IUSE-CUE: A Collaboratively Designed, Interdisciplinary Introduction to Computational Thinking in Data Science

Brendan Henrique, University of California, Berkeley

Brendan Henrique is a Ph.D. Student at the Berkeley School of Education who studies Computer Science Education. He received his M.A. in Urban Education from Loyola Marymount University and his B.A. in Cognitive Science from the University of California, Berkeley.

Lisa Yan, University of California, Berkeley

Lisa Yan is an Assistant Teaching Professor of Electrical Engineering and Computer Sciences at the University of California, Berkeley. She is a computer science and data science educator and researcher with expertise in data analysis of student learning in large courses, computing ethics education, and teaching assistant education. She received her Bachelor of Science degree in Electrical Engineering and Computer Sciences from UC Berkeley and her Master of Science and PhD degrees in Electrical Engineering from Stanford University.

Fernanda Guadalupe Cano Low, University of California, Berkeley
Frida Arriaga, University of California, Berkeley

IUSE-CUE: A collaboratively designed, interdisciplinary introduction to computational thinking in data science

Introduction

While core frameworks outline the content and skills for undergraduate data science curricula [1], [2], [3], efforts are still emerging to design introductory, interdisciplinary courses that foster broad data fluency [4], [5], [6]. This NSF IUSE-CUE project brings together multiple institutions to collaboratively develop and implement one such course informed by computational social science. Course designers and instructors are faculty across multiple disciplines—data science, statistics, computer science—and multiple institutions—large public universities, small private institutions as well as community colleges. The resulting course, Computational Thinking with Data and Society (CTDS), fosters student skills in *computational (CS) and social science (SS) thinking via a data science (DS)* approach with real-world applications. CTDS has been taught to over 300 students nationwide—across five institutions with varying class size, resources, and credit hours.

There have been major efforts to establish curricula and guidelines for undergraduate DS. [4] found that universities differ greatly in how they balance inference and modeling in introductory DS courses; they propose de-emphasizing statistical knowledge to include more computational learning. Both [2] and [3] argue for an interdisciplinary model that goes beyond an intersection of computer science and statistics. The NSF CISE report, Harnessing the Computational and Social Sciences to Solve Critical Societal [7] recommends exposing CS students to SS topics and vice versa—social science students to computational topics. Students from disadvantaged backgrounds can experience barriers to succeeding in STEM coursework [8], and this problem is particularly accurate for social science students in computing [9]. By framing data science within liberal arts education [5] or biology [6], a wide range of students can develop more nuanced perspectives on DS as intersecting multiple disciplines and multiple facets of society.

Introductory DS can therefore both facilitate interdisciplinary learning and broaden participation in computing and related STEM fields. As these courses emerge, course instructors and designers must consider: what curricula to include, what are the challenges that students might face in interdisciplinary learning, and what epistemelogical gaps exist between qualitatively oriented SS and quantitative DS. In this paper, we share our findings in the context of two participating institutions' Fall 2024 offerings of CTDS. We first conduct a curricular review of the two course offerings. Second, we analyze student learning and engagement by focusing on one implementation of the course at a large public university.

Phase 1: Curricular Review

For the first goal of defining and describing the core content of the versions of the course, we revisited every lecture, lab and assignment in two different offerings of CTDS. At both institutions, CTDS is a 4-unit undergraduate course that integrates computational thinking, computer programming and data science to examine social contexts and can act as an introduction for non-major students. The course is intended as a social-science-oriented precursor to advanced data science coursework. The course was organized around four

four-week modules: Fundamentals of Data Science, Data Visualizations, Control & Iteration, and Emerging Topics in DS (currently, Data and Privacy or Computational Social Science). Students had four class-hour meetings per week, though the proportion of between active learning and lecture varied by institution. Assignments included weekly programming labs, five homeworks, one major project, and one comprehensive final exam.

We focus on how the course effectively provides students with foundational programming concepts in preparation for future CS course work while simultaneously engaging students with SS lessons to target a myriad of DS applications. At the project's current stage, many lessons and assignments have been created and taught over the course's iterations to different cohorts of students. Due to the diversity of curricular materials developed, we began a curriculum and skill mapping project to catalogue and formalize what students are learning in each lecture, lab and discussion section. Alongside the core DS content, SS skills and content areas were also recorded to juxtapose how the course interweaved disciplines.

