Reimagining CS Pathways

Every student prepared for a world powered by computing

CSTA Computer Science Teachers Association

ACE INSTITUTE for ADVANCING COMPUTING EDUCATION

acm Association for Computing Machinery

CODE

CollegeBoard

CSforALL

ECEP Expanding Computing Education Pathways

Authors and Leadership

The project was primarily planned, facilitated, and coordinated by the project team.

Computer Science Teachers Association

Bryan Twarek (Principal Investigator)
Jake Koressel (Project Manager)

INSTITUTE for ADVANCING COMPUTING EDUCATION

Dr. Monica McGill (Co-Principal Investigator)
Dr. Julie Smith (Researcher)

Steering Committee

The steering committee had ultimate responsibility for all research, deliverables, and the overall project success.

Advisory Board

The advisory board evaluated progress towards project goals and provided regular feedback.

Funding Support

This project is supported by the National Science Foundation (NSF) under Grant No. 2311746. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF.

Suggested Citation: CSTA, IACE, ACM, Code.org, College Board, CSforALL, & ECEP Alliance. (2024). *Reimagining CS Pathways: Every student prepared for a world powered by computing.* Association for Computing Machinery. https://doi.org/10.1145/3678016

Report and additional resources are available at https://ReimaginingCS.org.

Table of Contents

Executive Summary

At a time when computing continues to gain importance in society, it is more crucial than ever to ensure that computer science education meets the needs of all students. To this end, the Computer Science Teachers Association (CSTA) is updating its K-12 computer science (CS) standards.

As a prelude to the standards revision, CSTA – working with many partners – has launched a project, *Reimagining CS Pathways: High School and Beyond*, to articulate what CS content is essential for all high school graduates to know and to establish pathways for continued study of CS beyond that foundational content.

The *Reimagining* project drew on the expertise and experiences of dozens of participants – including high school CS teachers, college CS faculty, state and local education leaders, CS education researchers, and those working for nonprofits and in the tech industry. These participants reflected diversity across many dimensions, including demographics, role, and expertise. They participated in focus groups, interviews, and in-person convenings, and they provided substantial asynchronous feedback.

The result of these extensive efforts is contained in this report, which articulates the foundational CS content and resulting pathways.

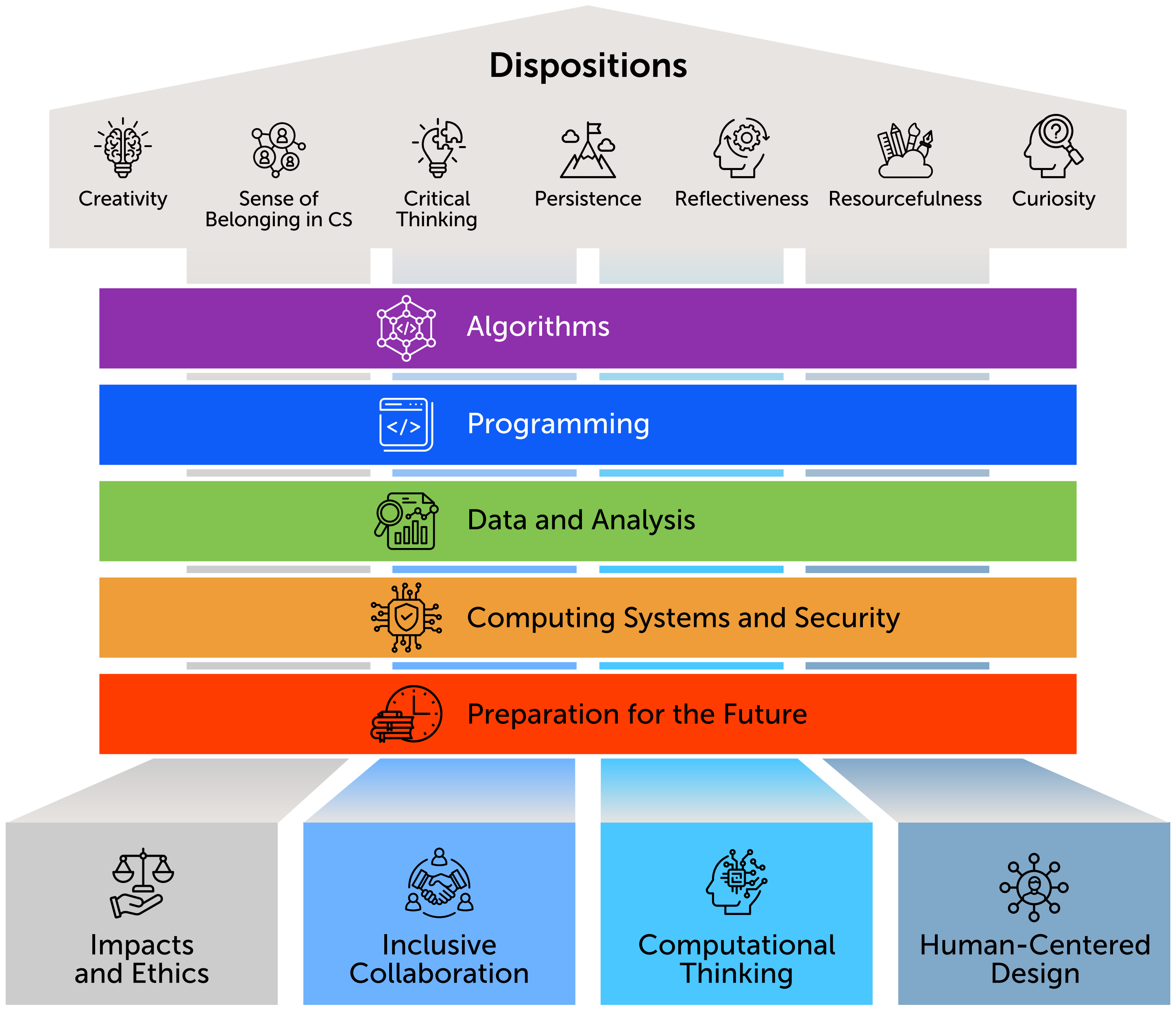
As the image on the next page illustrates, the foundational CS content is organized into Topic Areas, Pillars, and Dispositions. The Topic Areas, which reflect the content that is essential for all high school graduates, are 1) Algorithms, 2), Programming, 3) Data and Analysis, 4) Computing Systems and Security, and 5) Preparation for the Future. The Pillars, which reflect essential ideas and practices that cut across all of the Topic Areas, are 1) Impacts and Ethics, 2) Inclusive Collaboration, 3) Computational Thinking, and 4) Human-Centered Design. While they are not explicitly taught, the goal is to develop a set of specific dispositions in CS. *These Dispositions are persistence, reflectiveness, creativity, curiosity, critical thinking, resourcefulness, and sense of belonging in CS.*

There are many possible pathways stemming from this foundational content, ranging from Cybersecurity and Artificial Intelligence to X + CS (where another subject, such as Journalism or Biology, is integrated with the study of computing). Implementation of these pathways will vary significantly depending on community priorities and contexts. We recognize that schools will need to be selective in their implementation of CS pathways due to limited resources, and we make recommendations for how to select which options to implement.

Woven throughout this work is a commitment to improving equity in CS education. This commitment to equity is embedded throughout both the process and the outcome of the *Reimagining* project. It manifests in an effort to reimagine CS to ensure opportunities for all students and to prepare them for a world increasingly powered by computing.

Learn more at ReimaginingCS.org

Foundational Computer Science Content



1 Introduction

1 Introduction

Today's high school students will face pervasive questions that require foundational knowledge of computer science for them to answer. They will, for example, need to shape their views on the regulation of artificial intelligence (AI), have the ability to automate routine tasks, and analyze and visualize data in a variety of contexts. These situations illustrate the need for early, universal CS education, which will only become more important as society continues to increasingly rely on computing technologies.

Unfortunately, although CS education has undergone a rapid evolution over the last decade, what is taught and how it is taught has remained relatively the same. A critical evaluation of CS content knowledge, skills, and dispositions is necessary to ensure that students are prepared to understand the many facets of computing and how those facets can impact various aspects of people's lives.

On the workforce front, careers that involve processing and analyzing data – in other words, computing jobs – are projected to increase at more than ten times the average rate of overall employment, with a growth rate of 15% over the next decade (Bureau of Labor Statistics, 2023). At the same time, rapid advances in AI have led to concern that much of that human labor will be replaced (Kugler, 2023; Welsh, 2023), a concern grounded in the reality that nearly all professional software developers are already using AI tools in their workflow (Shani, 2023). Against this backdrop of increased workforce demand, rapid technological change, and persistent concerns with equity (Wang & Hejazi Moghadam, 2017) and ethics (Vakil, 2018), there is a growing recognition of the need to teach CS in grades K-12.

I don't know if my personal data is safe if I use this sleep app – Could I create my own app?
An ad just recommended that I try that bakery – Is something tracking my location?
Every student prepared for a world powered by computing
Tracking data for my soccer team takes a lot of time – Should I automate the process?
Should I vote for the candidate who promises to regulate AI?
VOTE
AI

1.1 Changes in K-12 CS Education

As reported in *2022 State of Computer Science Education: Understanding Our National Imperative* (Code.org et al., 2022), for the first time, over half of U.S. states now require CS to be taught in all of their high schools (although fewer than 6% of high school students take a CS course in a given year (Code.org et al., 2023)), with the majority of those states also requiring it in their elementary schools. Students enjoy CS and want to learn it – high school students like CS classes more than any other subject outside the arts (Code.org, 2016). And, as younger generations are born into and raised in a technologically advanced society, today's students understand that CS will be crucial for their careers and lives (Google & Gallup, 2017). As a result, more than half (57%) of high schools in the United States (U.S.) offer a foundational CS course, representing significant growth (Code.org et al., 2023) in a subject that is critical to the nation's economic health and security. Additionally, a joint task force from the Association for Computing Machinery (ACM), IEEE-Computer Society, and the Association for Advancement of Artificial Intelligence (AAAI) recently completed a revision of computing curriculum for higher education, which will impact what CS students should know in college (Kumar et al., 2024).

When the CSTA K-12 Computer Science Standards were last updated in 2017, only six states had K-12 CS standards; as of November 2023, 41 states had K-12 CS standards (and one had high school CS standards only). In 2016, no state had a high school CS graduation requirement; as of this report, nine states do (Code.org et al., 2023; Indiana HB1243, n.d.).

Reflecting on the wide variety of high school student experiences, many factors will likely shape the next decade of secondary (i.e., middle and high school) and postsecondary CS education:

- The recent K-12 CS movement has led to a population of secondary students interested in CS that are more diverse in demographics and interests and have more CS experience than previous generations of students (Liu et al., 2024).
- There is a growing significance of and need for CS skills including high-demand topics such as artificial intelligence, data science, and cybersecurity.
- A burgeoning number of secondary and postsecondary students is interested in minoring or majoring in CS, or just taking individual CS courses in college (National Academies of Sciences, 2018).
- States are increasingly adopting high school graduation requirements in CS (Bender, 2024; Comp-Sci Graduation Mandate Proposed in California, 2024).

1.2 CSTA's Vision for the Future of K-12 CS Education

CSTA aspires to have every student prepared for a world powered by computing.

Our vision for K-12 CS education is to ensure:

- All students are engaged and supported in learning CS, including its impacts on individuals, societies, cultures, democracies, and policies.
- Policies, pedagogies, and practices support all students learning CS.
- Standards align with the current and future needs for learning CS.

As a step toward this vision, CSTA spearheaded the *Reimagining CS Pathways: High School and Beyond* project to explore how CS learning opportunities can be reenvisioned for high school students. Co-led by the Institute for Advancing Computing Education (IACE), and partnering with ACM, Code.org, the College Board, CSforALL, and the Expanding Computing Education Pathways (ECEP) Alliance, our purpose of this National Science Foundation (NSF)–funded project is to develop community definitions that answer two key questions:

1. What CS content is essential for all high school graduates to know?
2. What pathways should exist to continue learning beyond the foundational high school content?

In a world increasingly powered by computing, students of all identities and chosen career paths need quality CS education to become informed citizens and confident creators of content and digital tools.

We aim not only to develop recommendations to inform the future of the CSTA K-12 Standards and Advanced Placement (AP) CS courses but also to clarify the alignment of and to develop example pathways for CS learning from high school through introductory computing experiences at the postsecondary level.

CS is often understood as synonymous with coding. But CS skills, especially in the age of generative AI, extend far beyond coding, and very few K-12 students will ultimately work as software developers. Instead of viewing CS as just coding, we adopt the view that CS is focused on developing individuals who not only have the necessary technical knowledge and skills but also are prepared to be responsible creators, citizens, workers, consumers, and policymakers in a variety of domains (Tissenbaum & Ottenbreit-Leftwich, 2020).

Ethics and social impacts of computing must also be integrated into K-12 CS at every point of instruction and in every grade level. Given the emerging technology and its implications on individuals and societies, it is critical for students to learn the role that technology has in their lives as it begins to shape it at every turn.

It is also important to take into consideration the growing body of research that points to dispositions as key to the intent to persist in the study of computing, particularly for students belonging to groups that have been historically marginalized in the field. For example, developing a sense that one belongs in a field is an important indicator of the intent to persist in that field (Baumeister & Leary, 1995). However, the relationship between a sense of belonging and an intent to persist has important equity implications, because girls and women often have less opportunity to develop a sense of belonging in CS (Lewis et al., 2017), a phenomenon that exists even among the youngest students (Master et al., 2016). Further, sense of belonging is correlated with better academic performance (Krause-Levy et al., 2021).

Given the dramatic changes forecasted in computing over the next decade, we engaged in this project using a concerted and community-driven effort to build capacity for the infrastructure and supports needed to accommodate the evolution of K-12 CS education over the next five to ten years. We sought to balance the needs and perspectives of K-12 instructors, higher education instructors, industry, researchers, and district- and state-level computing officials. This effort and our corresponding report have been designed to provide data and recommendations to inform a new version of the CSTA standards, reflecting the current and anticipated changes in computing so that high school students' learning needs are met, as well as provide recommendations for future iterations of AP CS courses and future work of various members of the CS education community.

1.3 The Project

The *Reimagining CS Pathways* project entailed the following:

- **Three convenings** of representatives from across the K-16 CS education landscape (including teachers, administrators, two- and four-year college instructors, curriculum developers, and industry), with written summaries shared with the public;
- The creation of a set of **recommendations on the foundational content** that should be included in experiences/courses satisfying a high school graduation requirement, and how future CSTA standards and AP CS courses might be adjusted to align with such a requirement;
- The creation of a set of **models of high school CS courses** (high-level course descriptions and outcomes) that create potential pathways beyond an introductory course for all students; and
- The creation of a **framework that enables a systematic and deliberate process** for examining and re-creating similar pathways in the future.

Feedback to inform recommendations and next steps was gathered from a diverse cross section of the CS education community and included both synchronous and asynchronous opportunities for interactive feedback (see Section 9.1).

1.3.1 Project Values

In support of the aims of this project, the following project values have been identified and were leveraged for continuous reflection on progress and refinement of deliverables.

Equity-centered. Promotes broad and equitable access, participation, and experiences in CS education among all high school students.

Community-generated. Meets the needs of the community, including K-12 educators, postsecondary institutions, students, parents, and industry.

Future-oriented. Anticipates future needs of current high school learners, and prepares them for a future that is increasingly reliant on computing.

Grounded in research. Reflects the evolving body of knowledge of how students learn CS.

Flexible in implementation. Considers multiple pathways for meeting individual needs of learners, including regional, cultural, ability, social, and economic factors.

The CSTA K-12 Computer Science Standards delineate a core set of learning objectives designed to provide the foundation for a complete CS curriculum and its implementation at the K-12 level. There have been four iterations published between 2003 and 2017, with a fifth iteration currently in development.

2003 — A Model Curriculum for K-12 Computer Science, 2003 (Tucker, 2003)

2006 — A Model Curriculum for K-12 Computer Science, Second Edition (Tucker et al., 2006)

2011 — CSTA K-12 Computer Science Standards, Revised 2011 (Seehorn et al., 2011)

2017 — CSTA K-12 Computer Science Standards, Revised 2017 (Seehorn et al., 2017)

2026 — CSTA K-12 Computer Science Standards, Revised 2026 (expected)

The content from this report will directly inform the fifth iteration of the CSTA K-12 Standards, with a planned release in summer 2026.

2 Foundational Content

2 Foundational Content

2.1 Priorities in the Foundational Content

What is prioritized in this report stems from the academic research as well as from what was prioritized by experts who participated in the convenings and provided feedback as the project progressed (see Section 9.1). These priorities are briefly described in the following sections.

Social Impacts and Ethics

Social impacts of computing and related ethical implications are of critical importance in the essential content. To reflect this prioritization, we included content related to societal impacts and ethical issues within each Topic Area. This content includes but is not limited to instruction related to social justice, equity, and, more generally, ensuring that computing benefits all members of society, especially the most vulnerable. We expect a foundational course to spend a substantial amount of time on these issues, and to do so in a way that integrates these ideas with more technical topics so that students understand the interwoven relationships between technical considerations (e.g., how data is represented in a system) and societal and ethical implications (e.g., whether a user can enter data into a form in a way that respects their identity, such as the use of characters not found in English, their preferred way of describing their gender).

Algorithms and Programming

Algorithms and programming are also prioritized, matching the significance placed in current curricula and high school instruction. However, there is a greater emphasis on algorithms and computational thinking and a reduced emphasis on programming, relative to what is typically taught today. This shift in emphasis is justified by emerging technologies, such as generative AI; the need for students to learn programming is thus balanced with skills in reading, modifying, and debugging code. There is ongoing importance of "the basics" in order to fully understand and leverage technological advancements. To reflect this prioritization, we defined two Topic Areas: (1) Algorithms and (2) Programming, and one Pillar: Computational Thinking.

Data and Analysis

Content related to data and its analysis is also a priority, reflecting the increased prevalence of data in daily aspects of life as well as the vast amount of data upon which emerging AI technologies are built. This trend also acknowledges data science as a burgeoning and increasingly important field with strong foundations in CS. We defined Data and Analysis as one of five Topic Areas to reflect this prioritization.

Inclusive Collaboration

The prioritization of inclusive collaboration as a Pillar recognizes equity as a value. Key skills include respecting diverse perspectives and experiences in CS, recognizing and addressing biases, and advocating for the needs of others.

Computing Systems and Security

Two of CSTA's concepts, (1) Computing Systems and (2) Networks and the Internet, were often grouped together when convening participants were indicating priorities across Topic Areas. Interrelated and complementary content within each of these concepts led us to combine them into one Topic Area: Computing Systems and Security.

Artificial Intelligence

AI is treated not as a discrete topic, but it is included in essential content organized within other Topic Areas. There is also some emphasis on emerging technologies, to account for significant advances in the future that cannot be predicted; we included this within a new Topic Area called Preparation for the Future.

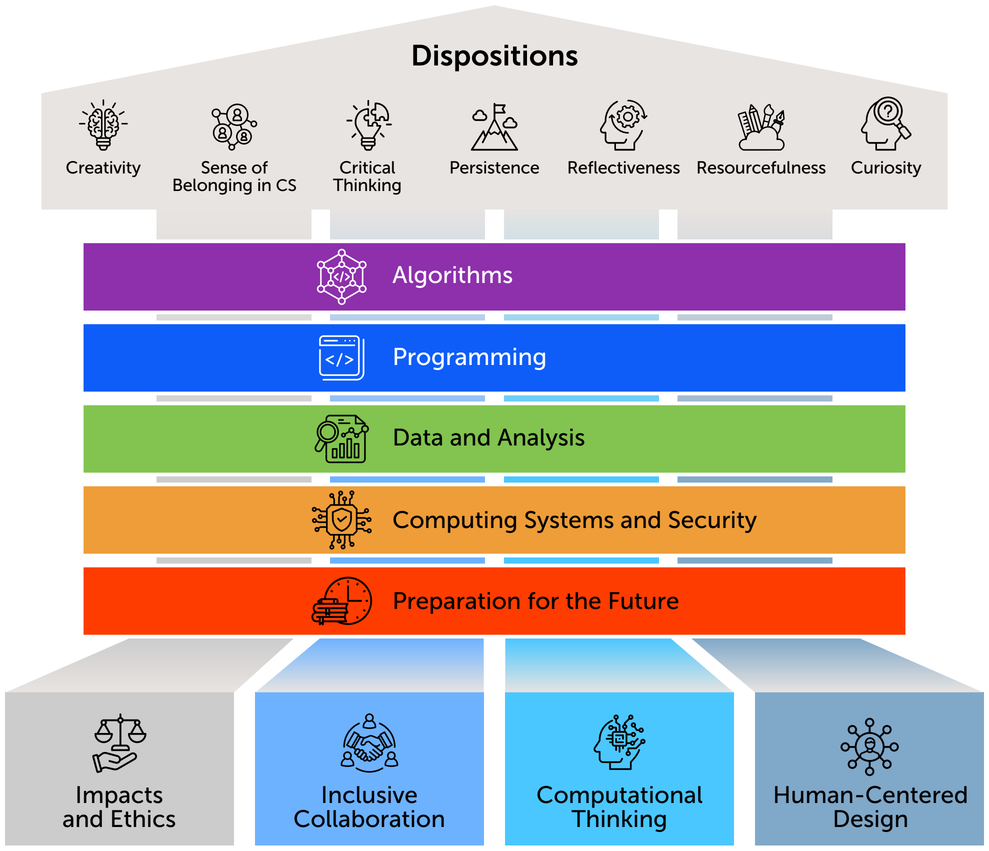
Careers

Knowledge of careers – both those in computing and those involving computing – are identified as part of essential CS content. This is perhaps the most significant departure from the 2017 CSTA K-12 Standards and current implementation. This priority on career awareness highlights that all disciplines and career fields are becoming increasingly reliant on or impacted by computing. To reflect this prioritization, career-related learning outcomes are included in the new Topic Area called Preparation for the Future.

2.2 Introduction to the Foundational Content

Recommended high-level topics emerged through analysis of convening data and review of relevant research. As described in Section 9.2.2 on the challenge of organizing content, this project began with CSTA's concepts and practices and adjusted them to accommodate participant priorities in a manner that minimized both gaps and overlaps. As shown in Figure 2.2, foundational content is organized into five Topic Areas, four Pillars, and seven Dispositions, each of which is described in more detail in this section.

Figure 2.2: Overview of foundational content.



2.3 Dispositions

In contrast to knowledge and skills, *dispositions* are broad-based habits of mind that are not discipline-specific (Claxton, 2009). Researchers have identified dispositions that are correlated (positively or negatively) with learning outcomes. For example, a disposition toward critical thinking can improve student learning (Bell & Loon, 2015).

Based on participants' contributions, we include the Dispositions in Table 2.3 as part of a foundational high school CS experience; note that these Dispositions are interrelated. For example, CS experiences that emphasize creativity have been shown to correlate with an increased sense of belonging in CS (Ryoo & Tsui, 2023). More importantly (and interesting from a theoretical standpoint), five of the seven map to self-regulated learning (persistence, reflectiveness, curiosity, critical thinking, and resourcefulness). Self-regulated learning is learning guided by a combination of the learner's metacognition (thinking about one's thinking), strategic action taken by the learner (planning, monitoring, and evaluating personal progress against a standard), and the learner's motivation to learn (Butler & Winne, 1995). Teaching activities that may support self-regulated learning may include learner self-assessment, reciprocal teaching where students who learned the material teach other students, and activities that encourage learners seeking help.

Table 2.3: An overview of the Dispositions.

Creativity	Creativity is "the interaction among aptitude, process, and environment by which an individual or group produces a perceptible product that is both novel and useful as defined within a social context" (Plucker et al., 2004, p. 90). Creativity is important to CS as a discipline (Giza, 2021), and incorporating opportunities for creativity into CS courses improves student learning (Sharmin, 2021).
Sense of Belonging in CS	A sense of belonging, or the "personal involvement (in a social system) to the extent that the student feels that they are an indispensable and integral part of the system" (Anant, 1967, p. 391), is one of the more widely researched dispositions in CS education. Its importance is linked to its relationship to a student's sense of their own ability in (Veilleux et al., 2013) and interest in persisting in their studies (Hansen et al., 2023). Sense of belonging is an important facet of ensuring equity in CS, since this sense often differs by student demographic group (Krause-Levy et al., 2021).
Critical Thinking	Critical thinking includes "the mental processes, strategies, and representations students use to solve problems, make decisions, and learn new concepts" (Sternberg, 1986, p. 3). A student engaged in critical thinking goes beyond lower-level thinking (such as memorizing and recalling information) to analyze, interpret, and synthesize information (Kennedy et al., 1991). Critical thinking ability is widely recognized as a crucial skill for modern workers (van Laar et al., 2020).
Persistence	Persistence is the "voluntary continuation of a goal-directed action in spite of obstacles, difficulties, or discouragement" (Peterson & Seligman, 2004, p. 229). Persistence has been identified as important to student success in CS courses. Research teasing out the differences between productive and unproductive forms of persistence is ongoing (Pinto et al., 2021).
Reflectiveness	Reflectiveness is the process of "turning experience into learning" (Boud et al., 2013) through the student thinking about the results of past actions and allowing that prior knowledge to inform predictions of possible outcomes of future actions. Reflective activities in CS courses have been shown to improve learning outcomes (Zarestky et al., 2022).
Resource-fulness	Resourcefulness refers to a student's strategic use of available resources, their ability to regulate their emotions and cognition, solve problems, and delay gratification for greater future rewards (Rosenbaum, 1989). Resourcefulness therefore invokes many components of self-regulated learning (Panadero, 2017; Zimmerman, 2008). As with curiosity, resourcefulness is not widely studied by CS education researchers, but its importance has been shown in other educational research (Dison et al., 2019; Kennett & Keefer, 2006).
Curiosity	Curiosity is the desire for new knowledge, information, experiences, or stimulation to resolve gaps or experience the unknown (Arnone et al., 2011; Berlyne, 1954; Litman, 2005). While the role of student curiosity is understudied in CS education, there is a broad base of educational research affirming the correlation between curiosity and learning outcomes (Stumm et al., 2011).

2.4 Pillars

The Pillars are neither Topic Areas nor Dispositions; rather, they are practices, methods, and approaches that are an integral part of each Topic Area. Previous research has shown that incorporating ideas from the Pillars, such as inclusive design, improves CS learning outcomes as well as improves students' ability to apply the concepts themselves (Garcia et al., 2023).

Pillars are designed to be embedded intentionally into all Topic Areas, with a focus on curricular design and instructional methods, that emphasize the Pillars throughout CS instruction. For example, a lesson on cybersecurity may include human-centered design practices to ensure the protocol is user-friendly and inclusive collaboration to ensure that it works for those with disabilities or non-English speakers, or may include incorporating recognition of the broader impacts of social media into the design of an app.

The following sections describe in more detail the four Pillars: Impacts and Ethics, Inclusive Collaboration, Computational Thinking, and Human-Centered Design.

