Designing High School Computer Science Curriculum to Address Real-World Biological Problems with Computational Thinking

A Major Qualifying Project
submitted to the Faculty of
Worcester Polytechnic Institute
in partial fulfillment of the requirements for the
Degree in Bachelor of Science
in
Computer Science.

By

Kirsten Hart

Date: 05/06/2021
Project Advisors:

Professor Carolina Ruiz, Advisor
Professor Elizabeth Ryder, Advisor

Abstract

All students need to have foundational computer science skills to contribute to our technology-centered society. This project supported the Bio-CS Bridge Project in promoting broader computer science interest among high school students by integrating scientific practices with computer science approaches to address real-world problems. Through backward curriculum design and incorporation of computational thinking practices, a new unit of computational problem-solving was created, and existing units were enhanced. The bridge between biology and computer science was strengthened throughout all computer science units in the Bio-CS Bridge curriculum by making connections to pollinator decline and loss of biodiversity.

Acknowledgments

I would like to thank my advisors, Professor Carolina Ruiz and Professor Liz Ryder, for their guidance and support throughout my MQP. I would also like to thank Shari Weaver of WPI's STEM Education Center for sharing her expertise on writing curriculum and reaching all learners. I would like to thank Kerri Murphy of Norton Public Schools for sharing her practical experience with teaching the Bio-CS Bridge curriculum and helping develop new lesson plan content. I would like to thank Professor Rob Gegear of UMass-Dartmouth for sharing his expertise on bumblebee ecology and conservation of pollination systems. I would like to thank Nathan Klingensmith and Joe Petitti of the Bio-CS Bridge software development team for their work designing and implementing the Bio-CS Bridge curriculum website. I would like to thank Sharada Vishwanath of Northborough Public Schools for piloting newly written learning activities. Finally, I would like to thank the rest of the Bio-CS Bridge team for providing feedback throughout my project and for developing the original curriculum content.

Table of Contents

List of Figures

List of Tables

1 Introduction

Computer science and computational thinking have become essential skills in the twenty-first century. Companies are fundamentally changing the way they operate by supporting evidence-based solutions (Dichev et al., 2016). In response to this change in the workforce, a change is also needed in K-12 curriculum. To better prepare students for proactively participating in making changes in today's digital world, all students need to have the foundational knowledge to do so. It can be beneficial to engage as many students as possible in the computational thinking skills required to solve real-world problems (Dichev et al., 2016).

According to Change the Equation's research, 7.7 million Americans use computers in complex ways every day to do their job (Education Commission of the States, 2015). Based on these results, it is apparent that the demand for advanced computing skills is great and young people need a strong foundation in computing skills to have a chance in the real world (Education Commission of the States, 2015). The educational system recognizes that there is a growing need for students to learn computer science concepts and in response school districts have been introducing computer science courses to their high schools. There has also been an increase in computer science undergraduates over the last decade (Camp et al., 2017). This seems promising that more individuals are capable of thinking computationally, but it is not enough. There is still limited opportunity to learn computer science concepts and to understand how computer science influences society; less than half of K-12 schools offer computer science courses and only 44% of high school seniors go to a school offering at least one computer science course (K12 Computer Science, n.d.). Students should not have to sign up for a computer science class in high school or university to learn computer science concepts for the first time; students should be engaged in computational problem solving and designing solutions for real-world problems in their general education K-12 courses.

Research has shown that providing students with opportunities to apply computer science skills and practices to solve real-world problems disrupts the common misconception that computer scientists solve theoretical problems in isolation (Aspray, 2016). Integrating computer science into existing STEM courses will allow more students to explore the applications of computer science. The need for computer science to be integrated into more courses in K-12 curricula is apparent, but this proves to be a challenge for educators. Designing and implementing a curriculum that effectively integrates STEM and computer science is difficult for educators because they have little understanding of the other disciplines' terminology, key concepts, tools, and approaches to learning (Yadav et al., 2016). With this understanding of the challenge educators are facing, the Bio-CS Bridge project formed a transdisciplinary team of university professors, high school teachers, and students from the computer science, biology, and education fields. The Bio-CS Bridge project is a National Science Foundation-funded project focused on building a bridge between biology and computer science courses in high school by developing an integrated curriculum that shows students how to apply computational thinking skills to real-world biological problems (Bio-CS Bridge, n.d.). Since there is a lack of computer science courses in high schools (K12 Computer Science, n.d.), the curriculum which integrates biology and computer science aims to integrate computer science standards and practices into the biology classroom and vice versa. If students are engaged in computer science within their general education courses, they will develop an understanding of the need for computational thinking in solving real-world problems (Aspray, 2016). This integration of subjects promotes broader student interest in computer science by showing students how computer science can be used to make a positive impact on their community.

The overarching goal of this project was to promote broader student interest in computer science by demonstrating how it can be used to solve real-world biological problems. More specifically, the project focused on improving the existing Bio-CS Bridge Curriculum through three main goals: 1) Strengthen the bridge between biology and computer science by making connections to pollinator decline and loss of biodiversity throughout all lessons of the computer science units. 2) Use pollinator decline and loss of biodiversity as context for expanding the coverage of core computer science concepts and standards throughout the curriculum. 3) Finalize computer science curriculum for dissemination to a wider audience via a curriculum website. To achieve these goals, we asked educators currently teaching the curriculum for feedback on how the curriculum could be improved and what other concepts students should be learning. Before this project, the computer science curriculum was very dependent on tutorials and less focused on student-teacher interactions. New learning activities were designed to supplement the tutorials to create unique and interactive lesson plans. Another focus of the efforts in this project was ensuring that the curriculum could be taught by someone with little to no computer science or biology knowledge. Lesson backgrounds and detailed lesson plans were developed to better prepare teachers to teach concepts outside of their discipline. Through working on the project, it was realized that some core computer science concepts like variables, functions, conditionals, loops, data structures, and algorithms were not explicitly addressed in one cohesive unit, so a new unit was designed to focus on computational problem-solving. As a result of this project, three modular, well-designed computer science units are ready to be implemented into high school classrooms and contributions were made to the design of a website for curriculum dissemination.

2 Background

Leading scientific organizations have been pushing for a fundamental change to biology education to better prepare students for solving real-world problems. The American Association for the Advancement of Science reported the need for students to learn how to think computationally in the context of biology (Vision & Change in Undergraduate Biology Education, 2011). Although these recommendations are directed towards the undergraduate level of education, K-12 educators support the need for change. Students are more engaged in the classroom when they are working on something that they can relate to the real world because they can see that it is important (Ryoo et al., 2013). The content students are learning becomes more relevant to them if they see connections between subjects; explicitly showing students how different subjects are connected can lead to a more engaging and meaningful educational experience (Stoll et al., 2006). To foster computational thinking in science classrooms, it makes sense to integrate across STEM curricula rather than teaching each subject in isolation from each other.

2.1 The Bio-CS Bridge Project

The Bio-CS Bridge project is a National Science Foundation-funded project seeking to develop, implement, and test a modular curriculum that integrates computational thinking with science content and practices. The curriculum is developed by a transdisciplinary team of university professors, high school teachers, and graduate, undergraduate and high school students with expertise in biology, computer science, software and database development, and education. The main goal of the curriculum design is to integrate computational thinking with other disciplines in solving real-world problems; this project specifically uses pollinator decline and loss of biodiversity as the motivating biological problem for all curricular units. Practicing biology and computer science teachers wrote the curriculum which consisted of three biology units and three computer science units. Although the curriculum was split up into six units, it was written so teachers can pick and choose which lesson plans or learning activities they would like to incorporate into their curriculum. In the biology units, students use computational tools to create models and test biological systems. In computer science units, students use biology topics as the domain for their projects (Bio-CS Bridge, n.d.).

2.2 The Beecology Project

The Beecology Project is a project which aims to develop effective conservation strategies by recruiting citizen scientists to collect ecological data on native pollinators. The decline of pollinators is a complex real-world biological problem that is the motivating problem behind the Bio-CS Bridge curriculum. Throughout the curriculum, students learn to use computer science practices to solve this ecological problem by collecting, analyzing, and interpreting data on pollinator-plant interactions. The Beecology project provides a suite of software tools for data collection, visualization, analysis, and modeling for solving this ecological problem (Beecology Project, n.d.).

2.3 Core Computer Science Concepts

Based on the Massachusetts Computer Science Standards and the Advanced Placement Computer Science Principles curriculum, a list of core computer science concepts has been identified (Massachusetts Department of Elementary and Secondary Education, 2016; College Board, 2020). No matter what computer science course students are enrolled in, they should be introduced to the following core concepts: *algorithm design, variables, functions, conditionals, loops, and complex data structures*. For algorithm design, students should understand what an algorithm is and they should be able to write an algorithm. In terms of variables, students should understand what a variable is and how to manipulate variables to solve problems. Students should also understand how to write efficient functions with parameters and return values. Program flow with conditionals is a concept students should understand as well. For iteration and loops, students need to understand the purpose of repetition of a process and be able to implement it. Finally, students should learn about complex data structures like lists and arrays; students should be able to select which data structure is most effective in solving a problem and access, update, and delete elements in these data structures with the help of conditionals and loops.

2.4 Massachusetts Computer Science Standards and Practices

The Massachusetts (MA) Department of Elementary and Secondary Education prepared a Digital Literacy and Computer Science (DLCS) Curriculum Framework in 2016 with the vision of inspiring “a larger and more diverse number of students to pursue innovative and creative careers in the future” (Massachusetts Department of Elementary and Secondary Education, 2016, 13). The ability to use and create technology to solve real-world, complex problems is now an essential literacy skill that students need to be taught during their K-12 education. The framework establishes standards for all age groups in the following core concepts organized in strands: *computing and society, digital tools and collaboration, computing systems, and computational thinking* (see Table 1). Along with the standards, there is a set of practices that are necessary for successfully meeting the state standards. These standards prepare students for success following their K-12 education by putting importance on digital literacy and computer science skills. Standards defined in this framework complement standards from other academic disciplines so they can be applied to other subjects like science, technology, engineering, and mathematics (Massachusetts Department of Elementary and Secondary Education, 2016).

2.4.1 Core Concepts

All standards in the DLCS framework are grouped into four core concepts or strands, and each strand is subdivided into topics composed of related standards (Massachusetts Department of Elementary and Secondary Education, 2016). Table 1 provides the strands and topics of the MA DLCS Curriculum Framework.

Table 1: Digital Literacy and Computer Science Standard Strands and Topics

Computing and Society [CAS]	Digital Tools and Collaboration [DTC]	Computing Systems [CS]	Computational Thinking [CT]
a. Safety and Security b. Ethics and Laws c. Interpersonal and Societal Impact	a. Digital Tools b. Collaboration and Communication c. Research	a. Computing Devices b. Human and Computer Partnerships c. Networks d. Services	a. Abstraction b. Algorithms c. Data d. Programming and Development e. Modeling and Simulation
Practices			
Connecting, Creating, Abstracting, Analyzing, Communicating, Collaborating, Research			

Adapted from the MA DLCS Curriculum Framework (Massachusetts Department of Elementary and Secondary Education, 2016)

2.4.2 Practices

The DLCS framework defines the following seven practices that students apply to solve problems: *creating, connecting, abstracting, analyzing, communicating, collaborating, and researching*. Creating is the practice focused on students engaging in designing and developing products to creatively solve problems. Connecting is focused on drawing connections between concepts. Abstracting is focused on using abstraction to make models and classify information. Analyzing is focused on being able to think critically to evaluate information. Communicating is focused on being able to accurately and concisely communicate information either through writing, visuals, or orally. Collaborating is focused on working in teams to achieve a task. Researching is focused on using digital tools to gather, evaluate, and use information.

2.4.3 Standards

Standards can be easily identified by the coding system that is used (see Appendix A). Each standard has a unique identifier comprising the grade span, strand, topic, and standard number. Throughout high school, all students should have the opportunity to demonstrate the abilities defined in the standards for their grade level. By studying these standards, students will be prepared for college and/or their career following high school.

2.5 Computational Thinking in a STEM Classroom

The Computational Thinking (CT) in Mathematics and Science Practices Taxonomy as described by Weintrop et al. is broken down into the following four categories of practices: *data, modeling and simulation, computational problem solving, and systems thinking* (2016). Each category is broken down further into a list of between five and seven practices. All practices in the taxonomy are interrelated and dependent on one another (Weintrop et al., 2016). Table 2 provides an overview of all of the practices of the taxonomy.

Table 2: Computational Thinking in Mathematics and Science Taxonomy

Data Practices	Modeling & Simulation Practices	Computational Problem Solving Practices	Systems Thinking Practices
❏ Collecting Data ❏ Creating Data ❏ Manipulating Data ❏ Analyzing Data ❏ Visualizing Data	❏ Using Computational Models to Understand a Concept ❏ Using Computational Models to Find and Test Solutions ❏ Assessing Computational Models ❏ Designing Computational Models ❏ Constructing Computational Models	❏ Preparing Problems for Computational Solutions ❏ Programming ❏ Choosing Effective Computational Tools ❏ Assessing Different Approaches/Solutions to a Problem ❏ Developing Modular Computational Solutions ❏ Creating Computational Abstractions ❏ Troubleshooting and Debugging	❏ Investigating a Complex System as a Whole ❏ Understanding the Relationships within a System ❏ Thinking in Levels ❏ Communicating Information about a System ❏ Defining Systems and Managing Complexity

Adapted from Weintrop et al., 2016.

These practices outlined in the CT Taxonomy complement the Next Generation Science Standards and state standards (*Next Gen Science Standards*, n.d.). Math and science lesson plans and learning activities can be enhanced to bring computational thinking into the classroom with these practices. The benefits of including computational thinking practices in STEM classrooms are: 1) it builds a relationship between computational thinking and the math and science domains, 2) it reaches all students, and 3) it aligns science and math education with the professional practices of these fields (Weintrop et al., 2016).

2.6 Understanding by Design - A Backward Curriculum Design Framework

Backward curriculum design is a method of developing a curriculum with the student's understanding at its core. Understanding by Design (UbD) is a framework for designing a curriculum (Wiggins & McTighe, 2011). The primary goal of UbD is developing and deepening students' understanding through purposeful curriculum planning. Traditionally teachers are pressured to address certain standards that will be assessed on state assessments and they may find themselves teaching to a textbook. Rather than relying on the structure of a textbook, backward curriculum design focuses on long-term desired results. For effective curriculum design, before planning out learning activities, curriculum designers go through a three-step process of defining desired results, specifying acceptable assessment evidence, and developing a learning plan. A successful unit plan begins with clarifying what the desired learning outcomes are and what evidence is needed to show that learning occurred. After a specific learning outcome has been identified, designers can focus on the instructional path to get learners to the desired learning destination. By following this framework when designing a curriculum, the specific lessons are developed in the context of a more comprehensive unit design.

2.6.1 The Three Stages of Backward Curriculum Design

There are three stages to the UbD framework which clearly outline the process of designing a curricular unit (Wiggins & McTighe, 2011). Figure 1 provides an overview of the UbD framework.