For the purposes of this paper, in Table 1 we share our review of the first introductory module of each course. We recorded CS and DS content and skill areas ("Students will be able to manipulate the columns of a table via Python") as well as SS content ("Students will be able to identify how public health scientists use data science". Once all content and skills were catalogued per lesson and assignment, a curriculum crosswalk was conducted to discover core similarities and differences of the pedagogy and curriculum across the two versions. This process helps us review and improve all course materials, identify the specific content areas for every aspect of the course, and generate public-facing, modular versions of the course that can be readily shared to instructors nationwide. Through this analysis of content areas and skills, we identified three core conceptual areas:

1. **Computing Science Concepts:** Introduction to Python Programming, Functions, Iteration, Control Structures
2. **Data Science:** Arrays, Tabular Data, Table Methods, NumPy
3. **Social Science and Data Sensemaking:** Making sense of the social/societal implications of data science and engaging in social science thinking, in concert with computational thinking.

Combined, the three areas support students in gaining introductory programming skills, fostering their data science understanding, and engaging in social science contexts across each of the four course modules. The course can then simultaneously serve two functions: to prepare students for more advanced DS or CS coursework, and to give SS students a background in computational approaches for their field.

Phase 2: Student learning, engagement and outcomes

We next examine the design and implementation of one CTDS course iteration in Fall 2024 at a large public R1 institution in California to study how students engaged across the disciplinary topics (Table 2). As such, we only report student demographic data from the Fall 2024 implementation of the course.

Table 1: Fundamentals of Data Science Module

	Title	**Data Science and Computing Content**	**Soc. Science Content**
Lecture 1	What is Data Science	This lecture introduces the course and data science (DS) as a field.	What are the social implications of DS and computing?
Disc. 1	Python Order of Operations	The discussion covers Python arithmetic and a review of mathematical order of operations.	What constitutes a college ranking system?
Lab 1	Jupyter Notebook	Students gain experience with Jupyter notebooks, python arithmetic, naming and assignment.	N/A
Lecture 2	Python and Jupyter Notebook	Lecture presents Python functionality through expressions, operators, and error messages	N/A
HW1	Introduction to Python via Public Health	Students write Python arithmetic expressions and explore data types, boolean expressions and functions, markdown language and assignment.	How do DS and public health intersect? What is the incidence vs. crude rate of a disease?
Disc. 2	Traffic Data and Measurement	Students explore traffic data and write code using variables and built in functions.	Affordances and constraints of different data presentations.
Lab 2	Variables and Python	Students practice using python division, data types, and built in functions. They explore variables, assignment and naming conventions.	N/A
Lecture 3	Evaluation, Names, Data Types	Students are introduced to functions, expressions, values and assignments. Naming conventions, data types and casting are covered.	N/A
Lecture 4	Working with Data: Arrays, NumPy, Indexes	Arrays are introduced and students are shown array and NumPy functions. Indexing and dot notation are introduced.	N/A
HW 2	Arrays and Table fundamentals	Students work with array functions, table attributes as well as data cleaning and table creation. They use weighted averages, frequency tables and population estimates.	Public health scientists use DS to analyze population estimates, and health disparities.
Disc. 3	Array Operations & Table Methods	Students practice using NumPy functionality with arrays, array operations, and table methods.	N/A
Lab 3	Print, Arrays, and Tables	Students program table and array manipulations and use table methods to query a table.	N/A
Lecture 5	Cause and Effect	This lecture covers the structure of tabular data and the types of variables and units of analysis that data scientists interact with.	The role of DS on social science research and the privacy implications of large data sets.

Table 2: Student demographics of one CTDS Fall 2024 offering (50 students).

Gender		Race*		First Gen		Major	
Male	51%	Asian	55%	Yes	31%	DS/CS**	29%
Female	31%	Hispanic / Latino	14%	No	55%	Non- major	71%
None/Other	18%	White	8%	Declined	16%		

**Race < 3% are not listed as to protect student privacy* ***DS/CS = Data Science/Computer Science*

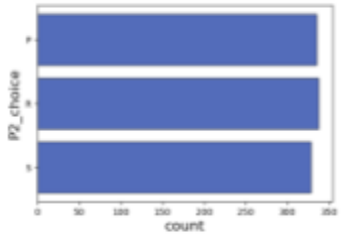
In this second phase, we critically evaluate the three core content areas in this course implementation to identify gaps in student learning that should be addressed in subsequent project activities. We collected anonymized course data from assignments—Jupyter notebook programming labs, homeworks and projects, and the final exam—and survey data from weekly course surveys and course evaluations. We present our analysis of concepts from two course modules: Data Visualizations and Control & Iteration. For each module, we first report summative student performance on specific final exam questions (Figure 1), tagged by CS concept. We then contextualize how students encountered concepts in formative assessments like homework, programming lab assignments, and class discussion. The latter helps us evaluate the efficacy of our curriculum in achieving CS and DS learning outcomes.

Figure 1: Summative final exam questions from CTDS Fall 2024.

