2021

The Impacts of Integrating Interdisciplinary, Introductory Computer Science in High School Courses

Jan Roddy

THE IMPACTS OF INTEGRATING INTERDISCIPLINARY, INTRODUCTORY COMPUTER SCIENCE IN HIGH SCHOOL COURSES

By

Jan Roddy

Bachelor of Arts, The University of Montana, Missoula, MT, 2015

Thesis

presented in partial fulfillment of the requirements
for the degree of

Master of Science
in Computer Science

The University of Montana
Missoula, MT

Autumn 2021

Approved by:

Ashby Kinch Ph.D., Dean
Graduate School

Yolanda Reimer Ph.D., Chair
Computer Science

Patricia Duce M.S.
Computer Science

Georgia Cobbs Ph.D.
Education

Roddy, Jan, M.S., November 2021 Computer Science

The Impacts of Integrating Interdisciplinary, Introductory Computer Science in High School Courses

Chairperson: Yolanda Reimer

Broadening the participation in Computer Science (CS) education is widely recognized as necessary to prepare students for the future however, in Montana, most High School students are not exposed to CS. In collaboration with High School teachers in Math, Science, and Business this project integrates interdisciplinary programming in non-CS courses. This preliminary study investigates the effects of interdisciplinary programming lessons on student interest in programming, basic learning of CS concepts, and learning of the non-CS material. Pre and post-lesson surveys were administered to the 132 student participants collecting quantitative and qualitative data. The results suggest learning gains in both CS and non-CS material post intervention, but student interest in programming did not change significantly. The results of this study provide motivation and can facilitate further analysis of using CS in high school courses to benefit student learning.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1 INTRODUCTION

Broadening the participation in Computer Science (CS) education is widely recognized as necessary to prepare students for the future [1, 2, 3]. Computational Thinking (CT), or "thinking like a computer scientist," is a foundational skill set that teaches creative problem solving and critical thinking [4]. Additionally, computer programming is relevant for nearly every profession and academic field [1, 2, 4]. Thus, providing more opportunities to expose students to CS and programming is essential to equip students with computational competencies across disciplines and for the workforce. However, most Montana high school students are never exposed to CS Education and lack a basic understanding of the CS field [1, 5]. Notably, low population and rural schools are even more affected by the lack of opportunity to study CS [5]. In Montana, only 36% of high schools teach a foundational computer science course [5]. Computer Science is an elective course for most students and does not always count towards graduation credit which means many students do not study CS even if it is offered to them [1, 5, 6]. In high school, many students first explore disciplines in-depth, consider future careers, and prepare for college which makes high school students an excellent target for the efforts of broadening participation in CS [2].

Even though the importance of CS education is widely recognized, there are considerable barriers and competing priorities to CS education to contend with. For one, high school curriculums are already overburdened with required standards. In addition, there are not enough teachers trained in teaching CS to fulfill the need, so not every school has the teaching expertise to allow for CS-specific courses [1, 5, 7]. With all of this in mind, how can we introduce students to programming who might not otherwise take a full year-long computer science course?; and what are teachers doing that could be modified and extended to include Computational Thinking and programming?

1.1 Overview and Motivation

In this paper, I will present a framework for interdisciplinary, introductory CS curriculum creation, instruction, and analysis. As one method, among many, of combating the lack of CS education for students, we propose injecting small doses of interdisciplinary CS into non-CS courses. In collaboration with teachers across Montana, we are working to integrate Python programming lessons into existing courses of various disciplines. By covering computational thinking in a class's existing curriculum, students can interactively reinforce important concepts and gain exposure to how computing can be used to solve problems in various fields. Through problem decomposition, discussion, and code examples students can engage with difficult concepts in a new way and discover the relevance of computing to their interests and strengths. A study regarding why students with an aptitude for CS, defined as taking advanced mathematics courses, do not major in CS found that 98% of students did not have a good grasp of what CS is [6]. Of the students in the study who were interested in studying CS, interest in applying CS to a different field was one of the top reasons cited [6].

There are no shortages of introductory CS curricula but this proposed model is different because of the individualized lessons, teacher support, and incorporation of current high school standards. The buy-in from high school teachers is also substantial because they reach many students in a variety of disciplines (some of which are core and required subjects). By integrating programming lessons into required and elective courses, more students are exposed to CS and the potential of programming to transform their world by solving complex problems that they care about. This project provides curricular materials, relevant resources, and lessons that teachers can tailor to their own needs. With the support of researchers at the University of Montana, teachers can connect CT and programming to learning goals already set by state standards and demonstrate the role of CT in non-CS disciplines. Along with developing the materials for this project, a preliminary research study regarding interdisciplinary CS as an introduction to the field was executed. The research study was motivated and designed according to the following questions.

1.2 Research Questions

The three following research questions motivate the data collection in this preliminary exploration into the impacts of the proposed interdisciplinary programming lessons developed for the study.

1. Does high school students' self-reported interest in programming increase after exposure to this study's interdisciplinary programming lessons?

2. Can students learn basic computational concepts through this study's interdisciplinary instruction?

3. Does including programming in Science, Math, and Business high school courses outside aid in student understanding of the host course material?

1.3 Hypothesis

The study's cross-disciplinary programming implemented in the 5 pilot high school courses in Math, Science, and Business will increase student's self reported interest in programming. Additionally, the student surveys will demonstrate student learning increases in the CS content questions and the core course material.

CHAPTER 2 LITERATURE REVIEW

This review of prior work on Computer Science Education begins by exploring what the CS Education community determines what Computer Science for K-12 is and the impacts of studying CS on student learning. Next interdisciplinary programming studies findings are summarized and reviewed to identify best practices and useful lessons. Then, students' reflections of programming the classroom during a prior research study are presented. Finally, the suggested benefits of programming as an active learning activity with code "notebooks," are presented.

2.1 Computational Thinking

There has been a significant amount of work and thought regarding the importance of Computational Thinking (CT) in K-12 education [1, 3, 4]. With the goal of incorporating CT and programming into high school student courses, the first step is identifying what defines CT. Computational thinking is a key 21st-century competence for developing a specific set of critical thinking skills [3, 7]. CT is often characterized by different concepts and specific practices for problem-solving. Some commonly regarded CT concepts and practices are listed in the table 2.1.

Computer programming is a natural and engaging vehicle for learning CT concepts and practices [8]. The concepts and practices listed in Table 2.1 inform the study and are incorporated in the programming modules created for the project. For instance, methods for solving math or science problems with programming can be broken down into steps to demonstrate problem decomposition and incremental development. When discussing formulas and procedures, algorithm design and logic are fitting topics to practice "thinking like a computer scientist" [1]. After the algorithms are

CT Concepts [4, 7]	CT Practices [7]
logic and logical thinking	problem decomposition
algorithm design	testing and debugging
pattern recognition	incremental development
evaluation	creativity
automation	
abstraction	

Table 2.1: A compiled list of CT Concepts and Practices examples.

discussed automation and abstraction can be explored with concepts such as functions, variables, and control structures. Computational Thinking can complement and enhance different areas of study [7]. For instance, programming in math and science is established and intuitive [4, 7]. It has been shown that extending coursework to include CT content enables deeper STEM learning [4, 7]. Another benefit of introducing programming to students in math and science courses is providing students a context for programming that they are already familiar with and have experience studying [4, 7].