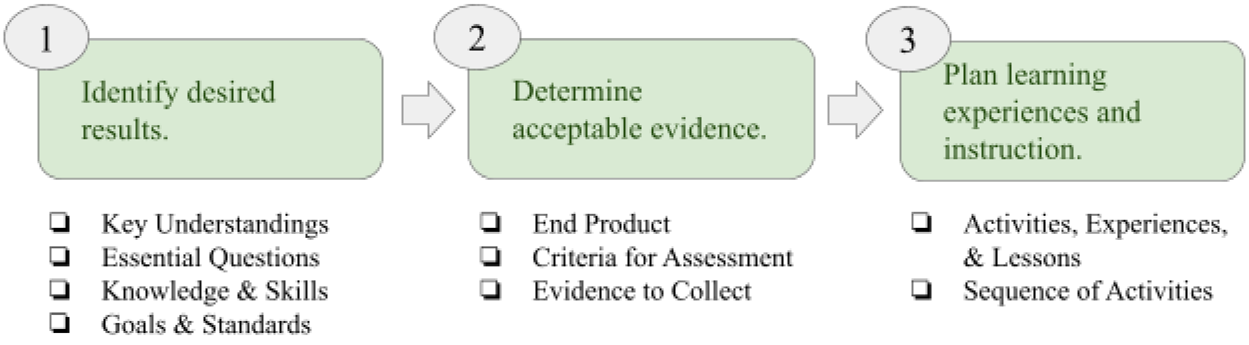

1
Identify desired results.
2
Determine acceptable evidence.
3
Plan learning experiences and instruction.
❏ Key Understandings
❏ Essential Questions
❏ Knowledge & Skills
❏ Goals & Standards
❏ End Product
❏ Criteria for Assessment
❏ Evidence to Collect
❏ Activities, Experiences, & Lessons
❏ Sequence of Activities

Figure 1. Backward Curriculum Design using the UbD Framework. Adapted from Wiggins & McTighe, 2011.

The first stage of UbD’s framework is identifying desired results. During this stage designers should determine what the long-term transfer goals are that should be targeted; what are some key skills that learners should take away from this unit? Next, designers should determine what key concepts students should understand as well as what essential questions students should be able to answer by the end of the unit. After determining long-term goals, understandings, and essential questions, designers should be able to identify which state standards will be addressed with this unit. It is important that the focus of this stage of design is not on what standards should be addressed, but rather what skills and understandings learners will take away from the unit. The standards should easily align with the skills and understandings that were identified.

The second stage of UbD’s framework is determining acceptable evidence. Now that the designer has decided what the desired results are, they can focus on what evidence is needed to prove that learners have reached the goals that have been set. This is when designers think about what products would reveal evidence that learners have gained the understandings and skills necessary. These products should be both formative and summative assessments of student learning. A clear rubric and set of specific criteria should be determined for each of the products. All assessments and product criteria should test key content and be aligned with the goals established in Stage 1.

The third stage of UbD’s framework is planning learning experiences and instruction according to the results and evidence determined in the previous two stages. Designers should think about what activities, experiences, and lessons will lead to the achievement of the results and success on the assessments. The main goal is helping students get to the desired results with increasing independence; by the end of the unit, students should need minimal assistance from their instructor with understanding how to apply key concepts to their assessment and products. Progress towards the desired results should be monitored throughout the unit so that the extent of success on the assessment and products is no surprise to the instructor or students. Most importantly, the lessons within the unit should be sequenced in a way to optimize achievement for all learners. Scaffolding is crucial to helping all learners succeed in a comfortable and welcoming learning environment.

If all three stages of UbD’s framework are strictly followed while planning the curriculum, students will be given multiple opportunities to apply their learning in meaningful contexts. Teachers will be teaching for understanding, rather than getting through all chapters of a textbook or addressing all standards to be assessed on state assessments. By the end of the unit, students should be able to autonomously explain, interpret, apply, shift perspective, empathize, and self-assess to demonstrate that they understood the long-term transfer goals set for them (Wiggins & McTighe, 2011).

2.6.2 Characteristics of Effective Curriculum Designs

Effective designs for learning include five common characteristics: *expectations, instruction, learning activities, assessment, and sequence and coherence*. The best learning designs provide clear goals and specific expectations for learners. Expectations should be communicated to students through models and examples. Instruction should be targeted in a way to facilitate, support, and guide learners to the expected performance. Learning activities should be accommodating of individuals with different learning styles, skill levels, and interests. Learning should be active and experiential to help students develop their understanding of complex content. An assessment should not be a mysterious entity because the learning activities should have prepared students to succeed in their performance goals. A good assessment should check for prior knowledge, skill level, and misconceptions while students demonstrate their understanding. The sequence and coherence of the curriculum should be cyclical to revisit ideas and have learners rethink and revise earlier ideas. Students should always be aware of the whole context, even when they are working on learning a specific key concept (Wiggins & McTighe, 2011).

2.7 Key Features of Curriculum Websites

Since the goal of the Bio-CS Bridge project is to make a difference in high school curriculum across the nation, the curriculum website must be designed to fit the needs of the teachers who will be utilizing it. There are many websites that focus on disseminating free STEM curricula. Evaluating these pre-existing curriculum websites is essential for establishing the key features that should be included in the Bio-CS Bridge curriculum website. The following bullet points outline some of the key features that teachers on the team expressed are of importance to the website design:

- ❑ Simple navigation between the unit and lesson plans.
- ❑ Filter and search through lesson plans by keyword, standard, practice.
- ❑ A roadmap of the unit (when viewing a lesson plan to see the lesson plan in a larger context).
- ❑ Clear and concise representation of lesson plan components.
- ❑ Downloadable unit or lesson plan toolkit (to include all necessary materials to teach).

3 Methodology

3.1 Project Overview

For this project, the main focus was to enhance the Bio-CS Bridge computer science curriculum and prepare it for dissemination via a website. Before this project, three computer science units were written by the Bio-CS Bridge transdisciplinary team of teachers and university faculty. During this project, two of these units (Unit 3: Web Design using HTML, CSS, and JS and Unit 4: Drawing and Animation using JavaScript) were reviewed and enhanced and an additional unit (Unit 2: Computational Problem Solving using Python) was written. Throughout this project, there was regular communication with the teachers of the Bio-CS Bridge project to discuss improvements to the curriculum. Through discussion it became apparent that the existing curriculum was too dependent on Khan Academy tutorials, more detail was needed in all aspects of the lesson plans to allow teachers with minimal computer science experience to teach the curriculum, and some core computer science concepts were not emphasized. These pieces of feedback guided this specific project's goals which were: 1) strengthen the bridge between biology and computer science by making connections to pollinator decline and loss of biodiversity throughout all lessons of the computer science units; 2) use pollinator decline and loss of biodiversity as context for expanding the coverage of core computer science concepts and standards throughout the curriculum, and 3) finalize computer science curriculum for dissemination to a wider audience via a curriculum website.

To achieve this project's goals and align with the goals of the Bio-CS Bridge project, when preparing curriculum for dissemination, each unit must contain the following elements:

- ❏ Integration between biology and computer science content throughout the majority of the lessons by emphasizing computational thinking practices.
- ❏ Precise and accurate computer science and science standards and practices to align with the learning activities of the lesson plan.
- ❏ Detailed description for lesson background, assessment evidence, and all learning activities.
- ❏ An online alternative method of instruction to accommodate remote learning (as a result of COVID-19's closure of schools).

To fulfill the requirements, as described above, for each unit, Weintrop et al.'s Computational Thinking in Mathematics and Science Taxonomy and Wiggins and McTighe's Understanding by Design Backward Curriculum Design Framework were utilized (Weintrop et al., 2016; Wiggins & McTighe, 2011).

3.2 Designing a Cohesive, Modular Curriculum "Backwards"

All seven units of the Bio-CS Bridge curriculum were designed "backward" using the Understanding by Design Curriculum Framework. The framework was adapted to ensure that the units achieved Bio-CS Bridge's goal of integrating biology and computer science standards and practices into a modular high school curriculum. Each unit was designed by following these steps:

Stage 1: Identifying Desired Results

1. Develop a unit project which uses biology content as the domain for demonstrating computer science understandings and skills.
    - Select a biology topic or concept that computer science students should learn and demonstrate in the project.
2. Determine understandings, essential questions, goals, and standards from both biology and computer science perspectives.

Stage 2: Determining Acceptable Evidence

3. Based on the goals and standards associated with the unit project, establish criteria or requirements for the project.

Stage 3: Planning Learning Experiences and Instruction

4. Generate a list of computer science concepts necessary for the project.
5. Distribute concepts into a sequence of proposed lesson plans by carefully thinking about the order that concepts should be taught.
6. Based on the proposed sequence of lesson plans, begin writing lesson plans and developing the necessary unit resources.
    - Each lesson plan is also designed based on the UbD framework where the desired result is established before determining what the specific learning activities will be.
    - To ensure that lesson plans can be stand-alone and therefore part of a modular curriculum, openings, learning activities, and closings of lesson plans must form a cohesive whole to address specific standards and objectives.
7. Once satisfied with lesson plans, review the entire unit to ensure that all necessary content is taught to prepare students for success with the unit project.

All of the steps above were followed for each of the three units that were prepared during this project. Although the two pre-existing units had projects and most of the resources were already developed, all parts of the unit were reviewed and evaluated for cohesiveness. The newly developed unit, Unit 2: Computational Problem Solving using Python, was a more extensive and step-by-step process through the UbD framework. The sequence of lesson plans and concepts was developed by finding other introductory Python curricula and consulting with the Bio-CS Bridge practicing teachers.

Lesson plans were all written using a template based on the UbD framework (see Appendix B). The Core Team of the Bio-CS Bridge project decided that all units should be broken down into multiple fifty-minute lesson plans; although class period times vary across districts and schools, a fifty-minute lesson plan should be easily adaptable to fit in a shorter or longer class period. Table 3 provides an overview of the basic structure of the lesson plan template.

Table 3: Lesson Plan Structure	
Header	Lesson Title, Unit Title, Subject, Lesson Background, Lesson Overview and Motivation, and Prior Knowledge
Desired Results	MA DLCS Standards and Practices, Science Practices, Understandings, Content Objectives, and Key Terms
Assessment Evidence	Performance Task and Key Criteria
Learning Plan	Learning Activities, Homework and Extensions, and Materials

Table 3. Condensed representation of the lesson plan template used to develop lesson plans for the Bio-CS Bridge curriculum.

3.3 Enhancing Curricular Units with Computational Thinking Practices

One of the main goals of the Bio-CS Bridge project is to bring together biology and computer science curriculum through computational thinking. Computational thinking is integrated throughout the curriculum by addressing standards from the computational thinking strand of the MA DCLS standards and practices from Weintrop's computational thinking taxonomy. When planning out a learning activity, a computational thinking practice from Weintrop et al.'s taxonomy was chosen and incorporated into the activity, if applicable (Weintrop et al., 2016). Since the curriculum is meant for a computer science classroom, most learning activities could be aligned to a practice. Specific examples of how learning activities use computational thinking will be further described in the results section of this paper.

3.4 Disseminating the Curriculum through a Website

Before this project, a prototype of the browsing page of the curriculum website was shown to the Bio-CS Bridge practicing teachers and feedback was collected. While the curriculum was being prepared for the website, a separate project was running in parallel to complete the design and implementation of the curriculum website. Although the main focus of this project was on the preparation of the curriculum, some feedback, mockups, and design advice was given to the curriculum website team. The flow of the website and key website features were designed according to the unit and lesson plan templates and teacher needs.

4 Results

4.1 Bio-CS Bridge Curriculum Structure

The structure of the Bio-CS Bridge curriculum is what makes it so unique compared to traditional high school STEM curricula; each unit of the curriculum has a connection to one or more units of the other discipline. In the biology units, students use computational tools to create models and test hypotheses about relationships between entities of an ecosystem, while in the computer science units, students use biology topics as the domain for their projects. The specific connections between units of the curriculum can be seen in Figure 2 below.

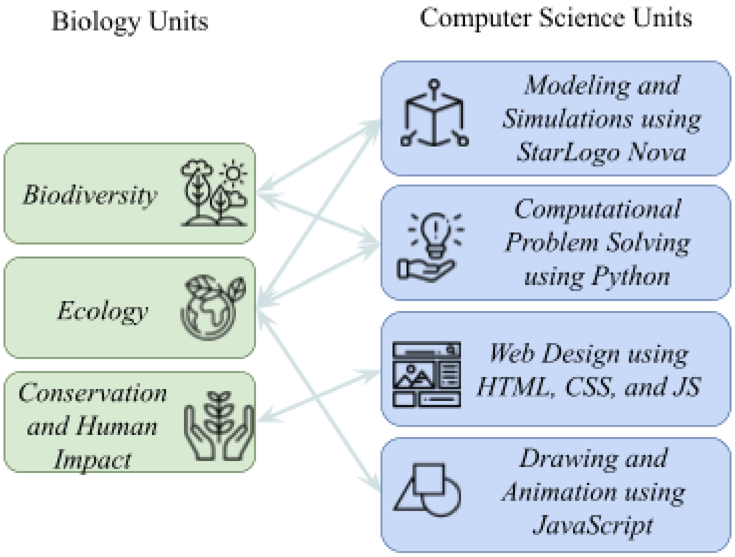

Biology Units
Computer Science Units
Modeling and Simulations using StarLogo Nova
Biodiversity
Computational Problem Solving using Python
Ecology
Web Design using HTML, CSS, and JS
Conservation and Human Impact
Drawing and Animation using JavaScript

Figure 2. A graphical representation of the Bio-CS Bridge Curriculum structure with emphasis on the bridged connections between biology and computer science units.

Since the units of the curriculum are so integrated, the coverage of standards and practices from both the biology and computer science domains is extensive. The extent of coverage of computer science standard topics and practices can be seen in Tables 4 and 5 below. For further detail in showing which lessons address specific standards within these standard topics refer to Appendix A.

Table 4: Massachusetts K-12 Computer Science Standard Strands by Topic

Standard Strand	Standard Topic	Unit 1	Unit 2	Unit 3	Unit 4
Computing and Society	Safety and Security				
	Ethics and Laws				
	Interpersonal and Societal Impact			✓	
Digital Tools and Collaboration	Digital Tools	✓		✓	✓
	Collaboration and Communication			✓	
	Research	✓		✓	
Computing Systems	Computing Devices			✓	
	Human and Computer Partnerships				
	Networks				
	Services				
Computational Thinking	Abstraction	✓			
	Algorithms		✓		
	Data		✓		
	Programming and Development		✓	✓	✓
	Modeling and Simulation	✓			

Table 4. Coverage of the Massachusetts K-12 Computer Science Standard Strands by Topic in the Bio-CS Bridge curriculum as defined in the MA DCLS Framework (Massachusetts Department of Elementary and Secondary Education, 2016).

Table 5: Massachusetts K-12 Computer Science Practices						
Creating	Connecting	Abstracting	Analyzing	Communicating	Collaborating	Researching
✓	✓	✓	✓	✓	✓	✓

Table 5. Coverage of the Massachusetts K-12 Computer Science Practices in the Bio-CS Bridge curriculum as defined in the MA DCLS Framework (Massachusetts Department of Elementary and Secondary Education, 2016).

During this project, two units (Unit 3: Web Design using HTML, CSS, and JS, and Unit 4: Drawing and Animation using JavaScript) were reviewed and updated and the curriculum's newest unit (Unit 2: Computational Problem Solving using Python) was written. In the following sections, these three units are described in-depth and the unit-to-unit connections that are shown in Figure 2 above will become evident.

4.2 The Process of Designing a Bridged Unit for the Bio-CS Bridge Curriculum

All three of the units addressed during this project were designed by following the Understanding by Design Backward Curriculum Design Framework (Wiggins & McTighe, 2011). This section will describe in detail how the steps from Section 3.2 were followed and how design choices were made. Specific examples from lesson plans will be used to help explain my thought process.

