access in California. The data is tracked by a coalition of stakeholders interested in expanding access to CS education in California [27] and was provided by authors of the cited report.

In the Fall term, when the course was mostly taken by declared computing majors, most programming assignments adhered to this over-arching context of CS education access. As their TypeScript knowledge and skills progressed, students were given progressively richer versions of the dataset. After the first assignment with only two data fields (county name and percentage of schools with CS), we moved on to compound objects with additional fields for gender representation in CS courses, and finally to more detailed records describing CS enrollments at the intersections of race and gender.

Students’ increasing familiarity with this context and data-set enabled us to have rich in-class discussions about representational disparities involved in secondary-level CS education, reasons for these disparities as discussed by Wang et al. [41], and consequences for representation in CS courses at the post-secondary level and in the workforce. Since most students were in-state students, they also explored the state of CS education access in their hometowns compared to the statewide average.

In the Winter term, when the course was taken by non-computing majors, students did not engage as deeply with the context of CS education access. After a class poll to solicit student input, this section explored more varied contexts in their programming assignments—two topics commonly preferred by students were housing equity
and food access. A significant multi-week project involved the discussion and implementation of an algorithm to rank applicants for public housing in which the students chose ranking criteria, reflected on those criteria with at least three community members outside the course, implemented an algorithm, and then reflected on which applicants were served and not well served by their choices. An entire lab period was dedicated to a class discussion on the power a developer has in making these algorithmic choices and the impacts on equity for a community. For example, one criterion that students debated is an applicant’s criminal record, which was discussed in the context of fairness and community impacts.

**Final project.** In all offerings, the course culminated with a final project in which students chose a data-set in consultation with the instructor, and created a web-based report containing visual and quantitative analyses about their chosen data-set. After creating and turning in their websites, students presented their findings to the rest of the class in the final week of the quarter.

Data-sets were either made available by course instructors (e.g., for data-sets provided by local non-profits) or obtained from CORGIS [3]. Some examples of the topics students explored include: local beach cleanup events (e.g., amount of trash and recycling collected, age ranges of participants), provided by a local non-profit organization; SAT exam attempts and scores and its relation with household income levels; drug use and age of drug users; fatal police shootings, including demographic variance; among others.

In conducting their analyses, students reflected on knowledge they gained about the varied contexts they were studying. For example, students expressed surprise at the large population of California, the relative dominance of the Soviet and US space programs in the history of space travel, inequities involved in the SAT exam, the lack of access to CS education in many California counties, and the ubiquitous use of concrete in construction despite its environmental cost. In this way, while students were learning to code, students were also “coding to learn” about societally relevant contexts that were interesting to them [34].

## 5 PRELIMINARY EVALUATION

Our course was offered for the first time in the Fall 2022 academic term, taught by one of the authors of this paper. The students were nearly all CS or SE majors. As discussed in previous sections, our goal with this initiative was to incorporate elements of socially responsible and socially relevant computing into CS0, while still achieving the course’s CS learning goals and preparing the students to take the subsequent required CS courses. We therefore investigate our students’ attainment of CS learning objectives by studying their performance and persistence in follow-on courses.

Every year, our department offers a number of thematic “flavors” of CS0, e.g., computational art, robotics, and now the present course, community-action computing. All courses are taught in sections of 30–35 students each. In the Fall 2022 term, five such sections were offered, one of which was the CS0 being described in this paper. We compared the performance of students who took our CS0 course (treatment group, n = 32) with that of students who took other CS0 courses in the same academic term (control group, n = 133). Note that rates of grade attainment or withdrawals are more volatile for the treatment group, since that group encompasses one section, while the control group encompasses four sections.

**Demographics.** The computer science first-time first year student population for the year of this work comprised 28% female students and 21% students identified by our University as having an identity from an underrepresented minority (“URM”).

We studied students’ performance in two follow-on courses that are required by computing majors: Fundamentals of Computer Science (“CS1”) and Data Structures (“CS2”). Both courses were taught using Python by instructors un-affiliated with our CS0 course and this paper. Data about grade attainment in CS0 and performance in follow-on courses is summarized in Table 2. Not all students from CS0 went on to take CS1 and CS2. This can happen for a number of reasons. For example, a total of 7 students (2 in the treatment group and 5 in the control group) skipped CS1 and went directly to CS2 after completing CS0.

**Grade attainment in follow-on courses.** For this analysis, we converted nominal course grades into “grade points” using the mapping that our university uses to compute GPAs. As can be

---

5 Our University uses the term “URM” to designate students “whose race/ethnicity is Hispanic/Latino/a/x, African American, Native American, Hawaiian/Pacific Islander, or multi-racial with at least one of those four ethnicities.”

---

### Table 2: For CS0, CS1, and CS2, the number of students, median grade attained, number of failing grades received, and the number of withdrawals. Data is present for the control group (C) and the treatment group (T).

