The College Board

AP Computer Science: Principles

Course Annotations

AP Computer Science: Principles is a pilot course under development. It is not an official Advanced Placement course currently being offered by the College Board.

This document is based upon work supported by the National Science Foundation, grant CNS-0938336. 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 National Science Foundation.

AP Computer Science: Principles
Annotated Course Outline

Computing and computer science are helping to shape and change our world. A solid understanding and facility with computational thinking, computing, and computer science is important, if not integral, to being part of a well-educated and informed citizenry. Although the computer science community has worked diligently to create intellectually rich and engaging courses, not all students have had the opportunity to take advantage of these courses. In particular, many students, particularly female students and those from underrepresented minority groups, are either choosing not to enroll in or do not have access to courses, materials, and role-models in computing and computer science. In response to this identified problem, members of the computer science community are developing a curriculum framework for a new *AP Computer Science: Principles* course, designed to complement the existing first course in computer science modeled by the AP Computer Science A course. This annotated course outline highlights the novel aspects of the *Principles* course and then provides a brief explanation of each Big Idea and Computational Thinking Practice that will comprise it.

AP Computer Science: Principles is designed to introduce students to the central ideas of computing and computer science, to instill ideas and practices of computational thinking, and to have students engage in activities that show how computing and computer science change the world. The proposed course is rigorous and rich in computational content, includes computational and critical thinking and skills, and engages students in the creative aspects of the field. Through both its content and pedagogy, this course aims to appeal to a broad audience.

A key theme of the *Principles* course is its focus on creativity. The Big Ideas and Computational Thinking Practices that follow hint at the creative nature of computing and computer science, yet alone they cannot truly convey how we hope creativity should be addressed in the course. It's not enough for students to know that "computing requires creativity." Rather, we want them to actually *be creative*: creating artifacts that they want to show off to their friends and family, using simulation to explore questions that interest them, and designing and implementing solutions employing the iterative and sometimes messy process that artists, writers, and engineers use to translate ideas into tangible form.

A second theme is the course's use of technology as a means for solving computational problems and exploring creative endeavors, rather than a focus on a specific tool or language. To that end, the course highlights programming as one of the seven big ideas of computer science, because programming is among the creative processes that help transform ideas into reality. Programming will be a tool students use to explore concepts and create exciting and personally relevant artifacts. In contrast to traditional college introductory CS courses and the current AP CS A course, the *Principles* course

will not focus on nor be organized around a specific language. The instructor of the course will select one or more languages, based on appropriateness for a specific project or problem and according to guidelines provided as part of the course specification. Language specifics will be taught only to the extent that students need them to produce their programs. Similarly, students in this course will work with "big-data"—to analyze it, to visualize it, to draw conclusions from trends in it—but the course itself does not specify particular tools for these explorations.

A third theme that will help the course appeal to a broad audience is the course's focus on people and society, not just on machines and systems. Students will explore computer science's relevance to and impact on the world today. They will investigate the innovations in other fields that computing and computer science have made possible. They will examine the ethical implications of new computing technologies. They will perform activities that develop their communication and teamwork skills. Students in this course will work individually and in teams to solve problems. They will talk and write about their solutions, the importance of these problems and their impact on the world.

The success of this course hinges on both a compelling curriculum and an engaging pedagogy. The following Big Ideas and Practices specify the course curriculum: the content, practices, thinking, and skills central to the discipline of computing and computer science. Course instructors will be key players in developing pedagogy that brings these to life. Of course we must do more than provide a curriculum framework on which a course can be built, we must ultimately provide the professional development and teacher education necessary to ensure the success of the course on a large scale. As this course develops, pilot instructors will create resources including reading materials, assignments, and lecture materials necessary to ensure the course's success. Instructor training sessions will help new instructors gain the knowledge they need to confidently teach the course.

Through this novel content and engaging pedagogy, we hope that students will experience the joy and beauty that permeates computing and computer science: the sense of community from connecting with friends on social networks, the "ah ha!" moment when an algorithm finally makes sense, the thrill of constructing a program and seeing it work, the pride of creating something for oneself, one's family or friends, or for the world.

AP Computer Science: Principles
Annotated Big Ideas

I. Computing is a creative human activity that engenders innovation and promotes exploration.

Creativity and computing are prominent forces in innovation; the innovations enabled by computing have had and will continue to have far-reaching impact. At the same time, computing and computer science facilitate exploration and the creation of knowledge. This course will emphasize these creative aspects of computing. Students in this course will create interesting and relevant artifacts with the tools and techniques of computing and computer science.

II. Abstraction reduces information and detail to focus on concepts relevant to understanding and solving problems.

Everyone uses abstraction on a daily basis to effectively cope with our world. In computer science, abstraction is a central problem-solving technique. It is a process, a strategy, and the result of reducing detail to focus on concepts relevant to understanding and solving problems. This course will include examples of abstractions used in modeling the world, in managing complexity, and in communicating with people as well as with machines. Students in this course will learn to work with multiple levels of abstraction while engaging with computational problems and systems.