4.2.1 Identifying the Desired Results

The first step for designing any unit is deciding what the end goal is. What should students understand by the end of this unit and why should they care? Specifically for the Bio-CS Bridge curriculum, the team is focused on allowing students to develop an understanding of the importance of computational thinking practices in solving real-world problems. Since the Bio-CS Bridge project is a public curriculum, there is also a focus on expanding the coverage of MA DCLS standards (Massachusetts Department of Elementary and Secondary Education, 2016). With these main focuses, the end goal of a unit can be decided on. For example, Unit 2: Computational Problem Solving using Python was written to address a section of standards that were not already addressed by the rest of the curriculum. The theme of the set of standards selected for this unit is computational thinking and algorithms, so the end goal of the unit is to provide students with experience in designing algorithms to solve real-world problems. Once an overarching end goal is decided on, a unit project is planned. The unit project is where the biology connection is truly integrated into the unit. Although goals and understandings are set for students in regards to computer science concepts, this is where the biology understandings are incorporated. For example, in Unit 2 students develop an understanding of how a combination of computer science skills can be used to analyze a large dataset while also developing an understanding of the importance of citizen science, data pooling, and data analysis in conservation. Following the design of the unit project, essential questions, understandings, and goals can be set for the unit.

4.2.2 Furthering the Design of the Unit Project by Determining Acceptable Evidence

Once a unit project, essential questions, understandings, and goals are decided on, the specific requirements, as well as formative and summative assessment evidence, can be determined. It is at this time that an example end product is made and a list of requirements is made based on this example. For instance, in Unit 2 a few versions of the unit project were made, and based on the intended difficulty of the project a set of requirements was written. This was a preliminary set of requirements and a final set of requirements was determined after learning activities were decided on.

4.2.3 Planning Learning Activities Based on the End Goal

The unit project should be a synthesis of what students learn throughout the unit. Students should not be surprised when they are given the project because it should be the application of the skills that they have been developing throughout the unit. With this in mind, a cohesive set of lesson plans can be designed. Based on the expected outcome of the unit, what skills should students develop during the unit to be successful in transferring their knowledge to the unit project? The preliminary list of requirements is useful in planning out learning activities because all of the requirements need to be addressed during the unit. A list of key concepts to be addressed in the unit is created and derived from the list of requirements. The sequence of concepts is put in a logical order by referring to other curricula; it is common for most computer science concepts to be taught in a certain order, so the sequence of concepts is relatively consistent throughout computer science curricula. For example, depending on the extent of coverage of concepts within the unit the sequence of concepts is usually something similar to the following: file input and output, variables and data types, operators and comparisons, conditionals, iteration, and functions. A list of concepts can then be divided up into days of lesson plans. The depth at which a concept can be addressed is dependent on the amount of time allotted for that concept. For the Bio-CS Bridge curriculum, the team understands that teachers may not necessarily have sufficient time to teach a full unit to their class, therefore the units are modular to allow teachers to easily integrate all or parts of a unit into their own curriculum.

With a layout of concepts into days, lesson plans can be planned out. Planning a single lesson plan follows the same process as planning an entire unit where the end goal is determined, then the assessment evidence, then the actual sequence of learning activities. With the idea of the concept to be addressed in the lesson plan, a set of understandings and objectives should be determined; the understandings and objectives for the lesson should be specific to what the lesson will be addressing but also related to the goals of the unit. Each lesson needs to have a purpose; when thinking about a lesson plan, it is important to answer the "why should my students care?" question because this will put the lesson plan in the context of the rest of the unit.

Students need to be assessed in every lesson, so evidence of learning should be collected during every lesson. The assessment of learning leads to planning the actual learning activities of the lesson plan.

All lesson plans have an opener, at least two learning activities, and a closing. The learning activities will correspond to the assessment evidence for the lesson, but the opener and closing are important for emphasizing the understandings of the lesson plan. I like to use the opener as an introduction to the concept that students will be learning during the subsequent learning activities. For example, the first lesson of Unit 2 is called What's an Algorithm? (which is shown in Figure 3 below). For the opener, students are given the task of solving a maze and writing out the steps to get from start to finish. When students are given this task, they may not understand the importance of it, however, in the closing of the lesson students discuss what an algorithm is and can make that connection back to the importance of the opening activity. The opener is a way to get students to think in a similar manner to how they will have to think during the rest of the lesson. For the closing activity, I like to have whole-class discussions so that students can share what they learned during the previous learning activities with the rest of the class; closing activities are important for collecting evidence of student understanding and determining if students successfully learned the concepts.

Most learning activity ideas in this unit came from other computer science curricula, tutorials, or personal experience. In general, most lesson plans go through the same sequence of learning activities; the opener is either a partner activity with a discussion or an individual exploratory activity which is followed by at least two learning activities that are either a student-driven template or teacher demonstration to learn a new concept which is followed by a closing whole-class discussion. This design of a single lesson plan makes the curriculum modular so that teachers can teach one lesson plan or learning activity without having to teach the previous or next one in the sequence. A complete lesson plan is shown below in Figure 3.

Lesson Plan Title: 1 - What's an Algorithm?

Subject/Course: Computer Science

Unit: Computational Problem Solving using Python **Grade Level: High School**

Lesson Background:

An **algorithm** is a set of unambiguous rules or instructions to achieve a particular objective. It is important to know that an algorithm is not necessarily for computers; there are many algorithms for other applications like mathematics. The goal of an algorithm is to be able to solve a problem in a repeatable way. For example, one might write an algorithm for solving a word search. This algorithm should be able to be followed by anyone and it should work for any given word search. In this unit, students will focus on learning core computer science concepts, while learning Python. **Python** is a general-purpose programming language that can be used for software development, math calculations, system scripting, and web development.

Overview of and Motivation for Lesson:

In this lesson, students will be introduced to Python and shown some of the capabilities of the language. Students will also be introduced to the concepts of an algorithm and pseudocode by solving simple search and sort problems (word search and card sorting).

Prior Knowledge:

N/A

Stage 1 - Desired Results

Standard(s):

- **9-12.CT.b.2** Represent algorithms using structured language, such as pseudocode.
- **9-12.CT.d.10** Use an iterative design process, including learning from making mistakes, to gain a better understanding of the problem domain.

Computer Science Practice(s):

- Creating
- Abstracting
- Analyzing

Science Practice(s):

- Using mathematics and computational thinking

Understanding(s):

Students will understand that . . .

- Algorithms are not always for computers.
- The goal of an algorithm is to be able to solve a problem in a repeatable way.

Content Objectives:

Students will be able to . . .

- List what Python can be used for.
- Define algorithm.
- Construct algorithms for completing simple puzzles.
- Identify situations where algorithms are used.

Key Vocabulary

- Python: a general-purpose programming language.
- programming language: formal language used to give computer instructions.
- computer science: the study of computers and algorithmic processes, including their principles, hardware and software designs, applications, and their impact on society.
- algorithm: a set of unambiguous rules or instructions to achieve a particular objective.
- pseudocode: an informal high-level description of the operating principle of a computer program or other algorithm.

Stage 2 - Assessment Evidence

Performance Task or Key Evidence

- Write and debug pseudocode for completing the assigned task (word search or sorting of cards).

Key Criteria to measure Performance Task or Key Evidence

- Pseudocode can be followed by a peer and the task can successfully be completed.

Stage 3 - Learning Plan

Do Now/Bell Ringer/Opener: What is Python? (5 - 10 mins)

- Demonstration: utilize a template (see Appendix C) to demonstrate some basic functionality of Python.
  - Make sure to mention what Python can be used for: software development, math calculations, system scripting, and web development. In this unit, we'll be focusing on solving problems with Python.
  - Use the template to guide your discussion of some of the functionality of Python. Feel free to make changes and encourage students to ask questions. Also ask students what they would expect the code to do before running it.
- Use W3Schools Introduction to Python as another resource for listing Python's functionality.

Online Version	
	• Create a video demonstrating the functionality of Python.

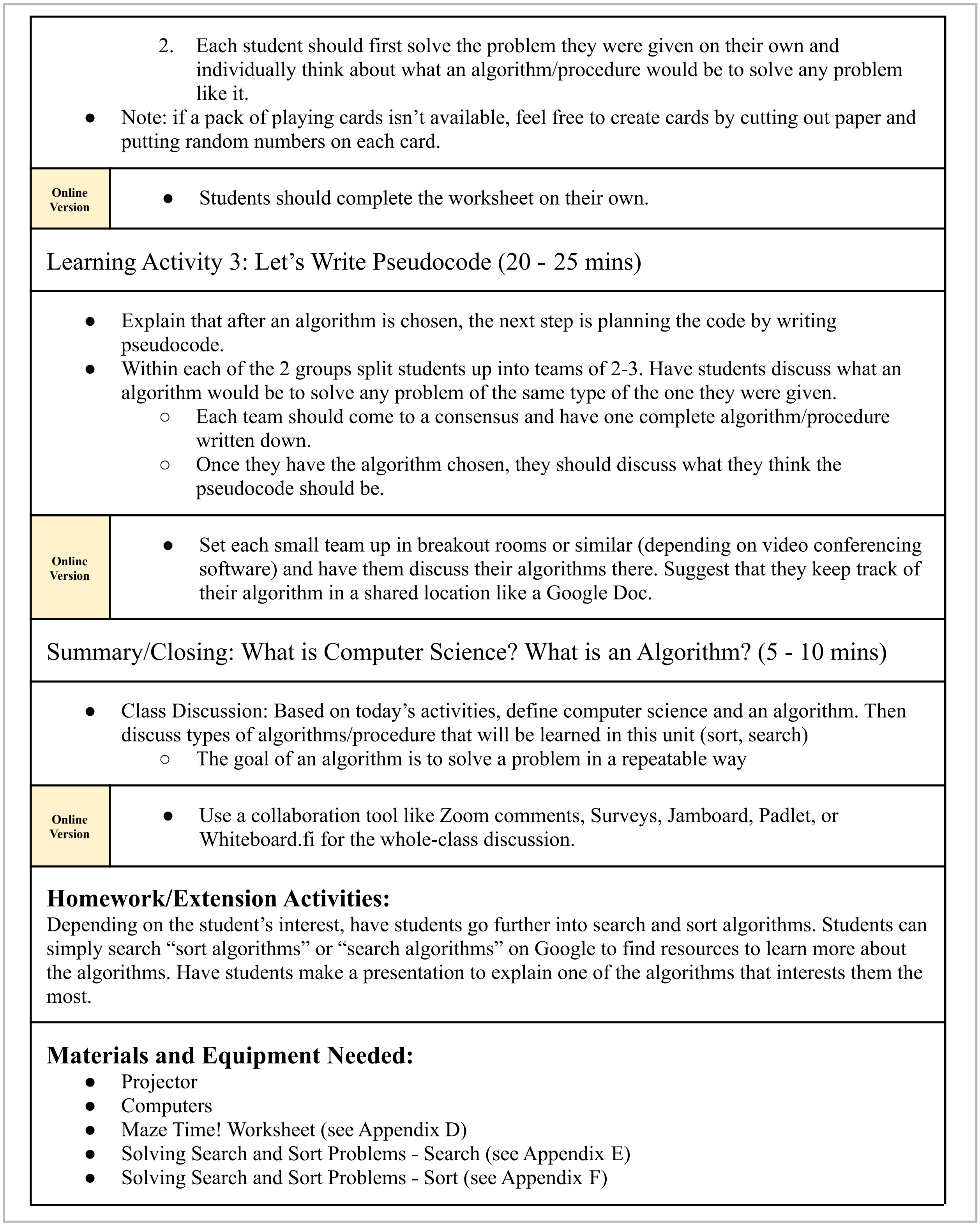
Learning Activity 1: Maze Time! (5 - 10 mins)

- Give students a worksheet (see Appendix D) to solve a simple maze. Students should make sure to write out steps to get from the entrance of the maze to the exit of the maze. They should use left, right, and straight as the possible steps.

Online Version	
	• Students should complete the worksheet on their own.

Learning Activity 2: Solving Search and Sort Problems (15 - 20 mins)

- Students will be working on writing simple algorithms for either a search or sort problem. Half the class will be assigned a word search and the other half will be asked to sort 5 numeric cards.
  1. Split the class up into 2 groups (one half will get a word search and the other half will get 5 cards each).



2. Each student should first solve the problem they were given on their own and individually think about what an algorithm/procedure would be to solve any problem like it.

- Note: if a pack of playing cards isn't available, feel free to create cards by cutting out paper and putting random numbers on each card.

Online Version	● Students should complete the worksheet on their own.

Learning Activity 3: Let's Write Pseudocode (20 - 25 mins)

- Explain that after an algorithm is chosen, the next step is planning the code by writing pseudocode.
- Within each of the 2 groups split students up into teams of 2-3. Have students discuss what an algorithm would be to solve any problem of the same type of the one they were given.
  - Each team should come to a consensus and have one complete algorithm/procedure written down.
  - Once they have the algorithm chosen, they should discuss what they think the pseudocode should be.

Online Version	● Set each small team up in breakout rooms or similar (depending on video conferencing software) and have them discuss their algorithms there. Suggest that they keep track of their algorithm in a shared location like a Google Doc.

Summary/Closing: What is Computer Science? What is an Algorithm? (5 - 10 mins)

- Class Discussion: Based on today's activities, define computer science and an algorithm. Then discuss types of algorithms/procedure that will be learned in this unit (sort, search)
  - The goal of an algorithm is to solve a problem in a repeatable way

Online Version	● Use a collaboration tool like Zoom comments, Surveys, Jamboard, Padlet, or Whiteboard.fi for the whole-class discussion.

Homework/Extension Activities:
Depending on the student's interest, have students go further into search and sort algorithms. Students can simply search "sort algorithms" or "search algorithms" on Google to find resources to learn more about the algorithms. Have students make a presentation to explain one of the algorithms that interests them the most.

Materials and Equipment Needed:

- Projector
- Computers
- Maze Time! Worksheet (see Appendix D)
- Solving Search and Sort Problems - Search (see Appendix E)
- Solving Search and Sort Problems - Sort (see Appendix F)

Figure 3. Lesson Plan for Unit 2: Computational Problem Solving using Python Lesson 1: What's an Algorithm?

As mentioned earlier, Unit 3: Web Design using HTML, CSS, and JavaScript and Unit 4: Drawing and Animation using JavaScript were very dependent on Khan Academy tutorials. The use of tutorials in computer science classrooms seems to be a common practice, however from personal experience with teaching and learning computer science it was decided that tutorials should not be the sole method of instruction for new concepts. With this decision, templates were created for the majority of Unit 3 and Unit 4 lesson plans. These templates serve as a guide for students to explore new concepts rather than being told what code they should be typing out. Students are not encouraged to grow and think outside of the box when going through a tutorial because the problem addressed is already solved. The supplementary templates that were created for these units encourage students to solve smaller tasks in more of an exploratory way; students need to be able to transfer the knowledge they are learning to new situations, so they should not be told explicitly what to do.

4.2.4 Finalizing the Project Description and Requirements

After all lesson plans are written, the project description and requirements are revisited to ensure that they match what students are taught during the unit. A challenge with writing lesson plans for a general audience is the level of prior knowledge and the learning abilities of the classroom are unknown. Not only do the lesson plans have to be scaffolded to reach all learners, but the projects also have to be scaffolded. All three of the unit projects have templates that students can utilize. To accommodate different levels of computer science knowledge, there is more than one template for each project. For example, in Unit 3 there is a template for biology students to complete a web page. Since biology students may complete the project with limited HTML, CSS, and JavaScript background, this template has comments to explain each line of code, and all of the needed tags are already there. On the other hand, there is a template for computer science students to use which has limited comments to instruct students. This cyclical “backward” curriculum design process ensures that students are well prepared for success in transferring knowledge to their unit project.