2.4.1 Impacts and Ethics

As computing becomes ever more pervasive, its social and ethical implications also become more important. Thus, it is crucial that students' understanding of these implications grows alongside their understanding of more technical concepts to ensure that computing benefits all members of society, especially the most vulnerable. Discussion of Impacts and Ethics includes but is not limited to:

- Societal impacts of computing
- Ethical issues in computing
- Social justice issues
- Access and equity concerns
- Safety and privacy

Content related to societal impacts and ethical issues of computing should be integrated into each Topic Area. Developing the capacity to reason about ethical issues related to computing is a key component of CS education ("Justice-Centered Computing," n.d.; Ko et al., 2024; Tissenbaum & Ottenbreit-Leftwich, 2020), and the development of citizenship skills is a crucial component of the study of computing (Yadav & Heath, 2022). We expect a foundational course (or a similar foundational experience) to devote a substantial amount of time to these issues, and to do so in a way that integrates these ideas with more technical topics so that students understand the interwoven relationships between technical considerations (e.g., how data is represented in a system) and societal and ethical implications. Specifically, students will develop the following skills:

- Recognize the ethical implications of design decisions
- Understand the societal impacts of computing technologies (e.g., social networks, facial recognition)
- Be able to articulate arguments for and against various policies and laws related to computing (e.g., net neutrality, limits on children's use of social media)
- Appropriately provide attribution for code that was produced by others or produced by AI

2.4.2 Inclusive Collaboration

As previously noted, one of the core values of the *Reimagining CS Pathways* project is equity-centeredness. Key skills within this practice include respecting diverse perspectives and experiences in CS, recognizing and addressing biases, and advocating for the needs of others. These skills overlap with other important skills related to communication and collaboration within CS. The core of inclusive collaboration is, as one participant phrased it, to *engage with diverse perspectives with respect and empathy.* Specifically, students will develop the following skills:

- Awareness of and Empathy with Others
    - Accommodate a variety of identities and perspectives, including from those with disabilities and from different cultural backgrounds.
    - Recognize and mitigate personal biases.
    - Provide support services to other people and groups via computing.
    - Support the learning of others.
    - Design and develop with accessibility in mind.
- Collaboration Skills
    - Recognize different roles on a team, and be able to assume different roles.
    - Seek out and use feedback from others.
    - Provide others with constructive feedback.
    - Advocate for the needs of others.
    - Use appropriate tools, including digital tools, for collaboration.
    - Use a variety of models and methods for collaboration, including pair programming.
    - Be able to communicate about technology in a variety of contexts, including with those with expertise in computing (by using precise, domain-specific vocabulary) and with those with limited technical knowledge (by translating technical concepts into common language).
    - Be able to document products and processes.
    - Apply principles of digital citizenship including data security, responsible communication, information evaluation, and respect for intellectual property.

2.4.3 Computational Thinking

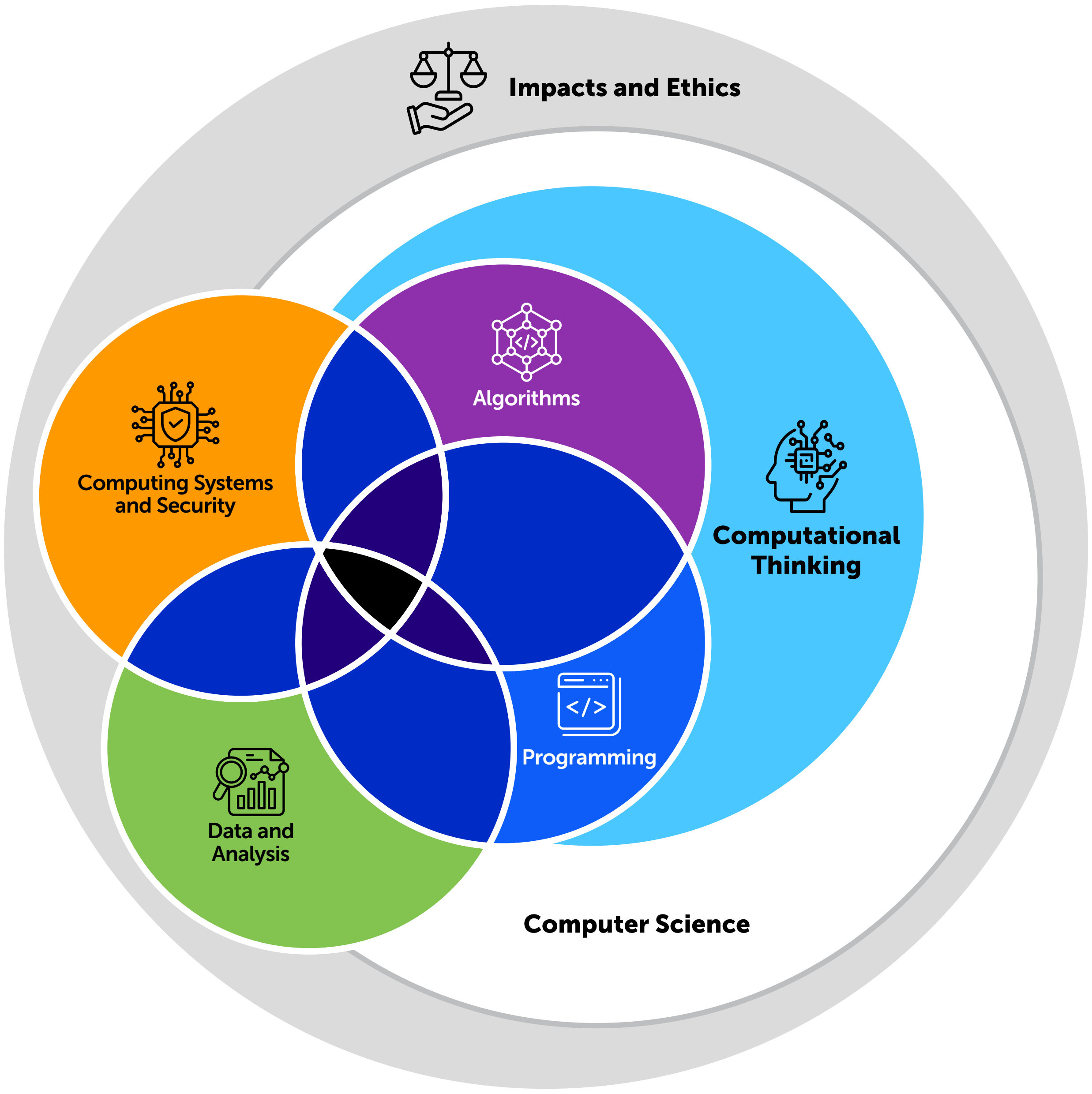
As mentioned in Section 9.2, one of the challenges of this project is the inability to anticipate which computing technologies will be central in the future (e.g., quantum computing, new advances in AI). An emphasis on computational thinking skills – as opposed to more technical implementation skills – can better position students to address whatever technologies become predominant in the future (Tissenbaum & Ottenbreit-Leftwich, 2020).

Computational thinking is most commonly defined as involving thinking processes that structure problems in a way that an information-processing agent (i.e., a computer) can solve them (Shute et al., 2017). Computational thinking includes algorithm development, decomposition, pattern recognition, and abstraction, and it is "a fundamental skill for everyone, not just for computer scientists" (Wing, 2006). The tenets of computational thinking should underpin instruction in each Topic Area and serve as connective tissue across CS learning experiences. Specifically, students will be able to (ISTE & CSTA, 2011):

- Formulate problems in a way that enables the use of a computer and other tools to help solve them.
- Decompose problems into smaller, more manageable subproblems.
- Identify, analyze, and implement possible solutions with the goal of achieving the most efficient and effective combination of steps and resources.
- Automate solutions through algorithmic thinking (a series of ordered steps).
- Recognize and describe patterns in problems, data, and programs.
- Represent data through abstractions such as models and simulations.
- Generalize and transfer this problem-solving process to a wide variety of problems.

Computational thinking does not stand apart from CS; computational thinking is embedded within CS. Similarly, algorithms cannot stand apart from computational thinking. Given the Topic Areas in the next section, we created a visual representation of our vision of how Computational Thinking, Algorithms, Programming, and CS overlap. While we debated frequently about how these areas should be represented within our findings, we maintain that separating them into Topic Areas and the Pillar of computational thinking made the most sense. However, we recognize that there are various ways to interpret how these concepts fit together. Figure 2.4.3 illustrates our vision of the overlap between CS, Computational Thinking, Algorithms, and Programming. We added two other Topic Areas – (1) Data and Analysis and (2) Computing Systems and Security – to show where they may fit as well. The importance of impacts and ethics is also noted.

Figure 2.4.3: Vision of the overlap between CS, Computational Thinking, Impacts and Ethics, and Topic Areas.



2.4.4 Human-Centered Design

With no-code and low-code environments expected to evolve rapidly over the next few years (Bock & Frank, 2021), there will be even more opportunities to infuse design thinking into high school CS learning experiences. Human-centered design will be critical as programming continues to be automated. Coupled with algorithmic thinking and auditing, humans will be needed to build empathy and understanding into the design, feed the design to the AI "programmer," and audit and refine the results.

Human-Centered Design encompasses aspects of planning that are user-focused. Specifically, students will be able to:

- Understand principles of effective design for people, including identifying problems and understanding underlying causes.
- Empathize with people impacted by the problems, including designing for accessibility and the socio-historical-cultural context.
- Think of everything (and approach solutions) as a system designed for humans.
- Generate ideas to solve problems, including considering who is affected by design choices and how they are affected.
- Prototype, test with users, and iterate solutions.
- Understand how design decisions shape the end user's experience.

We provide examples for many of the items that constitute each Topic Area and for how the Pillars and Dispositions might be treated in each Topic Area. It is important to note that these are examples meant to provide an indication of expected depth, breadth, and granularity – they are not requirements.

2.5 Topic Areas

We organized content into five Topic Areas: Algorithms, Programming, Data and Analysis, Computing Systems and Security, and Preparation for the Future (see Appendix A for the relationship between this organization and the concepts and practices found in the 2017 CSTA K-12 Standards). We then used Bloom's Revised Taxonomy (Forehand, 2010) to organize the CS content within each Topic Area. Bloom's taxonomy was developed to provide a common language for educators to communicate about learning and assessment methods through a framework for each stage of learning. Bloom's involves a progression from lower-level knowledge, such as remembering a definition, to higher-level application, such as creating a program. We have articulated the foundational content according to Bloom's Revised Taxonomy (in the "Level" column of the tables in this section), though we acknowledge that it is an imperfect tool, particularly in CS (Fuller et al., 2007). An ACM task force mapped verbs commonly used in computing (e.g., *deploy, model, script*) to Bloom's Revised Taxonomy (ACM Committee for Computing Education in Community Colleges (CCECC), 2023; Geissler et al., 2023). We leveraged their work as the basis for organizing the computing content within each Topic Area.

2.5.1 Algorithms

Algorithms – step-by-step processes to complete a task involving computation – are a fundamental part of CS, and understanding them is foundational for further work in computing. In this Topic Area, students are exposed to high-level concepts related to algorithms.

In the topic area tables, we use a system of superscripts to indicate which Pillars relate to which learning outcome:

- **Computational Thinking** ⟶ **CT**
- **Human-Centered Design** ⟶ **HCD**
- **Inclusive Collaboration** ⟶ **IC**
- **Impacts and Ethics** ⟶ **IE**

Table 2.5.1.1: Algorithms foundational content.

Level	Learning Outcome
Remember	AL.1 Define algorithm and explain what algorithms are used for^{CT}
	AL.2 Recognize that computational solutions take in information, store and process it, and produce a result
Understand	AL.3 Describe the difference between traditional algorithms and artificial intelligence/machine learning (AI/ML) algorithms and, at a high level, describe how AI/ML algorithms work^{CT}
	AL.4 Explain why/how sequence matters in an algorithm^{CT}
	AL.5 Interpret algorithms^{CT}
Apply	AL.6 Modify algorithms (e.g., to add functionality)^{CT}
	AL.7 Investigate what is inside of an opaque system (i.e., a system whose operation is not transparent) when it is necessary to do so
	AL.8 Apply principles of inclusive collaboration to a project involving the use of algorithms^{IC}
Analyze	AL.9 Compare (at a high level) the trade-offs (e.g., speed, memory) of different algorithms^{CT}
Evaluate	AL.10 Evaluate the appropriateness, reasonableness, and/or effectiveness of an algorithm for a specific task, including via algorithmic auditing^{CT}
	AL.11 Assess societal impacts of algorithms and related ethical issues (e.g., use of AI algorithms to choose job candidates, use of abstraction to obscure important context)^{IE}
Create	AL.12 Compose algorithms using sequence, selection, and iteration^{CT}
	AL.13 Design algorithms using principles of human-centered design^{HCD}

Table 2.5.1.2: Examples of integrating the Pillars and Dispositions into the Algorithms foundational content.

Impacts and Ethics	Inclusive Collaboration	Computational Thinking	Human-Centered Design	Dispositions
Create an algorithm that designs congressional districts fairly	Swap algorithms across project groups to debug	Decompose an algorithm	Create algorithms that solve problems relevant to the student's local context	Take an iterative approach to algorithm design – persist despite mistakes
Design algorithms for a chat room, considering content moderation policies	Design algorithms that support diverse identities (e.g., entering names in a form with various characters)	Analyze how abstractions can lead to biased results from algorithms	Test algorithm prototypes to ensure they meet users' needs	Reflect on personal experiences making trade-offs between privacy and transparency in a system's design

2.5.2 Programming

Programming is construed broadly to describe a variety of ways of generating computational artifacts. Programming, in the context of essential content for high school, is likely to include block-based and/or text-based programming languages. It may also include other computational artifacts, such as simulations, visualizations, robotic systems, or digital animations.

This Topic Area involves more technical content, and it is sometimes taught in ways that do not engage students' interest or imagination. Innovative and creative activities, such as creating programs to generate digital art or to meet a community need, may be more engaging. Using the development of dispositions as a lens when designing instruction may help address this.

Table 2.5.2.1: Programming foundational content.

Level	Learning Outcome
Remember	PR.1 Reference documentation and other online tools to assist with programming
Understand	PR.2 Convert an algorithm to code
	PR.3 Interpret the function of a segment of code
Apply	PR.4 Modify a program (e.g., add functionality or improve usability or accessibility)
	PR.5 Use prompt engineering, code generation tools, or other AI technologies to plan, write, test, and debug code[CT]
	PR.6 Document a program to clarify functionality (e.g., using comments within code)
	PR.7 Apply principles of inclusive collaboration to a programming project[IC]
Analyze	PR.8 Articulate whether a program solves a given problem[CT]
	PR.9 Use computational thinking principles to analyze a program[CT]
Evaluate	PR.10 Test and debug a program systematically[CT]
	PR.11 Evaluate whether and how computation can or cannot help solve a problem
	PR.12 Assess societal impacts of programming and related ethical issues (e.g., how might modifications to a program impact various groups of users?)[IE]
Create	PR.13 Design a program using principles of human-centered design[HCD]
	PR.14 Develop programs using sequence, selection, and iteration

Table 2.5.2.2: Examples of integrating the Pillars and Dispositions into the Programming foundational content.

Impacts and Ethics	Inclusive Collaboration	Computational Thinking	Human-Centered Design	Dispositions
Explore a program's implications for data privacy and security	Work in diverse teams to develop a program	Apply knowledge of programming patterns to new contexts	Interview users to understand their needs	Reflect on one's choice(s) to emphasize speed, cost, efficiency, accuracy, etc. in the design of a program
Examine how biases might arise in a program's output	Collaborate via peer code reviews	Explore whether and how a problem can be solved without computing, then transform into a program as appropriate	Develop an app that is accessible to low vision users	Apply universal design for learning concepts to improve sense of belonging

2.5.3 Data and Analysis

Data and Analysis involves understanding how computing systems collect, store, and process data and how people can use this data to make inferences and predictions. The increasing importance of data science and artificial intelligence points to the increasing need for understanding the basic elements of data and its analysis.

Table 2.5.3.1: Data and Analysis foundational content.

Level	Learning Outcome
Remember	DA.1 Identify and define data types (e.g., string, numeric, Boolean)
	DA.2 Identify basic data formats (e.g., tables, schemas, JSON)
Understand	DA.3 Describe, at a high level, the role of data in AI/ML applications
	DA.4 Understand the difference between data and metadata
	DA5 Describe how different types of data (e.g., audio, visual, spatial, environmental) can be collected computationally
Apply	DA.6 Transform and prepare (e.g., normalize, merge, clean) data
	DA.7 Apply principles of inclusive collaboration to a project involving the analysis of data^{IC}
Analyze	DA.8 Trace how data moves through a program^{CT}
	DA.9 Analyze data using computational thinking principles to make inferences or predictions^{CT}
Evaluate	DA.10 Evaluate approaches to cleaning data in a given context
	DA.11 Assess whether and how a given question can be answered using computational methods and data, and what specific data is needed
	DA.12 Assess societal impacts of data analysis and related ethical issues (e.g., biased data used to train AI systems, attribution related to products of generative AI)^{IE}
	DA.13 Evaluate data visualizations for clarity, potential biases, etc.
Create	DA.14 Select, organize, interpret, and visualize large datasets from multiple sources to support a claim and/or communicate information
	DA.15 Devise plans for using data to solve a problem
	DA.16 Create a data analysis artifact (e.g., a visualization) using principles of human-centered design^{HCD}

Table 2.5.3.2: Examples of integrating the Pillars and Dispositions into the Data and Analysis foundational content.

Impacts and Ethics	Inclusive Collaboration	Computational Thinking	Human-Centered Design	Dispositions
Consider data privacy issues related to a program	Ensure representation of diverse voices in data collection, visualization, and analysis	Identify what data is needed to solve a problem	Help address a community concern using data	Use critical thinking skills to test data models against real-world datasets
Consider ethical issues related to data visualization, such as bias, accessibility, etc.	Iteratively analyze data with feedback from family or community	Identify patterns in data to solve a problem	Use data from real-world contexts	Use creativity in designing data visualizations

2.5.4 Computing Systems and Security

Computing Systems and Security includes the broad categories of hardware, software, troubleshooting, networks, and cybersecurity, as well as the idea that systems have multiple levels or layers that impact each other. The increased interconnectedness of large systems and their impact on safety and security underscore the importance of this Topic Area.

Table 2.5.4.1: Computing Systems and Security foundational content.

Level	Learning Outcome
Remember	CS.1 Identify various types of hardware (including components) and software (including operating systems)
	CS.2 List security practices (e.g., safe passwords, two-factor authentication)
Understand	CS.3 Explain what networks (including the Internet) are and how they work
	CS.4 Explain how an operating system, other software, and hardware work together
	CS.5 Describe why cybersecurity is important
Apply	CS.6 Optimize operating systems and other software settings to achieve goals
	CS.7 Apply knowledge of the structure and function of various technologies (e.g., sensors, global positioning system (GPS), embedded/IoT, phones/tablets, medical devices, VR, robotics) to their use (e.g., explain why GPS can be used without Internet access)
	CS.8 Use documentation and other resources to guide tasks such as installation and troubleshooting
	CS.9 Apply principles of inclusive collaboration to a project involving computing systems and/or security^{IC}
Analyze	CS.10 Describe vulnerabilities in networks
	CS.11 Analyze a problem to determine appropriate troubleshooting strategies
	CS.12 Use computational thinking principles to analyze a computing system^{CT} (e.g., automate a security assessment)
Evaluate	CS.13 Assess societal impacts of networks and related ethical issues (e.g., digital divide)^{IE}
Create	CS.14 Design projects that combine hardware and software that collect and exchange data
	CS.15 Design a computing system or security protocol using principles of human-centered design^{HCD}

Table 2.5.4.2: Examples of integrating the Pillars and Dispositions into the Computing Systems and Security foundational content.

Impacts and Ethics	Inclusive Collaboration	Computational Thinking	Human-Centered Design	Dispositions
Discuss issues of privacy related to networks	Write a press release in response to a data breach	Troubleshoot a network	Understand the psychological aspects of threat reduction	Leverage curiosity to approach problems from different perspectives
Explore debates related to networks (e.g., Internet as a public utility, net neutrality)	Work as a team to conduct a security audit	Use decomposition to understand how a system's components work	Explore the trade-offs between security and usability	Be resourceful in solving networking challenges (e.g., reference documentation)

2.5.5 Preparation for the Future

Preparation for the future brings together two threads: 1) the student's own future, specifically pathways and careers that involve computing in some respect and 2) emerging technologies, including their societal implications and ethical issues. The inclusion of this Topic Area aligns with a major priority of convening participants and marks a shift from some existing CS standards and frameworks.

Table 2.5.5.1: Preparation for the Future foundational content.

Level	Learning Outcome	
Remember	PF.1	Identify pathways and careers that involve computing
Understand	PF.2	Explain how computing enables emerging technologies (e.g., autonomous vehicles) and how these emerging technologies are applied in various industries
Apply	PF.3	Apply computing concepts to other disciplines (e.g., investigate how to collect, process, and analyze heart rate sensor data in physical education)
	PF.4	Apply principles of inclusive collaboration when using emerging technologies^{IC}
Analyze	PF.5	Examine how emerging technologies are impacting a variety of practices (e.g., use of facial recognition in policing, AI-generated news products)
	PF.6	Analyze emerging technologies using computational thinking principles^{CT}
Evaluate	PF.7	Assess societal impacts and related ethical issues of emerging and future developments in computing (e.g., the impact of quantum computing on security)^{IE}
	PF.8	Evaluate the use of emerging technologies (e.g., generative AI) for accuracy and to meet specific needs^{IE}
Create	PF.9	Develop a personal career plan that highlights the use of computing
	PF.10	Create a plan to apply an emerging technology to meet a need using principles of human-centered design^{HCD}

Table 2.5.5.2: Examples of integrating the Pillars and Dispositions into the Preparing for the Future foundational content.

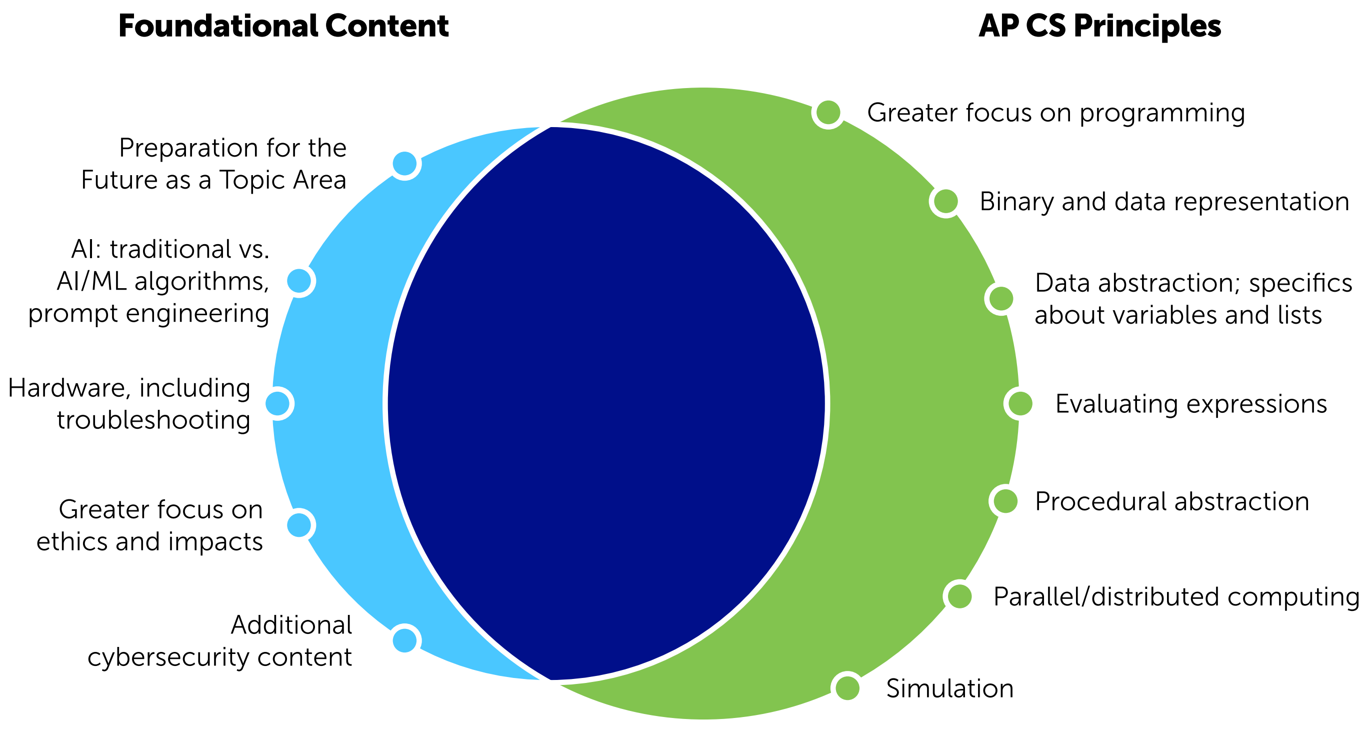
Impacts and Ethics	Inclusive Collaboration	Computational Thinking	Human-Centered Design	Dispositions
Consider the potential for future misuse of an application	Work collaboratively with diverse and distributed teams	Understand the components of emerging technologies	Use empathy to meet users' needs	Leverage resourcefulness to be flexible in the face of future changes
Explore how technology impacts interpersonal relationships	Follow best practices for accessibility and universal design	Recognize how the principles of CT can be applied across a variety of fields	Explore how design may change due to future innovations	Understand the qualities needed for success in CS careers

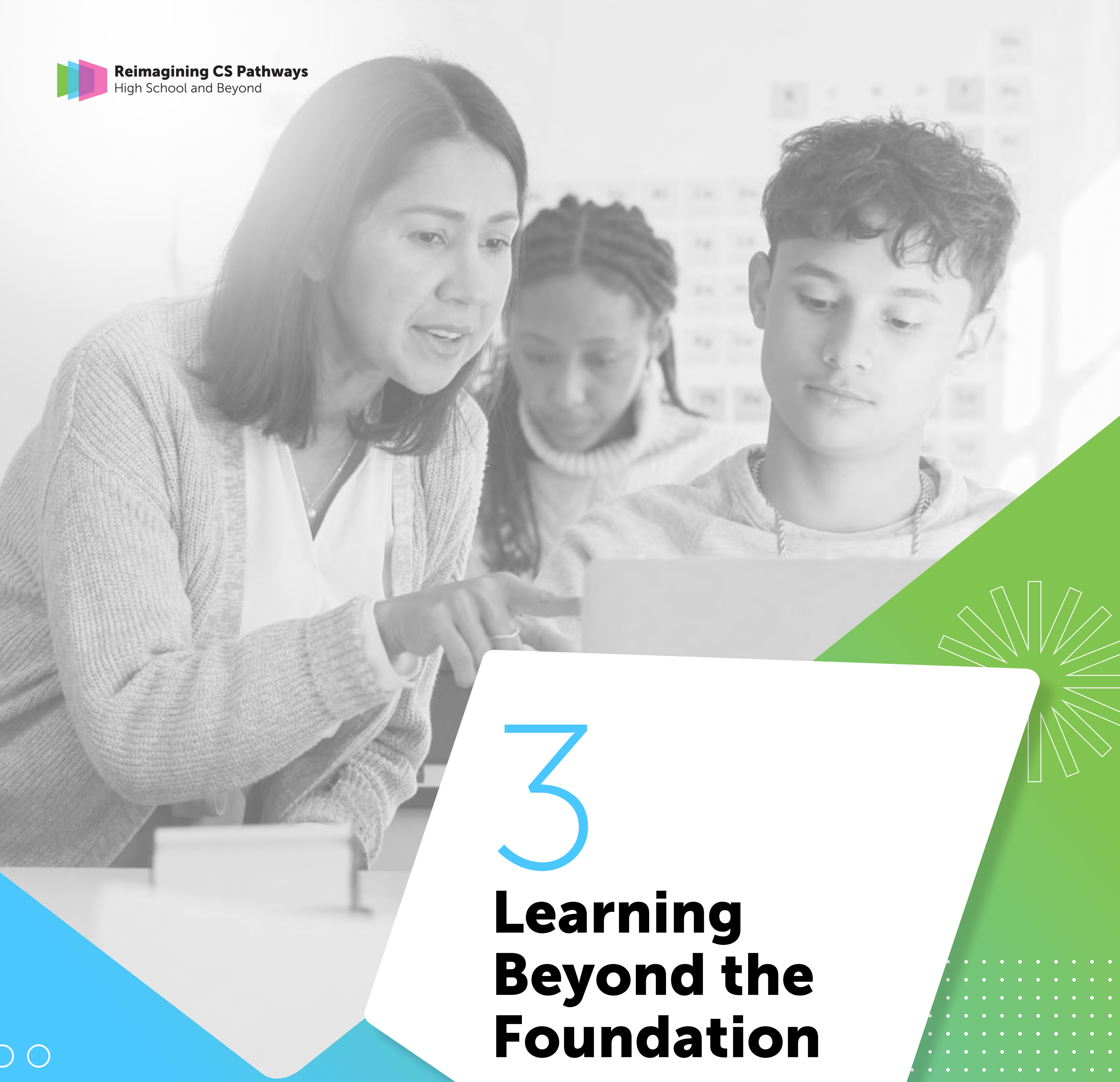
2.6 Alignment between Foundational Content and AP Computer Science Principles

One aim of this project is to identify the alignment between the foundational content outlined above and Advanced Placement (AP) coursework in CS. Through a high-level comparison of the foundational content to the learning objectives found in the 2023 AP Computer Science Principles (CSP) course framework (College Board, 2023), significant overlap between the two has been identified. Figure 2.6.1 illustrates the overlap and notable distinctions between the foundational content and the AP CSP course. For example, the foundational content includes AI, careers alignment, hardware, more cybersecurity, and a greater focus on ethics and impacts, whereas AP CSP includes a greater focus on programming, as well as binary, data and procedural abstraction, and parallel computing. At the time of this report's publication, the College Board is in the early stage of revising AP CSP. Recommendations for the AP CSP revision process are noted in Section 8.