2.2 Interdisciplinary CS

There have been many important efforts and research studies conducted on the viability of interdisciplinary CS. A few of the more pertinent efforts to this study are outlined regarding how they inform and motivate this study.

2.2.1 Math and Programming Can Reinforce Each Other

In one research study, Fisler et al. [8] regard how math and programming can reinforce each other and shares a few best practices regarding interdisciplinary programming. The study is a six-year reflection on an evolving curriculum for integrating computer science into mathematics by leveraging programming to help students explore mathematics [8]. The main lessons for integration

of the two disciplines include knowing your audience (teachers and students), teaching for knowledge transfer by explicitly linking concepts in CS and math, and adapting to the teaching practices and styles of the host class with flexible materials [8]. Additionally, Fisler et al. [8] suggests that relating CS to a teacher's interest in integrating computing into their home discipline allows for more success of the efforts [8]. This research project is motivated by the findings of this study by relying on the partner high school teachers who know their student and class dynamics to tailor the lessons for the audience. By working with partner teachers who have an interest in integrating computing into their discipline, we can focus on the material that the teachers want to link with CS concepts. Additionally, the Python modules created were set out to be flexible in length, difficulty, and subject matter.

2.2.2 Programming in STEM Courses Can Improve Students' Conceptual Learning

In another study, Jackson et al. [9], used an instructional model to apply programming to STEM courses [9]. The study participants were undergraduate students in lower-division Mathematics courses and High School teachers in professional development workshops [9]. The model used programming to study mathematical concepts with the underlying goal of increasing students' understanding of abstraction, an essential concept of CT [9]. The study included a control group of students who were not exposed to the CS content taking the same Math course. According to the research, the students who learned the content with computer programming improved 38% in their processing level, whereas the students in the control group improved 15% in processing level [9]. According to the authors, teachers, and students benefited from the approach and the results suggest that the integration of programming improved students' ability to abstract and generalize their conceptual learning [9].

2.3 Student Reflections on CS in the classroom

Celepkolu et al. [10] performed a year-long study with middle school science students engaged in CT activities and block-based programming. The study found significant learning gains in students' CS knowledge, and that the majority of students had positive sentiments regarding the integration of CS into their classrooms [10]. Some of the themes students reflected on after their experience was that computing can show the details of science processes, provide active learning opportunities, and aid in understanding science from a different perspective [10]. The findings of this study suggest that interdisciplinary programming may be interesting to students as well as beneficial to their learning in STEM courses.

2.4 Interactive Python Notebooks in Education Promotes Active Learning

iPython Notebooks are a web-based interactive computational environment for creating, executing, and visualizing code. They can be used to create a worksheet type lesson, text book and code editor for students. The use of iPython Notebooks in education has been shown to increase engagement and participation while also improving understanding of course content [11]. The notebooks can aid in conceptual learning and make lessons more relevant to diverse interests and different learning types [11]. The interactivity of the notebooks allows for students to be active participants in their learning by providing an avenue for exploring, analyzing, synthesizing, and evaluating the content. In active-learning, students are actively engaged by teaching strategies rather than passively participating in listening or taking notes [2]. Active learning methods have been shown to increase in STEM subjects [12]. The format offers a valuable tool for remote and asynchronous learning as a guided lesson.

2.5 Summary of Studies

The prior work in the area of interdisciplinary programming's impact on student learning suggests that it enhances STEM learning and problem solving skills. There is ample evidence to

support that further work in this area should be pursued and that there is more to learn. Additionally, the best practices identified by prior studies inform the methodology of this study. The studies do not however, provide findings regarding shorter exposure to CS and programming which this study explores.

CHAPTER 3 METHODOLOGY

The goal of this year long research study was to design, develop, and evaluate an interdisciplinary CS introductory curriculum. This curriculum was piloted in a variety of Montana high school courses that are not identified as programming or CS courses. The methods used to accomplish this goal are detailed below by demonstrating the curriculum development process, the instruction of the material, a description of the research study participants, the data collection procedures, and the data analysis methodologies.

3.1 Module Development

By partnering with high school teachers to develop modules that align with their specific curriculum, learning outcomes, and students' needs, we were able to integrate complementary programming exercises into classrooms across Montana. For instance, if a high school Chemistry teacher was already planning on covering Gas Law equations, then demonstrating how programming can be applied to solve gas law problems is a relevant and natural addition to the course. The complementary nature of using programming in pursuit of the learning outcomes that are already in place for a course benefits the students, teachers, and the goal of broadening the participation in CS education.

As high school teachers are an integral part of broadening CS Education, the first step in developing the modules was identifying teachers who are interested in linking programming to their core curriculum. We then asked for the teacher partner to identify a set of problems or lessons that students could benefit from learning with code. Teachers provided lessons on statistics, payroll math, arithmetic sequences, Gas Law formulas, molarity calculations, and more. Examples of

teacher provided problems are shown in figure 3.1 and Appendix C.

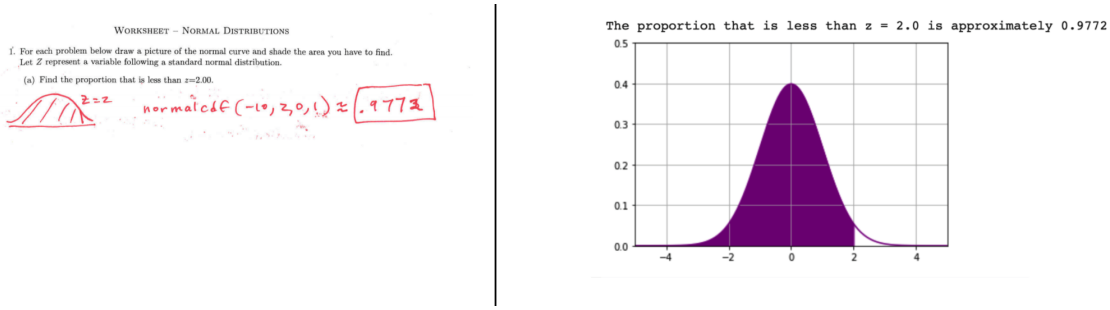

Figure 3.1: Example of a teacher provided statistics problem and programming assignment .

Next, we developed programming exercises based on the desired problems and worked with the teacher partners to make the module engaging, clear, and useful. Examples of programming exercises based on the teacher provided problems are shown in figures 3.1, 3.2, and 3.3 . Instructions provided step-by-step explanations on how to interact with the code examples. The instructions frequently include identifying certain parts of the program, for instance, identify a variable assignment in the code. Other student tasks include changing variable values, making function calls, writing part of a function, and checking outputs. The programming concepts that were introduced in the lessons vary based on the material, but generally include input/output, variables, functions, and using code libraries. Some include more advanced concepts such as branching and loops. In addition, different concepts and practices of computational thinking are used such as problem decomposition, algorithmic thinking. The lessons encourage student exploration by providing options and opportunities to play around with the code cells and visualize the output of their changes. Resources are often linked and ideas for developing more complexity in the programs are encouraged.