4.2.5 Incorporating Computational Thinking Practices

Computational thinking practices from Weintrop et al.’s Computational Thinking in Mathematics and Science Taxonomy were incorporated throughout the curriculum (Weintrop et al., 2016). Since these units are all meant for a computer science classroom, most of the learning activities were naturally accounting for at least one of the computational thinking practices. For example, in the first lesson of Unit 2: Computational Problem Solving using Python, the third learning activity that students complete is called Let’s Write Pseudocode (see Figure 3 above). In this activity, students have to write pseudocode for either a sort or search algorithm that they developed in the previous activity. This specific learning activity is an example of developing modular computational solutions and creating computational abstractions because students are representing the process of solving the problem in a more general form. By developing an algorithm and then writing the pseudocode, students are identifying, creating, and using computational abstractions to develop a modular computational solution. When creating a unit project, computational thinking practices are also incorporated naturally. The project for Unit 2 addresses many of the computational thinking practices, including manipulating data, analyzing data, visualizing data, programming, developing modular computational solutions, creating computational abstractions, troubleshooting and debugging, and understanding relationships. For this project, students are given the task of analyzing flower-bee interaction data to come to an evidence-based conclusion, so many of the data and computational problem-solving practices are addressed. Students are manipulating and analyzing data to develop a modular solution while creating abstractions to develop a better understanding of the relationship between bumblebees and flowers. Each unit design has the specific practices incorporated into the unit specified in Tables 6, 7, and 8.

4.3 Computational Problem Solving using Python Unit

Computational Problem Solving using Python is the newest unit of the Bio-CS Bridge curriculum. It was designed during this project. This unit focuses on teaching students core computer science concepts such as lists, arrays, dictionaries, loops, conditionals, and functions. During the unit students complete many small tasks to learn these concepts to prepare them for their unit project. To demonstrate their understanding and mastery of the concepts, students are given the task of analyzing flower-bee interaction data from the Beecology database to come to an evidence-based conclusion. The design of the unit based on the UbD Framework is below in Table 6.

Table 6: Computational Problem Solving using Python Unit Design	
Understanding by Design Framework: Stage 1 - Desired Results	
Key Understandings	❏ The goal of an algorithm is to be able to solve a problem in a repeatable way. ❏ Variables are used to store values and are important in many applications. ❏ Computers can be directed to make decisions to perform different tasks based on conditions in the algorithms. ❏ Loops can be used to repeat a block of code that needs to be executed multiple times. ❏ Functions can make groups of code reusable. ❏ A combination of variables, conditionals, loops, functions, and complex data structures can be used to analyze a large dataset.
Essential Questions	❏ What is a program? ❏ How does a computer operate? What are the 4 major steps of computer operation? ❏ How do the rules of logic apply to computer programming? ❏ What are the applications of programs that use conditional statements? ❏ What are the applications of programs that use loops? ❏ What is the relationship between functions and algorithms? ❏ What are some examples of some Python libraries?

	❑ What is debugging? ❑ How can the concepts learned in this unit be applied to solve real-world problems?
Goals & Standards	❑ **9-12.CT.b.1** Recognize that the design of an algorithm is distinct from its expression in a programming language. ❑ **9-12.CT.b.2** Represent algorithms using structured language, such as pseudocode. ❑ **9-12.CT.c.2** Create an appropriate multidimensional data structure that can be filtered, sorted, and searched (e.g., array, list, record). ❑ **9-12.CT.c.4** Analyze a complex data set to answer a question or test a hypothesis (e.g., analyze a large set of weather or financial data to predict future patterns). ❑ **9-12.CT.d.1** Use a development process in creating a computational artifact that leads to a minimum viable product and includes reflection, analysis, and iteration (e.g., a data-set analysis program for a science and engineering fair, capstone project that includes a program, term research project based on program data). ❑ **9-12.CT.d.2** Decompose a problem by defining functions, which accept parameters and produce return values. ❑ **9-12.CT.d.3** Select the appropriate data structure to represent information for a given problem (e.g., records, arrays, lists). ❑ **9-12.CT.d.5** Use appropriate looping structures in programs (e.g., FOR, WHILE, RECURSION). ❑ **9-12.CT.d.6** Use appropriate conditional structures in programs (e.g., IF-THEN, IF-THEN-ELSE, SWITCH). ❑ **9-12.CT.d.7** Use a programming language or tool feature correctly to enforce operator precedence. ❑ **9-12.CT.d.8** Use global and local scope appropriately in program design (e.g., for variables). ❑ **9-12.CT.d.9** Select and employ an appropriate component or library to facilitate programming solutions [e.g., turtle, Global Positioning System (GPS_, statistics library). ❑ **9-12.CT.d.10** Use an iterative design process, including learning from making mistakes, to gain a better understanding of the problem domain. ❑ **9-12.CT.d.11** Engage in systematic testing and debugging methods to ensure program correctness. ❑ **9-12.CT.d.12** Demonstrate how to document a program so that others can understand its design and implementation.
Understanding by Design Framework: Stage 2 - Assessment Evidence	
End Product	❑ Utilize variables, conditionals, functions, and complex data structures to analyze flower-bee interaction data to answer a question of interest.
Criteria for Product	❑ A well-written plan with pseudocode for all proposed functions. ❑ A written conclusion based on data analysis. ❑ Make sure to compare the conclusion with results from Floral Network Tool data analysis. ❑ Minimum Code Requirements: *at least 1* conditional, *at least 1* for loop, *at least 1* nested loop, and *at least* 1 function. ❑ All code should be sufficiently commented.
Evidence	❑ Implementation of various algorithms and functions throughout lessons. ❑ Sufficient analysis to provide an evidence-based conclusion on selected bee-flower data.

Understanding by Design Framework: Stage 3 - Learning Plan	
Learning Activities	1. What’s an Algorithm? a. What is Python? b. Maze Time! c. Solving Search and Sort Problems d. Let’s Write Pseudocode e. Class Discussion - What is Computer Science? What is an Algorithm? 2. Introduction to Variables, Input & Output, and Math Operators a. Basic Computer Operation b. Storing Values in Variables c. Make a Chat Bot with User Input d. Class Discussion - Computer Operation Process with Chat Bots 3. Boolean, Comparisons, Logical Operators & Conditionals a. Simon Says and Conditionals b. Comparison & Logical Operators c. Let’s Write Conditional Statements d. An Unpredictable Chat Bot e. Class Discussion - Custom Error Messages 4. Iteration with Loops a. Class Discussion - What’s a Flowchart? b. Flowchart to While Loop c. Writing While Loops d. Counters 5. More Data Types (Lists and Arrays) a. Why Lists? b. Let’s Make a List! c. Two-Dimensional Lists (aka Arrays) d. Class Discussion - Why Use Arrays? 6. Processing Lists and Arrays with Loops a. Sorting a List b. What’s a For Loop? c. Traversing through a 2D Array d. Utilizing 2D Arrays e. Class Discussion - For Loops vs. While Loops 7. Even More Data Types (Tuples and Dictionaries) a. Storing Larger Amounts of Data b. Let’s Make a Tuple! c. Let’s Make a Dictionary! d. Class Discussion - How to Select the Proper Data Structure for a Collection 8. Functions a. Class Discussion - What’s a Function? b. Introduction to Functions c. Writing Simple Functions d. Class Discussion - Why Use Functions? 9. Libraries, Debugging, & Documentation a. Explore Python Libraries b. How Should I Debug My Code? c. Debugging d. Writing Quiz Questions 10. End of Unit Project - Bee-Flower Interaction Data Analysis a. Introduction to the Project b. Creating a Plan

	c. Implementing the Plan d. End of Unit Closure
Computational Thinking in Mathematics and Science Taxonomy	
Practices	❑ Data: Manipulating Data, Analyzing Data ❑ Computational Problem Solving: Preparing Problems, Programming, Choosing Tools, Assessing Solutions, Developing Solutions, Creating Abstractions, Troubleshooting & Debugging ❑ Systems Thinking Practices: Understanding Relationships

Table 6. Representation of the Computational Problem Solving using Python Unit split into stages of the Understanding by Design Framework and the Computational Thinking in Mathematics and Science Taxonomy (Wiggins & McTighe, 2011; Weintrop et al., 2016).

Throughout this unit, students learn how to use variables, functions, conditionals, and loops to store and analyze data. Each of the requirements for the project is directly addressed in at least one lesson plan. To complete the project students go through a four-step process of 1) understanding the task, 2) making a plan, 3) implementing the plan, and 4) writing a conclusion. In the first step, students try to gain a good understanding of the task at hand which leads them into the second step which is coming up with a plan to complete the task. In this step, students have to write pseudocode for the functions they deem necessary for completing the task. With a well-developed plan, students are prepared for implementing their plan and writing the code. Once the code is written students are asked to write a conclusion and compare their results to their findings in the Floral Network Tool. By the end of this unit, students can demonstrate how to analyze a large dataset through the use of variables, functions, conditionals, and loops.

Students are introduced to the idea of citizen science and pollinator conservation during their unit project. To gain a better understanding of the dataset and the purpose of their project, students explore data visualization tools available on the Beecology website. After developing an understanding of the task, students review the columns of the dataset and come up with an investigable question to answer. For instance, a student may choose to compare bumblebee tongue length with flower shape to investigate their relationship. Once students have selected their columns of interest, they use their newly learned computational thinking skills to answer the question. Through the use of lists, arrays, conditionals, loops, and functions students manipulate the data to understand the relationship between the two columns. Students can replicate the results that they found on the Beecology website with their individually designed program. This unit project is a great example of how the Bio-CS Bridge curriculum puts computer science in context by using biology as the motivating problem.

4.4 Web Design using HTML, CSS, and JS Unit

Web Design using HTML, CSS, and JS is a unit that focuses on web design and implementation by teaching students the basics of HyperText Markup Language (HTML), Cascading Style Sheets (CSS), and JavaScript (JS). Students learn how to add content to web pages using HTML, how to style and format the content of a webpage using CSS, and how to make a web page interactive with JavaScript. During the unit, students complete work with a partner on a practice website to learn these concepts to prepare them for their unit project. To demonstrate their understanding and mastery of the concepts, students are given the task of designing and implementing an informative website based on their ecological research. The design of the unit based on the UbD Framework is below in Table 7.

Table 7: Web Design using HTML, CSS, and JS Unit Design

Understanding by Design Framework: Stage 1 - Desired Results	
Key Understandings	❏ HTML, CSS, and JavaScript can be used together to create a basic website ❏ HTML is used to set up the structure of a webpage with titles, headings, paragraphs, and other content like images and videos. ❏ CSS is used to edit the style of a webpage by changing the webpage's attributes - like fonts, layouts, and colors. ❏ JavaScript is used to add behavior to web pages like buttons. ❏ Web design can be used to present biological data and information.
Essential Questions	❏ What's a website and its purpose? ❏ What features should an effective website have? ❏ What ecology topic are you most passionate about? ❏ How can Hypertext Markup Language (HTML), Cascading Style Sheets (CSS), and JavaScript (JS) be used to create a basic, styled, interactive website based on an ecological issue of the student's choice? ❏ Which style properties work best to communicate the ecology topic of choice? ❏ Why should websites allow for user interactivity? ❏ What are the advantages of using CSS within a website's design? ❏ How does the use of design principles, website design analysis, and good communication skills help a web designer create an effective website for their audience?
Goals & Standards	❏ **9-12.CAS.c.7** Identify ways to use technology to support lifelong learning. ❏ **9-12.DTC.a.1** Use digital tools to design and develop a significant digital artifact (e.g., multi-page website, online portfolio, simulation). ❏ **9-12.DTC.b.1** Communicate and publish key ideas and details to a variety of audiences using digital tools and media-rich resources. ❏ **9-12.DTC.c.1** Generate, evaluate, and prioritize questions that can be researched through digital resources or tools. ❏ **9-12.DTC.c.4** Gather, organize, analyze, and synthesize information using a variety of digital tools. ❏ **9-12.DTC.c.5** Create an artifact that answers a research question, communicates results and conclusions, and cites sources. ❏ **9-12.CS.a.2** Examine how the components of computing devices

	are controlled by and react to programmed commands. ❏ **9-12.CT.d.10** Use an iterative design process, including learning from making mistakes, to gain a better understanding of the problem domain. ❏ **9-12.CT.d.11** Engage in systematic testing and debugging methods to ensure program correctness.
Understanding by Design Framework: Stage 2 - Assessment Evidence	
End Product	❏ An informative website based on ecological research
Criteria for Product	❏ HTML requirements: ❏ a title ❏ 2 headings ❏ 4 paragraphs ❏ 2 images ❏ 1 list ❏ 1 id tag ❏ 1 class ❏ at least 2 links to related websites (all of your sources should be referenced - where you get your information from) ❏ 2 internal links (hint: a link to another location within your website) ❏ 1 table ❏ comments describing major sections of the file ❏ use a span tag to apply specific CSS to a portion of a paragraph ❏ use 2 div tags to group several HTML elements and apply CSS to that div ❏ add a <script> tag to your index.html file to link to your external JavaScript (.js) file ❏ call your JavaScript function from the <body> of your index.html file ❏ CSS requirements: ❏ 2 text colors ❏ bold text ❏ italicized text ❏ 2 background colors ❏ incorporate a new CSS font-family ❏ use two different CSS font-size ❏ use two different CSS font-style ❏ use two different CSS line-heights ❏ use two different CSS text-align values ❏ use two different CSS text-decoration ❏ use width, height and overflow-x, and overflow-y properties ❏ use the margin, border, and padding properties to create colored borders around an HTML element ❏ use the float property to make text wrap around an image ❏ have elements of your page use fixed, relative, and absolute positioning ❏ use z-index to control how images are on top of each other ❏ move all CSS to an external style sheet ❏ JS requirements: ❏ use a button and JavaScript to change HTML content ❏ use a button and JavaScript to change an image as a result of a button press ❏ use a button and JavaScript to change the style of an HTML element ❏ use a button and JavaScript to hide an HTML element as a result of a button press