In general, there is relatively little content that is found in the foundation but not found in AP CSP. This suggests students enrolled in an AP CSP course might be able to experience all of the foundational content with some minor adjustments/additions to the course. It should be noted that while the AP CSP course might serve as a vehicle for bringing a foundational CS experience to students, equity concerns arise if it is the only available option for students to experience foundational CS content. Analogous circumstances exist in other content areas such as social studies – schools that offer AP U.S. History will offer another (non-AP) U.S. history course because the AP course may not meet the needs of all students. Similarly, schools that offer a foundational CS learning experience using AP CSP should also offer the foundational content in another format (e.g., non-AP course, integrated into other content area(s)/course(s)).

Figure 2.6.1: Overlap and notable distinctions between the foundational content and AP CS Principles.


Foundational Content
AP CS Principles
Preparation for the Future as a Topic Area
AI: traditional vs. AI/ML algorithms, prompt engineering
Hardware, including troubleshooting
Greater focus on ethics and impacts
Additional cybersecurity content
Greater focus on programming
Binary and data representation
Data abstraction; specifics about variables and lists
Evaluating expressions
Procedural abstraction
Parallel/distributed computing
Simulation



3 Learning Beyond the Foundation

3 Learning Beyond the Foundation

Stemming from the foundational CS content are numerous potential pathways for further study. While pathways that focus on preparing students for the study of computing in college tend to get the most attention, other postsecondary pathways can play important roles in broadening participation. For example, CS bootcamps tend to draw a relatively higher proportion of women and can meet the needs of those who developed an interest in computing at a time that they felt was too late to major in it (Lehman et al., 2020; Lyon & Green, 2020; Zhu et al., 2022). The pathways developed in this project are designed to meet the needs of all students, not just those who will study computing in college, and their communities. Framing pathways with the recognition that the vast majority of students will *not* major in computing in college helps encourage the identification of other pathway endpoints – endpoints that will likely impact a much larger number of students and may also boost civic engagement and personal empowerment (Tissenbaum et al., 2021).

The model courses and descriptions should continue to integrate the Dispositions and the Pillars that were articulated for the foundational content (see Sections 2.3 and 2.4).

The following subsections first describe a series of content progressions with particular focus areas (e.g., artificial intelligence, data science) that delineate high-level learning outcomes for students as they progress through computing coursework beyond the foundation. Example pathways are provided in Section 5.2, which suggest how the content progressions might be packaged into meaningful course sequences that can be offered in high schools. While it is unlikely that a school would be able to offer all of the content progressions detailed below, this section provides a breadth of options from which schools might be able to choose based on student interests, community needs, and resource availability.

In each of the content progressions, we use a three-column format:

Foundational CS Content:	Fundamentals:	Specialty:
This column includes the graphical overview of all foundational CS content as outlined in Section 2.5, as well as bullet points highlighting content that is particularly relevant to the given content progression. (Note that it is presumed that all students will have experienced all of the foundational content.)	This column includes content beyond the foundation that is essential for the given area of focus (e.g, AI, cybersecurity).	This column includes content that builds upon the fundamental content for the given area of focus (e.g., AI, cybersecurity).

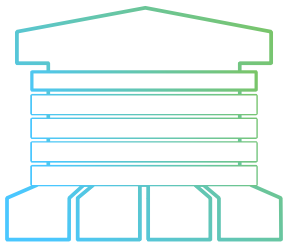
Possible careers

We also list possible careers stemming from each progression. Note, however, that the progression will also be an appropriate choice for many students who are interested in other career paths. For example, a student who plans to be an attorney or a business owner may find the cybersecurity pathway to be relevant to their career goals.

3.1 Programming Content Progression

Table 3.1 shows the content progression developed for students who are interested in continuing to learn programming or applications of CS more broadly, such as those intending to major in CS and potentially become a computer scientist or software engineer.

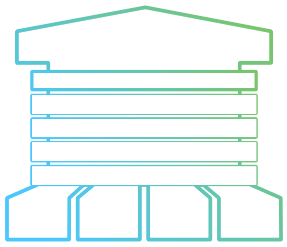
Table 3.1: Programming content progression.

Foundational CS Content	Fundamentals	Specialty
 **Prioritized foundational content specific to programming:** • Programming skills • Inclusive collaboration while programming • Ethics and social impact • Testing and debugging • Designing, composing, and interpreting algorithms • Preparation and use of data in programming • Cybersecurity basics • Hardware and devices	• Decomposition • Problem-solving • Conditions, iterations, selection, functions • Abstractions and models representing a system • Arrays and data structures • Unit testing • Debugging • Usage of integrated development environments (IDEs) • Algorithm optimization	• Programming skill development • Software development processes (e.g., Agile/Scrum) • Application development (e.g., mobile apps, virtual reality apps) • Team project skills • Collaborative source control • Correctness and provability of algorithms
Possible careers	• Computer Scientist • Software Engineer • Data Scientist • Artificial Intelligence Specialist	• Cybersecurity Specialist Network Specialist • Roboticist • CS Teacher/Instructor/Professor • CS Education Researcher

3.2 Cybersecurity Content Progression

Table 3.2 shows the content progression for acquiring more knowledge specifically related to cybersecurity. This content might lead to a major in cybersecurity or to earning industry certifications, followed by a career as a network technician, security analyst, or network systems administrator.

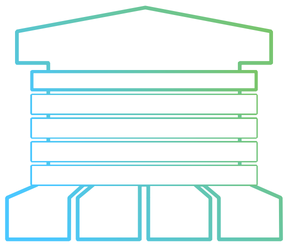
Table 3.2: Cybersecurity content progression.

Foundational CS Content	Fundamentals	Specialty
 **Prioritized foundational content specific to cybersecurity:** • Types of hardware and software (including operating systems) • How hardware/ software work together • Security practices (e.g., safe passwords, two-factor authentication) • Importance of cybersecurity • How networks work • Optimizing networking and operating system (OS) settings • Troubleshooting • Using documentation • Network vulnerabilities • Ethical issues (e.g., digital divide)	• CIA (confidentiality, integrity, and availability) triad, states of data, and types of controls • Basics of digital communication (open systems interconnection (OSI) model, protocols, ports, etc.) • Network fundamentals (protocols, topologies, and addressings; network hardware and their roles (servers, switches, routers, endpoints, firewalls)) • Command line in various operating systems • Network troubleshooting • Network management tools • Basic computing systems • Cybersecurity-related hardware/ software roles and components • Basic understanding of file systems • Impact of cybersecurity on society and critical infrastructure • Small office/home office (SOHO) / home networks • Types of attacks, threats, vulnerabilities, and basic remediation strategies • Wifi versus Internet • Public networks • Network addressing (Internet protocol (IP) addressing, medium access control (MAC) addressing) • Careers in cybersecurity • Scripting • Impact of AI on cybersecurity • Industry certification preparation	• High-level understanding of policies and why they matter • Basic application security • Basic hosting security • Scripting • Incident response • Ethical hacking and penetration testing basics • Risk management • Business continuity • More on organizational policies (e.g., impact of regulations and law) • Emerging technologies' impact on cybersecurity • Connecting to hardware through programming languages (e.g., C++, Python) • Database access controls • Model implementation of major networking protocols • Implications and impacts of different network topologies • Cloud computing • Communicating security policies to nonexperts • Network troubleshooting • Emerging technologies (e.g., blockchain) • Industry certifications • Lifelong learning in cybersecurity
Possible careers	• Network Technician • Security Analyst • Network Systems Administrator	• Risk Manager • Security Architect • Cybersecurity Specialist

3.3 Artificial Intelligence Content Progression

Table 3.3 shows the content progression for artificial intelligence. The AI content may require more prior mathematical knowledge than other pathways. This progression might lead to an AI major and to careers as a machine learning engineer, computer vision engineer, or AI ethics and policy analyst, among others.

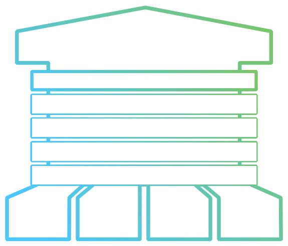
Table 3.3: Artificial Intelligence content progression.

Foundational CS Content	Fundamentals	Specialty
 **Prioritized foundational content specific to AI:** • How algorithms are used • Difference between traditional and AI/ML algorithms, including the role of data in AI/ML • Patterns/commonalities in problems, data, and programs • Evaluate outputs for biases and accuracy • Societal impacts of AI (e.g., biased data, attribution) • Basic data formats and metadata • Cleaning data • Visualizing data • Impact of emerging technologies	• What is AI: history, levels of AI, future careers, laws • Intro to AI programming and intro to prompt engineering • AI projects • Natural interaction, semantics, chatbots • Representation and reasoning, k-nearest neighbors (KNN), vectors • AI programming (projects), using AI tools to solve problems • Ethical frameworks, philosophy, psychology, bias • Sensors, perception, classification • Using datasets, regression, probabilistic thinking • Convolutional neural network (CNN), decision trees, bias • Ethical design and empathy interviews	• Fundamentals of electronics, mechanisms, circuits, gears, sensors • Computer vision, sensor applications, models, perceptions • Robot hardware manipulation (or software simulators) • Using data: collection, cleaning, data types, validity, bias • ML models: optimization, accuracy, decision-making, ethical considerations • Linear algebra, matrices, vectors, probability, statistics • Programming applications with math • Biases in data collection, analysis, and reporting • Preparation for industry certification
Possible careers	• Machine Learning Engineer • Data Scientist • AI Research Scientist • Computer Vision Engineer • Natural Language Processing Engineer	• Robotics Engineer • AI Ethics and Policy Analyst • Autonomous Vehicle Engineer • AI Cybersecurity Engineer

3.4 Physical Computing Content Progression

Table 3.4 shows the content progression for physical computing, which includes robotics. This content may lead to a physical computing or a robotics major and ultimately to careers as a robotics engineer, industrial automation specialist, control systems engineer, or human-robot interaction specialist, among others.

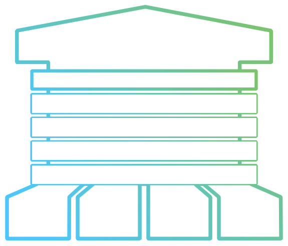
Table 3.4: Physical Computing content progression.

Foundational CS Content	Fundamentals	Specialty
 **Prioritized foundational content specific to physical computing:** • Programming basics • Social and ethical implications • Cybersecurity considerations • Use of documentation • Troubleshooting • Cleaning and using data • How networks work • Optimizing networking and OS settings • Troubleshooting • Using documentation • Network vulnerabilities	• Specifications and limitations of physical communication devices • Genesis of Internet of Things (IoT) from physical computing devices • Use of IoT devices • How to apply the engineering design process to physical computing, including debugging • Use a physical computing device to solve a real-world problem • Use sensors and peripherals appropriately as add-ons to physical computing devices • Communicate and present physical computing solutions so that others can understand the purpose and recreate the project • Security considerations for devices • Understanding and working with circuitry, including power systems, voltage, and batteries • Exposure to careers in physical computing and careers that involve physical computing	• Creating solutions to problems using physical computing • Programming for physical devices • Software development processes (e.g., Agile/Scrum) • Networking for physical devices • Application development (e.g., mobile apps, virtual reality apps) • Team project work • Collaborative source control • Working with motors, microcontrollers
Possible careers	• Robotics or Embedded Systems Engineer • Robotics Research Scientist • Industrial Automation Specialist • Control Systems Engineer • Automation Engineer	• Mechatronics Engineer • Robotics Software Developer • Drone Engineer • Human-Robot Interaction Specialist • Biomechanics Engineer

3.5 Data Science Content Progression

Table 3.5 shows the data science content progression. This content may lead to a data science major and a career as, for example, a data scientist, data engineer, data modeler, statistician, or data ethicist.

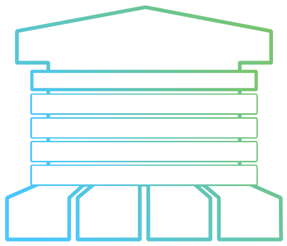
Table 3.5: Data Science content progression.

Foundational CS Content	Fundamentals	Specialty
 **Prioritized foundational content specific to data science:** • Programming basics • Cleaning and using data • Social and ethical implications • Data bias • Testing and debugging • Inclusive collaboration on data projects	• Data science tools • Transform and prepare data • Data validity (clean and accurate) • Statistics (e.g., normal distribution, descriptive statistics, regression analysis) • Data visualization • Extract meaning from tabular data using a function • Query formation (prompt engineering; Structured Query Language (SQL); elastic search) • Make predictions and determine generalizability • Data forms and bias (ethics) • Data fairness and bias (mitigating bias) • Data privacy, security, bias, missing data, ethics • Legal and ethical implications • Structured problem-solving (case studies; case analysis) • IDEs for data science (e.g., PyCharm, RStudio, Azure, Jupyter Labs) • Intersection of data science and other fields • Careers in data science	• Distributed cloud based systems • Data pipelines and transfer • Data modeling • Machine learning basics • Data validity, credibility, and reliability (data consciousness) • Advanced data visualizations • Data from wearables and its implications • Evaluating statistical conclusions (e.g., effect size) • Data privacy and security • Interface development for data analysis (e.g., business intelligence (BI) tools, such as PowerBI, Tableau) • Common algorithms for data science (e.g., linear regression, KNN) • Designing, imagining, and critiquing new ways to get, use, and restrict data
Possible careers	• Data Scientist • Data Security Analyst • Data Privacy Specialist	• Data Ethicist • Data Modeler • Statistician

3.6 Game and Interactive Media Design Content Progression

Table 3.6 shows the game and interactive media design content progression, which shares content with other 2D and 3D digital simulations. We acknowledge that there was limited consensus related to naming this pathway, and it could be named in many ways (e.g., Game Design and Development, Digital Innovation and Design).

Table 3.6: Game and Interactive Media Design content progression.

Foundational CS Content	Fundamentals	Specialty
 **Prioritized foundational content specific to game and interactive media design:** • Programming basics • Testing and debugging • Inclusive collaboration on projects • Social and ethical impacts • Cybersecurity basics	• Game design • Game, two-dimensional (2D), and three-dimensional (3D) Art • Game Sound • Interactive Design • User Interface • Psychology of Games • Storyboarding • Ethics • A/B testing • Graphical Processing Units (GPUs) • Interaction of physical devices with a program/game (e.g., joysticks, VR headsets) • Accessibility in game design • Inclusivity (broad cultural, religious, gender, physical, cognitive differences) • Social impact (games have power to influence culture, cultural values, and norms) • Physical modeling • Programming (e.g., interaction, navigation, world building) • Human Behavior/Safety in game environments and simulations • Debugging • Game/simulation pathways and careers	• Character and environment design • Art history and direction • 2D and 3D animation • Motion graphics • Simulations • Sound/music history • Encoding analog info (e.g., character state, mood) • Augmented Reality (AR)/Virtual Reality (VR)/Extended Reality (XR) • AI in game design • Object-oriented programming • Physics and states • Controller design • Integrating art and animation • Integrating sound/music • Encoding analog info • Source Control • Team Collaboration • Game development engines
Possible careers	• Game Designer • Game Developer • Graphic Designer • Concept Artist • Producer, Writer	• Level Designer • Game Tester • Sound Engineer • Simulation Engineer

3.7 X + CS Content Progression

Table 3.7 shows the content progression for X + CS. X can represent any subject area, including humanities. X + CS requires integration between the two or more subject areas. This content may lead to a major in CS, the "X" subject, or X + CS, followed by a wide variety of careers, including biomedical engineer, educational technologist, digital media specialist, or medical simulation specialist.

Table 3.7: X + CS content progression.

Foundational CS Content	Fundamentals	Specialty
Prioritized foundational content specific to X + CS: • Programming basics • Testing and debugging • Inclusive collaboration on projects • Social and ethical impacts • Cybersecurity basics	• Common themes, practices, and terminology between X and CS • Historical examples of X and CS, considering universal human endeavors as a bridge and identifying gaps and challenges • Data visualizations and computational models in X • Reframing problems in X using CS and in CS using X: decompose problem, translate into program, determine whether the program solves the problem • Exploration of multiple perspectives in X using programming skills	• Impact on CS of the evolution of X, and vice versa • Transforming data models to allow for utilization of source data from X • Evaluating and comparing algorithms that address problems in X • Contributing to the evolution of X in CS by creating an artifact • Developing a plan that uses algorithms in programming to address problems in X (student is selecting)
Possible careers	• Medical Simulation • Specialist • Biomedical Engineer • Business Data Analyst • Computing Ethicist • Neuroscientist	• Education Technologist • Digital Media Specialist • Digital Linguist • Human Language Technologist • Project Manager • CS Teacher

4 Moving Toward Implementation

4 Moving Toward Implementation

Beyond determining what knowledge, skills, and dispositions will be taught and supported in CS education, there are many complexities to consider when bringing educational opportunities to fruition. School size, available resources, and potential community partnerships are a few examples of factors that necessarily inform implementation strategies. This section offers a starting point for considerations related to structure, organization, and equitable implementation of foundational CS content, as well as opportunities for continued learning. See Section 7 for additional equity-related considerations.

4.1 Teaching the Foundation and Beyond

Schools could teach the foundational high school CS content and pathways for continued learning in several (potentially complementary) ways. The following are listed in no particular order:

Offer a discrete course(s). Many schools may opt to offer a discrete course for students to access foundational CS content as well as opportunities for continued learning beyond the foundation. Common and traditional infrastructure exists for this strategy, and as such, this may be viewed as the most straightforward means for making CS content accessible to students.

- A discrete foundational course can focus solely on CS content and satisfy the school's CS graduation requirement, if one exists.
    - If there is computing offered in middle school and/or if some foundational content is supplemented in other high school courses and/or experiences, then this might be a semester-long course.
    - An extended version of this course would allow for exposure to different pathways so that students can make informed decisions on subsequent learning experiences.
- A series of discrete courses can create a pathway.

Integrate foundational content into other subject areas. Integration of CS content into other disciplines can be an authentic and engaging approach to the provision of CS learning experiences. This approach may also require the most planning and coordination across educators sharing the responsibility of integration.

- Schools may distribute foundational CS content across other classes, based on strategic alignment. There are opportunities for interesting collaborations, such as with data analysis integrated into social studies or ethnic studies and programming topics integrated into art courses.
- Comprehensive integration may be challenging: there is a potential need for pairing integration with a discrete course, allowing integration to focus on authentic application.
- Supplement classroom learning with informal learning opportunities, like out-of-school time.
- Creating integrated pathways may require intensive integration planning, including an analysis of what content beyond the foundation does not require prerequisite knowledge to identify potential points for integration.

Offer content as part of (or in tandem with) one or more existing programs. It may be advantageous for schools to build on existing infrastructure in support of CS programs. This can reduce barriers to entry and infuse some level of familiarity into CS implementation. Existing programs may include:

- Advanced Placement (AP)
- Career and Technical Education (CTE)
- International Baccalaureate (IB)
- Partnership with an institution of higher education for early college credit (i.e., dual enrollment, dual credit)
- Relevant out-of-school programming

Provide flexible options. Where possible, accommodating the various needs and scheduling constraints of students, teachers, and schools will maximize access and likely increase the number of students participating in CS.

- Allow students to take the foundational high school CS course in middle school, while satisfying a CS graduation requirement, if one exists.
- Provide access to a virtual or online course.
- Create work-based, service-based, and/or project-based learning integration.
- Teach specific content progressions on a rotating basis to maximize teaching capacity.

4.2 Pathway Endpoints

High school pathways may lead to postsecondary studies and eventual careers related to the specialty area. However, there are many potential endpoints for a pathway, including:

- Certifications
- Internships
- Apprenticeships
- Student-directed capstone courses
- Certificate or specialty at a two-year institution
- Minor or major at a four-year institution
- CS bootcamp
- Direct entry career
- Enlistment leading to service

Given these varied endpoints, CS pathways need to support career exploration and industry awareness. This highlights the importance of the Preparation for the Future Topic Area in the foundation, as well as the intentional inclusion and support of Pillars and Dispositions in CS instruction.

4.3 Implementation Planning

Before implementing a new CS pathway, it is important to define current CS offerings at a school or district and then to identify areas for development based on relevance to the community, available resources, and desired outcomes for students.

Ramping up a robust CS program takes time, and it may require establishing a multiyear plan that involves assessing teacher interest, evaluating possible professional development opportunities, training teachers (including non-CS teachers), and recruiting additional teachers.

Due to resource constraints, very few schools will be able to offer the full range of content contained within the example CS pathways from this project. However, combining several ways to offer content can extend beyond the classroom; for example, small schools may offer out-of-school activities like robotics club or e-textiles. They may also establish partnerships within the community to offer summer enrichment camps and programs, apprenticeship programs, and other resources to support students in their learning. The process of determining implementation pathways from the content progressions described in Section 3 is not simple, as a wide variety of factors – from teacher capacity to student interest to local needs – must be considered.

Figure 4.3 shows an example of how a school may design and implement a set of relevant CS pathways over the course of five years.

Figure 4.3: Example five-year implementation plan for a high school implementing new CS pathways.

Year 1	Year 2	Year 3	Year 4	Year 5
Survey teachers to learn about their interest and past experience with CS	Offer foundational CS course	Offer courses for one or more pathways	Offer at least one full pathway	Offer one or more full pathways
Choose pathways based on district/ community needs	Provide training for non-CS teachers and CS teachers to continue to grow course offerings	→	→	→
Research sources for quality professional development	Make decisions about curriculum	Support interdisciplinary collaboration around CS integration	Authentically integrate CS into non-CS subjects	→
Plan training to prepare one or more teachers to teach foundational courses	Recruit additional teachers to attain required credentials	→	Offer a capstone or dual credit option	Ensure appropriate credentialing of CS teachers
Develop and implement student recruitment strategies	→	→	→	→

Historically, the implementation of CS pathways has often introduced barriers to participation for some students, particularly those from minoritized backgrounds. Thus, ensuring that pathways provide equitable access is a key concern. Please see Section 7.2, which discusses equity considerations for CS pathways.

5 Example Courses and Pathways

5 Example Courses and Pathways

Whereas Section 3 describes content progressions, this section details how the content from each specialty area could be packaged into course pathways, provides a holistic view of relationships across pathways, and describes each example course identified in the pathways. This collection of courses is designed to showcase the breadth of opportunities for learning beyond foundational high school CS content.

5.1 Example Courses at a Glance

Figure 5.1 illustrates a sample set of course implementation pathways. Actual pathways can and should differ widely based on local needs and resources. However, this sample is meant to suggest what a relatively full implementation might look like for a large school positioned to deliver a comprehensive set of CS pathways. It is not expected that schools would necessarily have the capability of offering all (or even multiple) of the pathways represented. Note that the content of the courses shown in the diagram and described below are intended to align with content from the content progressions in Section 3.

In the model shown in Figure 5.1, students experience the foundational CS content through taking a course called *Computer Science Foundations* (or an equivalent CS experience may be substituted).

Fundamental content from different specialty areas is packaged both discretely (e.g., *Physical Computing* and *Game Design & Digital Innovation*) or in a combined fashion (e.g., the *Programming the Future* course includes fundamentals from CS, AI, data science, and cybersecurity). Since students may take courses in different grades, the columns represent levels of experience, rather than specific grade levels; some students may only complete a foundational learning experience, whereas others may complete two, three, or even four experiences.

There are many opportunities for schools and students to create a pathway that makes the most sense for their unique interests, experiences, and resources. For example, after taking *Game Design & Digital Innovation*, a student may choose to take *Game Development or Application Development*, which flow most seamlessly from a content perspective. But a student may also decide that they prefer a breadth of experience and take another fundamentals course such as *Programming the Future*. Similarly, a student may have participated in an after-school robotics activity the summer after taking *Game Design & Digital Innovation* and determined that they are adequately prepared to take *Robotic Systems* as a follow-on course. These examples demonstrate the intended versatility and flexibility of the sample pathways presented in this report.

Most schools will not be able to offer many options for specialized CS learning, so they may select a relevant subset of these pathways or substitute other areas of specialty. We also note that it may be possible to offer any of these courses for dual credit if an appropriate agreement with an institution of higher education can be forged.

Figure 5.1: Example implementation pathways. Descriptions of these courses can be found in Section 5.3.

Foundation	Fundamentals	Specialty	Advanced Application
Computer Science Foundations	Ethics of Computing	AI and ML Programming	Apprenticeship
	Game Design & Digital Innovation	Application Development	Internship
	Physical Computing	Data Science and Analytics	Capstone
	Programming the Future	Game Development	
	Computational Art	Information and Network Security	
	Computational Biology	Robotic Systems	
	Computational Journalism	Software Development	
	Digital Humanities		

● Computer Science courses

● Integrated courses

5.2 Example Course Pathways by Focus Area

Section 4 delineates high-level content progressions to continue learning beyond the foundational content. In this section, we present examples of how to package this content into meaningful course sequences across particular specialty areas. The boxes in each diagram represent discrete courses. The content of those courses aligns with columns two and three of the content progression tables in Section 3 and descriptions for each course can be found in Section 5.3.

Figure 5.2.1: Example pathways aligned with the content progressions found in Section 3.