A goal of the project was to provide flexibility for teacher participation, whether that be running a 30-minute module in a class remotely, writing code and exercises for the project, or somewhere in between. Teacher education and involvement are essential to solving the CS education problem, thus their involvement and buy-in are just as important as the students.' The amount of training that teachers have regarding CS and programming varied and the individualized support allowed for teachers with no formal education in CS or experience in programming to run the curriculum in their classrooms.

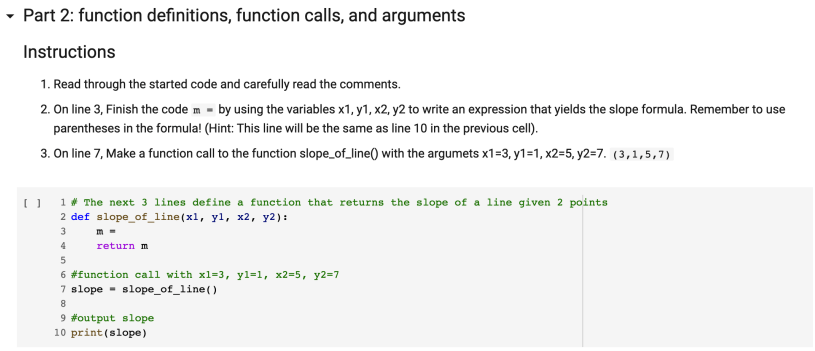

Figure 3.2: Example of a slope calculation programming problem introducing functions.

3.1.1 Google Colab and Interactive Python Notebooks

The lessons are developed in the Python programming language and distributed in interactive Python notebooks with the Google Colab environment (see figure 3.3). Python was chosen for the project because it is a popular high-level, programming language with a readable syntax which makes it great for beginning programmers [12]. The Python programming language is also powerful, widely used, and has many relevant libraries for interdisciplinary programming. The Google Colab IDE runs Python code in an internet browser. Google Colab renders interactive Python notebooks (Jupyter Notebooks) which allow for the the integration of formatted text, code, inputs, and outputs in the same window. Google Colab requires no installation or set up to run code and is simple to learn how to use. It provides a clean and user-friendly interface to provide instructions and code blocks which allows for an instructional problem-solving narrative to progress throughout the lesson.

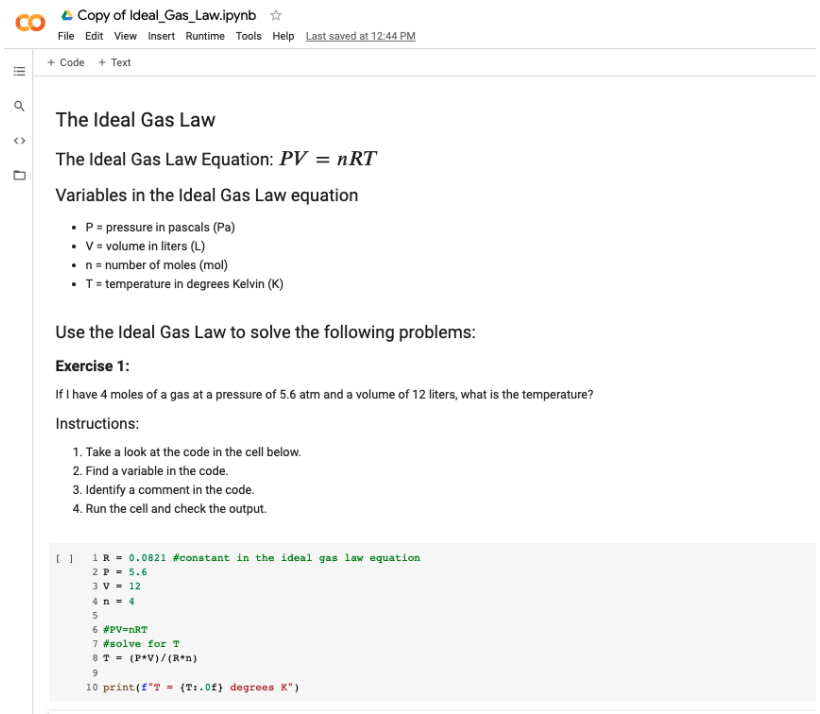

Figure 3.3: Ideal Gas Law, Chemistry programming exercise excerpt from the beginning of the module. The instructions in the beginning are more explicit, include more comments in the code and ease the students into reading and running code. The full module is included in Appendix D.

3.2 Module Instruction

After developing the content, the modules were ideally co-taught in the classroom with the partner teacher and researchers from the University of Montana. Given the need for distance learning, because of the COVID-19 pandemic, the Python modules were also integrated into remote classrooms and asynchronous learning schedules. We wanted to create useful resources for teachers and students while creating a community surrounding the larger goal of broadening participation in CS.

Initially, the instructors guided the students through using the Google Colab environment. Students began with an introductory module that covered basic coding structures such as input, output, functions, and parameters. Then, the students worked through the course-specific modules at their own pace, discussing questions regarding the CS and non-CS material with classmates and troubleshooting together to debug their code. The modules allowed students to read and run complete code snippets, change values, and then create their own programs. CT concepts and practices were immersed in the student instructions, allowing for understanding to broaden as confidence levels increased and the exercises became more challenging.

3.3 Data Collection

The University of Montana Institutional Review Board approved the collection of data for this project from the student and teacher participants. To ensure in-depth analysis, both qualitative and quantitative data in the form of pre-and post-lesson surveys were collected. The student survey was designed to collect data regarding the research questions of interest. The student participants took pre and post intervention surveys. The pre and post-instruction survey. The full surveys are included in Appendix A and Appendix B. The surveys included multiple-choice, short answer, and ranking questions. The questions regarding the host-discipline subject matter were designed by the high school teachers to track learning goals relevant to their courses and specific lessons. The primary research tool used to collect the data was Google Forms. Because of the COVID-19 pandemic, researchers were not able to be present in the classrooms when the instruction or data

collection happened which may have affected the consistency of the data collection.

3.4 Study Participants

3.4.1 Teacher Participants

The most important factor for selecting the study's teacher participants was their interest in incorporating CT and/or programming into their classrooms. The four high school teacher participants in the study are Montana educators, teaching grades 9-12, in Biology, Chemistry, Algebra, and Business. A majority of the partner high school teachers participated in a professional development course on integrating Python into their Math and Science courses. The teachers self-identified as having a little to a moderate amount of programming experience.

3.4.2 Student Participants

The 132 student participants in the study were selected based on the teacher partnerships developed. All of the students were enrolled in high schools in Missoula, Great Falls, or Corvallis taking a course in Biology, Chemistry, Business, or Algebra. 71.76% of the student participants had not heard of the Python programming language before the study and the majority of students identified having no programming experience at all (see table 3.1).

Table 3.1: Student Participants Response to: How much programming experience do you have?