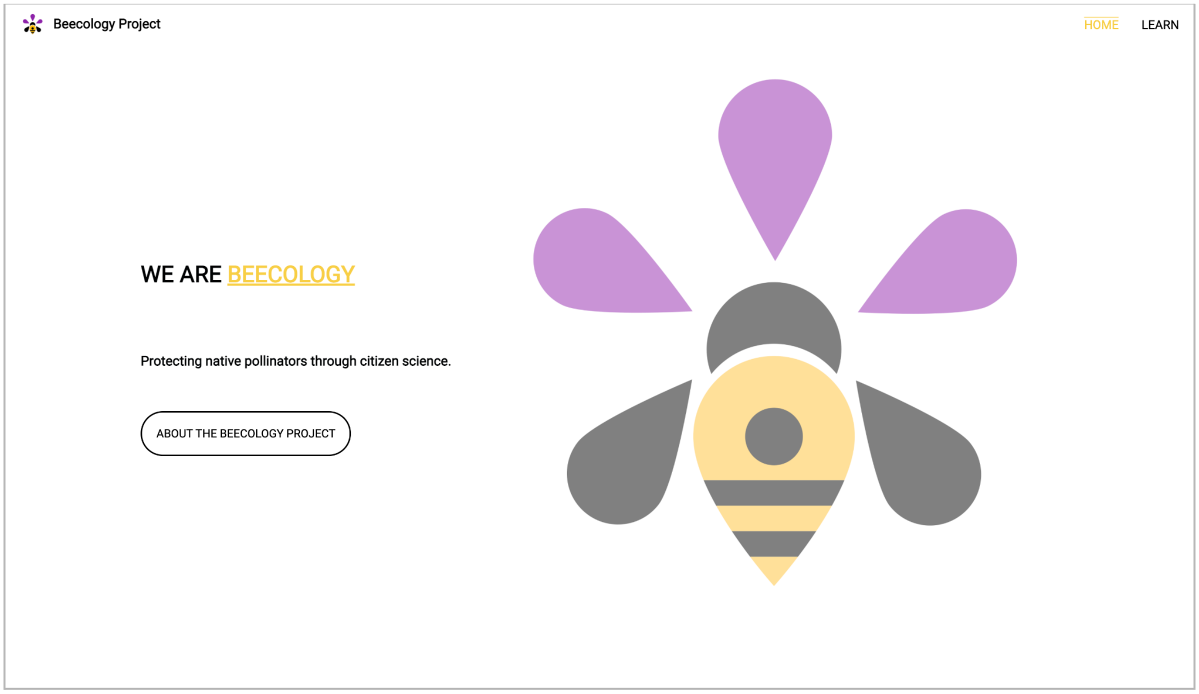
	❑ use a button and JavaScript to display hidden HTML as a result of a button press ❑ add your Javascript function to the external JavaScript file (script.js)
Evidence	❑ Partner designs and websites with new content added from each sequential lesson. ❑ Beneficial peer feedback creates a learning community and improves outcomes. ❑ Sufficient and correct content resulting from biology research. ❑ Culminating individual website unit project including biology content and all web design criteria.
Understanding by Design Framework: Stage 3 - Learning Plan	
Learning Activities	1. Introduction to Ecology Website a. Class Discussion - What's a Website? b. Brief Introduction to HTML, CSS, and JS Unit c. Begin Web Design d. Class Discussion - Pair Programming 2. Introduction to Web Design a. Finalize Design b. Feedback for Design c. How to Use Resources to Learn CS Skills d. Importance of Informational Websites 3. Introduction to HTML a. Practice using Resources b. Introduction to repl.it for Web Development c. Exploring the Structure of HTML Files d. Introduction to HTML (General Structure & Tags) e. Class Discussion - HTML Tags 4. More HTML (List, Links, & Tables) a. Explore More Website Examples b. Exploring HTML Tags c. More HTML Tags d. Class Discussion - Usage of Tags 5. Introduction to CSS a. Evaluate Website Examples b. How CSS Works to Style a Website c. Styling Feedback and Discussion 6. More CSS (Text & Layout Properties) a. CSS Style Brainstorming b. Explore More About CSS c. CSS Text Properties d. CSS Layout Properties e. Class Discussion - CSS Design Discussion 7. Introduction to JavaScript a. Interactive Website b. What is JavaScript? c. Exploring JS d. Adding a Push Button with JS e. Catch-Up f. Gallery Walk 8. End of Unit Project - Ecology Website a. Demonstrate How to Inspect HTML & CSS in a Browser

	b. Select a Research Topic c. Web Design d. Build the Website e. Class Discussion - End of Unit Closure
Computational Thinking in Mathematics and Science Taxonomy	
Practices	❑ Computational Problem Solving: Programming, Developing Solutions, Troubleshooting & Debugging

Table 7. Representation of the Web Design using HTML, CSS, and JS Unit split into stages of the Understanding by Design Framework and the Computational Thinking in Mathematics and Science Taxonomy (Wiggins & McTighe, 2011; Weintrop et al., 2016).

Throughout this unit, students learn new biology content by completing two separate ecological research projects while simultaneously learning how to communicate their findings on a web page. Each of the requirements for the unit project is directly addressed in at least one lesson plan. To complete the unit project students go through a three-step process of 1) researching the ecological topic of choice, 2) designing the webpage, and 3) implementing the design. Once students are ready to implement their design, they complete that one step in four stages which are aligned with what they learn during the learning activities. In stage one, students add basic content to their webpage like a title, headings, paragraphs, images, lists, and then add some styling to this content by adding text colors, bold and italicized text, and background colors. In stage two, students add more content with links and tables and then more complex formatting by adding text properties like font size, font style, line height, among others. Students also add behavior to their website, at this point, by adding in buttons. In stages three and four, students finish their website by adding in spans and divs, layout properties, and separation of HTML, CSS, and JavaScript into separate files. By the end of this unit, students can demonstrate how HTML, CSS, and JavaScript can be used together to create a well-formatted, interactive website.

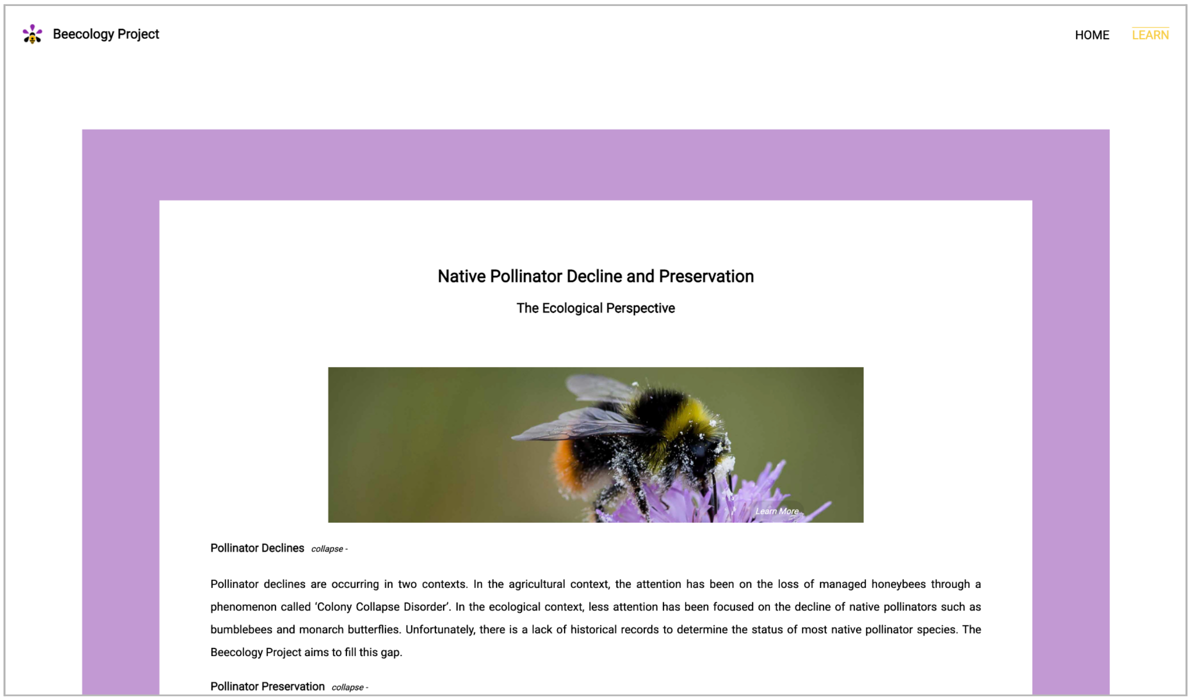
To demonstrate exemplary high expectations for the unit project, students are shown an example project created by the Bio-CS Bridge team. This example project is a website inspired by the Beecology website. It shows students how all project requirements can be met by making an appealing, interactive, and informative website. In Figure 4 below, some of the key features of the example project are shown. There is a navigation bar that fulfills the requirement of internal links and also makes the website look more like an actual website. In Figure 5 below, there are more interactive features like the ‘Learn More’ button on the bee picture and the ‘collapse -’ buttons near the section headers. The many features of this example website are given to students so that they can think of ways to make their website look appealing and easily interactive for users. This unit project is another good example of how the Bio-CS Bridge curriculum puts computer science in context by giving students a topic from the biology domain as the focus of their project. Figures 4 and 5 show the example project that students are given for Unit 3.



WE ARE BEECOLOGY

Protecting native pollinators through citizen science.

Figure 4. The “Home” page of the Unit 3 Example Project.



Native Pollinator Decline and Preservation

The Ecological Perspective

Pollinator Declines

Pollinator declines are occurring in two contexts. In the agricultural context, the attention has been on the loss of managed honeybees through a phenomenon called ‘Colony Collapse Disorder’. In the ecological context, less attention has been focused on the decline of native pollinators such as bumblebees and monarch butterflies. Unfortunately, there is a lack of historical records to determine the status of most native pollinator species. The Beecology Project aims to fill this gap.

Pollinator Preservation

Figure 5. The "Learn" page of the Unit 3 Example Project.

4.5 Drawing and Animation using JavaScript Unit

Drawing and Animation using JavaScript is a unit that focuses on the use of the ProcessingJS library to make simple drawings and animations. During this unit, students learn about variables, conditionals, and functions while illustrating and animating. Students complete many small tasks to learn these concepts to prepare them for their unit project. To demonstrate their understanding and mastery of the concepts, students are given the task of drawing and animating a phase of either the bumblebee or the flower life cycle. At the end of the unit, all phases of each cycle are brought together and students analyze the life cycles to see where they overlap and how dependent bees are on flowers and vice versa (see Figure 6 below). The design of the unit based on the UbD Framework is presented below in Table 8.

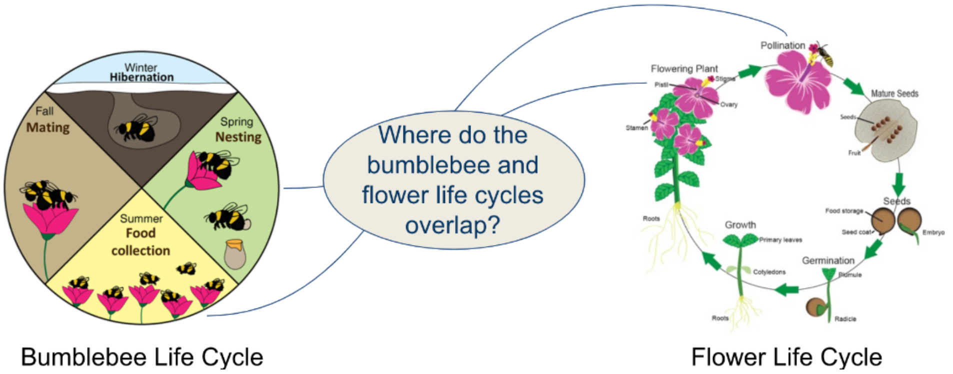


Figure 6. Drawing and Animating a Life Cycle Project.

Table 8: Drawing & Animation using JavaScript Unit Design	
Understanding by Design Curriculum Framework: Stage 1 - Desired Results	
Key Understandings	❏ Variables are used to store values and are important in many applications, such as animation. ❏ Functions can make groups of code reusable. ❏ Drawings can be animated in JavaScript by using the draw() function. ❏ If statements let a program decide whether or not to execute a block of code.
Essential Questions	❏ What is programming and what are some applications of programming? ❏ How can JavaScript be used to draw and animate simple drawings? ❏ What is the use and importance of If and If/Else Statements? ❏ How can variables be used to store values and why are variables useful in programming? ❏ How can the use of functions make code reusable? ❏ How can JavaScript be used to demonstrate scenarios in a biological application?
Goals & Standards	❏ **9-12.DTC.a.1** Use digital tools to design and develop a significant digital artifact (e.g., multi page website, online portfolio, simulation).

	❏ **9-12.CT.d.2** Decompose a problem by defining functions, which accept parameters and produce return values. ❏ **9-12.CT.d.6** Use appropriate conditional structures in programs (e.g., IF-THEN, IF-THEN-ELSE, SWITCH). ❏ **9-12.CT.d.7** Use a programming language or tool feature correctly to enforce operator precedence. ❏ **9-12.CT.d.8** Use global and local scope appropriately in program design (e.g., for variables).
Understanding by Design Framework: Stage 2 - Assessment Evidence	
End Product	❏ An animation representing a phase of either the bumblebee or flower life cycle.
Criteria for Product	❏ Use the background command. ❏ Use at least 4 colors. ❏ Use at least 2 lines. ❏ Use at least 2 ellipses. ❏ Use at least 2 triangles. ❏ Use at least 2 rectangles. ❏ At least 2 shapes filled in. ❏ At least 2 shapes not filled in. ❏ At least 2 arcs. ❏ Use of strokeWeight or noStroke to change the outline of a figure. ❏ Define at least 6 variables. ❏ Use at least 3 math operators (+, -, *, /, %). ❏ Define at least 1 function.
Evidence	❏ Drawing and animation of given drawing of flower and bee with new features added from each sequential lesson. ❏ Drawing and animation of the bumblebee life cycle with new features added from each sequential lesson.
Understanding by Design Framework: Stage 3 - Learning Plan	
Learning Activities	1. Introduction to Drawing with JavaScript a. Class Discussion - What's Programming? b. Introduction to Programming c. Introduction to Drawing with JavaScript d. Adding Color to JS Drawings e. Class Discussion - Let's Discuss Some Applications of JS 2. JS Variables a. Class Discussion - What are Variables? b. Drawing and Animating a Life Cycle: Part 1 c. Introduction to Variables d. Class Discussion - Why Use Variables? 3. Resizing Objects, Text, and Strings a. Interactive Programs b. Using Math Expressions in JS c. Text and Strings in JS d. Let's Combine Variables and Strings! 4. JS Functions a. Class Discussion - What's a Function?

	b. Introduction to JS Drawing Functions c. Drawing and Animating a Life Cycle: Part 2 d. Class Discussion - Why Use Functions? 5. Introduction to Animation a. How Does Animation Work? b. How to Animate with JS c. Drawing and Animating a Life Cycle: Part 3 d. Interacting with JS Animations 6. Conditionals a. Simon Says and Conditionals b. Logic and If Statements c. Unit Wrap-Up
Computational Thinking in Mathematics and Science Taxonomy	
Practices	❏ Computational Problem Solving: Programming, Developing Solutions, Troubleshooting & Debugging

Table 8. Representation of the Drawing and Animation using JavaScript Unit split into stages of the Understanding by Design Framework and the Computational Thinking in Mathematics and Science Taxonomy (Wiggins & McTighe, 2011; Weintrop et al., 2016).

Throughout this unit, students learn how to draw and animate using the ProcessingJS library by incrementally adding to a drawing of a flower and bee. Each of the requirements for the unit project is directly addressed in at least one lesson plan. To complete the project students go through a three-step process of 1) creating a static drawing, 2) simplifying the code, and 3) animating the drawing. In the first step, students add a background, colors, lines, ellipses, triangles, rectangles, and arcs to their drawings. In the second step, students modify the code of their drawing by adding in variables, math operators, and at least one function. In the final stage, students bring their drawings to life by animating their drawings. By the end of this unit, students can demonstrate how functions, conditionals, variables, and the ProcessingJS library can be used to draw and animate a process, in this case, that process is a stage of a life cycle.

As explained earlier in Section 4.2.3, this unit was originally very dependent on Khan Academy tutorials. Rather than relying on Khan Academy tutorials, students learn new concepts with our own templates and incrementally build upon the outcome of their previous learning activity. Since students are not expected to learn HTML or CSS in this unit, all necessary HTML and CSS components are added for them, then a detailed list of instructions is written for them in the ReadMe.md file of each template. Figure 7 below shows the ReadMe.md students are provided with to learn about conditionals and Figure 8 shows the outcome of this template.

Using Logic Operators and Conditional Expressions in JS

Today you'll be learning about logic operators and conditional statements.