III. Data and information facilitate the creation of knowledge.

Computing enables and empowers new methods of information processing that have led to monumental change across disciplines, from art to business to science. Managing and interpreting an overwhelming amount of raw data is part of the foundation of our information society and economy. People use computers and computation to translate, process, and visualize raw data, creating information. Computation and computer science facilitate and enable a new understanding of data and information that contributes knowledge to the world. Students in this course will work with data using a variety of tools and techniques to better understand the many ways in which data is transformed into information and knowledge.

IV. Algorithms are tools for developing and expressing solutions to computational problems.

Algorithms are fundamental to even the most basic everyday tasks. Algorithms realized in software have affected the world in profound and lasting ways. The

development, use, and analysis of algorithms is one of the most fundamental aspects of computing. Students in this course will work with algorithms in many ways: they will develop and express original algorithms, they will implement algorithms in some language, and they will analyze algorithms both analytically and empirically.

V. Programming is a creative process that produces computational artifacts.

Programming and the creation of software have changed our lives. Programming results in the creation of software, but it facilitates the creation of more general computational artifacts including music, images, visualizations, and more. In this course programming will enable exploration as well as being the object of study. This course will introduce students to the concepts and techniques used in writing programs and to the ways in which programs are developed and used by people; the focus of the course is not programming *per se*, but on all aspects of computation. Students in this course will create programs, translating human intention into computational artifacts.

VI. Digital devices, systems, and the networks that interconnect them enable and foster computational approaches to solving problems.

Digital devices and the Internet have had a profound impact on society. Computer networks support communication and collaboration. The principles of systems and networks that helped enable the Internet are also critical in the implementation of computational solutions. Students in this course will gain insight into how systems and networks operate, to the principles that facilitate their design, and will analyze the effects of systems and networks on people and society.

VII. Computing enables innovation in other fields including science, social science, humanities, arts, medicine, engineering, and business.

Computation has changed the way people think, work, live, and play. Our methods for communicating, collaborating, problem-solving, and doing business have changed and are changing due to innovations enabled by computing. Many innovations in other fields are fostered by advances in computing. Computational approaches lead to new understandings, new discoveries, and new disciplines. Students in this course will become familiar with the many ways in which computing enables innovation in other fields.

AP Computer Science: Principles
Computational Thinking Practices

1. **Analyzing effects of computation**

   Developments in computing have far-reaching effects on society and have led to significant innovations. These developments have implications for individuals, for society, for commercial markets, and for innovation. Students in this course will study these effects—intended or unintended, beneficial or harmful—and learn to analyze, critique, and discuss the ethical, legal and social implications of computing.

2. **Creating computational artifacts**

   Computing is a creative discipline in which the creation takes many forms, ranging from remixing digital music to generating animations to developing websites to writing programs and more. Students in this course will engage in the creative aspects of computing by designing and developing interesting computational artifacts as well as applying computing techniques to creatively solve problems.

3. **Using abstractions and models**

   Computational thinking requires understanding and applying abstraction at multiple levels ranging from privacy in social networking applications to logic gates and bits to the human genome project and more. Students in this course will use abstraction to develop models and simulations of natural and artificial phenomena, use them to make predictions about the world, and analyze their efficacy and validity.

4. **Analyzing problems and artifacts**

   The results and artifacts of computation, and the computational techniques and strategies that generate them, can be understood both intrinsically for what they are as well as for what they produce. They can also be analyzed and evaluated by applying aesthetic, mathematical, pragmatic, and other criteria. Students in this course will design and produce solutions, models, and artifacts and will evaluate and analyze their own computational work as well as the computational work that others have produced.

5. **Communicating processes and results**

   Students in this course will describe computation and the impact of technology and computation, will explain and justify the design and appropriateness of their computational choices, and will analyze and describe both computational artifacts and the results or behaviors of such artifacts. Communication will include written and oral descriptions supported by graphs, visualizations, and computational analysis.

6. **Connecting computation with mathematics, science, and engineering.**

   Mathematics, science, engineering, and computation are closely connected. Students in this course will relate and connect ideas, concepts, and thinking in these areas. For example, students will use mathematical and scientific methods to classify and characterize computation, will use randomness and simulation to create computational artifacts, and will leverage connections between these disciplines in designing, implementing, and analyzing programs and algorithms.

7. **Working effectively in teams**

   Innovation occurs through the work of individuals and teams. Individuals working effectively in teams can sometimes achieve more than individuals working independently. Students in this course will learn about effective teamwork and collaborate in the production of computational artifacts, by applying effective team practices, and by working to understand the different roles that are important in designing, building, and improving computational artifacts.