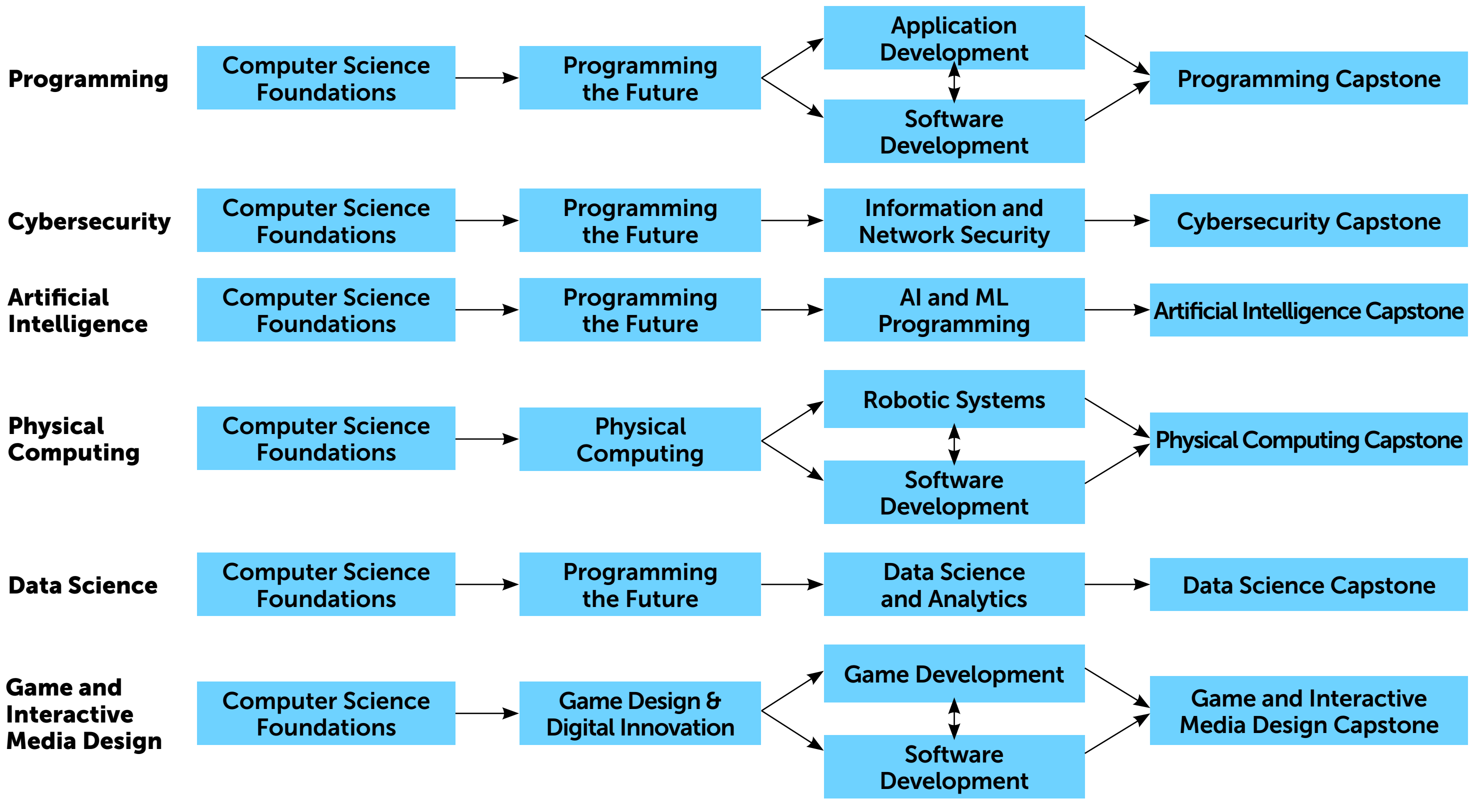
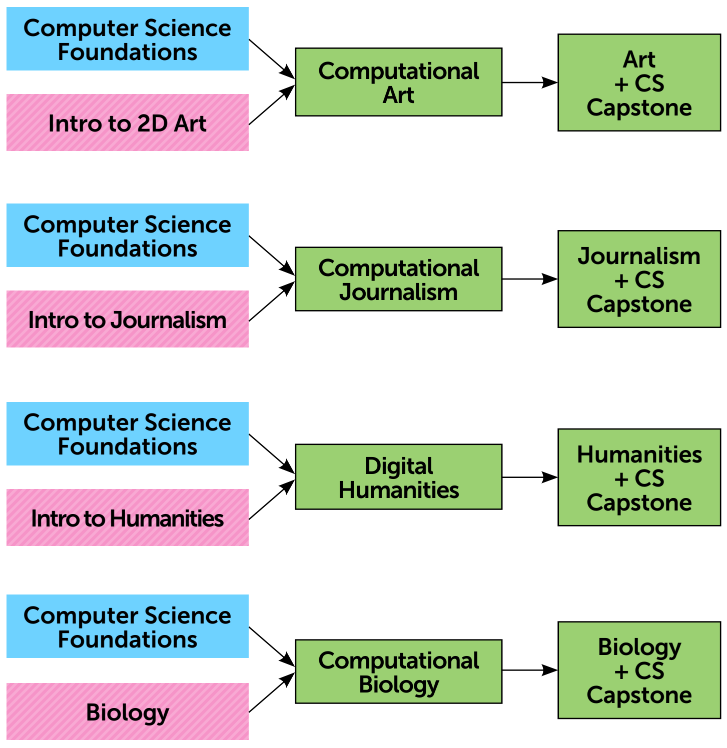

Programming
Computer Science Foundations
Programming the Future
Application Development
Software Development
Programming Capstone
Cybersecurity
Computer Science Foundations
Programming the Future
Information and Network Security
Cybersecurity Capstone
Artificial Intelligence
Computer Science Foundations
Programming the Future
AI and ML Programming
Artificial Intelligence Capstone
Physical Computing
Computer Science Foundations
Physical Computing
Robotic Systems
Software Development
Physical Computing Capstone
Data Science
Computer Science Foundations
Programming the Future
Data Science and Analytics
Data Science Capstone
Game and Interactive Media Design
Computer Science Foundations
Game Design & Digital Innovation
Game Development
Software Development
Game and Interactive Media Design Capstone

Figure 5.2.2: Example integrated pathways aligned with the X + CS content progression.



● Computer Science courses ● Integrated courses ● Non-CS courses

5.3 Example Courses and Descriptions

This section contains example courses and descriptions, with the assumption that individual schools and districts may modify the offerings to meet local contexts and needs. Regardless, we recommend that the Pillars and Dispositions articulated in the foundational content continue to be woven throughout these courses. Content to be covered within these courses aligns with fundamentals or specialty content from the content progressions in Section 3.

Computer Science Foundations

Computer Science Foundations supports all high school students, regardless of postsecondary goals, in developing the knowledge, skills, and dispositions necessary to navigate and understand the technology-driven world in which they live. Course content, organized into five Topic Areas (Algorithms, Programming, Data and Analysis, Computing Systems and Security, and Preparing for the Future), rests upon four Key Pillars (Computational Thinking, Inclusive

Collaboration, Human-Centered Design, and Impacts and Ethics). Topic Areas and Pillars are essential components of this course and the student experience (see Section 2 of this report for more details).

Ethics of Computing

Previously, technology has been considered an inherently neutral tool that has benefits and drawbacks that can be leveraged for better or for worse by the user or creator. Yet, many scholars – including Ruha Benjamin (2019), Safiya Noble (2018), and Joy Buolamwini (2017) – have cataloged the ways in which computing technologies have embedded and extended biases. In *Ethics of Computing*, students explore the implications, and potential harm, for users and nonusers. Further, students consider how this knowledge translates into being a critical consumer and responsible creator of technology, weighing pros and cons and recognizing intended and unintended consequences. Note that offering this course is not a replacement for including ethics and impacts throughout all CS courses/ instruction. *Ethics of Computing* simply provides an opportunity for deep, sustained, and focused learning specifically around ethics.

Other titles may help capture student interest as well as reflect specific focal points and/or relevant current events. For example:

“Game Design and Digital Innovation”	→	**“Digital Storytelling”**
“Physical Computing”	→	**“Digital Fashion”**
“Information and Network Security”	→	**“Hacking for Good”**
“Ethics of Computing”	→	**“Should We Ban TikTok?”**

Programming the Future

Programming the Future provides students who have a foundational understanding of computer science with an opportunity to explore various topics such as cybersecurity, artificial intelligence, and data science. While developing their programming skills, students will apply fundamental ideas in these areas to solve meaningful and interesting problems. Content covered in this course aligns with fundamentals content from the Programming, Cybersecurity, Artificial Intelligence, and Data Science content progressions as defined in Sections 3.1, 3.2, 3.3, and 3.5.

Game Design and Digital Innovation

Game Design and Digital Innovation is an ideal course for students who have a foundational understanding of computer science and a particular interest in applying that knowledge within the context of developing games or other 2D and 3D media, such as simulations. Students will learn aspects of the design process and leverage them in one or more projects of interest. Content covered in this course aligns with fundamentals content from the Game and Interactive Media Design content progression as defined in Section 3.6.

Game Development

Game Development is ideal for students who have already taken *Game Design and Digital Innovation* and have an interest in bringing their designs to life. Students will engage in advanced study of programming and may apply those skills to create games and simulations in traditional, augmented reality (AR), virtual reality (VR), and/or extended reality (XR) environments. This course is intended to involve extensive collaboration through an intentional development process. Content covered in this course aligns with specialty content from the Game and Interactive Media Design content progression as defined in Section 3.6.

Physical Computing

Physical Computing is a course for students who have a foundational understanding of computer science and want to learn more about applying CS ideas to robots, sensors, and IoT devices. Students will use the engineering design process to address an individual/community need to solve an authentic problem. Content covered in this course aligns with fundamentals content from the Physical Computing content progression as defined in Section 3.4.

Robotic Systems

Robotic Systems is designed to be a follow-on course to *Physical Computing*. Students build upon existing knowledge of physical devices such as robots, sensors, and IoT devices in an effort to solve meaningful problems through thoughtful design and implementation processes. Content covered in this course aligns with specialty content from the Physical Computing content progression as defined in Section 3.4.

Information and Network Security

Innovations in artificial intelligence and quantum computing underscore the importance of securing information, programs, and applications for both personal and societal safety. *Information and Network Security* is intended to follow foundational programming and introductory cybersecurity learning experiences and prepare students for advanced study or workplace application of cybersecurity principles. The course involves learning about and applying security practices in authentic environments and contexts where possible. Content covered in this course aligns with specialty content from the Cybersecurity content progression as defined in Section 3.2.

AI and ML Programming

AI and ML Programming is intended to follow foundational programming and introductory AI learning experiences. Students will build upon this prerequisite knowledge to leverage AI in practical and innovative applications as well as to interrogate when opportunities to use AI may be unsafe or unreliable. This course includes a significant emphasis on data and needs to be paired with appropriate math learning. Content covered in this course aligns with specialty content from the Artificial Intelligence content progression as defined in Section 3.3.

Data Science and Analytics

In a world that is increasingly informed and driven by data, it is necessary to understand data, where it comes from, how it is leveraged, and how it can impact life and work. *Data Science and Analytics* is a first in-depth course for students to investigate the various ways that data can be stored, accessed, modified, and visualized. Students will consider impacts and ethical considerations related to ownership and bias in data as well as how data visualizations can be misleading. While this course focuses on the computer science context, data science is increasingly interdisciplinary, and students will be afforded opportunities to apply analysis and visualization techniques in fields/topics of personal interest. Content covered in this course aligns with specialty content from the Data Science content progression as defined in Section 3.5.

Application Development

Application Development involves advanced study related to programming with a focus on developing mobile and desktop applications. Students will engage in collaborative development processes to solve a problem or address a personal or community need. In addition to development, students will test and refine their products to ensure usability and quality user experience. Ethical issues will also be considered. Content covered in this course aligns with specialty content from the Programming content progression as defined in Section 3.1.

Software Development

Software Development provides opportunities for extensive study in one or more programming languages, ideally that students have not experienced in previous coursework. Students learn about uses and advantages of particular programming languages and understand commonalities and differences across them. Students will engage in collaborative development processes to solve a problem or address a personal or community need using their programming skills. This course aligns with common first-year postsecondary programming courses (i.e., CS1, including AP CSA). Content covered in this course aligns with specialty content from the Programming content progression as defined in Section 3.1.

Computational Art

This course builds upon the student's previous experiences in computing and in art in order to provide opportunities for students to exercise their creativity and develop their portfolio in art by leveraging computing technologies. Content covered in this course may include:

- AI art generation, including prompt engineering
- Pixel-based art

- Ethical issues related to digital art
- Creative app development
- E-textiles
- Artistic applications of physical computing

Computational Journalism

Designed for students who have completed a foundational computing course as well as a foundational journalism course, this class exposes students to computational techniques and issues related to journalism, including:

- Ethical issues, including data privacy and security
- Language processing and text analysis
- Reporting on technology and the technology industry
- Computing-based investigative techniques
- Data journalism, including data visualization

Digital Humanities

This course, designed for students who have completed a foundational computing course as well as an introduction to the humanities, explores various techniques of digital humanities. Topics considered within the humanities are vast and thus, this course can be offered thematically. Students might, for example, develop digitized topographical maps to better understand historical battles (history lens), migration patterns (sociology lens), or artistic works (fine arts lens). Content covered in this course may include:

- Text mining and analysis, including via natural language processing
- Social network analysis
- Working with digital archives
- Digital mapping
- Audio, image, and video analysis
- New media studies, including software studies
- Ethical issues in the digital humanities

Computational Biology

This course builds on students' previous experiences in an introduction to biology and a foundational computing course in order to develop knowledge and skills related to computational biology, including:

- Genetics and evolution
- Personalized medicine
- Digital pathology
- Data visualization
- Systems and networks in biology
- Algorithms in nature
- Ethical issues in computational biology

A Note on Integrated Courses

Integrated courses necessitate a very collaborative and intentional planning process to ensure proper footing and representation of the disciplines at play. Collaborative planning is critical for instruction in integrated pathways (e.g., Computational Art, Computational Biology), as well as in more computing-intensive courses that heavily rely on other disciplinary content (e.g., data science draws heavily from math). Additionally, curriculum planners and key educator support roles are crucial in the thoughtful design and successful implementation of such courses in a manner that is accessible to all students who wish to take them. These educator support roles include instructional coaches, multilingual learner teachers or English language development specialists, and special education teachers.

In a traditional high school department structure, it may also be unclear through which department courses such as these might be offered. We recommend careful consideration to situating and presenting these courses in such a way that the intent is clear, all related disciplines are appropriately honored, and students understand how the course might align with their interests.

6 Integrating CS into Other Subject Areas

6 Integrating CS into Other Subject Areas

There are some substantial challenges to integration, including gaining support from teachers in other subject areas, who may have little interest in adding CS content to already-full curricula. It is also important to note that this project did not feature participants from other disciplines (e.g., biology or even computational biology), which left many perspectives unrepresented. Further, integration is highly under-researched, making it difficult to comment on even promising practices for this approach.

However, integrative approaches may also have some advantages over a stand-alone CS course, including that the use case of computing is more obvious when it is applied to another subject area, and a task may naturally be more authentic when embedded into another subject (Ko et al., 2024). Further, integrating CS can improve learning achievement in the discipline into which CS is integrated (Century et al., 2020) and can increase student engagement (Strickland et al., 2021). Integrating CS into other subject areas creates opportunities for projects that are more meaningful for students (Tissenbaum & Ottenbreit-Leftwich, 2020). It may also be logistically simpler for some schools to integrate CS content into other courses than to offer a stand-alone CS course, perhaps through small exercises that use programming to meet learning objectives in the other course (Guzdial, 2022).

Table 6: Examples of integrating CS into other subject areas.

Subject	Example of Integration		
Language Arts	**ELA concepts:** close reading for meaning and tone	**CS concepts:** types of data, data cleaning, data analysis and visualizationt	**Activity:** Using a text file of *Romeo and Juliet*, students record counts for each character's dialogue and then visualize that data. Using the visualization, students look for patterns in the data and then use the patterns to confirm what is known about the play and to generate new questions about the text. Students also assess word frequency per scene to look for patterns in the text. **Source:** Integrated Computational Thinking
Math	**Math concepts:** ratios, coordinates, scaling	**CS concepts:** functions, decomposition, image manipulation, comments	**Activity:** Students digitally replicate flags using a combination of math and programming skills. First, students sketch the image on graph paper. Then, they experiment with predefined functions to decompose elements of national flags and then compose additional flags. **Source:** Bootstrap
Science	**Science concepts:** ecosystems, evolution, patterns and systems, using models	**CS concepts:** cleaning, analyzing, and visualizing data	**Activity:** Students develop and experiment with computational models to explore the behavior of a forest fire and its impact on the forest ecosystem. **Source:** CT-STEM
Social Studies	**Social studies concepts:** population growth patterns, data literacy	**CS concepts:** function parameters, data visualization	**Activity:** Students explore patterns in population change across countries and time spans. They create multiple data visualizations by using a specialized tool to adjust parameters to generate the appropriate visualization, which can then be analyzed. **Source:** Data Visualization for Learning tool
Fine Arts	**Music concepts:** elements of a song (tempo, measures, sections)	**CS concepts:** functions, parameters	**Activity:** Students create a song by using predefined functions with the appropriate parameters, as they practice using music concepts and terminology. **Source:** EarSketch

Table 6 showcases some examples of lessons that integrate CS concepts into other disciplinary content. Because integrating CS into other subject areas is a relatively new approach, resources – including research on best practices, implementation guidance, and curricular materials – should all be expanded to best support student success. However, recent work by Weisberg et al. (2024) presents an overview of research literature on integrating CS into the arts. Their work highlights ways in which integrating CS and the arts can better promote equitable CS programs by leveraging student interest in creative self-expression and providing multiple entry points into the study of computing. They identified research on arts and CS integration that involved the visual arts, music, dance, and dramatic arts, with activities ranging from e-textiles to music composition to "robot theater." Activities resulted in increasing CS skills and positive shifts in attitude toward CS.

6.1 Integrating the Foundation

There are many options for integrating the foundational CS content into other subject areas. In this section, we offer some general observations about when and how CS might integrate into other subject areas (see also Integrated Computational Thinking, n.d.).

Foundational content such as generating a personal career plan may fit well within a language arts course, where students are already engaged in exploring issues of identity, future plans, and similar issues. More specifically, the development of a personal career plan might mesh well with a creative writing assignment that encourages the student to envision their future self or as part of an autobiographical writing assignment. Similarly, decomposing a problem into multiple subproblems is an exercise that aligns well with critical thinking, logical reasoning, and expository writing exercises.

A math course is a logical fit for many CS topics. For example, students study the order of mathematical operations, which has substantial overlap with explaining why/how sequence matters in an algorithm. Similarly, evaluating data visualizations for clarity and potential biases leverages numeracy skills as well as computing skills. A model assignment might integrate math skills related to interpreting charts and graphs alongside CS skills for generating and assessing them.

Several foundational CS skills could fit well in a science course. For example, students might collect data describing a natural phenomenon and then use programming skills to prepare to analyze that data. Similarly, modeling a system (such as a local waterway) could involve both science content and CS skills.

Social studies courses may be a good fit for some CS content, particularly the ethical issues related to computing. For example, considering societal impacts of social media networks – from their possible impact on the well-being of teenagers to their influence on political discourse – could productively occur in a social studies course.

Because integrating CS into another subject area requires subject matter expertise in both domains, it can be difficult to envision how exactly CS could be productively integrated into another subject. Table 6.1.1 provides a sample of how such integration might be accomplished: it takes one learning outcome from the Algorithms Topic Area, "AL.12 - Compose algorithms using sequence, selection, and iteration," and shows how it might be taught in lessons in various other disciplines.

Table 6.1.1: Samples of CS integration into the subject areas for the item "AL.12 - Compose algorithms using sequence, selection, and iteration."

Subject	Integration Ideas for "AL.12 - Compose algorithms using sequence, selection, and iteration"
Language Arts	• Write a paragraph outlining how you decide what to wear each day. Use each of these words at least once: *then, if, repeat*. • Write a persuasive essay describing what you think the consequences should be for online bullying. Use each of these words at least once: *next, while, again*. • Create a flowchart of the plot for a short story where the reader makes choices about what event will happen next.
Math	• Write out all of the steps to find the volume of a cone. • Describe the steps to determining whether it is better to lease or purchase a car, using information from a local car dealer's website. • You have learned three different methods for solving quadratic equations. Create a flowchart describing how you would decide which method to use.
Science	• List the steps for how each of the three major types of rocks are formed. • Write a list of procedures that describe the life cycle of recyclable materials in your community. • Create a flowchart that presents all possible outcomes in an offspring for a characteristic that is determined by two different genes.
Social Studies	• Create a flowchart that showcases the process of a bill becoming – or not becoming – a law. • Write a paragraph that describes a historical counterfactual. Include at least three if-then statements in your description. • Graphically depict a cost-benefit analysis for a topic of your choice related to a decision that a small business may encounter.
Fine Arts	• Write pseudocode for the creation of pixel art. • Create a dance routine; list the steps. • Create a flowchart that includes at least six principles for floral design.

In addition to outlining examples of how content items from the Topic Areas might be integrated into other subject areas, we also offer some more general principles, concerns, and recommendations for integrating CS content. The major challenge to integration is that it requires the support of teachers from other subject areas, who will require professional learning and perhaps persuasion in order to be able to successfully implement CS into their classrooms. They may feel that they do not have the time or energy to do this work (Lee et al., 2022). Effective strategies to address this sentiment are described in Table 6.1.2.

Table 6.1.2: Approaches for garnering support for integration.

Role	Approaches
Curriculum designers and teachers	• Encourage innovative pedagogies and activities, such as task-specific programming languages (Guzdial & Naimipour, 2019). • Introduce promising practices for computing education pedagogy, such as pair programming (Bishop-Clark et al., 2006). • Infuse "across the curriculum" approaches. • Identify content to be experienced in younger grades as students continue to be exposed to CS at earlier ages.
Administrators	• Frame CS integration as part of the effort to prepare students for careers of the future. • Offer substantive professional development that is targeted to the subject area in which CS will be integrated. • Showcase easy-to-implement lessons that cover key concepts in the subject domain as well as key CS concepts, such as developing a text-based story or game that meets ELA and social studies standards related to Native American history. • Frame CS as more than just programming. For example, emphasize that computational thinking is a framework for activities that are likely already being taught. This can provide students with opportunities to see connections across disciplines.
For professional development providers	• Address concerns about teacher self-efficacy, which may be aided by analog or unplugged activities, co-teaching, an instructional coach model, and/or games and other easier-to-implement approaches. • Adopt approaches and insights from adult learning theory, such as focusing on the benefits of CS integration to teachers themselves; for example, frame professional learning around the persona of a social studies teacher who wants to teach students enough about algorithms for students to understand the role that social media plays in political polarization. • Focus on aspects and framings of CS that are more approachable, such as design thinking and computational thinking.

6.2 Integrating CS Content Beyond the Foundation

When all students develop a consistent foundation in CS (whether through a stand-alone course, integration into other subject areas, or both), teachers are able to leverage and extend student CS knowledge and skills beyond the foundation and into other discrete CS courses. Another option is to integrate advanced CS content in other subject areas. This may be achieved through an approach similar to the integrated (X + CS) courses and pathways detailed in Sections 5.2 and 5.3.

Another approach is outlined in Table 6.2, which shows how fundamental and specialized AI content (see Section 3.3: AI Content Progression) can be integrated into other courses that students take after their foundational CS learning experience. This example can be extrapolated and applied to other specialty areas such as data science and physical computing.

Table 6.2: Example integration of fundamental and specialized AI content into other subject areas.

Subject Area	Example Integration of AI Content	Content Alignment Example
Social Studies (including Civics and Ethnic Studies)	• What is AI?: history, levels of AI, future careers, laws • Ethical frameworks, philosophy, psychology, bias • Ethical design and empathy interviews • Biases in data collection, analysis, and reporting • Using AI tools to solve problems	Exploration of AI ethics aligns with this item in the New York Learning Standards for Social Studies: "Prepare a plan of action that defines an issue or problem, suggests alternative solutions or courses of action, evaluates the consequences for each alternative solution or course of action, prioritizes the solutions based on established criteria, and proposes an action plan to address the issue or to resolve the problem." **Sample activity:** Students develop a plan of action related to the environmental costs of developing LLMs.
Mathematics	• Representation and reasoning, KNN, vectors • Using datasets, regression, probabilistic thinking • Using AI tools to solve problems • Linear algebra, matrices, vectors, probability, statistics • Programming applications with math	Using datasets aligns with this item in the Texas Mathematics Essential Knowledge and Skills: "Students will extend their knowledge of data analysis and numeric and algebraic methods." **Sample activity:** Students analyze the output of unsupervised learning models that categorize data.
Language Arts	• Natural interaction, semantics, chatbots • Intro to prompt engineering • Using AI tools to solve problems	An introduction to prompt engineering aligns with this item from the Illinois English Language Arts Learning Standards: "Produce clear and coherent writing in which the development, organization, and style are appropriate to task, purpose, and audience." **Sample activity:** Students write prompts using techniques such as few-shot prompting.
Science	• Sensors, perception, and classification • Using AI tools to solve problems • Using data: collection, cleaning, data types, validity, bias • Fundamentals of electronics, mechanisms, circuits, gears, sensors • Robot hardware manipulation (or software simulators)	Using sensors (and resultant data) and using AI tools to solve problems aligns with this item from the Mississippi Career-Readiness Standards for Science: "Students will use mathematical and computational analysis to evaluate problems." **Sample activity:** Students use sensors to gather data about a chemical process and then analyze it using an AI library.
Computing	• Intro to AI programming • Convolutional neural networks (CNN), decision trees, bias • AI programming project • Computer vision, sensor applications, models, perceptions • ML models: optimization, accuracy, decision-making, ethical considerations • Preparation for industry certification	Some content may be most appropriate or feasibly implemented in a discrete computing course. **Sample activity:** Create a program that uses a decision tree to decide which students will be granted a scholarship, using a fictitious dataset. Then, use the decision tree to assess the fairness of the results.
Fine Arts	• Biases in data collection, analysis, and reporting • AI programming (project)	Exploring biases in data collection aligns with this item from the Nevada Visual Arts Standards: "Demonstrate awareness of ethical implications of making and distributing creative work. **Sample activity:** Students explore similarities and differences between how AI models and artists make use of others' intellectual property, as well as the ethical and legal ramifications of such use.

7 Centering Equity in High School CS Education

7 Centering Equity in High School CS Education

Throughout its history, CS and CS education have not been representative of the broader population. This includes students with disabilities, who constitute over 15% of the student population (National Center for Education Statistics, 2023), particularly since people with disabilities are underrepresented in computing (Burgstahler & Ladner, 2007). Students from minoritized racial and ethnic backgrounds often have more interest in but less access to computing than other students (Wang & Hejazi Moghadam, 2017).

Similarly, students who identify as women or girls experience discouraging stereotypes about who belongs in computing (Master et al., 2021; Sax et al., 2018). Lack of access is also an issue for students from rural areas (Google & Gallup, 2017), and students who are multilingual learners often face various challenges using tools for learning programming (Vogel et al., 2021). Significantly, students who lack previous experience in computing often struggle to succeed in classrooms geared toward their better-prepared peers (Margolis et al., 2008).