	None at all (1)	A little (2)	Moderate (3)	A lot (4)	A great deal (5)
%	57.14	29.76	8.33	3.57	1.19

N=132

3.5 Data Analysis

The data was analyzed across all participating classes and summarized to make estimates about the target population of Montana high school students. Additional analysis of discipline-specific data was analyzed for statistical significance. Statistical tests completed on the quantitative data include correlated two-tailed T-tests, percent change between pre and post-intervention results, and measures of central tendency, and variance. A thematic analysis was performed on the qualitative data to find patterns and label recurring concepts by grouping them into themes.

CHAPTER 4 RESULTS

The results from the data analysis are provided below to indicate whether this method of broadening the participation of CS education is worth further study. The results are broken into three main sections: interest in programming, CS learning, and discipline specific learning goals correlating with the three research questions.

4.1 Student Interest in Programming

To address the research question of whether students' interest in programming increased after being exposed to interdisciplinary, introductory CS, the participants were asked to identify how much interest in programming that they had pre and post-intervention (see table 4.1, and table 4.2). Additionally, to better understand the reasons behind the quantitative data collected, participants were asked to explain why they were or were not interested in learning to program (see table 4.3).

Table 4.1: Student Interest in Programming Survey Results

Question	Time of Response	None (1)	A little (2)	Moderate (3)	A lot(4)	A great deal (5)
How much interest in learning to program do you have?	Pre (%)	25.76	38.64	27.27	6.06	2.27
	Post (%)	42.03	24.64	24.64	4.35	4.35

N=132

Table 4.2: Change in Student Interest in Programming Survey Results

Question	Pre Mean	Post Mean	Percent Change	T-value	P-value
How much interest in learning to program do you have?	2.2	2.04	-0.16	-1.06	0.29

N=132

Interest in learning to program decreased slightly across the pilots after the lessons. For this study, there is no supportive data to show that students were more interested in studying or pursuing Computer Science after the intervention, however, it would be worth studying more in the future to better understand students' lack of interest in programming/CS.

The reasons that students provided (see table 4.3) as to why they were interested in programming did not vary much from the pre and post-survey results. The reasoning provided by the students could help to improve further curricula. Considering how to frame computer science education as applicable, fun, and important for everyone should continue to be studied and incorporated into curricula.

Table 4.3: Why Students are Interested or not Interested In Learning to Program

	Interested in CS	Not interested CS
Pre-intervention	• fun • important • useful • cool • job prospects • scientific applications • web programming • trying new things • application to the Arts	• too busy • difficult, confusing • boring • I don't understand it • there are more important things to learn • the job I want doesn't require programming • I don't think that I will ever use it outside of school • I'm not good with technology • I don't like computer work • I would not be good at it
Post-intervention	• fun • jobs • interested in computers and code • liked the assignment • see an application to a personal project • can streamline processes • I want to make my own programs	• don't enjoy/ care about it • I'm not smart enough • boring, confusing, complicated • I don't think that I a good at it • I don't like computer stuff • I don't understand how it applies to "real life" • I don't think that I will use it

A compiled list (with duplicates removed and themes combined) of reasons students provided for their interest or lack of interest in learning to program

4.2 Student CS Concepts Learning

Learning gains by the percentage of correct answers regarding CS material and the change pre and post are presented in table 4.4. All four of the questions had a significant increase in correct answers after the lesson, and students' answering "I don't know," decreased significantly. These findings suggest that it is possible to teach students basic computational concepts with an interdisciplinary curriculum.

Table 4.4: CS Content Survey Question Results

Question	Pre Correct (%)	Post Correct (%)	(%) Change
Which of these Python code blocks would output a "Hello!" message to the screen?	17.74	76.19	+58.45
Which statement best describes what a function in Python is?	42.11	68.12	+26.01
Which statement best describes what an argument in Python is?	14.3	31.88	+17.58
Which statement best describes a variable in Python?	22.73	57.35	+34.62

N=132

4.3 Host-discipline Learning

Learning gains by percent change in Host-discipline learning concept correctness after the intervention in 4.5. The questions asked were specific to the lessons being taught and written by the high school partner teachers. For instance, in the Molarity Module one of the questions asked was:

> One mole of H2O is made up of 2 moles of hydrogen atoms and 1 mole of the oxygen atom. The mass of 1 mole of Hydrogen atoms= 1 g /mol and the mass of 1mole of Oxygen atoms = 16 g/mol. What is the Molar mass of water? (Answer "I don't know" if you do not know)

Table 4.5: Host Discipline Content Survey Question Results

Course Discipline	Mean (%) Change	Sample Size
Biology	+14.47	22
Chemistry (Class 1)	+52.97	28
Chemistry (Class 2)	+54.90	20
Algebra	+29.35	35
Business	+13.4	27
All Classes	+33.66	132

The two Chemistry teachers had the greatest percent change in core course content, which suggests that it was an appropriate fit for the project's goals. All of the courses provide significant gains for the host-discipline learning goals. The students' perception on the usefulness of the assignment varied greatly, as well as their opinion on programming being relevant to their coursework in the host-disciplines. All the questions can be referenced in Appendix A and Appendix B.

CHAPTER 5 DISCUSSION

The following sections will discuss the study's results, in more depth, and implications of the three research queries regarding: student interest in programming, CS concepts, and discipline specific learning. Next, the sections will discuss the study's limitations, partner high school teachers' perceptions of the lessons, recommendations for future work, and final conclusions.

5.1 Student Interest in Programming

Contrary to the hypothesized association, student interest in programming decreased slightly after the intervention. The percent change in the data and the qualitative responses to why the student participants were, or were not, interested in programming were fairly consistent both pre and post-intervention. Without more data, it is difficult to understand why the results indicate less interest or the same amount of interest in programming for the students' post-intervention. One factor that may have influenced the results is the short amount of time and exposure to a new academic field and practice. Students may not have had enough time to experience any change in their interest in CS/programming. Another important factor to consider is the instruction and content varied between pilots. Some of the students had the lessons taught remotely and asynchronously, while other lessons were taught in the classroom and in group settings. Remote learning is difficult for many students and teachers. The teaching style and level of expertise in programming also varied by instructor.

The results of this question matter because CS is an elective course for most students, if they are not interested in the subject then they will likely not elect to take CS courses. Having a better understanding of what influences a student's interest in CS is essential for working towards the

goal of all students having a basic understanding and experience with the field. For instance, for the students who did indicate interest in the field many noted that the programming assignment was fun, useful, and learning to the program could provide job opportunities. Interviews with the students could provide a more thorough understanding of this area, and provide important context for improving student interest. Focusing further projects with this in mind could help to encourage more students.

In reflection of the results regarding student interest in programming, interesting lines of inquiry to include in future data collection may be:

1. Did you enjoy the programming lesson more or less than paper assignments? Why or why not?

2. What is your interest level in including programming in their class in the future?

3. How interested are you in the host discipline subject area?

5.2 CS Concepts

In accordance with the hypothesis, students appeared to learn basic CS concepts from the lessons based on the data collected. There was a significant percent change in correct answers of the CS concept questions post-intervention. These results lend themselves to the conclusions that students picked up the programming concepts quickly, and that an introduction to programming with interdisciplinary material that students can provide a basic understanding of the field. These results suggest that interdisciplinary programming is a promising method for introductory CS exposure. As the questions were simple, and multiple choice, we were not able to test in depth understanding of CS/ CT concepts. It may be interesting in the future to include some more complex questions in the data collection and expand upon the CS concepts taught in the modules.