Logic Operators

Logical operators are used to determine the logic between variables or values. The most common logic operators are AND, OR, and NOT which in JavaScript are &&, ||, and !, respectively.

Here are some examples to show how these operators work:
Let's say x = 3 and y = 5.

AND Operator
(x < 10 && y > 4) evaluates to TRUE because both conditions are true (3 is less than 10, and 5 is greater than 4).
(x < 2 && y > 4) evaluates to FALSE because only 1 of the conditions is true (5 is greater than 4, but 3 is not less than 2).

OR Operator
(x < 10 || y > 4) evaluates to TRUE because at least 1 of the conditions is true (3 is less than 10 and 5 is greater than 4).
(x < 2 || y > 4) evaluates to TRUE because at least 1 of the conditions is true (5 is greater than 4, even though 3 is not less than 2).
(x < 2 || y > 10) evaluates to FALSE because neither condition is true (3 is not less than 2 and 5 is not greater than 10).

NOT Operator
!(x == y) evaluates to TRUE because 3 is not equal to 5.
!(x < 2 || y > 10) evaluates to TRUE because it is the opposite of what (x < 2 || y > 10) evaluates to.

Conditional Expressions and If Statements

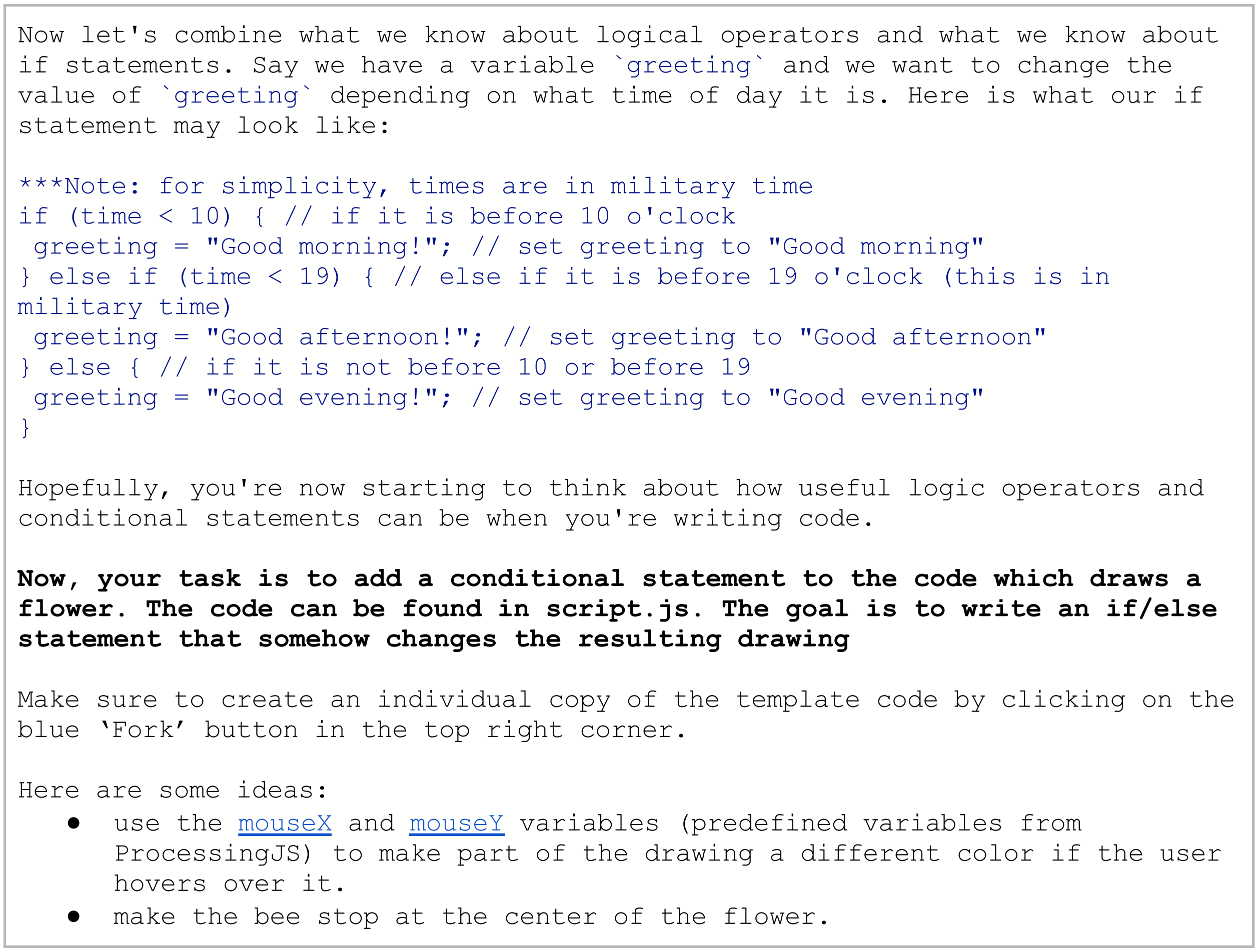
When writing code there's often a situation when you want to perform different actions based on different decisions. To do this, you can use *conditional expressions*.

JavaScript has the following conditional statements:

- "if": to specify a block of code to be executed if a given condition evaluates to true
- "else": to specify a block of code to be executed if the condition given in the if statement is evaluates to false
- "else if": to specify a new condition to test if the previous if the condition is false

Here is the syntax of the conditional statements explained above:

```
if (condition1) {
 //  block of code to be executed if condition1 is true
} else if (condition2) {
 //  block of code to be executed if the condition1 is false AND condition2 is
true
} else {
 //  block of code to be executed if the condition1 is false AND condition2 is
false
}
```



Now let's combine what we know about logical operators and what we know about if statements. Say we have a variable `greeting` and we want to change the value of `greeting` depending on what time of day it is. Here is what our if statement may look like:

```
***Note: for simplicity, times are in military time
if (time < 10) { // if it is before 10 o'clock
 greeting = "Good morning!"; // set greeting to "Good morning"
} else if (time < 19) { // else if it is before 19 o'clock (this is in
military time)
 greeting = "Good afternoon!"; // set greeting to "Good afternoon"
} else { // if it is not before 10 or before 19
 greeting = "Good evening!"; // set greeting to "Good evening"
}
```

Hopefully, you're now starting to think about how useful logic operators and conditional statements can be when you're writing code.

Now, your task is to add a conditional statement to the code which draws a flower. The code can be found in script.js. The goal is to write an if/else statement that somehow changes the resulting drawing

Make sure to create an individual copy of the template code by clicking on the blue 'Fork' button in the top right corner.

Here are some ideas:

- use the mouseX and mouseY variables (predefined variables from ProcessingJS) to make part of the drawing a different color if the user hovers over it.
- make the bee stop at the center of the flower.

Figure 7. ReadMe.md for teaching students about conditionals.

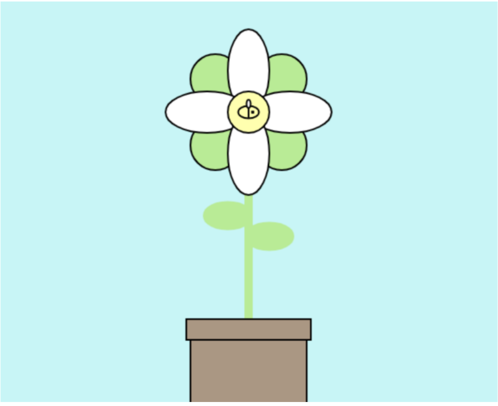


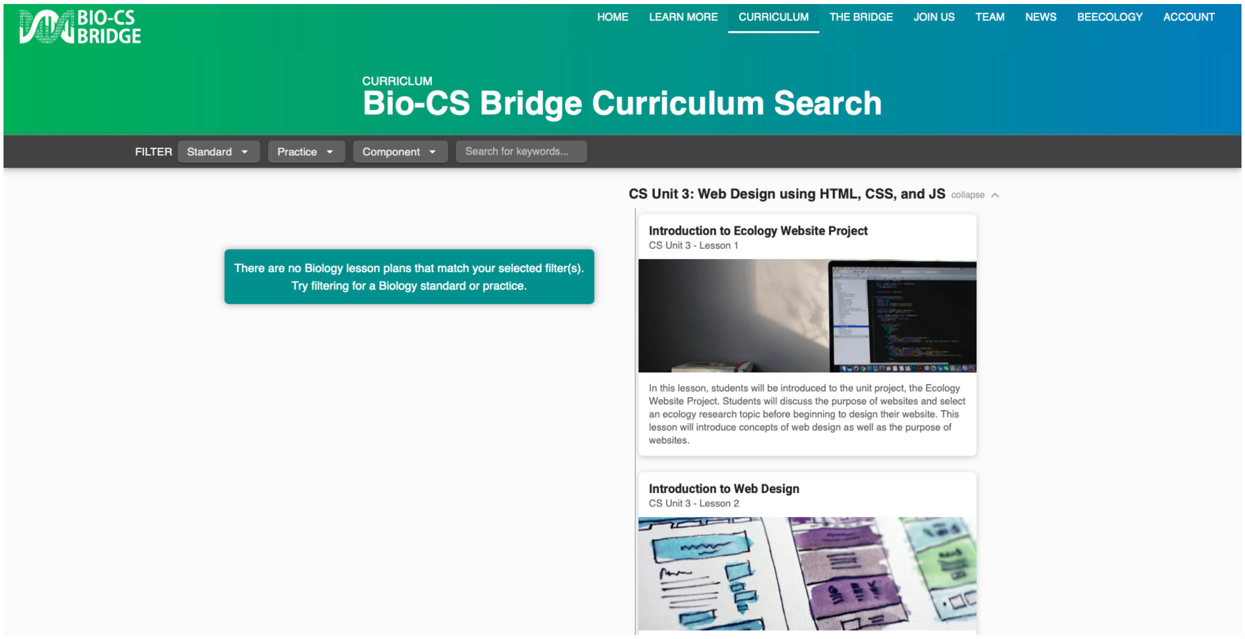
Figure 8. The outcome of conditionals learning activity.

4.6 Bio-CS Bridge Curriculum Website Design

In parallel to this project, the Bio-CS Bridge software development team worked on designing and implementing the Bio-CS Bridge Curriculum website. The new Bio-CS Bridge Curriculum website is designed to fit the needs of teachers. After research and discussion with teachers, a preliminary website was created. The website has many features including the ability to easily navigate between the unit and lesson plans, to look up lesson plans by standard, practice, component, and/or keywords, to see other lessons in a unit while viewing lesson plan components, and to download all necessary materials. In this section, the different pages of the website will be explained in more detail.

4.6.1 Bio-CS Bridge Curriculum Website: Search Page

When visiting the Bio-CS Bridge main website, a user can click on the curriculum tab to access the curriculum website. The curriculum tab brings the user to the Bio-CS Bridge Curriculum Search page which is where the user can browse through the curriculum. The filter bar contains four filters that the user can use to filter the curriculum by standard, practice, component, and/or keyword. If no filters are specified, the entire curriculum can be seen by lesson here. If filters are specified, the specific lessons of the curriculum can be seen here. A screenshot of the Bio-CS Bridge Curriculum Search page can be viewed in Figure 9 below.



CURRICULUM

Bio-CS Bridge Curriculum Search

There are no Biology lesson plans that match your selected filter(s). Try filtering for a Biology standard or practice.

CS Unit 3: Web Design using HTML, CSS, and JS

Introduction to Ecology Website Project
CS Unit 3 - Lesson 1

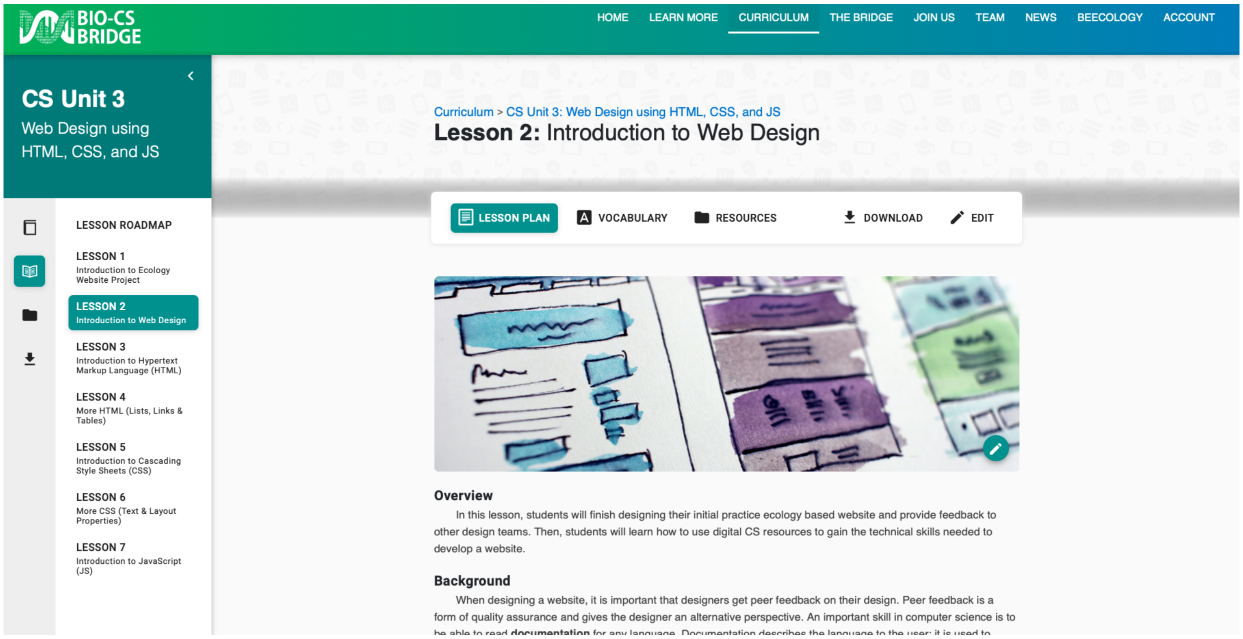
In this lesson, students will be introduced to the unit project, the Ecology Website Project. Students will discuss the purpose of websites and select an ecology research topic before beginning to design their website. This lesson will introduce concepts of web design as well as the purpose of websites.

Introduction to Web Design
CS Unit 3 - Lesson 2

Figure 9. A screenshot of the Bio-CS Bridge Curriculum Website Search page showing results from a query for web development lesson plans.

4.6.2 Bio-CS Bridge Curriculum Website: Lesson Page

Once the user finds a lesson of interest, they can click on the lesson card from the Bio-CS Bridge Curriculum Search page (from Figure 9 above) which will bring them to the lesson page. On the lesson page, a user can see all information necessary to teach the selected lesson plan. Below the title of the lesson page, the user can click on three tabs: lesson plan, vocabulary, or resources. On the lesson plan tab, users can read the overview, background, understandings, prior knowledge, and lesson activities. On the vocabulary tab, users can read all of the key terms for the lesson plan. On the resources tab, users can preview all of the resources necessary for the lesson plan. To the right of these three tabs are a download button that allows the user to download the lesson plan and all resources for the lesson plan with the press of one button. Also, on the left side panel, the user can view the rest of the lessons in the unit. A screenshot of the lesson page can be viewed in Figure 10 below.