In contrast, as described in Section 1.2, CSTA's vision is for all students to be supported in learning CS, including those from groups that have historically been marginalized in computing. Additionally, a key value of the *Reimagining CS Pathways* project is that it strives to be equity-centered, with the goal of promoting broad and equitable access, participation, and experiences in CS education among all high school students (see Section 1.3.1).

For purposes of this project, we define equity following Madkins et al. (2020, p. 3):

"Working towards equity means supporting minoritized students in:

1. **engaging in meaningful and rigorous instruction;**
2. **grappling with and challenging systemic racism, power, and oppression; and**
3. **using STEM and CS to empower themselves and their communities.**

As such, equity is defined as intentionally facilitating justice-oriented learning experiences for minoritized students. This requires viewing teaching and learning as *inseparable* from pursuing justice while attending to students' access to rigorous instruction and equitable outcomes."

As a result, we centered equity throughout every part of the articulation of foundational content and resulting pathways. For example, the inclusion of a sense of belonging as a Disposition reflects the importance of its cultivation for persistence in computing – and recognition that students from minoritized backgrounds often face barriers in developing that sense (see Section 2.3). Similarly, the emphasis on Inclusive Collaboration as a Pillar (see Sections 2.1 and 2.4.2) reflects the importance of creating a computing culture that fosters equity.

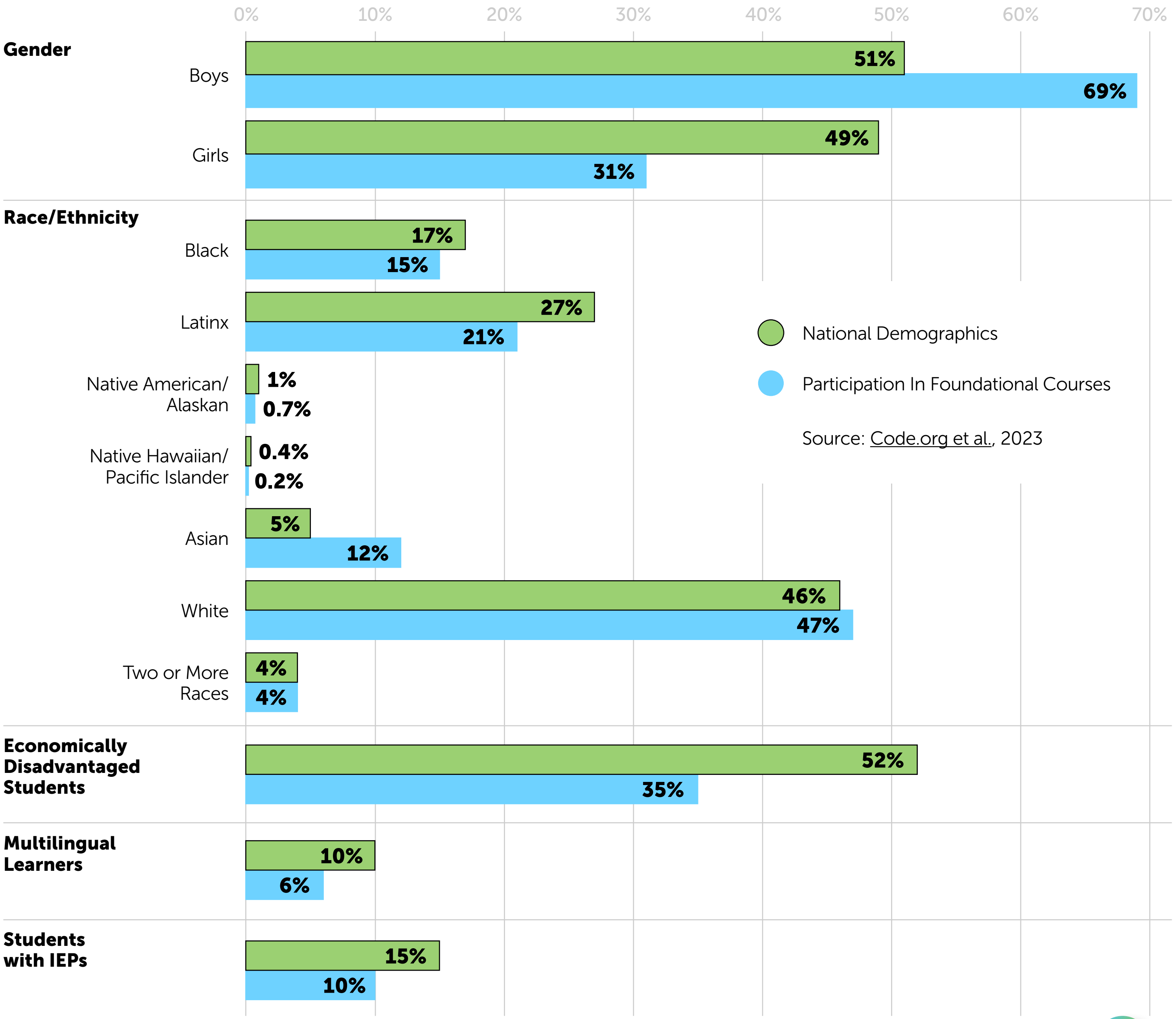
We centered equity in the process used by this project; we strove to create a participant group that reflected diversity across several dimensions: demographic, expertise, role, and geography. Another key factor in participant selection was experience in supporting students with diverse identities and backgrounds (e.g., girls and nonbinary students, students who identify as LGBTQ+, economically disadvantaged students, students from various racial/ethnic backgrounds, students with disabilities, students experiencing homelessness, multilingual learners, migrant students, and students living in rural communities). See Appendix D for more information on project participants.

In the rest of this chapter, we articulate several considerations for decision-makers related to reimagining high school CS education in ways that are more equitable.

Persistent Disparities

The *State of CS Report from Code.org*, CSTA, and ECEP Alliance provides an annual update on national and state-level CS education policy and implementation trends. The 2023 report illuminates persistent disparities in foundational high school CS course participation for many student groups, including girls; Black, Latinx, Native American/Alaskan, and Native Hawaiian/Pacific Islander students; economically disadvantaged students; English language learners; and students with disabilities who have Individualized Education Plans, or IEPs (Code.org et al., 2023).

Figure 7.1: Participation in foundational high school CS Courses by subgroup.



7.1 Equity Considerations for the Foundational CS Content

In planning how to implement the foundational CS content, educators and leaders can ensure all students' needs are met by considering the following equity-related issues:

More than workforce preparation. It has often been the case that high school CS standards, curricula, pathways, and programs focused, at least implicitly, on preparing students to study computing in college and then to work in the tech or related industry.

In contrast, the foundational content is focused on the experiences of all high school students – only a tiny fraction of whom will specialize (e.g., major) in computing as part of postsecondary education. Thus, the foundational CS content is designed to support the future needs of all students, not just those who will continue to formally study computing. It prepares all members of society to understand the issues related to computing that are necessary for navigating life in the middle of the twenty-first century.

Systematic approach. Decision-makers can think systematically about designing CS learning experiences that support all students, including those traditionally marginalized in CS education. They may find tools such as the CAPE Framework (Fletcher & Warner, 2021) helpful, thinking of CS equity in terms of *capacity* to offer CS, student *access* to CS, student *participation* in CS, and student *experience* in CS. Or, they might use the approach articulated by Santo et al. (2019), which focuses on asking *who* CS is for, *how* CS is taught, and *what* CS is taught. Additionally, the Alliance for Identity-Inclusive Computing Education (AiiCE) delineates tenets for curriculum, pedagogy, professional development, policy, and research that supports increasing the representation, power, and protection of marginalized people in CS (AiiCE, n.d.).

Accessibility for all students. About 15% of students in the U.S. have a disability (National Center for Education Statistics, 2023), and it is unfortunately common to pull out students with disabilities for specialized services during CS instruction (Blaser et al., 2024). It is crucial to design a foundational CS experience that is accessible and appropriate for students with disabilities (Moreno Sandoval et al., 2021). Similarly, especially where CS is a graduation requirement, it is important for schools to ensure that all students have access to the foundational content.

Developing access opportunities for students who enter a school system at a point after the foundational content is taught is crucial. For example, a district that covers some of the foundational content in middle school will need to ensure opportunities for those students who transfer into the district in high school. And while some schools may want to create opportunities for students to learn the foundational content outside of the school day, it is important to ensure that there are alternatives for students who are unable to access out-of-school opportunities (e.g., due to cost, transportation).

While AP CSP significantly overlaps with the foundation (see Section 2.6), it must be supplemented to include all foundational content. Additionally, schools must offer options beyond *only* AP CS Principles (or IB Computer Science, or other advanced options) as a way to learn the foundation. This is due to both real and perceived challenges with taking AP courses (e.g., belief that one can succeed in a college-level course, breadth content and pace of content, cost of exam).

Dispositions. Dispositions are a key component of equitable CS education, and those involved with making decisions about what and how to teach foundational content can intentionally incorporate them into their curriculum. As described in Section 2.3, research shows over and over again that a sense of belonging in CS is a key determinant in students' interest in continuing to study CS, and sense of belonging often differs by demographic group.

Pedagogy. While pedagogy is beyond the scope of this project, there are some instructional methods that are more welcoming to students traditionally left out of CS education, and schools can ensure that educators have access to professional development that prepares them to teach according to these best practices. For example, research has shown that girls will, on average, find activities that use computing for storytelling more motivating than generic activities (Kelleher et al., 2007). Similarly, educational leaders will need to carefully attend to the climate in their CS courses since the elimination of stereotypically "geeky" elements has been shown to encourage more students to study computing (Cheryan et al., 2015). The foundational content can be taught in ways that are culturally relevant (Ladson-Billings, 1995), culturally sustaining (Paris, 2012), and culturally responsive (Scott et al., 2015).

The framework for Culturally Responsive-Sustaining Computer Science Education (Kapor Center, 2021) from the Kapor Center is a useful resource. In short, classroom activities can be created so that they relate to student interests and life experiences (Madkins et al., 2020).

A consistent focus on equity. Note that while equity is not explicitly mentioned in every item in each Topic Area, it is presumed that all topics are *to be implemented in an equitable manner* and that equitable CS requires a critical approach to CS content. For example, one of the learning outcomes in the Algorithms Topic Area is "AL.2 - Recognize that computational solutions take in information, store and process it, and produce a result." Part of this recognition includes learning to challenge the common understanding that an algorithm itself cannot be racist (Madkins et al., 2020).

7.2 Equity Considerations for CS Pathways

When designing implementation pathways, educators and leaders must consider many implications for promoting educational equity, including flexibility, resource limitations, and program alignment.

The importance of flexibility. Flexibility in implementation better supports students who, for example, move into a school district in the middle of high school to participate in the pathway. Flexibility is also useful for students who choose to change pathways, have differing prior experience, extend learning outside of school, or complete self-guided learning. Further, pathways can be created to accommodate a variety of postsecondary plans, including not just higher education but also industry certifications, direct entry into the workforce, and military service. And pathways can be created to accommodate students with a range of prior experience – including no prior experience – in CS, as well as a range of prior math knowledge and English language fluency.

Resource limitations. While highly resourced schools may be able to implement a wide variety of CS pathways and options, students in other types of schools may have fewer opportunities to exercise choice in what CS content to study. Education leaders can make every effort to ensure that resources are available to implement appropriate CS pathways that meet student interests, their community needs, and available resources. Innovative solutions may need to be implemented to overcome barriers specific to rural and urban contexts (e.g., teacher sharing programs, transportation for after-school programs).

Alignment with other programs and entities. Course offerings are often connected to teacher certification/credentialing requirements, which may limit a school's ability to offer specific courses. For example, high school CS courses are often classified as either CTE or traditional academic courses. In some states, dual coding is permitted, and in others, it is not. Offering CTE courses may qualify schools for Perkins V funding to support software, hardware/equipment, curricular materials, teacher professional development, and hiring of new teachers and administrators for up to three years. Opportunities for postsecondary credit (e.g., dual enrollment) and placement in advanced coursework (e.g., after passing AP exams) may be limited by students' ability to pay for college credits, exams, and certifications. Finally, communities place differing priorities on higher education versus certifications, which will impact schools' selection of programs.

Gate-opening vs. gatekeeping. CS teachers identify a lack of support, interest, or knowledge by administrators and counselors as one of the greatest challenges to teaching and promoting equity in CS education (Koshy et al., 2022). Those who schedule courses have a tremendous impact on student participation. For example, misunderstandings lead counselors and administrators to not suggest or recommend CS to students with disabilities (Blaser et al., 2024). It is critical that educators view CS as foundational for all students and support them in pursuing relevant pathways of study.

Course names and descriptions matter. Choosing names for CS courses has been identified as a promising practice for encouraging students from traditionally underrepresented groups to pursue computing (Arnston, 2016). At the same time, there is often a tension between choosing names that are familiar to most students (e.g., "Game Design") and choosing names that may be more appealing to students less likely to fit stereotypes about who CS is for (e.g., "Interactive Media"). Regardless of the name chosen, it is important to ensure that courses appeal widely and that all students, teachers, administrators, and counselors understand what the courses offer.

Collaborating around Computing
P3 Recognizing and Defining Computational Problems
P4 Developing and Using Abstractions
P5 Creating Computational Artifacts
P6 Testing and Refining Computational Artifacts
P7 Communicating about Computing
AP Computer Science Principles

8 Key Recommendations for Future Work

8 Key Recommendations for Future Work

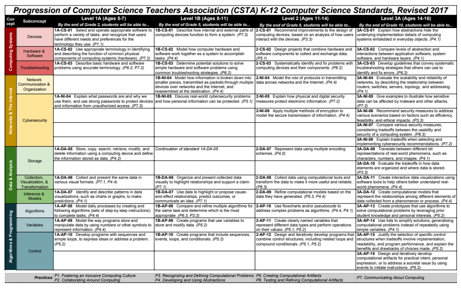
A primary objective of the *Reimagining CS Pathways* project is to formulate recommendations for future work, including recommendations for writers involved in the upcoming CSTA K-12 Standards revision process and those who lead the process of revising Advanced Placement (AP) courses in CS. The following sections delineate recommendations for those specific audiences, as well as others, based on findings from this project and the experiences of those who were involved.

8.1 Recommendations for Standards Writers

For broad implementation, K-12 students will learn content that is included within their adopted curricula, and this content is defined by the CSTA or state-adopted K-12 standards. Thus, it is critical that the K-12 standards are revised to incorporate the foundational content and recommendations from this project. The following recommendations relate to the structure and design of updated standards:

Recommendations related to the structure and design of updated standards	A. Learn from states about obstacles and opportunities related to structuring standards based on grade bands versus discrete grade levels. B. Use relevant research around standards and CS learning to inform design decisions. C. Write standards in a manner that supports both stand-alone and integrated implementation strategies. D. Review high-quality standards from other content areas to determine ideal characteristics of updated CSTA standards. E. Compare current CSTA standards with newer, related frameworks (e.g., cybersecurity) to see what content has withstood the test of time and should be considered foundational. F. Consider including content limits/boundaries to clarify the level of depth intended by a standard. G. Write standards that: i. Explicate connections between Pillars and Topic Areas. ii. Raise issues of bias early and often. iii. Address identity and how it shapes bias. iv. Address accessibility (e.g., learner variability and access to resources) across Topic Areas and progressions. H. Adopt or create a framework that informs how issues of ethics/bias and social impacts appear in the standards. I. Indicate when content builds on prior learning or when it does not necessarily require prerequisite knowledge, particularly within high school pathways. J. Consider how Dispositions might be incorporated into or inform the standards.

Recommendations to support standards implementation and increase usability	A. Create a progression document similar to the existing CSTA progression document for an easily digestible version of the standards.



Progression of Computer Science Teachers Association (CSTA) K-12 Computer Science Standards, Revised 2017

Con cept	Subconcept	Level 1A (Ages 5-7)	Level 1B (Ages 8-11)	Level 2 (Ages 11-14)	Level 3A (Ages 14-16)
		By the end of Grade 2, students will be able to...	*By the end of Grade 5, students will be able to...*	*By the end of Grade 8, students will be able to...*	*By the end of Grade 10, students will be able to...*
Computing Systems	Devices	**1A-CS-01** Select and operate appropriate software to perform a variety of tasks, and recognize that users have different needs and preferences for the technology they use. *(P1.1)*	**1B-CS-01** Describe how internal and external parts of computing devices function to form a system. *(P7.2)*	**2-CS-01** Recommend improvements to the design of computing devices, based on an analysis of how users interact with the devices. *(P3.3)*	**3A-CS-01** Explain how abstractions hide the underlying implementation details of computing systems embedded in everyday objects. *(P4.1)*
	Hardware & Software	**1A-CS-02** Use appropriate terminology in identifying and describing the function of common physical components of computing systems (hardware). *(P7.2)*	**1B-CS-02** Model how computer hardware and software work together as a system to accomplish tasks. *(P4.4)*	**2-CS-02** Design projects that combine hardware and software components to collect and exchange data. *(P5.1)*	**3A-CS-02** Compare levels of abstraction and interactions between application software, system software, and hardware layers. *(P4.1)*
	Troubleshooting	**1A-CS-03** Describe basic hardware and software problems using accurate terminology. *(P6.2, P7.2)*	**1B-CS-03** Determine potential solutions to solve simple hardware and software problems using common troubleshooting strategies. *(P6.2)*	**2-CS-03** Systematically identify and fix problems with computing devices and their components. *(P6.2)*	**3A-CS-03** Develop guidelines that convey systematic troubleshooting strategies that others can use to identify and fix errors. *(P6.2)*
Networks & The Internet	Network Communication & Organization		**1B-NI-04** Model how information is broken down into smaller pieces, transmitted as packets through multiple devices over networks and the Internet, and reassembled at the destination. *(P4.4)*	**2-NI-04** Model the role of protocols in transmitting data across networks and the Internet. *(P4.4)*	**3A-NI-04** Evaluate the scalability and reliability of networks, by describing the relationship between routers, switches, servers, topology, and addressing. *(P4.1)*
	Cybersecurity	**1A-NI-04** Explain what passwords are and why we use them, and use strong passwords to protect devices and information from unauthorized access. *(P7.3)*	**1B-NI-05** Discuss real-world cybersecurity problems and how personal information can be protected. *(P3.1)*	**2-NI-05** Explain how physical and digital security measures protect electronic information. *(P7.2)*	**3A-NI-05** Give examples to illustrate how sensitive data can be affected by malware and other attacks. *(P7.2)*
				2-NI-06 Apply multiple methods of encryption to model the secure transmission of information. *(P4.4)*	**3A-NI-06** Recommend security measures to address various scenarios based on factors such as efficiency, feasibility, and ethical impacts. *(P3.3)*
					3A-NI-07 Compare various security measures, considering tradeoffs between the usability and security of a computing system. *(P6.3)*
					3A-NI-08 Explain tradeoffs when selecting and implementing cybersecurity recommendations. *(P7.2)*
Data & Analysis	Storage	**1A-DA-05** Store, copy, search, retrieve, modify, and delete information using a computing device and define the information stored as data. *(P4.2)*	*Continuation of standard 1A-DA-05*	**2-DA-07** Represent data using multiple encoding schemes. *(P4.0)*	**3A-DA-09** Translate between different bit representations of real-world phenomena, such as characters, numbers, and images. *(P4.1)*
					3A-DA-10 Evaluate the tradeoffs in how data elements are organized and where data is stored. *(P3.3)*
	Collection, Visualization, & Transformation	**1A-DA-06** Collect and present the same data in various visual formats. *(P7.1, P4.4)*	**1B-DA-06** Organize and present collected data visually to highlight relationships and support a claim. *(P7.1)*	**2-DA-08** Collect data using computational tools and transform the data to make it more useful and reliable. *(P6.3)*	**3A-DA-11** Create interactive data visualizations using software tools to help others better understand real-world phenomena. *(P4.4)*
	Inference & Models	**1A-DA-07** Identify and describe patterns in data visualizations, such as charts or graphs, to make predictions. *(P4.1)*	**1B-DA-07** Use data to highlight or propose cause-and-effect relationships, predict outcomes, or communicate an idea. *(P7.1)*	**2-DA-09** Refine computational models based on the data they have generated. *(P5.3, P4.4)*	**3A-DA-12** Create computational models that represent the relationships among different elements of data collected from a phenomenon or process. *(P4.4)*
Algorithms & Programming	Algorithms	**1A-AP-08** Model daily processes by creating and following algorithms (sets of step-by-step instructions) to complete tasks. *(P4.4)*	**1B-AP-08** Compare and refine multiple algorithms for the same task and determine which is the most appropriate. *(P6.3, P3.3)*	**2-AP-10** Use flowcharts and/or pseudocode to address complex problems as algorithms. *(P4.4, P4.1)*	**3A-AP-13** Create prototypes that use algorithms to solve computational problems by leveraging prior student knowledge and personal interests. *(P5.2)*
	Variables	**1A-AP-09** Model the way programs store and manipulate data by using numbers or other symbols to represent information. *(P4.4)*	**1B-AP-09** Create programs that use variables to store and modify data. *(P5.2)*	**2-AP-11** Create clearly named variables that represent different data types and perform operations on their values. *(P5.1, P5.2)*	**3A-AP-14** Use lists to simplify solutions, generalizing computational problems instead of repeatedly using simple variables. *(P4.1)*
	Control	**1A-AP-10** Develop programs with sequences and simple loops, to express ideas or address a problem. *(P5.2)*	**1B-AP-10** Create programs that include sequences, events, loops, and conditionals. *(P5.2)*	**2-AP-12** Design and iteratively develop programs that combine control structures, including nested loops and compound conditionals. *(P5.1, P5.2)*	**3A-AP-15** Justify the selection of specific control structures when tradeoffs involve implementation, readability, and program performance, and explain the benefits and drawbacks of choices made. *(P5.2)*
					3A-AP-16 Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue by using events to initiate instructions. *(P5.2)*
	Practices	*P1. Fostering an Inclusive Computing Culture* *P2. Collaborating Around Computing*	*P3. Recognizing and Defining Computational Problems* *P4. Developing and Using Abstractions*	*P5. Creating Computational Artifacts* *P6. Testing and Refining Computational Artifacts*	*P7. Communicating About Computing*

B. Curate and/or develop standards-aligned lesson and assessment exemplars (e.g., pre-/post-assessments, project-based units, high-quality integration).

C. Develop standards rubrics for evaluating the level of alignment between curricula and new standards.

D. If using a grade band structure, develop an ideal vertical progression within the grade bands (e.g., within the K-2 grade band, differentiate what a particular standard should look like in Kindergarten versus Grade 1 versus Grade 2).

E. Create guidance on standards implementation for a variety of users (e.g., teacher, building principal, state department of education) with a particular focus on equitable and flexible implementation.

F. Crosswalk updated standards with the CSTA K-12 Standards, 2017 (Seehorn et al., 2017) and other relevant frameworks and standards (e.g., Common Core State Standards (Common Core State Standards Initiative, n.d.), Next Generation Science Standards (Next Generation Science Standards, 2013), Advanced Placement coursework).

G. Curate a list of pedagogical approaches and planning strategies intended to underpin standards implementation (e.g., inquiry-based instruction; Universal Design for Learning (UDL); PRIMM (Predict, Run, Investigate, Modify, Make); Use, Modify, Create).

H. Offer guidance on how to leverage standards to engage student populations that have been traditionally underrepresented in computing (e.g., culturally responsive and sustaining pedagogy (Kapor Center, 2021).

I. Identify best practices for building, supporting, and reinforcing Dispositions through standards implementation.

J. Create a glossary where terminology in the standards is explicitly defined.

K. Provide guidance on effective and equitable assessment practices in CS, including considerations for how educators assess CS learning in an age of AI.

8.2 Recommendations for College Board and ACM/IEEE/AAAI

The College Board offers a well-established way for high school students to earn college credit, offering a bridge from high school to college-level learning. ACM, IEEE, and AAAI have jointly created many versions of the ACM/IEEE-CS/AAAI Computer Science Curricula (Kumar et al., 2024) that describes the content that they recommend be covered in computer science, with the latest being released in 2023. When reimagining CS pathways for high school students, the roles of College Board, ACM, IEEE, and AAAI are important to consider, including how the proposed foundational Topic Areas, Pillars, and Dispositions might impact both.

Recommendations for College Board	A. Align AP CS Principles with (or make it inclusive of) the foundational content, as delineated in Section 2. Add the following content to the AP CSP course framework to ensure that all foundational content exists within the AP CSP course: i. Preparation for the Future (e.g., careers alignment, emerging technologies) ii. Inclusion of AI: e.g., traditional vs. AI/ML algorithms, prompt engineering iii. Hardware and software, including troubleshooting iv. Additional cybersecurity content v. Greater focus on ethics and impacts B. Include assessment items related to Ethical and Social Implications of Computing Systems to ensure that this content is actually taught in classrooms in AP CSA. (This is an existing topic in the course framework but is not included in the exam.) C. Consider developing an AP course focused on the *impacts and ethics* of computing. Such an exam would elevate this critical area of knowledge and would encourage students to pursue learning in this area. It could include the following areas: i. Recognize the ethical implications of design decisions. ii. Understand the societal impacts of computing technologies (e.g., social networks, facial recognition). iii. Be able to articulate arguments for and against various policies and laws related to computing (e.g., net neutrality, limits on children's use of social media). iv. Appropriately provide attribution for code that was produced by others or found in various resources. D. Include items to authentically assess *inclusive collaboration*, focusing on the practice of inclusiveness on software development teams and developing software that meets the needs of all users, including the need to: i. Accommodate a variety of identities and perspectives, including from those with disabilities and from different cultural backgrounds. ii. Advocate for the needs of others. iii. Design and develop with accessibility in mind. E. Consider developing courses beyond AP CSA as content is pushed down to earlier grades. F. More broadly consider treating the computing field like the science field. There are many areas of computing (like cybersecurity and artificial intelligence) and consider AP exams for subfields (like science has for physics, chemistry, biology). G. Partner with ACM to 1) define the scope and sequence for what is commonly referred to as "CS0" and 2) provide guidance on how postsecondary institutions can provide course credit for AP CS Principles.

Recommendations for ACM, IEEE, and AAAI	A. Delineate CS Curriculum (e.g., CS2023) content for early CS major courses (e.g., CS1, CS2), to better support vertical alignment with K-12 content. B. Embed in future CS Curricula and develop guidance to teach *ethics and impacts of computing* content throughout, especially within early CS coursework (e.g., CS1, CS2). This course content would elevate a critical area of knowledge and would encourage students to pursue learning in this area. It could include the following areas: i. Recognize the ethical implications of design decisions. ii. Understand the societal impacts of computing technologies (e.g., social networks, facial recognition). iii. Be able to articulate arguments for and against various policies and laws related to computing (e.g., net neutrality, limits on children's use of social media). iv. Appropriately provide attribution for code that was produced by others or found in various resources. C. Embed in future CS Curricula and develop guidance to teach *inclusive collaboration* throughout, especially within early CS coursework (e.g., CS1, CS2), focusing on the practice of inclusiveness on software development teams and developing software that meets the needs of all users, including the need to: i. Accommodate a variety of identities and perspectives, including from those with disabilities and from different cultural backgrounds. ii. Advocate for the needs of others. iii. Design and develop with accessibility in mind. D. Consider removing the discreteness around how course content can be delivered (maybe creating innovative pathways for achieving the desired learning outcomes). E. Consider how new students are increasingly holding knowledge introduced in K-12 about computer science and how that might impact CS1 (Ko et al., 2024). F. Partner with the College Board to 1) define the scope and sequence for what is commonly referred to as "CS0" and 2) provide guidance on how postsecondary institutions can provide course credit for AP CS Principles. G. Partner with K-12 educators in the next curriculum revision.