5.3 Host-discipline Concepts

Intending to meaningfully bridge concepts across different disciplines, programming reinforcing important concepts in other disciplines seems to be the most promising result for this study. As with the CS concepts, there was a significant percent change in correct answers post-intervention for the host-discipline questions as well. These results suggest that interdisciplinary programming may be beneficial to learning in other subjects as well. As the literature suggests, CS/CT provides a set of tools and ways of thinking to solve problems with decomposition. This exposure aimed to provide the context for the broad applications of programming. As there is no way to fairly compare the questions for different classes and account for the variation of student experience I recommend the continuation of the project to examine this research question more.

5.4 Limitations of the Study

This study had a few limitations to note. Some of the variables were difficult to control for including variance in instruction, with the different teachers, subjects, cities, grade levels. Additionally, some students who took the pre-instruction survey did not take the post-instruction survey. Because of the COVID-19 pandemic, researchers were not able to be present in the classrooms when the instruction or data collection happened which may have affected the consistency of the data collection. As the teacher participants noted in their exit survey, asynchronous and remote learning is difficult. Specifically, one teacher noted that"it would have been better if I could have completed the assignments in person so students could get help and not get too frustrated." The teachers also noted that trying something new in the classroom can be difficult and that more practice would help the success of the project.

5.5 Teacher Perception of the Lessons

Additionally, after reflection on the student data collected in the study, input from the high school teacher partners was collected. The teachers found the exercises as useful as an average

of 3.5 / 5 (with 1 = "not useful" and 5 = "very useful"). Some of the teachers reported that the assignment aided in their students' understanding of the learning goals for the course, while the other teachers were unsure. One teacher reported that the exercises helped students "improve their problem solving skills." The teachers liked the formatting of the assignments and the Google Colab environment. They reported that they are likely to use programming in their classrooms again. Time restraints were cited as barriers to integrating CS/ programming into their classes; "the students loved it and wanted more, we just didn't have time." Another teacher commented that "I think they thought it was a complicated way to do something that they could do on a calculator." This comment touches on an important issue with the introductory interdisciplinary CS: how do you explain or discuss the benefit of learning CT. Further, exploration of the topic would be worthwhile for the next iterations of the project. Just as teacher partnerships for this project were essential developing the curricula and instruction, teacher perceptions are equally important. Examining what works well in the classroom from the teacher perspective could be valuable in CS educational research and important in the efforts of expanding the reach of the future iterations of the project.

5.6 Recommendations for Future Work

Based on the results of this study, I have a few recommendations regarding further research in introductory CS education, and specifically for the continuation of this project. Most importantly, the modules should be co-taught, in the classroom, with host-discipline teachers and CS researchers from the University. As a researcher, it was difficult not to have been in the classroom to observe the students and talk with them. Having researchers from the University trained and interested in CS and especially CS education more involved in the instruction of the modules could be very beneficial to further research and student learning. Additionally, I recommend adding more time for students to become comfortable with the programming material and allow for more practice with the environment and tools. One way to do this would be to work with the teachers who want to integrate programming into multiple lessons and have a dedication to the project beyond one

assignment. Finally, interviews of the students, especially regarding what they liked and disliked about the lessons and their interest in CS, could provide a better understanding of the research. Interviews with the students could provide a more thorough understanding of the lack of interest that high school students may show in CS.

5.7 Conclusions

This research study examined the effects of integrating interdisciplinary CS education into high school courses. The three specific effects of the programming integration that the study was designed to research were students' interest in programming, understanding of basic computational concepts, and the understanding of host-discipline learning goals. The data shows promising, preliminary results for two of the three research questions: 1) learning CS concepts from interdisciplinary computing, and 2) increased performance of host discipline learning goals. The results of the third research question about student interest in programming proved inconclusive, but there were qualitative insights gained. These results build on existing evidence of STEM courses and programming reinforcing each other in the classroom. However, the results do not fit with the theory that students enjoy CS/programming or at least in the manner that was practiced in the study.

The data contributes a clearer understanding of how CS can be introduced to a larger section of students who may not choose to take an elective CS course. While previous research has focused on longer term integration of programming in the classrooms, these results demonstrate that short term lessons can also be effective in teaching concepts and introducing programming to students. The experiment provides a new insight into the efficacy of small doses of programming and non-CS STEM high school courses. Overall, the results indicate that more research in this area would be worthwhile. These results should be taken into account when considering how to tackle the CS education problem in Montana. By exposing more students and teachers to CS with diverse methods, less students will be left out of learning important skills and ways of thinking.

The results of this study provide motivation and can facilitate further analysis of using CT in

high school courses to benefit student learning. During this research study, many students were exposed to programming for the first time, and they experienced the wide applications of computing. A diverse group of students, including underrepresented groups in computing, were exposed to CS's relevance to their academic interests, goals, and communities. Further, the teachers in the study reported that they are interested in incorporating programming into their courses in the future.

Approaches and insights that emerge from this preliminary study can inform further exploration of this topic and the continuation of the project. Future work with researchers in the field is highly encouraged and would provide more opportunities to understand the CS problem in Montana. Research in integrating CS into the humanities, arts, and social sciences would be an interesting continuation of the project. It may also be beneficial to understand how appropriate the curriculum is for middle school or elementary school students. Undergraduate and graduate students could be recruited in co-teaching modules K-12 classrooms creating supportive and encouraging mentorship opportunities. Other recommendations for building on this study include conducting student and teacher interviews and considering the impact of classrooms that can work on the project for more than one lesson or class period.

To the benefit of the field of CS education, the analysis and execution of this project contribute to a better understanding of how students learn CS; their perceptions of CS and its possibilities; and the efficacy of cross-disciplinary CS education as an introduction to the field. As a result, hopefully more students consider studying CS, schools without CS-specific courses can offer students programming opportunities and increased awareness of CS applications to STEM fields. Just as society wants all students to have an understanding of English, Math, History, and the Sciences, all students should have an understanding of Computer Science.

APPENDIX A Pre-instruction survey questions included in pilots

1. How much programming experience do you have?
    - A great deal
    - A lot
    - A moderate amount
    - A little
    - None at all
2. Have you heard of the Python programming language before?
    - Yes
    - No
3. How much interest in learning to program do you have?
    - A great deal
    - A lot
    - A moderate amount
    - A little
    - None at all
4. Can you explain why you are (or why you are not) interested in learning to program?

