CS1 course description

CMU CS Academy CS1 Introduction to Programming and Computer Science

Philosophy

Computer Science and computational problem solving are fundamental skills for engaging the 21st-century marketplace of ideas and economies. We believe that all students should have the opportunity to learn these skills as they will use them in whatever career they are likely to enter.

This free CS1 curriculum is designed for students in 8th or 9th grade with algebra readiness skills. No prior programming experience is required. It is inspired by a highly successful Intro Computing course (15-112, Fundamentals of Programming and Computer Science) that has been taught at Carnegie Mellon University for the past 10+ years. It is predicated on the notion that learning about programming and computer science should be fun and engaging. This requires interesting problems to solve, as computational problem-solving is the core of computer science. It is why we choose to first expose students to graphical problems in CS1: they are visually engaging, allow for multiple correct solutions, and provide visual cues when a solution goes awry.

There are 12 Units to the course, the course is split up into two parts, CS1a (units 1-7) and CS1b (units 8-12) so that it can be taught as a year long course or two semester based courses. We believe the best way to learn this material is to do it, so each unit provides content for the topic to be investigated, a worked problem(s) to illustrate and let students explore the topic, a set of exercises to hone their mastery of the topic, some end-of-unit exercises that require students to use and synthesize all the topics found in that Unit, and a creative task that lets them further explores the topics in the Unit in a manner driven by their interests.

The course provides its own browser-based Integrated Development Environment (IDE) that the students will use to create and run their programs. It encompasses an editor and compiler, a custom graphics package, and an autograder that is capable of grading not only textual problems and solutions but also a broad range of graphics problems and solutions.

Standard Alignment

For state specific computer science education standard alignment visit **here**.

CSTA (Computer Science Teachers Association) Standards Aligned

3A-AP-139-10 Create prototypes that use algorithms to solve computational problems by leveraging prior student knowledge and personal interests.
Algorithms & Programming Creating

3A-AP-149-10 Use lists to simplify solutions, generalizing computational problems instead of repeatedly using simple variables.
Algorithms & Programming Abstraction

3A-AP-169-10 Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue by using events to initiate instructions.
Algorithms & Programming Creating

3A-AP-179-10 Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.
Algorithms & Programming Computational Problems

3A-AP-199-10 Systematically design and develop programs for broad audiences by incorporating feedback from users.
Algorithms & Programming Creating

3A-AP-219-10 Evaluate and refine computational artifacts to make them more usable and accessible.
Algorithms & Programming Testing

3A-AP-229-10 Design and develop computational artifacts working in team roles using collaborative tools.
Algorithms & Programming Collaborating

3A-AP-239-10 Document design decisions using text, graphics, presentations, and/or demonstrations in the development of complex programs.
Algorithms & Programming Communicating

K-12 CS Framework Practices

Practice 1. Fostering an Inclusive Computing Culture

Building an inclusive and diverse computing culture requires strategies for incorporating perspectives from people of different genders, ethnicities, and abilities. Incorporating these perspectives involves understanding the personal, ethical, social, economic, and cultural contexts in which people operate. Considering the needs of diverse users during the design process is essential to producing inclusive computational products.

By the end of Grade 12, students should be able to:

- 1. Include the unique perspectives of others and reflect on one's own perspectives when designing and developing computational products.

Practice 2. Collaborating Around Computing

Collaborative computing is the process of performing a computational task by working in pairs and on teams. Because it involves asking for the contributions and feedback of others, effective collaboration can lead to better outcomes than working independently. Collaboration requires individuals to navigate and incorporate diverse perspectives, conflicting ideas, disparate skills, and distinct personalities. Students should use collaborative tools to effectively work together and to create complex artifacts.

By the end of Grade 12, students should be able to:

- 1. Cultivate working relationships with individuals possessing diverse perspectives, skills, and personalities
- 3. Solicit and incorporate feedback from, and provide constructive feedback to, team members and other stakeholders.

Practice 5. Creating Computational Artifacts

The process of developing computational artifacts embraces both creative expression and the exploration of ideas to create prototypes and solve computational problems. Students create artifacts that are personally relevant or beneficial to their community and beyond. Computational artifacts can be created by combining and modifying existing artifacts or by developing new artifacts. Examples of computational artifacts include programs, simulations, visualizations, digital animations, robotic systems, and apps.

By the end of Grade 12, students should be able to:

- 1. Plan the development of a computational artifact using an iterative process that includes reflection on and modification of the plan, taking into account key features, time and resource constraints, and user expectations.
- 2. Create a computational artifact for practical intent, personal expression, or to address a societal issue.
- 3. Modify an existing artifact to improve or customize it.

Practice 6. Testing and Refining Computational Artifacts

Testing and refinement is the deliberate and iterative process of improving a computational artifact. This process includes debugging (identifying and fixing errors) and comparing actual outcomes to intended outcomes. Students also respond to the changing needs and expectations of end users and improve the performance, reliability, usability, and accessibility of artifacts.

By the end of Grade 12, students should be able to:

- 2. Identify and fix errors using a systematic process.
- 3. Evaluate and refine a computational artifact multiple times to enhance its performance, reliability, usability, and accessibility.

ISTE Standards Alignment

ISTE Standards for Students

1. Empowered Learner

Students leverage technology to take an active role in choosing, achieving and demonstrating competency in their learning goals, informed by the learning sciences.

Students:

A. articulate and set personal learning goals, develop strategies leveraging technology to achieve them and reflect on the learning process itself to improve learning outcomes.

B. build networks and customize their learning environments in ways that support the learning process.

C. use technology to seek feedback that informs and improves their practice and to demonstrate their learning in a variety of ways.

4. Innovative Designer

Students use a variety of technologies within a design process to identify and solve problems by creating new, useful or imaginative solutions.

Students:

A. know and use a deliberate design process for generating ideas, testing theories, creating innovative artifacts or solving authentic problems.

B. develop, test and refine prototypes as part of a cyclical design process.

C. exhibit a tolerance for ambiguity, perseverance and the capacity to work with open-ended problems.

5. Computational Thinker

Students develop and employ strategies for understanding and solving problems in ways that leverage the power of technological methods to develop and test solutions.
Students:

A. break problems into component parts, extract key information, and develop descriptive models to understand complex systems or facilitate problem-solving.

6. Creative Communicator

Students communicate clearly and express themselves creatively for a variety of purposes using the platforms, tools, styles, formats and digital media appropriate to their goals.

Students:

A. create original works or responsibly repurpose or remix digital resources into new creations.

B. communicate complex ideas clearly and effectively by creating or using a variety of digital objects such as visualizations, models or simulations.

C. publish or present content that customizes the message and medium for their intended audiences.