8.3 Recommendations for K-12 Educators

There are many roles that K-12 educators play, and we defined recommendations across these various roles. For example, teachers who teach CS can participate in ongoing professional learning (formal or informal), focusing on the foundational CS content. Counselors can reference the various pathways and linked careers to help guide students and develop their interest in CS.

Role	Recommendations
CS Teachers	A. Advocate to teach the foundational CS content to all students. B. Recommend, and advocate for, new pathways and courses that align with student interests and community needs. C. Participate in ongoing professional learning. D. Ensure selected curriculum aligns with reimagined CS and related standards. E. Connect students, particularly those from marginalized communities, to out-of-school learning opportunities and enrichment programs. F. Connect with the larger CS teacher community (locally, through CSTA, or other professional organizations) for support, learning, and collaboration.

Teachers of Subject Areas Outside of CS	A. Identify opportunities to integrate or reinforce foundational CS knowledge and skills, in collaboration with other teachers. B. Connect students to out-of-school learning opportunities and enrichment programs, particularly historically marginalized students. C. Participate in ongoing professional learning. D. Connect with the larger CS teacher community (locally, through CSTA, or other professional organizations) for support, learning, and collaboration.
Instructional Coaches	A. Develop strong familiarity and fluency with K-12 CS standards and the CSTA Standards for Teachers. B. Deepen understanding of how to support CS teachers (e.g., reference the CS coaching toolkit). C. Participate in ongoing professional learning. D. Encourage, engage, and empower all teachers to teach CS. E. Support collaboration between CS and non-CS teachers. F. Connect with the larger CS teacher community (locally, through CSTA, or other professional organizations) for support, learning, and collaboration.
Counselors	A. Learn more about CS by observing CS classes, talking with CS teachers, and attending professional development (such as Counselors 4 Computing). B. Examine biases for who "belongs in CS," and develop understanding of the impact of bias and stereotype threat and how it impacts student advisement. C. Ensure no (intentional or unintentional) gatekeeping of those who are ready to take CS. D. Identify and eliminate barriers to students taking CS. E. Introduce/reinforce CS as a subject for all students. F. Work to help parents navigate CS misconceptions. G. Review/troubleshoot course scheduling (e.g., ensure English learners, students with disabilities, students in AVID, and students in band/orchestra are able to take CS). H. Participate in ongoing professional learning. I. Connect with the larger CS teacher community (locally, through CSTA, or other professional organizations) for support, learning, and collaboration.
Administrators (e.g., Principals, CTE directors)	A. Communicate CS initiatives with families and community members. B. Ensure that all students learn the foundational CS content. C. Practice shared decision-making with teachers when selecting curriculum resources, determining course offerings, etc. D. Select and/or develop relevant new CS pathways and courses that align with student interests and community needs. E. Participate in ongoing professional learning. F. Build enough familiarity with CS content to oversee implementation (e.g., understand CS beyond coding). Means of building familiarity may include attending professional development (PD), talking with a CS teacher, reviewing CS standards, and observing CS classes. G. Allocate resources (funding, time for PD, materials) to support CS. H. Align new or existing CTE programs of study to related/relevant content progressions. I. Connect with the larger CS teacher community (locally, through CSTA, or other professional organizations) for support, learning, and collaboration.

8.4 Recommendations for Curriculum Providers, PD Providers, and School of Education Faculty

Curriculum providers and PD providers play a crucial role in providing learning experiences and teaching strategies for teachers. Similarly, preservice teaching faculty in schools of education share a role in developing the knowledge and skills of preservice teachers.

Role	Recommendations
Curriculum Providers	A. Develop both discrete and integrated curricula that align to the foundational CS content, including Dispositions and Pillars. In particular, include content related to: i. Ethics and impacts of computing ii. Inclusive collaboration B. Develop advanced curricula that align to content progressions and example pathways and that integrate the Pillars.
PD Providers	A. Provide professional learning that supports reimagined CS, the foundational CS content, and example pathways (e.g., develop content that includes emerging areas, fosters Dispositions, integrates with other subject areas, and/or fosters an inclusive classroom environment). In particular, include content related to: i. Ethics and impacts of computing ii. Inclusive collaboration
School of Education Faculty	A. Develop faculty's knowledge and skills related to K-12 CS education, particularly as the foundational content and revised standards are implemented. B. Develop or update programs to prepare K-12 CS teachers that align to the revised CSTA K-12 Standards including relevant pedagogical content knowledge. (See CSTA's Schools of Education Guidance.) C. Include foundational CS content in required coursework. D. Support preservice teachers of all disciplines in understanding connections between CS and their primary discipline (and how they might integrate CS into their instruction).

8.5 Recommendations for Policymakers and Funders

Policymakers and funders are unique audiences that play a strategic role in building the capacity for K-12 CS education. With respect to the standards work, we offer some recommendations for each in the below table.

Role	Recommendations
Policymakers	A. Adopt policies to ensure universal learning of the foundational CS content as defined in Section 2 (e.g., graduation requirement). B. Learn more about CS as a K-12 discipline (e.g., CS is more than coding, AI is a part of CS, ethics and impacts are taught alongside technical content). C. Use student access, participation, and achievement data to inform additional policy related to ensuring all students are able to learn CS and related to implementation of curriculum. D. Adopt policies that are specifically related to ethics in computing and AI as content to be taught. E. Consider policies around preservice teacher preparation in CS. F. Invest in teacher PD and capacity building for implementing foundational CS content and pathways that incorporate the foundational high school CS content.
Funders	A. Fund curriculum and PD programs that align to this vision of Reimagining CS (e.g., providing universal learning of the foundational high school CS content) and that prioritize equity. B. Support strong local CS ecosystems by fostering collaboration among schools, informal learning opportunities, institutions of higher learning, researchers, and nonprofits. C. Fund initiatives that support the integration of ethics in computing and AI into CS curriculum.

8.6 Recommendations for Other Community Members

While updated standards and AP course frameworks were of primary interest throughout this project, it will take action from a broader swath of the CS education community to bring the vision of this project to fruition. The following table outlines a set of recommendations for ideal action by additional roles within the CS education community.

Role	Recommendations
Researchers	Consider answering research questions related to the revised standards, including: A. How do the revised standards impact participation among all students, including historically marginalized groups and students with disabilities? B. How can the ethics and impacts of computing content be assessed in K-12 classrooms? C. How can the proposed Dispositions be integrated into and assessed in K-12 classrooms? D. What effective teaching strategies align with the revised standards? What student populations do they support? E. What learning progressions have been developed to incorporate the revised standards and how do they impact student learning? F. What are impactful ways to integrate instruction that simultaneously meet the revised standards and the standards of the other disciplinary subject? G. How do the revised standards compare to current state standards? H. How have the revised standards been adopted by states? I. How have the revised standards impacted policy, teacher certifications, teacher training, etc.? J. What are unique ways in which various schools incorporate the revised standards? K. What gaps still exist in the revised standards?
Higher Education CS Faculty	A. In postsecondary contexts, develop and implement pedagogies that foster scaffolded, inclusive, collaborative, and relevant instruction, aligned to the vision of reimagining CS. B. Develop vertical K-16 alignment with local school districts and organizations. C. Align entry-level postsecondary courses with advanced content in the high school CS content progressions, including through dual enrollment. D. Update credit or placement policies to reflect the growing CS experience among incoming students (e.g., add AP credit/placement policies, create placement exams), while at the same time making it possible for students who do not have prior CS experience to pursue CS in college.
Industry	A. Develop additional certifications for students that are aligned to the CS content progressions and example pathways. B. Support career exploration (e.g., through guest speaking) and work-based learning (e.g., through mentoring, on-site training, job shadowing) in local schools. C. Develop paid internship or apprenticeship programs for students and teachers.
Families	A. Foster confidence and encourage creativity with CS. Ask questions about what your children are learning and encourage them to take CS. Encourage them to think about how CS is connected to their personal and/or career interests. B. Advocate for CS instruction in your schools.

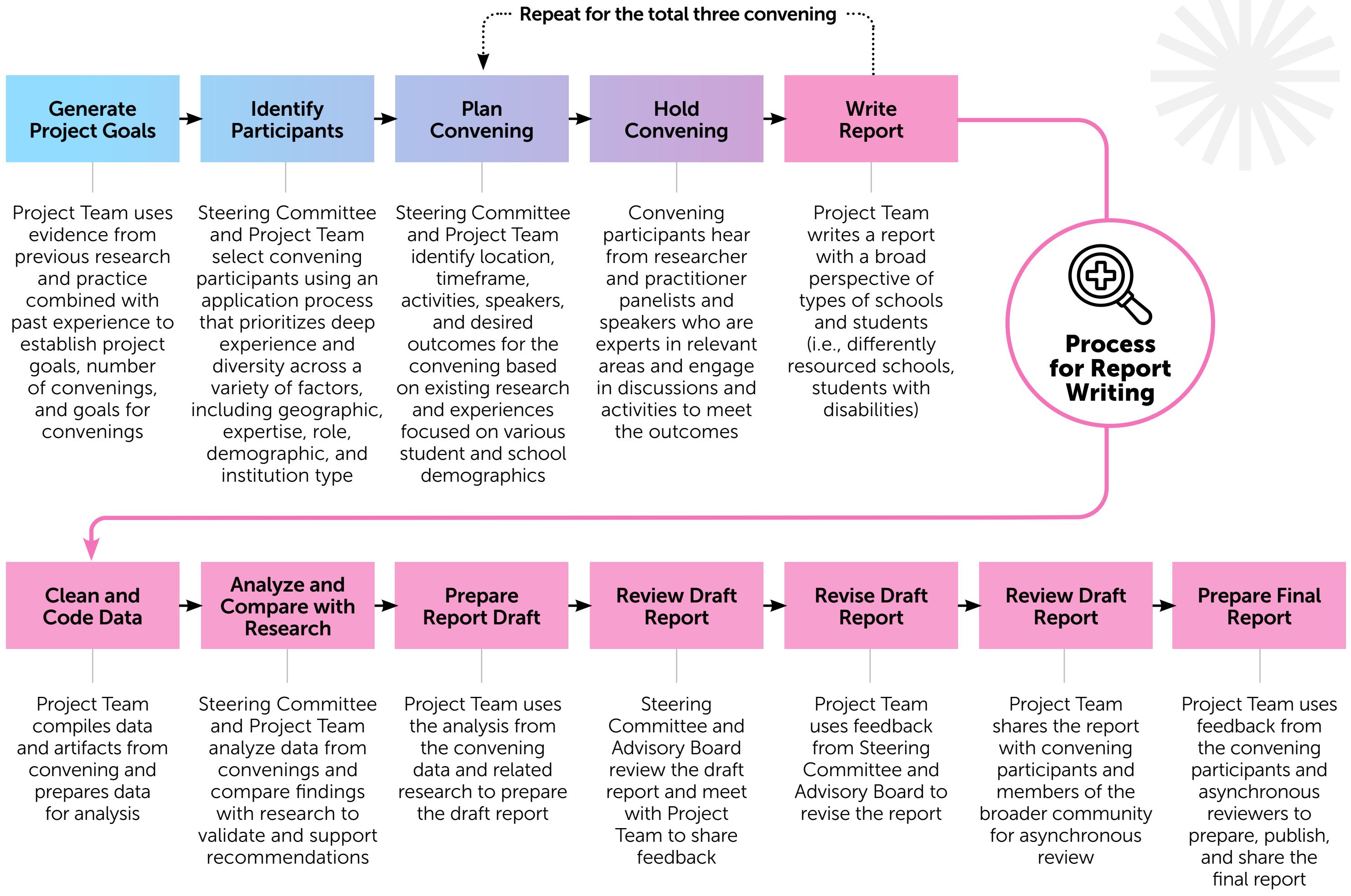
9 Process and Challenges to Reimagining CS Pathways

9 Process and Challenges to Reimagining CS Pathways

We engaged in this project using a concerted and community-driven effort to ensure that proper infrastructure and supports are in place to accommodate the evolution of K-12 CS education over the next five to ten years. This section explains the specific process used in the *Reimagining* project, as well as the challenges we experienced. Section 10 provides a toolkit for replicating this process in the future and in local contexts.

The process to reimagine CS pathways was centered around hosting a series of three in-person convenings with experts from K-12, higher education, and industry. These convenings were complemented by other research, including focus groups, interviews, and literature reviews. After synthesizing data from multiple sources, we drafted and refined reports, with several rounds of feedback. The process used in the *Reimagining* project is explained in Figure 9.1.

Figure 9.1: Process used in the Reimagining CS project.


Repeat for the total three convening
Process for Report Writing

Generate Project Goals: Project Team uses evidence from previous research and practice combined with past experience to establish project goals, number of convenings, and goals for convenings

Identify Participants: Steering Committee and Project Team select convening participants using an application process that prioritizes deep experience and diversity across a variety of factors, including geographic, expertise, role, demographic, and institution type

Plan Convening: Steering Committee and Project Team identify location, timeframe, activities, speakers, and desired outcomes for the convening based on existing research and experiences focused on various student and school demographics

Hold Convening: Convening participants hear from researcher and practitioner panelists and speakers who are experts in relevant areas and engage in discussions and activities to meet the outcomes

Write Report: Project Team writes a report with a broad perspective of types of schools and students (i.e., differently resourced schools, students with disabilities)

Clean and Code Data: Project Team compiles data and artifacts from convening and prepares data for analysis

Analyze and Compare with Research: Steering Committee and Project Team analyze data from convenings and compare findings with research to validate and support recommendations

Prepare Report Draft: Project Team uses the analysis from the convening data and related research to prepare the draft report

Review Draft Report: Steering Committee and Advisory Board review the draft report and meet with Project Team to share feedback

Revise Draft Report: Project Team uses feedback from Steering Committee and Advisory Board to revise the report

Review Draft Report: Project Team shares the report with convening participants and members of the broader community for asynchronous review

Prepare Final Report: Project Team uses feedback from the convening participants and asynchronous reviewers to prepare, publish, and share the final report

9.1.1 Convenings

The project held three in-person convenings across 2023-24. The first convening was held in Chicago in November 2023. Its focus was to gather participant input to define what CS content is essential for all high school students. Interim Report #1 summarizes this process. The second convening was held in Atlanta in January 2024, and it focused on articulating pathways stemming from the previously defined essential content. Its work is summarized in Interim Report #2. The final convening, held in Portland, Oregon, in March 2024, revisited the topics of the first two convenings in light of the work produced thus far; it also explored related questions covered in this report, such as how CS content might be integrated into other subject areas.

Convening #1 Nov. 13-14 Chicago, IL	What CS content is essential for all high school graduates?
Convening #2 Jan. 25-26 Atlanta, GA	What content and pathways for continued CS learning should exist for high school students and their postsecondary lives?
Convening #3 Mar. 19-20 Portland, OR	How can we move toward this vision? (recommendations)

These convenings were highly collaborative and generative. For example, to answer the question *What CS content is essential for all high school graduates*?, a variety of activities were designed to gain participant input concerning key CS content for all high school students as well as the level of priority associated with that content. At the first convening, ideas were generated using the lens of several personas, considering what CS knowledge, skills, and dispositions those students would need to experience/develop in high school to be successful in their life and career in the year 2037. Day two of this convening centered on refining and prioritizing the ideas generated on day one. This included identifying gaps and necessary refinements, prioritizing content within categories defined by the concepts and practices from the K-12 CS Framework, and proposing how instructional time might ideally be distributed across these high-level categorizations. A portion of day two was also dedicated to the exploration of dispositions. Throughout both convening days, data was collected via artifact creation (e.g., posters, sticky notes, dot voting) and an online, interactive polling platform (e.g., regular temperature checks, ranking questions, word clouds).

Nearly 300 people expressed interest in joining the project in a call for participation distributed in September 2023. The steering committee and project team selected 42 convening participants from 26 states, via a process that prioritized

deep experience and diversity across a variety of factors, including geography (i.e., U.S. region as well as urban/suburban/rural), expertise, role, demographic, and institution type. For instance, 73% of selected participants identify as women; 21% identify as Black, 17% as Latinx, 12% as Asian, and 2% as Native; and 14% have a disability or chronic condition. Participants included teachers, district and state administrators, K-12 nonprofit leaders, higher education faculty, researchers, and industry partners. The vast majority of participants have experience teaching K-12 CS (68%), developing K-12 CS PD or curriculum (78%), and conducting CS education research (71%). Additionally, 36% have experience teaching postsecondary CS, and 46% have experience working in CS-related industry roles. A breakdown of convening participants by primary professional role and relevant experience can be found below. More detailed demographics are presented in Appendix D.

Figure 9.1.1.1: Convening participants by primary professional role.

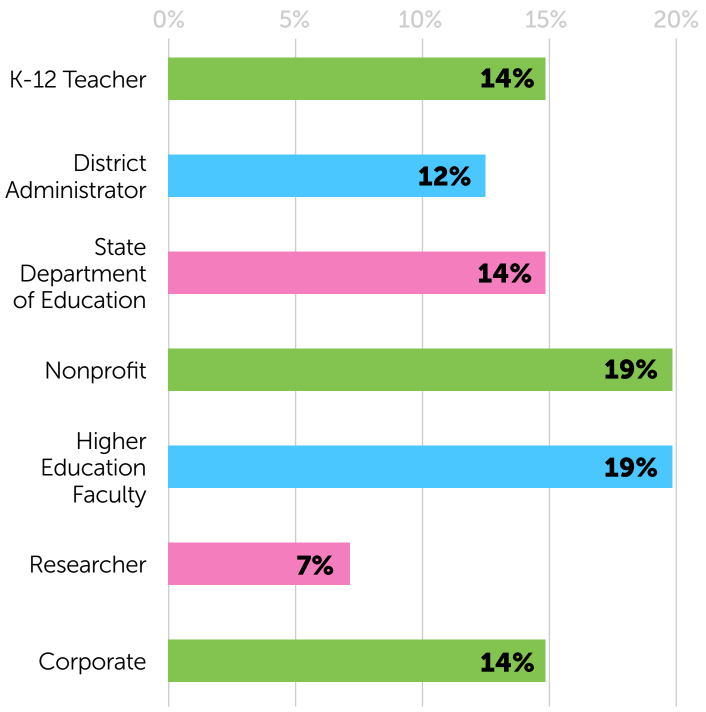
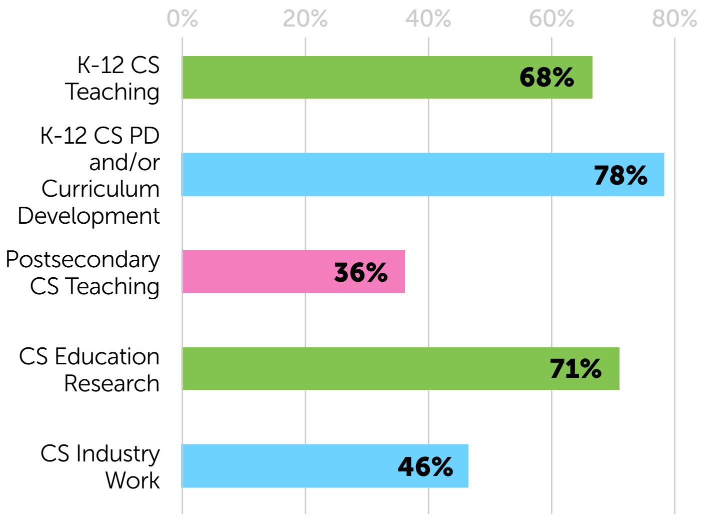


Figure 9.1.1.2: Convening participants' experience related to CS education and industry.



9.1.2 Focus Groups and Interviews

In addition to in-person convenings, we also solicited feedback from focus groups, interviews, and asynchronous reviews on what CS content should be prioritized in a foundational CS course, anticipated changes in the computing industry, anticipated changes in higher education CS courses, and potential pathways for high school CS. From the applicant pool, we hosted a series of focus groups for high school CS teachers, higher education CS instructors, and industry representatives.

We also conducted interviews with several young adults who will be invited to reflect on their experiences with learning CS in high school and/or in postsecondary.

9.1.3 Asynchronous Feedback

We solicited asynchronous feedback from others interested in this work. We asked these participants to vote and comment on what they believe is essential content; over 135 people participated. We also asked asynchronous reviewers to provide feedback on early drafts of interim reports following the first and second convening, as well as early drafts of the final report; 55 people provided written comments on report drafts (see Acknowledgments for a full list). Additional sources of input included interactive conference sessions and industry events.

9.1.4 Synthesizing and Refining Ideas

During the convenings, the participants were advised that providing detailed input was more important than achieving consensus. As a result, participants recorded detailed notes, capturing a multiplicity of voices. Other feedback mechanisms, such as digital voting and commenting, were also used to document ideas. Then, the project team analyzed and synthesized these artifacts, incorporating findings from research into promising practices for teaching CS. Further synthesis and refinement involved input from the steering committee and advisory board. Then, through an iterative process, the project team created report drafts, and participants provided asynchronous feedback on those drafts for validation and refinement.

9.2 Challenges

This section describes the main challenges that convening participants negotiated throughout the process of determining the foundational content and the resultant pathways.

Future forecasting. As the education sector is still working to understand how to grapple with recent advancements in computing (e.g., use of generative AI in education), there was hesitation among participants around how to predict what changes might be on the horizon and how education should adjust accordingly. While it is clear that generative AI and other AI-based tools will have a substantial impact on computing education (Kim, 2023), it is not entirely clear what that impact will be. Given these broader uncertainties, it was difficult for participants to answer the two guiding questions of this project (regarding essential content and pathways) in light of the impact that AI will have on computing education. Similar uncertainties exist – although perhaps to a lesser extent – around other emerging technologies such as quantum computing.

Organizing content. After the first convening, the major challenge that the project leadership encountered when transforming participant feedback into a set of recommendations was how to organize the feedback in a coherent way. Questions around content organization persisted throughout all of the convenings. For example, there is substantial overlap between Computational Thinking (a Pillar) and Algorithms and Programming (two Topic Areas). There is no perfect way to organize material under these headings, and there are advantages and disadvantages to collapsing them into just one or two groups.

Idealism versus pragmatism. Tensions between what is ideal and what is practical manifested in several ways across this project. For example, should the project describe elaborate CS pathways that are not realistic for most schools, or should it describe simple pathways that do not reflect many options for CS study?

Another venue where the tension between idealism and pragmatism came into play was in determining how much content to include in the foundational course. Similarly, there was tension articulating alignment with K-8 and/or postsecondary CS experiences: there is currently a gap between the study of CS in higher education and what is needed to prepare a student for work in industry (Craig et al., 2018; Garousi et al., 2019; Oguz & Oguz, 2019). Also, many in CS education feel that the introductory college-level CS course (often called CS1) needs fundamental reconsideration (Ko, 2022; Luxton-Reilly, 2016; Settle et al., 2015; Sibia et al., 2024), making it difficult to determine whether essential content should anticipate that reconsideration or prepare students for CS1's current implementation.

Meeting the Needs of All Students. Another challenge for the project was balancing the tension between two hypothetical students: one who will pursue a major and a career in computing and another who will follow a very different path. What content is considered essential for a future CS major may well be very different from what is foundational for a future attorney, welder, or nurse, and the same is true of high school pathways stemming from that foundational content.

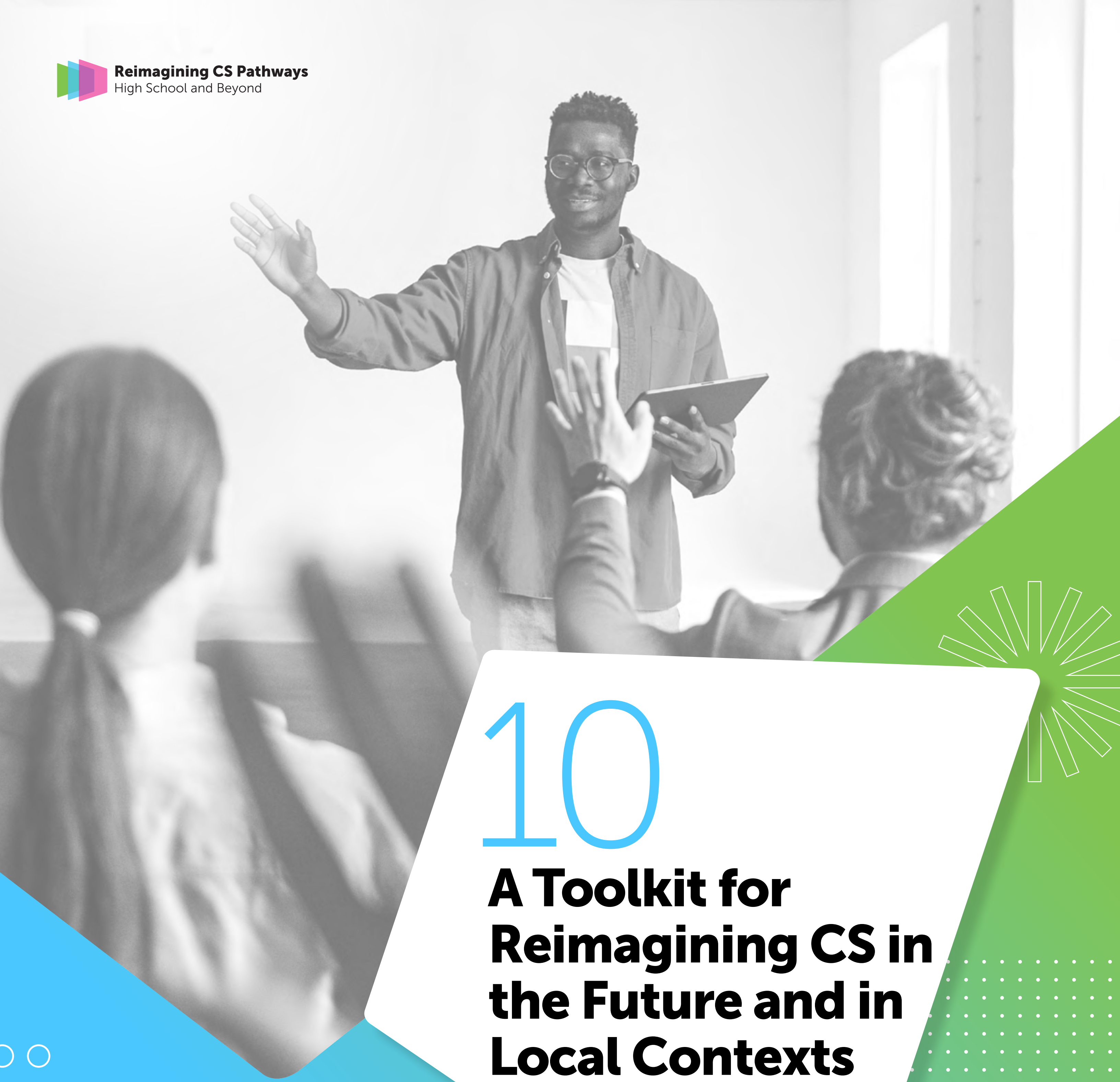
Alignment. The content of a foundational high school course and resultant pathways needs to be aligned with previous experiences (e.g., K-8 computing) and possible future experiences (e.g., higher education, industry, other opportunities). Both K-8 and postsecondary CS education are in flux for a variety of reasons and will likely continue to be. There is also immense variation in K-8 and postsecondary experiences. This combination of variety and change makes it very difficult to map out high school CS.