    Open-ended response

5. Do you think programming is relevant to your professional and/or academic goals?

    - Yes
    - No
    - Maybe

6. Which of these Python code blocks would output a “Hello!” message to the screen?

    - output(”Hello!”)
    - (“Hello!”)
    - print(“Hello!”)
    - I don’t know

7. Which statement best describes what a function in Python is?

    - a block of code that when called can take inputs and return outputs
    - code that executes a mathematical operation
    - a description of the program’s purpose
    - I don’t know

8. Which statement best describes what an argument in Python is?

    - two ways to code the same result
    - a value passed to a function
    - the final answer to an equation
    - I don’t know

9. Which statement best describes a variable in Python?

    - a container for a value
    - a data type

- a library of functions
- I don't know

10. How comfortable do you feel with understanding [*core course lesson]?

    - Very comfortable
    - Somewhat comfortable
    - A little uncomfortable
    - Very uncomfortable

11. How much interest in learning to program do you have?

    - A great deal
    - A lot
    - A moderate amount
    - A little
    - None at all

12. Do you think that programming is relevant to your course work in [*core course]?

    - Yes
    - No
    - Maybe

A.1 Biology Content Questions: Graph Choice and Scientific Analysis

1. What type of graph is best for data that compares how one numeric value is related to another (for example height and foot length)?

    - a histogram plot
    - a pie chart

- a Scatter Plot
- I don't know

2. What type of graph is best to display data from an investigation that compares the variability within a set of data (for example eagle wingspans in inches)?

   - a histogram plot
   - a pie chart
   - a Scatter Plot
   - I don't know

3. For the following data set: (3,5,6,7,9,6,8), which sequence of numbers accurately represents the mode? (Answer "I don't know" if you do not know)

   Open-ended response

4. For the following data set: (3,5,6,7,9,6,8), which sequence of numbers accurately represents the median? (Answer "I don't know" if you do not know)

   Open-ended response

5. For the following data set: (3,5,6,7,9,6,8), which sequence of numbers accurately represents the mean? (Answer "I don't know" if you do not know)

   Open-ended response

6. Which measure can be used to determine how much a value differs from the rest of the data points?

   - true
   - false
   - I don't know

7. What type of graph is best to display data from an investigation that compares the variability within a set of data (for example eagle wingspans in inches)?

   - arithmetic mean
   - percent deviation
   - outlier
   - I don't know

A.2 Chemistry Content Questions: Molarity

1. Which statement best defines molar mass?

   - the mass of a given substance divided by its amount of substance (mol), in g/mol
   - the amount of a substance dissolved in a certain mass of solvent
   - the volume that one mole of a substance takes up
   - I don't know

2. One mole of a substance is equal to $6.022 \times 10^{\mathbf{23}}$ units of that substance (such as atoms, molecules, or ions). What is this number known as?

   - The molar constant
   - Aufbau's number
   - Avogadro's number
   - I don't know

3. One mole of H2O is made up of 2 moles of hydrogen atoms and 1 mole of the oxygen atom. The mass of 1 mole of Hydrogen atoms= 1 g /mol and the mass of 1mole of Oxygen atoms = 16 g/mol. What is the Molar mass of water? (Answer "I don't know" if you do not know)

   Open-ended response

molar mass of water is 18 g/mol. How many grams are in 3 moles of water(H2O)?

Open-ended response

A.3 Chemistry Content Questions: Ideal Gaw Law

1. What is the formula for the Ideal Gas Law?

   - P=VnRT
   - PV=nRT
   - RV=nPT
   - I don't know

2. What does the R variable in the Ideal Gas Law represent?

   - The Ideal Gas Law constant
   - The rate of gas change
   - The Ideal Gas Law real number
   - I don't know

3. What is 67 degrees Celsius in Kelvin? (Answer "I don't know" if you do not know)

   Open-ended response

A.4 Algebra Content Questions: The Slope of a Line

1. What is the formula for the Point-slope formula?

   - m = (y2-x2) / (y1-x1)
   - m = (y2+y1) / (x2+x1)
   - m = (y2-y1) / (x2-x1)
   - I don't know

2. What is 67 degrees Celsius in Kelvin? (Answer "I don't know" if you do not know)

   Open-ended response

APPENDIX B Post-instruction survey questions included among all pilots

1. How much programming experience do you have?
    - A great deal
    - A lot
    - A moderate amount
    - A little
    - None at all
2. Have you heard of the Python programming language before?
    - Yes
    - No
3. How much interest in learning to program do you have?
    - A great deal
    - A lot
    - A moderate amount
    - A little
    - None at all
4. Can you explain why you are (or why you are not) interested in learning to program?

    Open-ended response

5. Do you think programming is relevant to your professional and/or academic goals?

    - Yes
    - No
    - Maybe

6. Which of these Python code blocks would output a “Hello!” message to the screen?

    - output(”Hello!”)
    - (“Hello!”)
    - print(“Hello!”)
    - I don’t know

7. Which statement best describes what a function in Python is?

    - a block of code that when called can take inputs and return outputs
    - code that executes a mathematical operation
    - a description of the program’s purpose
    - I don’t know

8. Which statement best describes what an argument in Python is?

    - two ways to code the same result
    - a value passed to a function
    - the final answer to an equation
    - I don’t know

9. Which statement best describes a variable in Python?

    - a container for a value
    - a data type

- a library of functions
- I don't know

10. How comfortable do you feel with understanding [*core course lesson]?

    - Very comfortable
    - Somewhat comfortable
    - A little uncomfortable
    - Very uncomfortable

11. How much interest in learning to program do you have?

    - A great deal
    - A lot
    - A moderate amount
    - A little
    - None at all

12. How much do you think that the programming exercises helped you to understand [*insert host-disciple course content here]?

    - A great deal
    - A lot
    - A moderate amount
    - A little
    - None at all

13. Do you think that programming is relevant to your course work in [*core course]?

    - Yes
    - No
    - Maybe

14. Can you explain why (or why not) that you think that programming is relevant to your course work in [*insert course name here]?

    Open-ended response

APPENDIX C Example of Ideal Gas Law problem set selected by Chemistry teacher for the project

Name:_____ Date:_____ Per:_____

12B - Ideal Gas Law Problems

Use the ideal gas law to solve the following problems:

1) If I have 4 moles of a gas at a pressure of 5.6 atm and a volume of 12 liters, what is the temperature?

2) If I have an unknown quantity of gas at a pressure of 1.2 atm, a volume of 31 liters, and a temperature of 87 ^{0}C, how many moles of gas do I have?

3) If I contain 3 moles of gas in a container with a volume of 60 liters and at a temperature of 400 K, what is the pressure inside the container?

4) If I have 7.7 moles of gas at a pressure of 0.09 atm and at a temperature of 56 ^{0}C, what is the volume of the container that the gas is in?

5) If I have 17 moles of gas at a temperature of 67 ^{0}C, and a volume of 88.89 liters, what is the pressure of the gas?

6) If I have an unknown quantity of gas at a pressure of 0.5 atm, a volume of 25 liters, and a temperature of 300 K, how many moles of gas do I have?

❑ I checked my answers

Name:_____ Date:_____ Per:_____

7) If I have 21 moles of gas held at a pressure of 78 atm and a temperature of 900 K, what is the volume of the gas?

8) If I have 1.9 moles of gas held at a pressure of 5 atm and in a container with a volume of 50 liters, what is the temperature of the gas?