<table>
<thead>
<tr>
<th>Course</th>
<th>CS0</th>
<th>CS1</th>
<th>CS2</th>
</tr>
</thead>
<tbody>
<tr>
<td># Students</td>
<td>C</td>
<td>133</td>
<td>125</td>
</tr>
<tr>
<td>T</td>
<td>32</td>
<td>29</td>
<td>23</td>
</tr>
<tr>
<td>Median grade</td>
<td>C</td>
<td>A</td>
<td>B+</td>
</tr>
<tr>
<td>T</td>
<td>A</td>
<td>A–</td>
<td>B+</td>
</tr>
<tr>
<td># Failing Grades</td>
<td>C</td>
<td>4 (3%)</td>
<td>18 (14.4%)</td>
</tr>
<tr>
<td>T</td>
<td>1 (3.1%)</td>
<td>4 (13.7%)</td>
<td>0 (0%)</td>
</tr>
<tr>
<td># Withdrawals</td>
<td>C</td>
<td>0</td>
<td>1 (0.8%)</td>
</tr>
<tr>
<td>T</td>
<td>1 (3.1%)</td>
<td>1 (3.4%)</td>
<td>0 (0%)</td>
</tr>
</tbody>
</table>
seen in Figure 1, students generally performed well in all flavors of CS0—the median score was a 4 out of 4.

In the CS1 course, students from the treatment CS0 course performed slightly better on average than students from the control CS0 course. The median student in the treatment group scored a 3.7 in CS1 (A–), while the median student in the control group scored a 3.3 in CS1 (B+). A Mann Whitney U test indicated that this difference was not statistically significant ($U = 2105.5, p = 0.83$).

In the CS2 course, the median student in both the treatment and control groups achieved a grade of 3.3 (B+).

**Withdrawal and failure rates in follow-on courses.** In addition to grades, we consider the number and rate of students who failed or withdrew from follow-on courses. A failing grade is one that would require the student to re-take the course before moving on in the major (D+ or lower). Table 2 shows that failure and withdrawal rates were largely similar between our new CS0 course and other CS0 courses taken in the same term. This is a positive result considering that a previous (anonymized) longitudinal study [42] suggests that our menu of CS0 courses are generally beneficial in the long term with regards to students’ GPAs and persistence.

**Outcomes for the non-major section.** We also consider CS0 grade attainment for non-computing majors who took the course in the Winter 2023 term. We cannot report on follow-on courses for this section because these students are not required to take the follow-on computing courses. For this section, 94% of students received a grade of D or better (2 students withdrew from the course), with 58% achieving A grades. This is a similar success rate to the prior four offerings of an alternative CS0, computational art, for the same non-major population which had an average of 95%.

### 6 INSIGHTS AND CONCLUSION

In this paper, we have described the goals and implementation of a CS0 course focused on socially responsible computing, aimed at students with little-to-no prior programming experience. Preliminary results suggest that students’ attainment of CS learning objectives was not negatively impacted by the sustained focus on socially relevant contexts. While we cannot yet report on student belonging and persistence in the major (since not enough time has passed), overall students in our CS0 performed equally well in follow-on computing courses as students who took other CS0 courses in the same term. Non-computing majors who do not take a follow-on course also performed equally well in the treatment CS0 compared to non-major students in other CS0 courses.

In general, both instructors found that students were energetic, engaged with the material, and expressive. The ability for students to choose a context and data-set and to exercise some aesthetic choice in graphing (particularly for the graphic design majors in the Winter term) allowed for a degree of self-expression that is often missing in introductory programming courses. Speaking from 20 years of experience teaching novice programming, the author who taught the Winter section notes that students were deeply engaged and excited to work through labs to produce results from the data to reflect upon (notably the "coding to learn" moments were frequent and engaging for students).

While in these first offerings, students mostly experienced an insulated context of presenting findings to other students in the class, our data-centric community focus has opened doors for possible collaborations with local non-profit organizations in future offerings. For example, members from the cat shelter that provided a data-set for the course expressed an interest in having students present their findings to shelter organizers. Though we were not able to organize this in the first offering of the course, it is a goal in a future iteration, enabling students to experience direct communal impacts through their coursework.

We made a few atypical technological choices for this course, notably the Web as a platform and TypeScript as a programming language. Though TypeScript is not a pedagogically designed language, features like its static type system and ability to be embedded in a web-page meant that it was more appropriate than other commonly used introductory languages like, say, Python. Additionally, pre-configured development environments in Replit meant that much of the complexity of web development was hidden from students. (Though learning about the different pieces involved in a web application is a worthwhile eventual objective, it is not a learning objective for this CS0 course.) As students generally performed well in our course and in subsequent (Python) courses (Section 5), these choices do not appear to have had particularly pernicious effects, and helped us meet our stated design goals (Section 3). We also found that students in general did not express any concerns about whether the course was “real” CS or used a “real” programming language, as often occurs in other CS0 settings. In fact, several upper division computing students such as the course learning assistants and those working with the local non-profits expressed that they wished they had been able to take a CS0 like the current course due to the value of learning some web development early.

That said, end-of-term evaluations suggested that some students found the course pace challenging. Both computing majors in the Fall term and non-computing majors in the Winter term mentioned that the course moved through topics quickly. So, although CS0 grades and outcomes in follow-on courses were satisfactory, students felt challenged enough in the course to specifically mention it in their comments. We plan to pare down some content in future iterations, particularly in terms when the course is mostly taken by non-computing majors.

Overall, we feel this curricular initiative was successful in its first year and look forward to teaching it again and to future evaluations of student outcomes.

### ACKNOWLEDGMENTS

This work was supported by National Science Foundation award 2216687. We thank Dr. J. Clements for help in retrieving data about follow-on courses, and the anonymous peer reviewers for their feedback.

### REFERENCES