BIO-CS BRIDGE
HOME
LEARN MORE
CURRICULUM
THE BRIDGE
JOIN US
TEAM
NEWS
BEECOLOGY
ACCOUNT
CS Unit 3
Web Design using HTML, CSS, and JS
LESSON ROADMAP
LESSON 1
Introduction to Ecology Website Project
LESSON 2
Introduction to Web Design
LESSON 3
Introduction to Hypertext Markup Language (HTML)
LESSON 4
More HTML (Lists, Links & Tables)
LESSON 5
Introduction to Cascading Style Sheets (CSS)
LESSON 6
More CSS (Text & Layout Properties)
LESSON 7
Introduction to JavaScript (JS)
Curriculum > CS Unit 3: Web Design using HTML, CSS, and JS
Lesson 2: Introduction to Web Design
LESSON PLAN
VOCABULARY
RESOURCES
DOWNLOAD
EDIT

Overview

In this lesson, students will finish designing their initial practice ecology based website and provide feedback to other design teams. Then, students will learn how to use digital CS resources to gain the technical skills needed to develop a website.

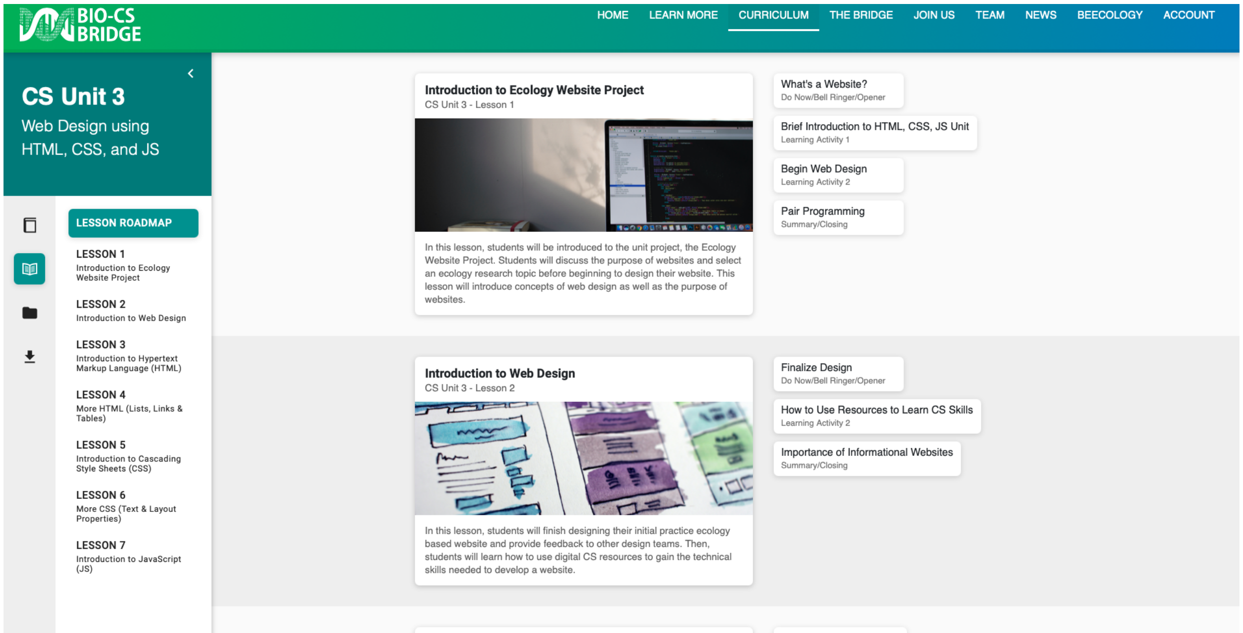
Background

When designing a website, it is important that designers get peer feedback on their design. Peer feedback is a form of quality assurance and gives the designer an alternative perspective. An important skill in computer science is to be able to read **documentation** for any language. Documentation describes the language to the user; it is used to

Figure 10. A screenshot of the Bio-CS Bridge Curriculum Website Lesson page.

4.6.3 Bio-CS Bridge Curriculum Website: Lesson Roadmap Page

If a user would like to see a lesson plan in the context of the rest of the unit, they can visit the lesson roadmap page. From the left side panel, users can click on 'LESSON ROADMAP' which will show the user the overview of the lesson as well as a list of learning activities within the lesson. Users can view multiple lesson overviews at once to develop a better understanding of each lesson plan in the context of the rest of the unit. If a user would like to look at a specific activity, they can click on the activity card on the right side and it will bring them to the description of the learning activity on the lesson plan page (as pictured in Figure 10 above). A screenshot of the lesson roadmap page can be viewed in Figure 11 below.


BIO-CS BRIDGE
HOME
LEARN MORE
CURRICULUM
THE BRIDGE
JOIN US
TEAM
NEWS
BEECOLOGY
ACCOUNT
CS Unit 3
Web Design using HTML, CSS, and JS
LESSON ROADMAP
LESSON 1
Introduction to Ecology Website Project
LESSON 2
Introduction to Web Design
LESSON 3
Introduction to Hypertext Markup Language (HTML)
LESSON 4
More HTML (Lists, Links & Tables)
LESSON 5
Introduction to Cascading Style Sheets (CSS)
LESSON 6
More CSS (Text & Layout Properties)
LESSON 7
Introduction to JavaScript (JS)

Introduction to Ecology Website Project
CS Unit 3 - Lesson 1

In this lesson, students will be introduced to the unit project, the Ecology Website Project. Students will discuss the purpose of websites and select an ecology research topic before beginning to design their website. This lesson will introduce concepts of web design as well as the purpose of websites.

What's a Website?
Do Now/Bell Ringer/Opener

Brief Introduction to HTML, CSS, JS Unit
Learning Activity 1

Begin Web Design
Learning Activity 2

Pair Programming
Summary/Closing

Introduction to Web Design
CS Unit 3 - Lesson 2

In this lesson, students will finish designing their initial practice ecology based website and provide feedback to other design teams. Then, students will learn how to use digital CS resources to gain the technical skills needed to develop a website.

Finalize Design
Do Now/Bell Ringer/Opener

How to Use Resources to Learn CS Skills
Learning Activity 2

Importance of Informational Websites
Summary/Closing

Figure 11. A screenshot of the Bio-CS Bridge Curriculum Lesson Roadmap page.

4.6.4 Meeting the Teachers' Needs

Some of the key features that teachers expressed are of importance to the Bio-CS Bridge Curriculum website were:

- ❑ Simple navigation between the unit and lesson plans.
- ❑ Filter and search through lesson plans by keyword, standard, practice.
- ❑ A roadmap of the unit (when viewing a lesson plan to see the lesson plan in a larger context).
- ❑ Clear and concise representation of lesson plan components.
- ❑ Downloadable unit or lesson plan toolkit (to include all necessary materials to teach).

All of these features were integrated into the website's design. The simple navigation between the unit and lesson plans and roadmap of the unit was addressed by adding the left side panel to the lesson page. The ability to filter and search through lesson plans by keyword, standard, and practice was included in the Bio-CS Bridge Curriculum Search page. All components of the lesson plan can be viewed from the lesson page and all necessary materials can be viewed in the resources section of the lesson page. The design of the Bio-CS Bridge Curriculum webpage was very well-thought-out with the end-user in mind for all design decisions. The team is excited to release the site to the public in the upcoming months.

5 Conclusions and Future Work

5.1 Conclusions

One of the underlying goals of the Bio-CS Bridge project is to promote broader student interest in computer science by demonstrating how it can be used to solve real-world biological problems. To address this goal, this project focused specifically on the following three goals: 1) strengthen the bridge between biology and computer science by making connections to pollinator decline and loss of biodiversity throughout all lessons of the computer science units; 2) use pollinator decline and loss of biodiversity as context for expanding the coverage of core computer science concepts and standards throughout the curriculum; and 3) finalize computer science curriculum for dissemination to a wider audience via a curriculum website. This was accomplished by utilizing the Understanding by Design Curriculum Framework and Computational Thinking in Mathematics and Science Taxonomy. In addition to these methods, there was frequent communication with computer science educators currently teaching the curriculum. By the end of this project, three modular, well-designed computer science units are ready for being implemented into high school classrooms and a website for curriculum dissemination has been designed and is under implementation by the Bio-CS Bridge software development team.

The two computer science units that were enhanced during this project have been taught by teachers in the Bio-CS Bridge team, however, the newly added templates for learning activities have not been used yet by teachers. All new materials for these units will be released once the curriculum website is complete and teachers will be able to implement them in their classroom. The new computer science Python unit has not been piloted by any teachers yet, however, the plan is to have teachers of the Bio-CS Bridge project pilot this unit in the upcoming school year. Once teachers of the Bio-CS Bridge project can successfully implement this unit in their classroom, the plan is to release the unit to the public via the curriculum website. In addition to the implementation of the newly enhanced computer science units, the Bio-CS Bridge team would like to expand the curricular design approaches used during this project to other STEM subjects and lower grade levels.

The aim is that more and more students will be introduced to computational thinking skills during their general education courses which will better prepare them for participating in the ever-changing digital society. The Bio-CS Bridge curriculum provides educators and districts with a model of how computational thinking skills can be incorporated into an integrated curriculum that connects multiple domains. After implementing the Bio-CS Bridge curriculum into schools, the team expects that schools and districts can see the benefits of an integrated curriculum. With the success of implementation, we expect that there will be an increase in the number of districts that integrate computational thinking into other subjects and therefore more students will be prepared for success in today’s data-driven society after their public educational career.

5.2 Future Work

Based on the success of the current Bio-CS Bridge curriculum, the group would like to explore the opportunity to expand the curriculum. One path of expansion would be adapting it so that it can be taught to lower grade levels. A key component of curriculum design is making sure that it is scaffolded to enhance learning and mastery of skills for all students in the classroom. In the Bio-CS Bridge computer science curriculum, most learning plans are scaffolded to allow for all learners to be able to succeed. To expand this curriculum to lower grade levels like middle school, more scaffolding may need to be added. Since middle schoolers do not address the same biology standards as high schoolers do, the biology content might have to be changed or explicitly taught to middle schoolers. Unit 3: Web Design using HTML, CSS, and JavaScript and Unit 4: Drawing and Animation using JavaScript could easily be integrated into a middle school STEM classroom as long as the biology content is changed to content that students are currently learning. Unit 2: Computational Problem Solving using Python may be more difficult to integrate into a middle school STEM classroom because students have less experience with drawing conclusions from data. With some additional scaffolding and further instruction on how to analyze the data, middle school students should be able to succeed in Unit 2. All three of these computer science units could be easily adapted to be taught in a middle school STEM classroom.

Expanding the computer science curriculum to other STEM subjects should be fairly simple as well. In all three units addressed during this project, the biology content was integrated throughout the lesson plans. This was done by first figuring out what the end product would be and then deciding how to prepare students for success with the end product. As long as this same procedure is followed, all three of these units could be expanded to different STEM subjects. By following the Understanding by Design curriculum design framework educators should decide on what students should understand by the end of the unit in both domains, computer science, and the other STEM subject; the computer science understandings are already decided on, so just the other STEM subject's understandings would need to be determined. Once all understandings are determined, integrating those understandings into the end product needs to be changed from the existing unit. The integration of the new STEM-related understandings into the end product is most likely the hardest part. It can be difficult to gauge how much background students will need in the other domain, however that can be determined by understanding how much background they already have. The expansion of the Bio-CS Bridge curriculum to other STEM subjects may be more challenging than expanding to lower grade levels, but by determining all necessary content understandings first, it should be straightforward.

References

Aspray, W. (2016). *Women and Underrepresented Minorities in Computing* (1st ed.). Springer. https://www.springer.com/gp/book/9783319796819

Beecology Project. (n.d.). *Beecology Project*. Beecology Project. https://beecology.wpi.edu/

Bio-CS Bridge. (n.d.). *The Bio-CS Bridge*. Bio-CS Bridge. https://biocsbridge.wpi.edu/

Camp, T., Adrion, W. R., Bizot, B., Davidson, S., Hall, M., Hambrusch, S., Walker, E., & Zweben, S. (2017, May). Generation CS: the growth of computer science. *ACM Inroads*. https://dl.acm.org/doi/pdf/10.1145/3084362

College Board. (2020). *AP Computer Science Principles Course and Exam Description, Effective Fall 2020*. AP Central. https://apcentral.collegeboard.org/pdf/ap-computer-science-principles-course-and-exam-description.pdf?course=ap-computer-science-principles

Dichev, C., Dicheva, D., Cassel, L., Goelman, D., & Posner, M. (2016, March). Preparing All Students for the Data-driven World. *The Association of Computer Science Departments at Minority Institutions*. http://www.admiusa.org/admi2016/Papers_Faculty/ADMI2016_DichevEtAll.pdf

Education Commission of the States. (2015, December 7). *The Hidden Half*. Change the Equation. http://ecs.force.com/studies/rstempg?id=a0r0g000009TLfB

K12 Computer Science. (n.d.). *Equity in Computer Science Education*. K-12 Computer Science Framework. https://k12cs.org/equity-in-computer-science-education/

Massachusetts Department of Elementary and Secondary Education. (2016). *2016 Digital Literacy and Computer Science Curriculum Framework*. Massachusetts Department of Elementary and Secondary Education. https://www.doe.mass.edu/stem/dlcs/

Next Gen Science Standards. (n.d.). Next Generation Science Standards. http://www.nextgenscience.org/massachusetts

Ryoo, J., Margolis, J., Lee, C., Sandoval, C., & Goode, J. (2013, January). Democratizing computer science knowledge: transforming the face of computer science through public high school education. *Learning, Media and Technology*, *38*(2), 161-181. https://www.tandfonline.com/doi/pdf/10.1080/17439884.2013.756514

Stoll, L., Bolam, R., Mcmahon, A., Wallace, M., & Thomas, S. (2006). Professional Learning Communities: A Review of the Literature. *Journal of Educational Change*, *7*(4), 221-258. https://www.researchgate.net/publication/226457350_Professional_Learning_Communities_A_Review_of_the_Literature

Vision & Change in Undergraduate Biology Education. (2011). *About V&C: A Call to Action*. Vision & Change in Undergraduate Biology Education. https://www.visionandchange.org/about-vc-a-call-to-action-2011/

Weintrop, D., Beheshti, E., Horn, M., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining Computational Thinking for Mathematics and Science Classrooms. *J Sci Educ Technol*, (25), 127-147. 10.1007/s10956-015-9581-5

Wiggins, G., & McTighe, J. (2011). *The Understanding by Design Guide to Creating High-Quality Units*. Association for Supervision and Curriculum Development. http://www.ascd.org/Publications/Books/Overview/The-Understanding-by-Design-Guide-to-Creating-High-Quality-Units.aspx

Yadav, A., Gretter, S., Hambrusch, S., & Sands, P. (2016). Expanding computer science education in schools: understanding teacher experiences and challenges. *Computer Science Education*, *26*(4), 235-254. https://www.tandfonline.com/doi/full/10.1080/08993408.2016.1257418

Appendix A: MA Digital Literacy and Computer Science Standards

Strand & Topic	#	Bio-CS Bridge Lessons				Description
		Unit 1	Unit 2	Unit 3	Unit 4	
Computing and Society [CAS]						
Safety & Security 9-12.CAS.a	1					Evaluate and design an ergonomic work environment.
	2					Explain safe practices when collaborating online, including how to anticipate potentially dangerous situations.
	3					Construct strategies to combat cyberbullying/harassment.
	4					Identify the mental health consequences of cyberbullying/harassment.
	5					Explain how peer pressure in social computing settings influences choices.
	6					Apply strategies for managing negative peer pressure and encouraging positive peer pressure.
Ethics & Laws 9-12.CAS.b	1					Model mastery of the school's Acceptable Use Policy (AUP).
	2					Identify computer-related laws and analyze their impact on digital privacy, security, intellectual property, network access, contracts, and consequences of sexting and harassment.
	3					Discuss the legal and ethical implications associated with malicious hacking and software piracy.
	4					Interpret software license agreements and application permissions.
Interpersonal & Societal Impact 9-12.CAS.c	1					Explain the impact of the digital divide on access to critical information.
	2					Discuss the impact of computing technology on business and commerce (e.g., automated tracking of goods, automated financial transaction, e-commerce, cloud computing).