9) If I have 2.4 moles of gas held at a temperature of 97 ^{0}C and in a container with a volume of 45 liters, what is the pressure of the gas?

10) If I have an unknown quantity of gas held at a temperature of 1195 K in a container with a volume of 25 liters and a pressure of 560 atm, how many moles of gas do I have?

11) If I have 0.275 moles of gas at a temperature of 75 K and a pressure of 1.75 atmospheres, what is the volume of the gas?

12) If I have 72 liters of nitrogen gas held at a pressure of 3.4 atm and a temperature of 225 K, how many grams of gas do I have?

❑ I checked my answers

Name:_____ Date:_____ Per:_____

Ideal Gas Law Problems – Solution Key

1) If I have 4 moles of a gas at a pressure of 5.6 atm and a volume of 12 liters, what is the temperature? **205 K**

2) If I have an unknown quantity of gas at a pressure of 1.2 atm, a volume of 31 liters, and a temperature of 87 ^{0}C, how many moles of gas do I have? **1.26 moles**

3) If I contain 3 moles of gas in a container with a volume of 60 liters and at a temperature of 400 K, what is the pressure inside the container? **1.64 atm 166 kPa**

4) If I have 7.7 moles of gas at a pressure of 0.09 atm and at a temperature of 56 ^{0}C, what is the volume of the container that the gas is in? **2310 L**

5) If I have 17 moles of gas at a temperature of 67 ^{0}C, and a volume of 88.89 liters, what is the pressure of the gas? **5.34 atm 540.3 kPa**

6) If I have an unknown quantity of gas at a pressure of 0.5 atm, a volume of 25 liters, and a temperature of 300 K, how many moles of gas do I have? **0.51 moles**

7) If I have 21 moles of gas held at a pressure of 78 atm and a temperature of 900 K, what is the volume of the gas? **19.9 L**

8) If I have 1.9 moles of gas held at a pressure of 5 atm and in a container with a volume of 50 liters, what is the temperature of the gas? **1603 K**

9) If I have 2.4 moles of gas held at a temperature of 97 ^{0}C and in a container with a volume of 45 liters, what is the pressure of the gas? **1.62 atm 164 kPa**

10) If I have an unknown quantity of gas held at a temperature of 1195 K in a container with a volume of 25 liters and a pressure of 560 atm, how many moles of gas do I have? **143 moles**

11) If I have 0.275 moles of gas at a temperature of 75 K and a pressure of 1.75 atmospheres, what is the volume of the gas? **0.97 L**

12) If I have 72 liters of gas held at a pressure of 3.4 atm and a temperature of 225 K, how many moles of gas do I have? **372.4 grams**

❑ I checked my answers

APPENDIX D Full example of Chemistry programming Module based off the Ideal Gas Law problem set provided by Chemistry teacher

The Ideal Gas Law

The Ideal Gas Law Equation: $PV = nRT$

Variables in the Ideal Gas Law equation

- P = pressure in pascals (Pa)
- V = volume in liters (L)
- n = number of moles (mol)
- T = temperature in degrees Kelvin (K)

Use the Ideal Gas Law to solve the following problems:

Exercise 1:

If I have 4 moles of a gas at a pressure of 5.6 atm and a volume of 12 liters, what is the temperature?

Instructions:

1. Take a look at the code in the cell below.
2. Find a variable in the code.
3. Identify a comment in the code.
4. Run the cell and check the output.

```
R = 0.0821 #constant in the ideal gas law equation
P = 5.6
V = 12
n = 4

#PV=nRT
#solve for T
T = (P*V)/(R*n)

print(f"T = {T:.0f} degrees K")
```

Exercise 2:

If I have an unknown quantity of gas at a pressure of 1.2 atm, a volume of 31 liters, and a temperature of 87 0C, how many moles of gas do I have?

Instructions:

1. Fill in the values for P, V, and T (don't forget to convert to K) on lines 2 - 4 based on exercise 2.
2. Run the cell and check the output.
3. Solve the equation by hand or with a calculator and compare the answers. If it is not the same, then check with a classmate and together try to figure it out.

```
R = 0.0821 #constant in the ideal gas law equation
P =
V =
T =

#PV=nRT
#solve for n
n = (P*V)/(R*T)

print(f"n = {n:.2f} moles")
```

Exercise 3:

If I contain 3 moles of gas in a container with a volume of 60 liters and at a temperature of 400 K, what is the pressure inside the container?

Instructions:

1. Take a look at lines 2 and 3. What do you think this code does?
2. Fill in the variables on lines 5-8.
3. Check the answers again on paper.

```
#function to convert standard atmospheres to kilopascals
def atm_to_kPa(atm):
    return atm * 101.325

R =
V =
n =
T =

#PV=nRT
#solve for P
P = (n*R*T)/V

print(f"P = {P:.2f} atm. and {atm_to_kPa(P):.0f} kPa")
```

Exercise 4:

If I have 7.7 moles of gas at a pressure of 0.09 atm and at a temperature of 56 degrees C, what is the volume of the container that the gas is in?

Instructions:

1. Fill in the variables P and n based on exercise 4.
2. Use a function call to assign T to the correct temperature. Take a look at the function definition on line 2 for C_to_K(deg_C). It takes one argument deg_C which is the degrees Celsius that you want to convert to Kelvin. You can assign a variable to a function call. Try "T = C_to_K(56)" on line 12.
3. On line 13, let's test that our function is working as we would expect. Write print(T). Is it right?
4. Check the answer to exercise 4.

```
 1 #function to convert degrees Celsius to degrees Kelvin
 2 def C_to_K(deg_C):
 3     return deg_C + 273.15
 4
 5 R = 0.0821 #constant in the ideal gas law equation
 6 P =
 7 n =
 8 T =
 9
10
11 #PV=nRT
12 #solve for V
13 V = (n*R*T)/P
14
15 print(f"V = {V:.2f} liters")
```

Exercise 5:

If I have 17 moles of gas at a temperature of 67 degrees Celsius, and a volume of 88.89 liters, what is the pressure of the gas?

Instructions:

1. Fill in the variables on lines 5-8, you can use a function call to convert the temperature to K if you'd like.
2. Solve for P on line 12 (don't forget parenthesis).
3. Check your answer

```
#function to convert degrees Celsius to Kelvin
def C_to_K(deg_C):
    return deg_C + 273.15

R =
V =
n =
T =

#PV=nRT
#solve for P
P =

print(f"P = {P:.2f} atm")
```

Exercise 6:

If I have an unknown quantity of gas at a pressure of 0.5 atm, a volume of 25 liters, and a temperature of 300 K, how many moles of gas do I have?

Instructions:

1. solve for n on line 6 after assigning the known variables from exercise 6.

```
R =
P =
V =
T =

#PV=nRT
#solve for n
n =

print(f"n = {n:.2f} moles")
```

Exercise 7:

If I have 21 moles of gas held at a pressure of 78 atm and a temperature of 900 K, what is the volume of the gas?

Instructions:

1. Solve for the V and don't forget to check your work.

```
R =
```

```
P =
n =
T =

V =

print(f"V = {V:.2f} liters")
```

Exercise 8:

If I have 1.9 moles of gas held at a pressure of 5 atm and in a container with a volume of 50 liters, what is the temperature of the gas?