Granularity. Determining the appropriate level of granularity for each portion of this project was challenging. For example, one of the Topic Areas for the foundational course is Algorithms (see Section 2.5.1), but there is a wide range of possible algorithms that a student might learn about, as well as a range of accompanying learning activities (from understanding an algorithm at a high level to modifying it to implementing it from scratch). Being too specific in this work runs the risks of making the project less amenable to local contexts and future changes in technology, but being too vague can make implementation tricky, create equity issues, and lead to difficulties for curriculum developers, especially in terms of course alignment and progression.

Scope. It was difficult to confine this work to the prescribed tasks of determining foundational CS knowledge and articulating pathways after that foundation. This was particularly true because a robust CS program will include other crucially important facets. For example, we know that dispositions such as a student's sense of belonging are a crucial component of their CS experience and persistence (Moya et al., 2023). But it is rarely appropriate to incorporate an explicit focus on dispositions directly into content standards; rather, dispositions such as a sense of belonging are likely best addressed via other avenues, such as teacher professional development focused on pedagogical practices (Ryoo & Tsui, 2023) or eliminating stereotypical elements from classrooms (Cheryan et al., 2011). Similarly, defining the boundaries of CS was at times challenging. For example, to what extent should digital literacy and digital citizenship be included in essential content?

Forming Consensus. Consensus was achieved across many items during the convenings. Unsurprisingly, participants did not initially reach a consensus on all of the discussion items, including how long the foundational course should be (i.e., one year or one semester) and whether AI tools for programming should be introduced in the foundational course. Generally, the project team used the accompanying detailed feedback to synthesize what was generated by participants. We then provided that synthesis back to participants for further validation, feedback, and refinement.

Reimagining CS Pathways
High School and Beyond



10 A Toolkit for Reimagining CS in the Future and in Local Contexts

10
A Toolkit for Reimagining CS in the Future and in Local Contexts

Based on our experience and reflections from this project, we acknowledge that we will likely need to continue to revise or re-create various community definitions over time as context, research, and content evolve. As such, the following toolkit may be extrapolated to inform myriad planning and contexts at various organizational levels (e.g., local, state, national), including planning the process, running convenings, building in research into the process, and synthesizing and reporting outcomes.

While we engaged in this process of participatory action research, we also built our understanding of how future revisions of CSTA standards could follow similar approaches. In the first convening, for example, it was challenging for the project team to translate the results of participant work into recommendations for foundational content because there was no clear or obvious method for organizing the content. In contrast, we intentionally designed the activities for the second convening via a backward design approach, where we articulated that the goal of the convening was for participants to create multiple pathways stemming from the foundational content. With that goal in mind, we planned activities leading to that outcome. While there were still challenges in synthesizing the content, the task was more manageable. Ultimately, we learned that using a backward design approach made it much easier for participants to understand where the process was leading and it also made the process of synthesizing the data collected in the convenings easier.

Small design decisions (e.g., agenda setting, activities) for the convenings often led to significant impacts. For example, we developed personas of future high school graduates with different backgrounds, experiences, and interests and placed them in their future lives in the year 2037. This provided participants with grounding and brought future high school students along with their future lives into the fold. This led participants to start with a blank slate and dream big about what an ideal high school CS education could be that would support future students' lives and career preparation. This open-ended and forward-looking activity seemed to have encouraged folks to dream more and root the discussions in the future than if we had chosen an approach of taking the existing standards and modifying them. Thus, future projects should carefully consider small design decisions that might shape the course of their work and provide opportunities for participants to set aside the standards that were created years ago.

With technology advancing so rapidly, it will continue to be necessary to reimagine CS education for all grade levels. How often *Reimagining* convenings are held is dependent on the three typical project constraints of time, resources, and financial support, as well as an anticipated cadence of standards revisions. Holding such convenings every five to ten years seems to be a reasonable cadence given the rapid advancements in computing technology.

In this section, we provide a framework that can be used in the future when revised standards are being considered and that can describe how the process of gaining collective input from a wide range of interested parties can unfold. While we provide recommendations here, we encourage the teams to view them just as that rather than being prescriptive. Circumstances of our world and our communities may necessitate different considerations and steps to be taken.

Table 10.1: Planning for community-driven content.

Key steps	We learned that...
Identify and secure funding sources	Of particular concern was ensuring that all participants had travel costs covered. We also recommend providing a stipend to cover participants' time if attending is not part of their regular job roles.
Identify steering committee and advisory board members	Building a diverse committee and board with representatives across vested organizations (such as nonprofit organizations, individuals who have worked on previous versions of standards, College Board, and postsecondary representatives) is beneficial in soliciting critical feedback.
Define the outcomes for the project	Clearly defined outcomes will help guide the project and provide direction for the convenings.
For physical convenings, choose venues with ample physical space	Our space in one hosted building was ample. Our other two spaces were either awkward or too small for all participants, causing crowding and unease.
Determine the number of convenings to hold and the format (in-person, virtual, or hybrid)	Our in-person convenings were held in November 2023, January 2024, and March 2024. No virtual convenings were held, although several participants mentioned that these could have been used to supplement the in-person meetings. Holding virtual convenings in the alternating months (e.g., December, February, and April) could have provided opportunities for participants to reflect more on the convenings. Given the growing importance of integrated CS, consider adding one convening that is dedicated to integration.
Identify dates for the convenings	Consider weekend or summer meetings, which may be easier for teachers to schedule.
Define the convening schedule and topics to be covered at each convening	We spent considerable time aligning the scheduled topics with the data needed to answer our research questions and address project objectives. This was somewhat hampered by project goals that were not as clearly defined as they could have been and personnel change on the project. Ensuring these are well-defined is imperative. We also ensured that the convenings were highly interactive. By the third convening, we incorporated multiple "norms" checks throughout to keep participants on task (rather than checking emails and conducting other work).
Determine the number of convening participants and their roles and demographic information	Approximately 40 people participated in each convening, including the leadership team and steering committee members. While there are pros and cons to any group size, 40 feels large enough to ensure diverse representation on the project while still being small enough to allow for meaningful discussion across participants with consensus being achievable. Consider including guidance counselors, policymakers, administrators, K-8 CS teachers, high school CS teachers, postsecondary faculty, K-12 CS education researchers, CSTA Equity Fellows, CTE teachers, those who studied CS through alternative pathways (i.e., military, trade school, or certification). Perspectives that were not included were non-CS teachers and those who use computing in another discipline (e.g., computational biologist). In retrospect, those perspectives could have enriched the work. Consider that job requirements (e.g., teaching) may interfere with in-person convenings, and hybrid or virtual meetings may need to be incorporated to include all voices.

Table 10.2: Preparing for convenings.

Key steps	We learned that...
Prepare convening agenda and share with steering committee and advisory board members	We needed sufficient time to design agendas that were inclusive and engaging for all participants and to discuss the activities and areas of exploration with both the steering committee members and the advisory board members. Their critical feedback made our agenda and the activities stronger.
Prepare pre-read/prework materials and send along to convening participants	It is important to include critical research on topics related to the agenda items as well as guiding questions for the prework reading. Participants also suggested including survey data from teachers on topics such as what topics they teach that are not in the 2017 standards, what they struggle with, and other data that could inform revised foundational content. Participants also suggested that for integration, share models for integrating CS into other subject areas.
Prepare slides and interactive activities	Building in breaks is important since the time together is all-consuming and requires attention throughout. When appropriate and there is sufficient time, invite participants to review the slides before each convening.

Table 10.3: Activities during convenings.

Key steps	We learned that...
Provide clear context about the goals of each activity	It is easy to conflate aspirational visions of CS education with the realities of CS education within the context of the current education ecosystem. Clearly stating when open-ended and forward-looking discussions are happening that provide space for participants to dream big and brainstorming that takes into account specific parameters will help participants delineate between the two.
Establish group norms early and continue emphasizing group norms, including checking in with participants throughout the event	During the first and second convening, at times we were competing with distractions such as email and other participant obligations. After some small adjustments to keep participants engaged, we landed on intermittent checks throughout the third convening to help with engagement. This was welcomed by participants and appeared to keep the group focused on the tasks at hand.
Provide informal group activities (e.g., meals) as these often included conversations that directly shaped the convening work	Providing informal times for participants to carry on various discussions, some related to the convening topics, enabled reflection and discussion that was brought to the larger group later. These must be accompanied by norms and be attuned to participants' needs so that they are safe, welcoming places for everyone.
Create explicit opportunities for participants to debate about topics and provide clear ways to record divergent views	Divergent views are critical since they offer perspectives that we may not have thought of. Ensuring that there is time for debate and ways to record differing views is necessary to reflect on these when final decisions are being made.
At the end of each convening, provide a preview of the topics/activities of the next convening so participants can begin to think about them	When possible, participants wanted clear outcomes of the convening, including what the leadership team will do with the convening information and how they can further prepare for the next convening.

Table 10.4: Reporting on findings.

Key steps	We learned that...
Summarize each convening and the findings in a draft report after each convening	It is important to synthesize the data from the convening, but also to consider data from research. Balancing dozens of participants' thoughts (as well as additional feedback from focus groups and surveys) is challenging, and having someone on the team that is detail oriented to synthesize the data in a meaningful way is critical. Capturing the key findings is also essential; likewise, adding details about dissenting opinions and challenges is necessary to reflect the reality of the various experiences reflected by participants.
Share the report with the steering committee for feedback; then revise and share with the advisory board for feedback (or vice versa)	Before sharing reports with the broader group, leverage the perspectives and experiences from both the steering committee and project advisory board to ensure the report is meaningful and accurate.
Revise and share with convening participants for feedback	Share the reports asynchronously with participants. Using a platform that masks others' comments is beneficial in ensuring that each person can give thoughtful comments without bias that may form from others' comments. We also found that sharing with a wider group than just participants (like those in the community who were not selected to be part of the convening but who indicated interest in providing asynchronous feedback) also yielded important comments.
Revise based on the additional feedback	Ultimately, the final decision of what to include lies with the leadership team. Carefully considering and providing rationale for why decisions were made, particularly with sticky or controversial topics, is important when framing the final report.

11 Conclusion

11 Conclusion

Reimagining pathways for all high school students has entailed a mixing of visions, hopes, and dreams among a cadre of a diverse set of students, teachers, academics, researchers, and other involved community members. It has required our community to be honest, bold, and radical, as well as hold challenging discussions about current thinking, mindsets, and practices that do not always align with our aspirations for students.

This process has required us to push boundaries where we believe it will be helpful in ultimately achieving our vision for each and every student learning CS in ways that lead to equitable outcomes. This is why, throughout the process, choices have been carefully made to elevate new ways of thinking, such as integrating CS into different subject areas, while also pushing our predictions of what the future holds within the relentlessly fluctuating field of technology.

As educators responsible for shaping the next generation of students, the end of this particular phase of standards revisions brings us to another visioning exercise: *What is possible in a world where all students learn the foundational content?*

It's fair to say that the future of technology comes with many unknowns and cautionary warnings. Even so, barriers to actualization of CS-driven solutions continue to be lowered, and with all students learning CS, the ideation processes for using CS in solving problems can rapidly expand.

> **Some of the seemingly intractable problems of our current and future generations may be tackled by the very students who sit in the classrooms today learning the future of technology and weighing its ethical implications.**

Cures to chronic diseases, improved agricultural techniques that ensure food security for everyone, and the development and proliferation of sustainable and clean energy are just a few areas that can be achieved faster by students who are prepared to understand them and tackle them. However, even everyday problems, like those related to personal data and privacy, precision location tracking used for individual targeted marketing, and upholding our democracy are all imminent issues that can be better addressed by a computer science educated citizenry.

Whether students choose paths that tackle such issues or they choose paths that are differently suited for their life goals, they all will need a background in computing that enables them to make sound decisions and respond to the forces of computing that explicitly and implicitly impact their daily lives. Through this reimagining, current and future students, some of which will inevitably serve as teachers and policymakers in the future, will carry key computing knowledge, skills, and dispositions forward as they serve the next generation of students.

References

ACM Committee for Computing Education in Community Colleges (CCECC). (2023). *Bloom's for Computing: Enhancing Bloom's Revised Taxonomy with Verbs for Computing Disciplines.* Association for Computing Machinery.

Alliance for Identity-Inclusive Computing Education (AiiCE). (n.d.) Identity-Inclusive Computing (IIC) Tenets. Retrieved May 31, 2024, from https://identityincs.org/resources/iic-tenets/

Anant, S. S. (1967). Belongingness and mental health: Some research findings. *Acta Psychologica*, 26, 391–396.

Arnone, M. P., Small, R. V., Chauncey, S. A., & McKenna, H. P. (2011). Curiosity, interest and engagement in technology-pervasive learning environments: A new research agenda. *Educational Technology Research and Development,* 59(2), 181–198. https://doi.org/10.1007/s11423-011-9190-9

Arnston, L. "Harvey Mudd College Confronts Lack of Female Computer Science Majors." *Higher Education Today*, 2 Mar. 2016, www.higheredtoday.org/2016/03/02/harvey-mudd-college-confronts-lack-of-female-computer-science-majors/.

Baumeister, R. F., & Leary, M. R. (1995). The need to belong: Desire for interpersonal attachments as a fundamental human motivation. *Psychological Bulletin, 117*(3), 497–529. https://doi.org/10.1037/0033-2909.117.3.497[2]

Bell, R., & Loon, M. (2015). The impact of critical thinking disposition on learning using business simulations. *The International Journal of Management Education*, 13(2), 119–127.

Bender, R. (2024, April 15). *Washington Bill Proposes New Computer Science Graduation Requirements.* https://www.channelonline.tv/washington-bill-proposes-new-computer-science-graduation-requirements/

Benjamin, R. (2019). *Race after technology: Abolitionist tools for the new Jim code.* John Wiley & Sons. https://www.wiley.com/en-us/Race+After+Technology%3A+Abolitionist+Tools+for+the+New+-Jim+Code-p-9781509526437

Berlyne, D. E. (1954). A Theory of Human Curiosity. *British Journal of Psychology. General Section, 45(3)*, 180–191. https://doi.org/10.1111/j.2044-8295.1954.tb01243.x

Bishop-Clark, C., Courte, J., & Howard, E. V. (2006). Programming in Pairs with Alice to Improve Confidence, Enjoyment, and Achievement. *Journal of Educational Computing Research, 34*(2), 213–228. https://doi.org/10.2190/CFKF-UGGC-JG1Q-7T40

Blaser, B., Ladner, R. E., Twarek, B., Stefik, A. and Stabler, H. (2024). Accessibility and Disability in PreK-12 CS: Results from a Landscape Survey of Teachers. *Proceedings of the 2024 RESPECT Annual Conference (RESPECT 2024)*. Association for Computing Machinery, New York, NY, USA, 13–20. https://doi.org/10.1145/3653666.3656071

Bock, A. C., & Frank, U. (2021). Low-Code Platform. *Business & Information Systems Engineering, 63*(6), 733–740. https://doi.org/10.1007/s12599-021-00726-8

Boud, D., Keogh, R., & Walker, D. (2013). *Reflection: Turning experience into learning.* Routledge. https://www.taylorfrancis.com/books/edit/10.4324/9781315059051/reflection-david-boud-rosemary-keogh-david-walker

Buolamwini, J. A. (2017). *Gender shades: Intersectional phenotypic and demographic evaluation of face datasets and gender classifiers* [PhD Thesis, Massachusetts Institute of Technology]. https://dspace.mit.edu/handle/1721.1/114068

Bureau of Labor Statistics. (2023). *Employment Projections: 2022-2032 Summary - 2022 A01* Results. https://www.bls.gov/news.release/ecopro.nr0.htm

Burgstahler, S. E., & Ladner, R. E. (2007). Increasing the participation of people with disabilities in computing fields. *Computer, 40*(5), 94–97.

Butler, D. L., & Winne, P. H. (1995). Feedback and Self-Regulated Learning: A Theoretical Synthesis. *Review of Educational Research, 65*(3), 245-281. https://doi.org/10.3102/00346543065003245

Century, J., Ferris, K. A., & Zuo, H. (2020). Finding time for computer science in the elementary school day: A quasi-experimental study of a transdisciplinary problem-based learning approach. *International Journal of STEM Education, 7*(1), 20. https://doi.org/10.1186/s40594-020-00218-3

Cheryan, S., Master, A., & Meltzoff, A. N. (2015). Cultural stereotypes as gatekeepers: Increasing girls' interest in computer science and engineering by diversifying stereotypes. *Frontiers in psychology*, 6, 49.

Cheryan, S., Meltzoff, A. N., & Kim, S. (2011). Classrooms matter: The design of virtual classrooms influences gender disparities in computer science classes. *Computers & Education, 57*(2), 1825-1835.

Claxton, G. (2009). Cultivating positive learning dispositions. *In Educational Theories, Cultures and Learning.* Routledge.

Code.org. What Classes Do Students like the Most?, 16 June 2016, blog.code.org/post/146020540698/what-classes-do-students-like-the-most.

Code.org, CSTA, & ECEP Alliance. (2023). *2023 State of Computer Science Education.* https://advocacy.code.org/stateofcs

College Board. (2023). *AP Computer Science Principles: Course and Exam Description.* https://apcentral.collegeboard.org/media/pdf/ap-computer-science-principles-course-and-exam-description.pdf`

Common Core State Standards Initiative. (n.d.). *Common Core State Standards Initiative.* Retrieved May 20, 2024, from https://www.thecorestandards.org/read-the-standards/

Comp-Sci Graduation Mandate Proposed in California. (2024, February 26). GovTech. https://www.govtech.com/education/k-12/comp-sci-graduation-mandate-proposed-in-golden-state

Craig, M., Conrad, P., Lynch, D., Lee, N., & Anthony, L. (2018). Listening to early career software developers. *J. Comput. Sci. Coll, 33*(4), 138-149.

Dison, L., Shalem, Y., & Langsford, D. (2019). Resourcefulness matters: Student patterns for coping with structural and academic challenges. *South African Journal of Higher Education, 33*(4). https://doi.org/10.20853/33-4-2831

Forehand, M. (2010). Bloom's taxonomy. *Emerging Perspectives on Learning, Teaching, and Technology, 41*(4), 47–56.

Fuller, U., Johnson, C. G., Ahoniemi, T., Cukierman, D., Hernán-Losada, I., Jackova, J., Lahtinen, E., Lewis, T. L., Thompson, D. M., Riedesel, C., & Thompson, E. (2007). *Developing a computer science-specific learning taxonomy. ACM SIGCSE Bulletin, 39*(4), 152–170. https://doi.org/10.1145/1345375.1345438

Garcia, R., Morreale, P., Letaw, L., Chatterjee, A., Patel, P., Yang, S., Escobar, I. T., Noa, G. J., & Burnett, M. (2023). "Regular" CS × inclusive design = smarter students and greater diversity. *ACM Transactions on Computing Education, 23*(3), Article 34, 35 pages. https://doi.org/10.1145/3603535

Garousi, V., Giray, G., Tuzun, E., Catal, C., & Felderer, M. (2019). Closing the gap between software engineering education and industrial needs. *IEEE software, 37*(2), 68-77.

Geissler, M., Koumadi, K., Schmelz, P., Servin, C., Tang, C., & Tucker, C. (2023). Designing Learning Outcomes and Competencies Using Bloom's for Computing. *Journal of Computing Sciences in Colleges, 38*(7), 86–88.

Giza, P. (2021). Creativity in computer science. *Creativity Studies, 14*(2), 444–460. https://journals.vilniustech.lt/index.php/CS/article/view/14699

Google, & Gallup. (2017). *Computer Science Learning: Closing the Gap Rural and Small-Town School Districts.* https://services.google.com/fh/files/misc/computer-science-learning-closing-the-gap-rural-small-town-brief.pdf

Guzdial, M. (2022). Teaspoon Languages for Integrating Programming into Social Studies, Language Arts, and Mathematics Secondary Courses. *Proceedings of the 53rd ACM Technical Symposium on Computer Science Education* V. 2, 1027. https://doi.org/10.1145/3478432.3499240

Guzdial, M., & Naimipour, B. (2019). Task-Specific Programming Languages for Promoting Computing Integration: A Precalculus Example. *Proceedings of the 19th Koli Calling International Conference on Computing Education Research*, 1–5. https://doi.org/10.1145/3364510.3364532

Hansen, M. J., Palakal, M. J., & White, L. J. (2023). The Importance of STEM Sense of Belonging and Academic Hope in Enhancing Persistence for Low-Income, Underrepresented STEM Students. *Journal for STEM Education Research.* Scopus. https://doi.org/10.1007/s41979-023-00096-8

Indiana HB1243. (n.d.). LegiScan. Retrieved April 25, 2024, from https://legiscan.com/IN/text/HB1243/id/2956230

Integrated Computational Thinking. (n.d.). Integrated Computational Thinking. Retrieved May 23, 2024, from http://projects.ctintegration.org

Justice-Centered Computing. (n.d.). *Kapor Center*. Retrieved April 27, 2024, from https://www.kaporcenter.org/justicecs/

Kapor Center. (2021). *Culturally responsive-sustaining CS education: A framework.*

Kelleher, C., Pausch, R., & Kiesler, S. (2007). Storytelling Alice motivates middle school girls to learn computer programming. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 1455–1464). Association for Computing Machinery. https://doi.org/10.1145/1240624.1240844

Kennedy, M., Fisher, M. B., & Ennis, R. H. (1991). Critical thinking: Literature review and needed research. *Educational Values and Cognitive Instruction: Implications for Reform, 2*, 11–40.

Kennett, D. J., & Keefer, K. (2006). Impact of Learned Resourcefulness and Theories of Intelligence on Academic Achievement of University Students: An integrated approach. *Educational Psychology, 26*(3), 441–457. https://doi.org/10.1080/01443410500342062

Kim, D. Y. (2023). Redefining Computer Science Education: Code-Centric to Natural Language Programming with AI-Based No-Code Platforms. arXiv preprint arXiv:2308.13539.

Ko, A. (2022). On Pacing and Programming Pedagogy. https://medium.com/bitsand- behavior/on-pacing-and-programming-pedagogy-12c25f9b2c9d.

Ko, A., Beitlers, A., Wortzman, B., Davidson, M., Oleson, A., Kirdani-Ryan, M., Druga, S., & Everson, J. (2024). *Critically Conscious Computing: Methods for Secondary Education.* https://criticallyconsciouscomputing.org/introduction

Koshy, S., Twarek, B., Bashir, D., Glass, S., Goines, R., Cruz Novohatski, L., & Scott, A. (2022). Moving Towards a Vision of Equitable Computer Science: Results of a Landscape Survey of PreK-12 CS Teachers in the United States. Computer Science Teachers Association. https://landscape.csteachers.org/

Krause-Levy, S., Griswold, W. G., Porter, L., & Alvarado, C. (2021). The Relationship Between Sense of Belonging and Student Outcomes in CS1 and Beyond. *Proceedings of the 17th ACM Conference on International Computing Education Research*, 29–41. https://doi.org/10.1145/3446871.3469748

Kugler, L. (2023, March 30). *Will AI Replace Computer Programmers? – Communications of the ACM.* https://cacm.acm.org/news/will-ai-replace-computer-programmers/

Kumar, A. N., Raj, R. K., Aly, S. G., Anderson, M. D., Becker, B. A., Blumenthal, R. L., Eaton, E., Epstein, S. L., Goldweber, M., Jalote, P., Lea, D., Oudshoorn, M., Pias, M., Reiser, S., Servin, C., Simha, R., Winters, T., & Xiang, Q. (2024). *Computer Science Curricula 2023*. Association for Computing Machinery.

Ladson-Billings, G. (1995). But that's just good teaching! The case for culturally relevant pedagogy. *Theory Into Practice, 34*(3), 159–165. https://doi.org/10.1080/00405849509543675

Lee, V. R., Clarke-Midura, J., Shumway, J., & Recker, M. (2022, July). "Design for Co-Design" in a Computer Science Curriculum Research-Practice Partnership. In *Proceedings of the 16th International Conference of the Learning Sciences-ICLS 2022*.

Lehman, K. J., Wofford, A. M., Sendowski, M., Newhouse, K. N. S., & Sax, L. J. (2020). Better Late Than Never: Exploring Students' Pathways to Computing in Later Stages of College. *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, 1075–1081. https://doi.org/10.1145/3328778.3366814

Lewis, K. L., Stout, J. G., Finkelstein, N. D., Pollock, S. J., Miyake, A., Cohen, G. L., & Ito, T. A. (2017). Fitting in to Move Forward: Belonging, Gender, and Persistence in the Physical Sciences, Technology, Engineering, and Mathematics (pSTEM). *Psychology of Women Quarterly, 41*(4), 420–436. https://doi.org/10.1177/0361684317720186

Litman, J. A. (2005). Curiosity and the pleasures of learning: Wanting and liking new information. *Cognition and Emotion, 19*(6), 793–814. https://doi.org/10.1080/02699930541000101

Liu, J., Conrad, C., & Blazar, D. (2024). *Computer Science for All? The Impact of High School Computer Science Courses on College Majors and Earnings.* https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4709691

Luxton-Reilly, A. (2016, July). Learning to program is easy. In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education* (pp. 284-289).

Lyon, L. A., & Green, E. (2020). Women in coding boot camps: An alternative pathway to computing jobs. *Computer Science Education, 30*(1), 102–123. https://doi.org/10.1080/08993408.2019.1682379

Madkins, T. C., Howard, N. R., & Freed, N. (2020). Engaging Equity Pedagogies in Computer Science Learning Environments. *Journal of Computer Science Integration, 3*(2), 1. https://doi.org/10.26716/jcsi.2020.03.2.1

Margolis, J., Estrella, R., Goode, J., Holme, J. J., & Nao, K. (2008). *Claimed spaces:"Preparatory privilege" and high school computer science.*

Master, A., Cheryan, S., & Meltzoff, A. N. (2016). Computing whether she belongs: Stereotypes undermine girls' interest and sense of belonging in computer science. *Journal of Educational Psychology, 108*(3), 424–437. https://doi.org/10.1037/edu0000061

Master, A., Meltzoff, A. N., & Cheryan, S. (2021). Gender stereotypes about interests start early and cause gender disparities in computer science and engineering. *Proceedings of the National Academy of Sciences, 118*(48), e2100030118. https://doi.org/10.1073/pnas.2100030118

Moreno Sandoval, C. D., Hernández Saca, D. I., & Tefera, A. A. (2021). *Intersectional Rights of Teachers and Students in Computer Science and Special Education: Implications for Urban Schooling. Urban Education, 56*(5), 675–704. https://doi.org/10.1177/0042085917714512

Moya, J., Flatland, R., Matthews, J. R., White, P., Hansen, S. R., & Egan, M. L. (2023, March). " I Can Do That Too" Factors Influencing a Sense of Belonging for Females in Computer Science Classrooms. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1* (pp. 680-686).