	3					Describe the role that assistive technology can play in people's lives.
	4					Create a digital artifact that is designed to be accessible (e.g., closed captioning for audio, alternative text for images).
	5					Analyze the beneficial and harmful effects of computing innovations (e.g., social networking, delivery of news and other public media, intercultural communication).
	6					Cultivate a positive web presence (e.g., digital resume, portfolio, social media).
	7			3.1, 3.E		Identify ways to use technology to support lifelong learning.
	8					Analyze the impact of values and points of view that are presented in media messages (e.g., racial, gender, political).
	9					Discuss the social and economic implications associated with malicious hacking, software piracy, and cyber terrorism.
Digital Tools and Collaboration [DTC]						
Digital Tools 9-12.DTC.a	1	1.5, 1.6A, 1.6B, 1.7		3.2, 3.3, 3.4, 3.5, 3.6, 3.7	4.2, 4.4, 4.5	Use digital tools to design and develop a significant digital artifact (e.g., multi page website, online portfolio, simulation).
	2					Select digital tools or resources based on their efficiency and effectiveness to use for a project or assignment and justify the selection.
Collaboration & Communication 9-12.DTC.b	1			3.E		Communicate and publish key ideas and details to a variety of audiences using digital tools and media-rich resources.
	2					Collaborate on a substantial project with outside experts or others through online digital tools (e.g., science fair project, community service project, capstone project).
Research	1			3.1, 3.E		Generate, evaluate, and prioritize

9-12.DTC.c						questions that can be researched through digital resources or tools.
	2					Perform advanced searches to locate information and/or design a data-collection approach to gather original data (e.g., qualitative interviews, surveys, prototypes, simulations).
	3					Evaluate digital sources needed to solve a given problem (e.g., reliability, point of view, relevancy).
	4			3.1, 3.3, 3.4, 3.E		Gather, organize, analyze, and synthesize information using a variety of digital tools.
	5	1.6A, 1.6B, 1.7		3.2, 3.3, 3.4, 3.E		Create an artifact that answers a research question, communicates results and conclusions, and cites sources.
Computing Systems [CS]						
Computing Devices 9-12.CS.a	1					Select computing devices (e.g., probe, sensor, tablet) to accomplish a real-world task (e.g., collecting data in a field experiment) and justify the selection.
	2			3.5, 3.6, 3.7, 3.E		Examine how the components of computing devices are controlled by and react to programmed commands.
	3					Apply strategies for identifying and solving routine hardware and software problems that occur in everyday life (e.g., update software patch
	4					Explain and demonstrate how specialized computing devices can be used for problem solving, decision-making and creativity in all subject areas.
	5					Describe how computing devices manage and allocate shared resources [e.g., memory, Central Processing Unit (CPU)].
	6					Examine the historical rate of change in computing devices (e.g., power/energy, computation capacity,

						speed, size, ease of use) and discuss the implications for the future.
Human & Computer Partnerships 9-12.CS.b	1					Identify a problem that cannot be solved by humans or machines alone and design a solution for it by decomposing the task into sub-problems suited for a human or machine to accomplish (e.g., a human-computer team playing chess, forecasting weather, piloting airplanes).
Networks 9-12.CS.c	1					Explain how network topologies and protocols enable users, devices, and systems to communicate with each other
	2					Examine common network vulnerabilities (e.g., cyberattacks, identity theft, privacy) and their associated responses.
	3					Examine the issues (e.g., latency, bandwidth, firewalls, server capability) that impact network functionality.
Services 9-12.CS.d	1					Compare the value of using an existing service versus building the equivalent functionality (e.g., using a reference search engine versus creating a database of references for a project).
	2					Explain the concept of quality of service (e.g., security, availability, performance) for services providers (e.g., online storefronts that must supply secure transactions for buyer and seller).
Computational Thinking [CT]						
Abstraction 9-12.CT.a	1					Discuss and give an example of the value of generalizing and decomposing aspects of a problem in order to solve it more effectively.
Algorithms 9-12.CT.b	1		2.6, 2.8			Recognize that the design of an algorithm is distinct from its expression in a programming language
	2		2.1, 2.4, 2.6, 2.8			Represent algorithms using structured language, such as pseudocode.

	3					Explain how a recursive solution to a problem repeatedly applies the same solution to smaller instances of the problem.
	4					Describe that there are ways to characterize how well algorithms perform and that two algorithms can perform differently for the same task.
	5					Explain that there are some problems, which cannot be computationally solved.
Data 9-12.CT.c	1					Describe how data types, structures, and compression in programs affect data storage and quality (e.g., digital image file sizes are affected by resolution and color depth).
	2		2.5, 2.7			Create an appropriate multidimensional data structure that can be filtered, sorted, and searched (e.g., array, list, record).
	3					Create, evaluate, and revise data visualization for communication and knowledge.
	4		2.E			Analyze a complex data set to answer a question or test a hypothesis (e.g., analyze a large set of weather or financial data to predict future patterns).
	5					Identify different problems (e.g., large or multipart problems, problems that need specific expertise, problems that affect many constituents) that can benefit from collaboration when processing and analyzing data to develop new insights and knowledge.
Programming & Development 9-12.CT.d	1		2.E	3.3		Use a development process in creating a computational artifact that leads to a minimum viable product and includes reflection, analysis, and iteration (e.g., a data-set analysis program for a science and engineering fair, capstone project that includes a program, term research project based on program data).

	2		2.8		4.1, 4.2, 4.3, 4.4, 4.5	Decompose a problem by defining functions, which accept parameters and produce return values.
	3		2.7			Select the appropriate data structure to represent information for a given problem (e.g., records, arrays, lists).
	4					Analyze trade-offs among multiple approaches to solve a given problem (e.g., space/time performance, maintainability, correctness, elegance).
	5		2.4, 2.6			Use appropriate looping structures in programs (e.g., FOR, WHILE, RECURSION).
	6		2.3		4.6	Use appropriate conditional structures in programs (e.g., IF-THEN, IF-THEN-ELSE, SWITCH).
	7		2.2		4.3, 4.4	Use a programming language or tool feature correctly to enforce operator precedence.
	8		2.8		4.4, 4.5	Use global and local scope appropriately in program design (e.g., for variables)
	9		2.9			Select and employ an appropriate component or library to facilitate programming solutions [e.g., turtle, Global Positioning System (GPS), statistics library].
	10		2.1	3.2, 3.3, 3.6, 3.E		Use an iterative design process, including learning from making mistakes, to gain a better understanding of the problem domain.
	11		2.9	3.E		Engage in systematic testing and debugging methods to ensure program correctness.
	12		2.9			Demonstrate how to document a program so that others can understand its design and implementation.
Modeling & Simulation 9-12.CT.e	1	1.1, 1.2A, 1.2B, 1.3, 1.4, 1.5, 1.6A, 1.6B, 1.7				Create models and simulations to help formulate, test, and refine hypotheses.
	2	1.7				Form a model from a hypothesis

						generated from research and run a simulation to collect and analyze data to test that hypothesis.

Appendix B: Lesson Plan Template

Lesson Plan Title: # - *insert title here*

Subject/Course: Computer Science

Unit: **Grade Level: High School**

Lesson Background:

thinking about teacher's with little experience, explain what concepts they'll need to know before teaching this lesson

Overview of and Motivation for Lesson:

In this lesson, students will ...

Prior Knowledge:

Before starting this lesson, students should have a basic understanding of …
If students do not have this understanding, they should complete lesson # …

Stage 1 - Desired Results

Standard(s):
●
Computer Science Practice(s):
●
Science Practice(s):
●

Understanding(s): *Students will understand that . . .* ●
Content Objectives: *Students will be able to . . .* ●
Key Vocabulary ● *insert term here*: *insert definition here*
Stage 2 - Assessment Evidence
Performance Task or Key Evidence ● *insert task for students to be assessed on*
Key Criteria to measure Performance Task or Key Evidence ● *insert requirements for above task*
Stage 3 - Learning Plan

Do Now/Bell Ringer/Opener: *insert activity title here* (# - # mins)	
● *in-person activity description*	
Online Version	● *online activity description*
Learning Activity 1: *insert activity title here* (# - # mins)	
● *in-person activity description*	
Online Version	● *online activity description*
Learning Activity 2: *insert activity title here* (# - # mins)	
● *in-person activity description*	
Online Version	● *online activity description*
Summary/Closing: *insert activity title here* (# - # mins)	
● *in-person activity description*	
Online Version	● *online activity description*
Homework/Extension Activities: *insert extension activity - how can students further their learning?*	
Materials and Equipment Needed: ● Projector ● Computers (ideally 1:1 student to computer ratio) ● *insert additional materials needed for the lesson, i.e. link to the presentation	

Template modified from Wiggins and McTighe by Shari Weaver and Kirsten Hart (Wiggins & McTighe, 2011).

Appendix C: Teacher Demonstration for CSLP 2.1

```
# This is a comment. It is good to comment on your code so that others can understand it

# Below is a print statement that is useful for printing something out to the system console (on the right side of the screen)
print("Hello World!")

# MATH
# Below we initialize 2 variables x and y. Once those variables are initialized, we can perform math operations on them, like addition, subtraction, multiplication, and division.
x = 4
y = 2
print(x + y)

# LISTS
# Below we initialize 3 lists. list 1 and list 2 are 2 lists of numbers and list3 is list2 appended to list1. Notice how when list3 is printed out it is the contents of list1 before the contents of list2.
list1 = [1, 2, 3]
list2 = [4, 5, 6, 7]
list3 = list1 + list2
print(list3)

# Below we initialize 1 list of strings. Once a list is initialized, we can access elements in the list. Also, notice how strings can be contained in either double or single quotes.
list4 = ["a", 'b', "c"]
print(list4[1])

# LOOPS
# Below we loop through list4 and look at each letter in the list and print it out.
for letter in list4:
 print(letter)
```

```
# Below we have a conditional while x is greater than or equal to y, then
decrement x and print out x.
while (x >= y):
 x = x - 1
 print(x)
 # CONDITIONALS
# Below we have the same conditional as before, however it is not
contained in a loop, so it will only be checked once.
if (x >= y):
 x = x - 1
 print(x)
```

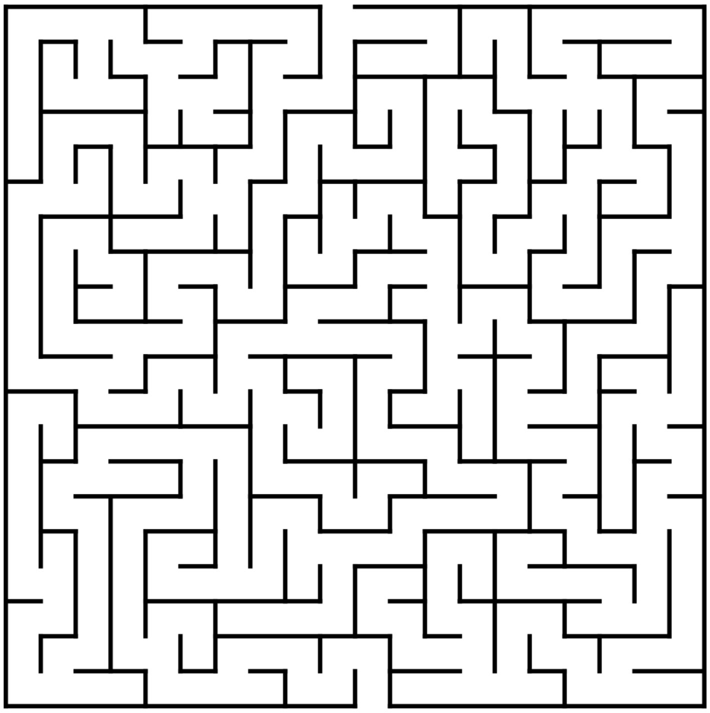
Appendix D: Maze Time! Worksheet

Maze Time!

Directions: Solve the maze below and write out all of the steps to get from the entrance to the exit on the lines below.

Steps to get from Entrance to Exit:
(left, right, straight)

ENTRANCE



EXIT

1. ______________________

2. ______________________

3. ______________________

4. ______________________

5. ______________________

6. ______________________

7. ______________________

8. ______________________

9. ______________________

10. ______________________

11. ______________________

12. ______________________

13. ______________________

14. ______________________

15. ______________________

16. ______________________

17. ______________________

18. ______________________

19. ______________________

20. ______________________

21. ______________________

22. ______________________

23. ______________________

24. ______________________

25. ______________________

26. ______________________

27. ______________________

28. ______________________

29. ______________________

30. ______________________

31. ______________________

32. ______________________

33. ______________________

Appendix E: Solving Search and Sort Problems Worksheet - Search

What’s an Algorithm?

Directions: Solve the word search below and *after discussion with your team* write out an algorithm (or procedure) for solving any word search. That is, given any table of letters (size n x n, where n >= 1) and a word, determine if the word appears horizontally on the table of letters.

Your algorithm must:

1. Stop at a certain point.
2. Have well-defined instructions with specific steps.
3. Be effective in solving the problem it was designed to solve (a word search).

N	N	U	O	D	I	O	T	U	A	R	H	R	R
I	D	V	A	T	D	B	R	N	G	P	M	L	M
M	R	T	L	B	A	P	A	U	P	G	N	V	O
H	A	C	F	U	N	C	T	I	O	N	R	T	T
T	P	P	R	O	G	R	A	M	M	I	N	G	P
I	A	T	N	A	I	H	E	P	R	N	E	E	A
R	N	C	P	N	O	T	E	O	T	R	P	H	R
O	O	T	T	I	D	I	E	U	S	R	O	H	A
G	H	N	A	N	O	T	N	R	U	T	E	R	M
L	T	F	H	H	T	A	O	E	A	T	U	I	E
A	Y	T	I	I	A	D	M	O	R	T	H	O	T
I	P	A	H	M	R	R	B	A	P	G	I	A	E
P	S	E	U	D	O	C	O	D	E	I	E	V	R
M	O	B	O	O	L	E	A	N	I	N	C	A	E

PSEUDOCODE
PROGRAMMING
RETURN
ALGORITHM
ITERATIVE
FUNCTION
BOOLEAN
PARAMETER
PYTHON

Write your thoughts for an algorithm to solve a searching problem here.

____

____

____

____

____

____

Think about this! What if we wanted to check if the word appears on the table in reverse order or vertically or horizontally? How would your algorithm have to change?

Appendix F: Solving Search and Sort Problems Worksheet - Sort

What's an Algorithm?

Directions: With the cards that you have been given, lay them out on the table in random order. Now sort the cards from smallest to largest by only moving one card at a time. Once the cards are sorted, discuss with your team an algorithm for solving a sorting problem. *After discussion with your team* write out an algorithm (or procedure) for solving any sorting problem. That is, given any set of (size n, where n >= 1) cards, sort them in increasing order.

Your algorithm must:

1. Stop at a certain point.
2. Have well-defined instructions with specific steps.
3. Be effective in solving the problem it was designed to solve (sorting numeric values).

Feel free to use the boxes below to draw out your thinking.

Smallest Largest

□ □ □ □ □

Write your thoughts for an algorithm to solve a sorting problem here.