Write an expression that creates a **horizontal bar chart**, where there is one bar for each type of choice ("R", "P", "S"), and the length of the bar represents the number of times that P2 made that choice.

Example solution:

```
tbl.group("P2_choice").barh("P2_choice")
```



Question A: Data Visualizations

Fill in the implementation of **draw_cooler_triangle**, which prints out a center-aligned triangle with height h, Each layer of the triangle increases by 2 instead of 1.

```
def draw_cooler_triangle(n):
    """
    >>> draw_cooler_triangle(1)
    *
    >>> draw_cooler_triangle(2)
     *
    ***
    """
    for layer in np.arange(1, h+1) :
        line = ""
        for i in np.arange(1, h+layer) :
            if i <= n - layer:
                line = line + " "
            else:
                line = line + "*"
        print(line)
```

Question B: Control & Iteration

In the Data Visualization module, students use Jupyter Notebooks and Python to create and analyze common visualizations: histograms, bar charts, line plots, and scatter plots. The Fall 2024 final exam had four visualization questions: three assessed conceptual understanding, and one assessed code-writing (Figure 1, Question A). On the final exam, students performed well on conceptual visualization questions (average 8.17/9 ≈ 91%; SD = 1.61) but comparatively worse on Question A (average 3.07/4 ≈ 77%, SD = 1.31). In Question A, students had to first group categorical data and visualize results with a horizontal bar chart. A curriculum review reveals that during formative assessment, students did well on both types of questions; however, targeted practice with horizontal bar charts was comparatively limited to a small number of exercises—1 lab, 1 homework, and 1 discussion worksheet problem. In its current form, the curriculum may not provide enough coverage of visualization coding in formative assessments, possibly contributing to lower performance on the coding exam question.

In the Control & Iteration module, students write conditional statements and for-loops in Python. On the Final 2024 exam, students averaged 11.98/18 ≈ 67% (SD = 5.63) on simple iteration questions involving single for-loops. By contrast, students performed slightly better on simple control-structure questions involving only if- and if-else statements, averaging 9.76/14 ≈ 70%

(SD = 3.20). Question B in Figure 1 combines both concepts and assesses a *nested* loop; on average, students scored just 2.46/6 ≈ 41% (SD = 1.98) on this question. Upon reviewing the curriculum in the course module, we found that the concept of nested loops was never explicitly covered. Assignments also included extensive "skeleton" code (i.e., instructor-written structural code or comments), and students were seldom asked to integrate both iteration and control structures. Exam Question B was at a much higher difficulty than the practice students had, suggesting a mismatch from instructor expectations.

Discussion and Future Work

Future research activities include using a learning sciences perspective to explore shifts in both student learning and instructor pedagogy. We plan to continue our research with two types of observations: classroom observations and instructor meetings. Through these observations, we hope to understand how the curriculum and course content is being implemented in a real classroom setting, what instructional strategies course instructors employ for interdisciplinary teaching, and whether instructor delivery aligns with the intended curricular design. In particular, our Phase 2 analysis revealed a mismatch between instructor expectations and student practice of core ideas. In our next set of project activities, we engage more with instructors to onboard them to this new curriculum. Engaging in observations of cross-institutional instructor planning meetings will help the project team gain insight into the instructional planning process. At present, we have conducted multiple observations of live-classroom teaching of the Fall 2025 CTDS offering at the R1 public university; these data will help us evaluate learning outcomes in the SS core conceptual area of the CTDS curriculum, which are most present in reading and in-class peer discussion and are minimally assessed on the CS-heavy final exam.

The two-phase analysis presented in this paper will continue to inform and improve the project's ongoing curriculum development. Findings from this work have already been shared with curriculum designers, who are readjusting the balance of how concepts are practiced across lab assignments, homework, and class discussion. The project team is also discussing how SS topics should be introduced, and what should be included in the fourth module, Emerging DS topics—in particular, how to expose contemporary ideas like Large Language Models.

In this paper, we have shared early findings on how the CTDS curriculum teaches an interdisciplinary blend of CS, DS, and SS concepts. Future work will evaluate how the CTDS course broadens participation in DS by improving student self-efficacy and fluency in CS and DS. We plan to perform an intersectional analysis of final course grades with course demographics. We have also developed a survey instrument to measure student self-efficacy before and after CTDS. At the core of this project is a dedication to developing an inclusive and interdisciplinary approach to teaching and learning data science by centering social science research methods. The curricular and outcome analysis in this paper has furthered this goal by prompting critical self evaluation and reflection on the curriculum while also informing the field on the challenges and promises of teaching data science through its intersection with other fields.

Acknowledgments
This material is based upon work supported by the National Science Foundation under Grant No. (NSF #2245877, #2245878, #2245879).