National Academies of Sciences. (2018). *Assessing and responding to the growth of computerscience undergraduate enrollments.* National Academies Press. https://nap.nationalacademies.org/catalog/24926/assessing-and-responding-to-the-growth-of-computer-science-undergraduate-enrollments

National Center for Education Statistics. (2023). *Students With Disabilities.* https://nces.ed.gov/programs/coe/indicator/cgg

Next Generation Science Standards: For States, By States. (2013). National Academies Press. https://doi.org/10.17226/18290

Noble, S. U. (2018). *Algorithms of oppression: How search engines reinforce racism.* New York university press.

Oguz, D., & Oguz, K. (2019). Perspectives on the gap between the software industry and the software engineering education. *IEEE Access, 7*, 117527-117543.

Panadero, E. (2017). A Review of Self-regulated Learning: Six Models and Four Directions for Research. *Frontiers in Psychology, 8*, 422. https://doi.org/10.3389/fpsyg.2017.00422

Paris, D. (2012). Culturally sustaining pedagogy: A needed change in stance, terminology, and practice. *Educational researcher, 41*(3), 93-97.

Peterson, C., & Seligman, M. E. (2004). *Character strengths and virtues: A handbook and classification* (Vol. 1). Oxford University Press. https://www.apa.org/pubs/books/4317046

Pinto, J. D., Zhang, Y., Paquette, L., & Fan, A. X. (2021). Investigating Elements of Student Persistence in an Introductory Computer Science Course. *CEUR Workshop Proceedings, 3051.* https://experts.illinois.edu/en/publications/investigating-elements-of-student-persistence-in-an-introductory-

Plucker, J. A., Beghetto, R. A., & Dow, G. T. (2004). Why Isn't Creativity More Important to Educational Psychologists? Potentials, Pitfalls, and Future Directions in Creativity Research. *Educational Psychologist, 39*(2), 83–96. https://doi.org/10.1207/s15326985ep3902_1

Rosenbaum, M. (1989). Self-control under stress: The role of learned resourcefulness. *Advances in Behaviour Research and Therapy, 11*(4), 249-258. https://doi.org/10.1016/0146-6402(89)90028-3

Ryoo, J. J., & Tsui, K. (2023). Defining a "Computer Science Person" and the Pedagogical Practices Supporting Positive Identification for Minoritized Youth. *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, 673–679. https://doi.org/10.1145/3545945.3569794

Santo, R., DeLyser, L. A., Ahn, J., Pellicone, A., Aguiar, J., & Wortel-London, S. (2019, February). Equity in the who, how and what of computer science education: K12 school district conceptualizations of equity in 'cs for all'initiatives. In *2019 research on equity and sustained participation in engineering, computing, and technology (RESPECT)* (pp. 1-8). IEEE.

Sax, L. J., Blaney, J. M., Lehman, K. J., Rodriguez, S. L., George, K. L., & Zavala, C. (2018). Sense of Belonging in Computing: The Role of Introductory Courses for Women and Underrepresented MinorityStudents. *Social Sciences, 7*(8), 122. https://doi.org/10.3390/socsci7080122

Scott, K. A., Sheridan, K. M., & Clark, K. (2014). Culturally responsive computing: a theory revisited. *Learning, Media and Technology, 40*(4), 412–436. https://doi.org/10.1080/17439884.2014.924966

Seehorn, D., Carey, S., Fuschetto, B., Lee, I., Moix, D., O'Grady-Cunniff, D., Owens, B. B., Stephenson, C., & Verno, A. (2011). *CSTA K-12 computer science standards: Revised 2011.* Association for Computing Machinery. ISBN: 9781450308816.

Seehorn, D., Primann, T., Lash, T., Twarek, B., Moix, D., Batista, L., Bell, J., Kuszmaul, C., O'Grady-Cunniff, D., Park, M., Pollock, L., Ray, M., Ryder, D., Sedgwick, V., Smith, G., & Uche, C. (2017). *CSTA K-12 Computer Science Standards Revised 2017.* Computer Science Teachers Association. https://members.csteachers.org/documents/en-us/46916364-83ab-4f51-85fb-06b3b25b417c/1/

Settle, A., Lalor, J., & Steinbach, T. (2015, February). Reconsidering the impact of CS1 on novice attitudes. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education* (pp. 229-234).

Shani, I. (2023, June 13). Survey reveals AI's impact on the developer experience. *The GitHub Blog.* https://github.blog/2023-06-13-survey-reveals-ais-impact-on-the-developer-experience/

Sharmin, S. (2021). Creativity in CS1: A Literature Review. *ACM Transactions on Computing Education, 22*(2), 16:1-16:26. https://doi.org/10.1145/3459995

Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review, 22,* 142–158. https://doi.org/10.1016/j.edurev.2017.09.003

Sibia, N., Bui, G., Wang, B., Tan, Y., Zavaleta Bernuy, A., Bauer, C., ... & Petersen, A. (2024, March). Examining Intention to Major in Computer Science: Perceived Potential and Challenges. *In Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1* (pp. 1237-1243).

Sternberg, R. J. (1986). *Critical Thinking: Its Nature, Measurement, and Improvement.* https://eric.ed.gov/?id=ED272882

Strickland, C., Rich, K. M., Eatinger, D., Lash, T., Isaacs, A., Israel, M., & Franklin, D. (2021). Action Fractions: The Design and Pilot of an Integrated Math+CS Elementary Curriculum Based on Learning Trajectories. *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, 1149–1155. https://doi.org/10.1145/3408877.3432483

Stumm, S. von, Hell, B., & Chamorro-Premuzic, T. (2011). The Hungry Mind. *Perspectives on Psychological Science.* https://doi.org/10.1177/1745691611421204

Tissenbaum, M., & Ottenbreit-Leftwich, A. (2020, May 1). *A Vision of K-12 Computer Science Education for 2030 – Communications of the ACM.* https://cacm.acm.org/opinion/a-vision-of-k-12-computer-science-education-for-2030/

Tissenbaum, M., Weintrop, D., Holbert, N., & Clegg, T. (2021). The case for alternative endpoints in computing education. *British Journal of Educational Technology, 52*(3), 1164–1177. https://doi.org/10.1111/bjet.13072

Tucker, A. (2003). *A model curriculum for K-12 computer science: Final report of the ACM K-12 task force curriculum committee.* Association for Computing Machinery. ISBN: 1581138377.

Tucker, A., Deek, F., Jones, J., McCowan, D., Stephenson, C., & Verno, A. (2006). *A Model Curriculum for K–12 Computer Science, 2nd Edition.* Association for Computing Machinery. https://members.csteachers.org/documents/en-us/89c434dc-a22a-449b-b398-87ab22cf2f1e/1/

Vakil, S. (2018). Ethics, identity, and political vision: Toward a justice-centered approach to equity in computer science education. *Harvard Educational Review, 88*(1), 26–52.

van Laar, E., van Deursen, A. J. A. M., van Dijk, J. A. G. M., & de Haan, J. (2020). Determinants of 21st-Century Skills and 21st-Century Digital Skills for Workers: A Systematic Literature Review. *Sage Open, 10*(1), 2158244019900176. https://doi.org/10.1177/2158244019900176

Veilleux, N., Bates, R., Allendoerfer, C., Jones, D., Crawford, J., & Floyd Smith, T. (2013). The relationship between belonging and ability in computer science. *Proceeding of the 44th ACM Technical Symposium on Computer Science Education,* 65–70. https://doi.org/10.1145/2445196.2445220

Vogel, S., Hoadley, C., Vogelstein, L., Barrales, W., James, S., Ascenzi-Moreno, L., Ma, J., Wu, J., Wu, F., & Marquez, J. (2023). A Translanguaging Approach to Computing Education: Language Justice, CS, and You. In CS Educational Justice Collective (Ed.), *Advancing Educational Equity in Computer Science*. https://edtechbooks.org/aeecs/chapter_6_a_translanguaging_approach_to_computing_education

Wang, J., & Hejazi Moghadam, S. (2017). Diversity barriers in K-12 computer science education: Structural and social. *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education,* 615–620.

Weisberg, L., Barrett, J., Israel, M., & Miller, D. (2024). A review of arts integration in K-12 CS education: gathering STEAM for inclusive learning. *Computer Science Education*, 1–30. https://doi.org/10.1080/08993408.2024.2359854

Welsh, M. (2023, January 1). *The End of Programming – Communications of the ACM.* https://cacm.acm.org/opinion/the-end-of-programming/

Wing, J. M. (2006). *Computational thinking. Communications of the ACM, 49*(3), 33–35. https://doi.org/10.1145/1118178.1118215

Yadav, A., & Heath, M. K. (2022). Breaking the Code: Confronting Racism in Computer Science through Community, Criticality, and Citizenship. *TechTrends, 66*(3), 450–458. https://doi.org/10.1007/s11528-022-00734-9

Zarestky, J., Bigler, M., Brazile, M., Lopes, T., & Bangerth, W. (2022). Reflective Writing Supports Metacognition and Self-regulation in Graduate Computational Science and Engineering. *Computers and Education Open, 3,* 100085. https://doi.org/10.1016/j.caeo.2022.100085

Zhu, J., Ross, M., & Patel, D. (2022, August 23). *Avoiding Barriers: A Literature Review on the Alternative Pathways for Women in Computer Science.* 2022 ASEE Annual Conference & Exposition. https://peer.asee.org/avoiding-barriers-a-literature-review-on-the-alternative-pathways-for-women-in-computer-science

Zimmerman, B. J. (2008). Investigating Self-Regulation and Motivation: Historical Background, Methodological Developments, and Future Prospects. *American Educational Research Journal, 45*(1), 166–183. https://doi.org/10.3102/0002831207312909

Appendix A:

2017 CSTA Standards and Reimagining Comparison

Our project began by using the organization of content from the CSTA K-12 Standards (2017), as aligned to the K-12 CS Framework (2016). This includes five core concepts and seven practices, listed in the left-most columns in the table below. We reorganized content into the Topic Areas, Pillars, and Dispositions as detailed in Section 2 (and noted in the Reimagining CS column below). A comparison between the concepts and practices from the 2017 CSTA K-12 Standards and the *Reimagining CS Pathways* project is summarized in the following table.

	CSTA Standards and K-12 Framework	Reimagining CS	Justification for Change
Concepts	Computing Systems	Computing Systems and Security	Combined to reflect the overlap in key content as well as participant priorities
	Networks and the Internet		
	Data and Analysis	Data and Analysis	No title change
	Algorithms and Programming	Algorithms	Separated to reflect the importance of algorithms and their distinction from programming
		Programming	
	Impacts of Computing	Impacts and Ethics	Integrated to reflect the importance of integrating consideration of impacts and included as a Pillar
Practices	Recognizing and Defining Computational Problems	Added to Algorithms	Added to other areas (as indicated) due to overlap in key content
	Developing and Using Abstractions	Added to Algorithms	
	Creating Computational Artifacts	Added to Programming	
	Testing and Refining Computational Artifacts	Added to Programming	
	Fostering an Inclusive Computing Culture	Inclusive Collaboration	Added as a Pillar due to overlap in key content and its relevance to all other areas
	Collaborating around Computing		
	Communicating about Computing		
New	N/A	Human-Centered Design	Added as a Pillar as a result of its importance in the context of accessibility and human-centered computing
		Dispositions	Added as a Pillar to reflect the importance of certain dispositions (e.g., persistence)
		Preparation for the Future	Added as a Topic Area to highlight the importance of learning about (1) pathways and careers in computing and (2) emerging technologies
		Computational Thinking	Added as a Pillar to reinforce the importance of developing computational thinking skills across Topic Areas

Appendix B:

Foundation Summary

Encompasses all content

Applies to each topic

Dispositions	Impacts and Ethics	Human-Centered Design	Inclusive Collaboration	Computational Thinking
Algorithms	• Define *algorithm*, including traditional and AI/ML algorithms • Compose, modify, and interpret algorithms • Decompose a problem into multiple subproblems • Evaluate aspects of different algorithms			
Programming	• Convert an algorithm to code • Modify a program • Articulate whether a program solves a given problem • Test and debug a program systematically			
Data and Analysis	• Describe, at a high level, the role of data in AI/ML applications • Prepare (e.g., normalize, transform, clean) data • Trace how data moves through a program • Evaluate data visualizations • Work with large data sets			
Computing Systems and Security	• Identify various types of hardware and software • Describe why cybersecurity is important • Explain what networks (including the Internet) are and how they work • Apply troubleshooting strategies to identify and fix problems • Use documentation and other resources to guide tasks			
Preparing for the Future	• Identify pathways and careers that involve computing • Apply computing concepts to other academic disciplines • Examine how emerging technologies are impacting a variety of practices • Evaluate the use of emerging technologies • Plan how an emerging technology could meet a need			

Appendix C: Supplemental Materials

Interim Report #1:

The first interim report from the Reimagining CS Pathways: High School and Beyond project provides a draft definition of the essential computer science content for all high school graduates. This was published in January 2024, following the first phase of the project. A primary source of data was an in-person convening of K-12 educators, higher education faculty, and industry held in November 2023.

Interim Report #2:

The second interim report from the Reimagining CS Pathways: High School and Beyond project provides draft pathways for continued computer science learning beyond a foundational high school course. It includes content progressions for seven specialty areas including programming, AI, cybersecurity, data science, physical computing, game design, and X+CS. It also includes example course pathways showing how these content progressions could be implemented as courses. This report was published in April 2024, following the second phase of the project. A primary source of data was the second in-person convening of K-12 educators, higher education faculty, and industry held in January 2024.

Personas:

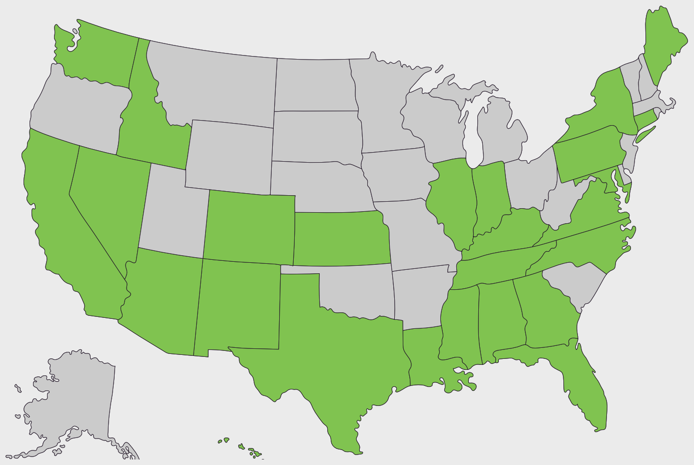
CSTA and IACE developed 13 personas of future high school graduates with different backgrounds, experiences, and interests and placed them in their future lives in the year 2037. These personas can be used to articulate the essential computer science content that all high school students learn. Small groups of participants are presented with these personas and asked to consider what CS content learned in high school would have best prepared these people for their current life circumstances, including but not limited to their occupations. This file includes the 13 personas, plus an explanation, suggested activity protocol, and facilitation guidance.

Appendix D: Participant Demographics and Experience

The steering committee and project team selected 42 convening participants via a process that prioritized deep experience and diversity across a variety of factors, including geographic (i.e., U.S. region as well as urban/suburban/rural), expertise, role, demographic, and institution type.

States

Participants represent 26 states: AL, AZ, CA, CO, CT, FL, GA, HI, ID, IL, IN, KS, KY, LA, MD, ME, MS, NC, NM, NV, NY, PA, TN, TX, VA, and WA.



Race/Ethnicity

The table below shows the distribution of participants' racial and ethnic identities. Several participants identify with multiple races or ethnicities, so the numbers and percentages do not sum to 42 and 100%, respectively.

Race/Ethnicity	Number	Percentage
White or Caucasian	22	52%
Black or African American	9	21%
Hispanic or Latinx	7	17%
Asian or Asian American	5	12%
Prefer not to answer	2	5%
American Indian or Alaska Native	1	2%
Native Hawaiian or Pacific Islander	0	0%
Another race or ethnicity	0	0%

Gender Identity

The majority of participants identify as women (n = 30, 71%), and the remainder identify as men (n = 12, 29%). No participants identify as nonbinary or another gender.

Gender Identity	Number	Percentage
Woman	30	71%
Man	12	29%
Nonbinary	0	0%
Another gender	0	0%
Prefer not to answer	0	0%

Disability Status

Approximately 14% of participants identify as having a disability or chronic condition. We did not collect data about specific types of disability or condition, though we did ask about and provide disability-related accommodations at convenings.

Identify as having a disability	Number	Percentage
No	27	75%
Yes	5	14%
Prefer not to answer	4	11%

Primary Professional Role

Participants' current and primary professional roles were relatively balanced across K-12 teachers, higher education faculty, district administrators, state departments of education, corporations, and K-12 CS education nonprofit organizations. While there are only three participants whose primary role is researcher, 71% of participants have experience with CS education research (as shown in the next table: Professional Experience).

Primary Professional Role	Number	Percentage
Higher Education Faculty	8	19%
Nonprofit	8	19%
Corporate	6	14%
K-12 Teacher	6	14%
State Department of Education	6	14%
District Administrator	5	12%
Researcher	3	7%

Professional Experience

Participants have wide-ranging experience across K-12 CS education, postsecondary CS education, and industry, with an average of 9 experience types listed in the table below.

Experience	Number	Percentage
K-12 CS professional development	29	76%
CS education research	27	71%
K-12 CS curriculum development	24	63%
9-12 CS teaching	21	55%
Teaching introductory high school CS courses	20	53%
K-12 CS standards development	17	45%
CS industry work	17	45%
Teaching AP CSP and/or AP CSA courses	16	42%
6-8 CS teaching	14	37%
K-12 district or local education agency leadership	13	34%
K-12 school leadership	12	32%
K-12 state education agency leadership	10	26%
K-5 CS teaching	9	24%
Postsecondary CS teaching at four-year primarily undergraduate institution	8	21%
Postsecondary CS teaching at four-year PhD-granting institution	8	21%
Teaching dual enrollment CS courses	5	13%
Postsecondary CS teaching at HSI	4	11%
Postsecondary CS teaching at two-year institution	3	8%
Postsecondary CS teaching at HBCU	1	3%
K-12 guidance counselor	0	0%
Postsecondary CS teaching at Tribal College/University	0	0%

Expertise Supporting Marginalized Groups

Participants have significant expertise serving student populations that are marginalized and underrepresented in CS education, as indicated in the following table.

Expertise Supporting Marginalized Groups	Number	Percentage
Girls and nonbinary students	29	76%
Economically disadvantaged students (or Title I schools)	23	61%
Latinx or Hispanic students	23	61%
Black or African American students	20	53%
Students with disabilities	19	50%
Bi-/multilingual learners (English learners)	16	42%
Rural communities	15	39%
Native or Indigenous students	9	24%
Students who identify as LGTBQ+	8	21%
Students who are experiencing homelessness	7	18%
Migrant students	7	18%

CS Content Teaching Experience

Participants have taught the following CS content in their classrooms. The most common topics were computational thinking, algorithms, programming, and impacts of computing.

CS Content Coverage	Number	Percentage
Computational thinking	27	71%
Algorithms and programming	25	66%
Impacts of computing	25	66%
Digital citizenship	24	63%
Computing systems (e.g., hardware/software)	21	55%
Data and analysis	21	55%
Networks and the Internet	21	55%
Ethics	21	55%
Accessibility	19	50%
Web development	19	50%
Physical computing	19	50%
App development	15	39%
Artificial intelligence (AI)	15	39%
Cybersecurity	15	39%
Robotics	14	37%
Data science	14	37%
Game design/development	14	37%
Internet of Things	13	34%
Quantum computing	3	8%

Acknowledgments

We acknowledge the many participants who added valuable insights to the project via focus groups and interviews. These contributors include those who had recently completed high school, high school teachers, higher education faculty, and those who work in tech-related industry positions. We also thank Microsoft for hosting our first participant convening and recognize the following individuals for their important contributions to the project:

Convening Participants

Julie Alano, High School CS Teacher and Department Chair, Hamilton Southeastern High School, Fisher, IN

Cathy Ammirati, STEM Outreach Manager, Micron, Boise, ID

Dr. Owen Astrachan, Professor of the Practice, Duke University, Durham, NC

Quiana Bannerman, Instructional Supervisor – CTE, Prince George's County Public Schools, White Plains, MD

Kris Beck, Director of CS, Chicago Public Schools, Chicago, IL

Dr. Mohsen Beheshti, CS Department Chair, California State University, Carson, CA

Darlene Bowman, Founder and CS Teacher, AusomeTech Industries, Staten Island, NY

Dr. Quinn Burke, Sr. Director of Computational Thinking Research, Digital Promise, San Francisco, CA

Maria Camarena, CS Teacher, Maywood Center for Enriched Studies, Maywood, CA

Justin Cannady, Northern Lights Collaborative for Computing Education, Birmingham, AL

Cindi Chang, Director of Teaching and Learning, Nevada DOE, Las Vegas, NV

Dr. Terry Coatta, Practitioners Board Chair, Association of Computing Machinery, Langley, BC

Jackie Corricelli, PreK-12 CS Curriculum Specialist and Teacher, West Hartford Public Schools, West Hartford, CT

Lien Diaz, Director and Sr. Research Associate, Constellations Center for Equity in Computing, Atlanta, GA

Becca Dovi, Chief CS Advocate, CodeVA, Richmond, VA

Charlotte Dungan, Program Architect, Mark Cuban Foundation, Durham, NC

Rachel Fenichel, Engineering Manager, Google, San Francisco, CA

Sara Frey, State Lead for K-12 CS Education, Pennsylvania DOE, Harrisburg, PA

Crystal Furman, Director of Java in Education, Oracle, Snellville, GA

Laura Gray, AI and CS Instructional Specialist, Gwinnett County Public Schools, Athens, GA

Dr. Christopher Harris, School Library System Director, Genesee Valley BOCES, LeRoy, NY

Wren Hoffman, Professional Development Lead, aiEDU, Timnath, CO

Dr. Sean Jackson, K-12 CS Lead, Kentucky DOE, Frankfort, KY

David Lockett, K-12 Outreach and Federal Programs Manager, Meharry School of Applied Computational Sciences, Nashville, TN

Dr. Michelle Magallanez, Head of Strategic Partnerships and Innovation, AVID, San Diego, CA

Dr. Janice Mak, Clinical Assistant Professor, Arizona State University, Tempe, AZ

Dr. Laura Malavé, College of CS and IT Faculty, St. Petersburg College, St. Petersburg, FL

Dr. Amanda Mason-Singh, Principal Data Scientist, The MITRE Corporation, McLean, VA

Sofia Mohammed, Executive Director, Raspberry Pi Foundation – North America, Madison, MS

Angela Oechslie, Director of Project Login, Educate Maine, Bangor, ME

Yolanda Payne, Research Associate, Georgia Tech, Northeast Georgia, GA

Dr. Rafi Santo, Principal Researcher, Telos Learning, New York, NY

Dr. Allison Scott, CEO, Kapor Foundation, Oakland, CA

Dr. Adam Smeets, Senior Product Manager, Microsoft, Chicago, IL

Carla Strickland, Digital Curriculum Develop- ment Manager, UChicago, Chicago, IL

Cat Tabor, CS Teacher, Canutillo Independent School District, El Paso, TX

Brett Tanaka, CS Education Specialist, Hawaii DOE, Honolulu, HI

Amy Traylor, CS Program Development Specialist, Albuquerque Public Schools, Albuquerque, NM

Dr. John Underwood, STEM Specialist, Louisiana Department of Education, Baton Rouge, LA

Thomas Wang, High School CS Teacher, LGSUHSD, Bay Area, CA

Perla Weaver, Associate Professor, Johnson County Community College, Overland Park, KS

Lavita Williams, CS Specialist, Georgia DOE, Atlanta, GA

Guest Speakers

Dr. Quinn Burke, Sr. Director of Computational Thinking Research, Digital Promise

Dr. Terry Coatta, Practitioners Board Chair, Association of Computing Machinery

Laura Gray, AI and CS Instructional Specialist, Gwinnett County Public Schools

Dr. Shanika Hope, Director of CS Impact and Outreach, Google

Dr. Manuel Pérez-Quiñones, Professor, University of North Carolina at Charlotte

Dr. Rafi Santo, Principal Researcher, Telos Learning, LLC

Dr. Matt Welsh, Chief Architect and Co-founder, Fixie.ai

Dr. Jason VanBilliard, Senior Director, AP Math and CS Department Head, College Board

National Science Foundation Contributors

Dr. Jeff Forbes, Program Director, National Science Foundation

Dr. Jill Denner, Program Director, National Science Foundation

Dr. Allyson Kennedy, Program Director, National Science Foundation

Dr. Laurel Watkins de Jong, AAAS Science Policy Fellow, National Science Foundation

Staff Support

Rachael Steacy, Event Planner, Rachael Steacy Events

Jake Baskin, Executive Director, CSTA

Michelle Burton, Director of Program Operations, CSTA

Shaina Glass, Director of Education, CSTA

Laycee Thigpen, Research Assistant, IACE

c|change, Report Design

Rae Ward, Web Presence

Reviewers

Leah Aiwohi, Kauai High School

Mayra Bachrach, Retired

Jan Barber-Doyle, Retired

Keith Barnes, Bullitt County Public Schools

Dr. Joanne Barrett, University of Florida

Greg Beutler, Hawaii Technology Academy

Dr. Sol Boucher, Jakarta Intercultural School

Lizzy Brooks, Lick-Wilmerding High School

R. Travis Burton, Dallas ISD

Josh Caldwell, Google

Bess Ann Camara, Ivy Tech Community College

Kate Cameron, Delta High School

Timothy Clark, Gilroy Unified School District

David Czechowski, Hyde Park Central School District

Dr. Melissa DeLaurentis, New Mexico Public Education Department

Zarek Drozda, Data Science 4 Everyone

Michael Fine, Fine Technology Solutions

Dr. Chuck Gardner, Cyber.org

Lori Goldade, Code.org

Rose Mary Gregitis, Unaffiliated

Dr. Yasemin Gulbahar, Teachers College, Columbia University

Poonam Gupta, Brunswick School

Eli Hamrick, North Carolina Department of Public Instruction

Victor Hicks, Coding with Culture

Jason Holt, Uplift Education

Elissa Hozore, Maryland Center for Computing Education

Dr. Mike Karlin, California State University, Dominguez Hills

Christopher Kerr, Newington Public Schools

Joe Kmoch, NCWIT-WI

Eve Kolitsky, American International School of Costa Rica

Jacob Landry, Denman Junior High School

Bill MacKenty, American School of Warsaw

Fred Major, Homewood High School

Dr. Sheri McGuffin, AdvanceKentucky

Kate McLeod, Belmont Public Schools

Kerri Murphy, Norton High School and Exploring Computer Science

Mark Newburn, Vizics Inc.

Katie O'Shea, CodeHS

Dr. Becky Odom-Bartel, Cleveland State University

Josh Paley, Palo Alto Unified School District

Dora Palfi, imagi

David Parker, Grand Canyon University

Laura Peters, Somerville Public Schools

Mike Phelan, First Flight High School

Dr. Philip J. Piety, Montgomery County Public Schools

Shanon Reckinger, University of Illinois Chicago

Bruce Regittko, Oracle Academy

Dr. Laurie Salvail, Cyber.org

Spruha Satavlekar, Indian Institute of Technology Bombay

Rob Schultz, Bellbrook High School

Abby Sheridan, Kentucky Country Day

Marti Shirley, Illinois Mathematics & Science Academy

Jeremiah Simmons, ZERO.health

Lawrence Tanimoto, CSTA Puget Sound

Terrill Thompson, University of Washington DO-IT Center

Dr. Marcelo Worsley, Northwestern University

ReimaginingCS.org