Instructions:

1. Solve for the temperature of the gas.

```


T =

print(f"T = {T:.0f} degrees K")
```

Exercise 9:

If I have 2.4 moles of gas held at a temperature of 97 degrees Celsius and in a container with a volume of 45 liters, what is the pressure of the gas?

Instructions:

1. solve for the pressure of the gas.

```
def atm_to_kPa(atm):
    return atm * 101.325

#function to convert degrees Celsius to degrees Kelvin
def C_to_K(deg_C):
    return deg_C + 273.15



P =
```

```
12
13 print(f"P = {P:.2f} atm and {atm_to_kPa(P):.0f kpa}")
```

Exercise 10:

If I have an unknown quantity of gas held at a temperature of 1195 K in a container with a volume of 25 liters and a pressure of 560 atm, how many moles of gas do I have?

Instructions:

1. Solve for the number of moles.

```
1
2
3
4
5
6 n =
7
8 print(f"n = {n:.0f} moles")
```

Exercise 11:

If I have 0.275 moles of gas at a temperature of 75 K and a pressure of 1.75 atmospheres, what is the volume of the gas?

Instructions:

1. Solve for the volume of the gas.

```
1 R =
2 P =
3 n =
4 T =
5
6 V =
7
8 print(f"V = {V:.2f} liters.")
```

Exercise 12:

If I have 72 liters of gas held at a pressure of 3.4 atm and a temperature of 225 K, how many moles of gas do I have?

Instructions:

1. Solve for the number of moles.

```
1
2
3
4
5
6 n =
7
8 print(f"n = {n:.0f} moles.")
```

Exercise 13:

Let's write a program that could solve any of the previous Ideal Gas Law questions.

Instructions:

1. Take a look at the program below. There will likely be a lot of things you don't understand, but what do you think is happening?
2. Let's start with the line 1, We are using a function called input to allow the user of the program to provide which variable we are solving for. Let's try it out.
3. In Python, "==" means equal to. If the unknown variable is equal to 'P' then the code to solve for P will run.
4. Try running the cell below and entering P for the unknown variable. Try exercise 2 again using this code: If I contain 3 moles of gas in a container with a volume of 60 liters and at a temperature of 400 K, what is the pressure inside the container? Did you get the same answer?
5. Now, try some of the other problems from above.

```
 1 unknown = input("What is the unknown variable? (P,V,n,T)")
 2 print (f"Ok, we will solve for {unknown}.")
 3
 4 R = 0.0821 #constant in the ideal gas law equation
 5
 6 if unknown == 'P' or unknown == 'p':
 7     V = float(input("What is the volume (in liters) of the gas?"))
 8     n = float(input("How many moles of the gas are in the problem?"))
 9     T = float(input("What is the temperature of the gas (in K?)"))
10     P = (n*R*T)/V
11     print(f"P = {P:.2f} atm.")
12
13 elif unknown == 'n' or unknown == 'N':
14     P = float(input("What is the pressure (in atm) of the gas?"))
15     V = float(input("What is the volume (in liters) of the gas?"))
16     T = float(input("What is the temperature of the gas (in K?)"))
```

```
    n =  (P*V)/(R*T)
    print(f"n = {n:.2f} moles.")

elif unknown == "V"or unknown == "v":
    P = float(input("What is the pressure (in atm) of the gas?"))
    n = float(input("How many moles of the gas?"))
    T = float(input("What is the temperature of the gas (in K?)"))
    V = (n*R*T)/P
    print(f"V = {V:.2f} liters.")

elif unknown == 'T' or unknown == 't':
    P = float(input("What is the pressure (in atm) of the gas?"))
    V = float(input("What is the volume (in liters) of the gas?"))
    n = float(input("How many moles of the gas are present?"))
    T = (P*V)/(R*n)
    print(f"T = {T:.2f} degrees K.")

else:
    print("Error, please try again.")
```

BIBLIOGRAPHY

[1] J. M. Wing, "Computational thinking," *Commun. ACM*, vol. 49, no. 3, p. 33–35, Mar. 2006. [Online]. Available: https://doi.org/10.1145/1118178.1118215

[2] J. Cuny, "Transforming high school computing: A call to action," *ACM Inroads*, vol. 3, no. 2, p. 32–36, Jun. 2012. [Online]. Available: https://doi.org/10.1145/2189835.2189848

[3] "Computer science initiative for all." [Online]. Available: https://obamawhitehouse.archives.gov/the-press-office/2016/01/30/fact-sheet-president-obama-announces-computer-science-all-initiative-0

[4] S. Grover and R. Pea, "Computational thinking in k–12: A review of the state of the field," *Educational Researcher*, vol. 42, no. 1, pp. 38–43, 2013. [Online]. Available: https://doi.org/10.3102/0013189X12463051

[5] "Code.org state facts mt." [Online]. Available: https://code.org/advocacy/state-facts/MT.pdf

[6] L. Carter, "Why students with an apparent aptitude for computer science don't choose to major in computer science," vol. 38, no. 1, 2006. [Online]. Available: https://doi.org/10.1145/1124706.1121352

[7] V. Barr and C. Stephenson, "Bringing computational thinking to k-12: What is involved and what is the role of the computer science education community?" *ACM Inroads*, vol. 2, no. 1, p. 48–54, Feb. 2011. [Online]. Available: https://doi.org/10.1145/1929887.1929905

[8] K. Fisler, E. Schanzer, S. Weimar, A. Fetter, K. A. Renninger, S. Krishnamurthi, J. G. Politz, B. Lerner, J. Poole, and C. Koerner, "Evolving a k-12 curriculum for integrating computer

science into mathematics," in *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '21. New York, NY, USA: Association for Computing Machinery, 2021, p. 59–65. [Online]. Available: https://doi.org/10.1145/3408877.3432546

[9] J. L. Jackson, C. L. Stenger, J. A. Jerkins, and M. G. Terwilliger, "Improving abstraction through python programming in undergraduate computer science and math classes," *J. Comput. Sci. Coll.*, vol. 35, no. 2, p. 39–47, Oct. 2019.

[10] M. Celepkolu, D. A. Fussell, A. C. Galdo, K. E. Boyer, E. N. Wiebe, B. W. Mott, and J. C. Lester, "Exploring middle school students' reflections on the infusion of cs into science classrooms," in *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, ser. SIGCSE '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 671–677. [Online]. Available: https://doi.org/10.1145/3328778.3366871

[11] L. J. B. Lorena A. Barba, "Teaching and learning with jupyter," 2019. [Online]. Available: https://jupyter4edu.github.io/jupyter-edu-book/

[12] S. Freeman, S. L. Eddy, M. McDonough, M. K. Smith, N. Okoroafor, H. Jordt, and M. P. Wenderoth, "Active learning increases student performance in science, engineering, and mathematics," *Proceedings of the National Academy of Sciences*, vol. 111, no. 23, pp. 8410–8415, 2014. [Online]. Available: https://www.pnas.org/content/111/23/8410