Greenwich Public Schools Curriculum Overview
COMPUTER SCIENCE PRINCIPLES

Personalized learning is achieved through standards-based, rigorous and relevant curriculum that is aligned to digital tools and resources.

Note: Teachers retain professional discretion in how the learning is presented based on the needs and interests of their students.

Course Description

Computer Science Principles
(Concentration in JavaScript)
Full Year
027400 6 Blocks 1 Credit
Prerequisite: Introduction to Computer Programming A or Introduction to Computer Programming B.
This course blends programming with an exploration of computer science fundamentals. Students will learn how to implement object-oriented data structures and algorithms, and learn the basics of computer science. In addition to coding projects in JavaScript, students will complete a study of data and information, the internet, and the global impact of computing.

Unit Guide

Unit 1 - Digital Information
Unit 2 - The Internet
Unit 3 - Intro to App Design
Unit 4 - Objects
Unit 5 - Lists, Loops, and Transversals
Unit 6 - Algorithms
Unit 7 - Performance Task Preparation
Unit 9 - Data
Unit 10 - Cybersecurity and Global Impacts
Unit 8 - Create Performance Task
* AP Exam Review
* AP Exam or Final Exam
Post-AP Exam: Partner project with Intro the Computer Programming B student or Coding TI Innovator Hub in Python or Lieutenant Governor's App Challenge or other similar topics.

Computational Thinking Practices

- Program Design and Algorithm Development
- Code Logic
- Code Implementation
- Code Testing
- Documentation

Mathematical Practices:

- Make sense of problems and persevere in solving them.
- Reason abstractly and quantitatively.
- Construct viable arguments and critique the reasoning of others.
- Model with mathematics.
- Use appropriate tools strategically.

- Attend to precision.
- Look for and make use of structure.
- Look for and express regularity in repeated reasoning.

Enduring Understandings

- Incorporating multiple perspectives through collaboration improves computing innovations as they are developed.
- Developers create and innovate using an iterative design process that is user-focused, that incorporates implementation/feedback cycles, and that leaves ample room for experimentation and risk-taking.
- The way a computer represents data internally is different from the way the data are interpreted and displayed for the user. Programs are used to translate data into a representation more easily understood by people.
- Programs can be used to process data, which allows users to discover information and create new knowledge.
- To find specific solutions to generalizable problems, programmers represent and organize data in multiple ways.
- The way statements are sequenced and combined in a program determines the computed result. Programs incorporate iteration and selection constructs to represent repetition and make decisions to handle varied input values.
- Programmers break down problems into smaller and more manageable pieces. By creating procedures and leveraging parameters, programmers generalize processes that can be reused. Procedures allow programmers to draw upon existing code that has already been tested, allowing them to write programs more quickly and with more confidence.
- There exist problems that computers cannot solve, and even when a computer can solve a problem, it may not be able to do so in a reasonable amount of time.
- Computer systems and networks facilitate the transfer of data.
- Parallel and distributed computing leverage multiple computers to more quickly solve complex problems or process large data sets.
- While computing innovations are typically designed to achieve a specific purpose, they may have unintended consequences.
- The use of computing innovations may involve risks to personal safety and identity.

Essential Questions

- What are some ways you currently plan your work before starting a project?
- What apps or programs have you stopped using because you didn't like the design of how you interacted with it?
- How can we use1s and 0s to represent something complex like a video of the marching band playing a song?
- How can you predict the attendance at a school event using data gathered from social media?
- When is it more appropriate to use a computer to analyze data than to complete the analysis by hand?
- How can we store data in a program to solve problems?
- What might happen if you completed the steps in your regular morning routine to get ready and go to school in a different order? How might the reordering affect the decisions you make each morning?
- How do video games group the different actions for a player based on what key is pressed on the keyboard or controller? How do apps group different actions together based on user interaction, such as pressing buttons?
- What types of problems can be solved more easily with a computer, and what types can be solved more easily without a computer? Why?

- Why are long text messages sometime delivered out of order?
- When an Internet service outage occurs in a different part of your town or city, how are you still able to access the Internet?
- What are the benefits of dividing tasks among group members?
- Is there a point where adding another group member would not make completing the task faster? Why?
- What app or computer software do you use most often and would have a hard time going without? How does this software solve a problem for you or benefit you?
- Are innovators responsible for the harmful effects of their computing innovations, even if those effects were unintentional? Why or why not?
- What data are generated by smart phones, and what are they being used for?

Resources and Assured Experiences

- Online curricular resource: code.org
- *Additional reading for AP level Course: *Blown to Bits*, Lewis, Ledeen, Abelson and Seltzer, Addison-Wesley Professional; 2nd edition (December 5, 2020), ISBN-13: 978-0134850016
- * AP Classroom
- Online code reference: W3schools.org, Khan Academy, MDN web docs
- Caret IDE or rep.it, once it competes digital tool approval process
- American Computer Science League
- TI nSpire and Innovator Hub

GHS Capstone Task, assessed quarterly

- Capstone Vision of the Graduate Capacity: #6 - Generates innovative, creative ideas and products

Quarterly Grading - Quarter Grades will be determined using the following components:

Non-AP Course

- Participation (includes Classwork) = 10%
- Preparation (includes Homework) = 10%
- Assessments (both Summative & Formative) = 80%

AP Course

- Participation (includes Classwork) = 5%
- Preparation (includes Homework) = 5%
- Assessments (both Summative & Formative) = 90%

CSTA Computer Science Standards

- 3A-CS-01 Explain how abstractions hide the underlying implementation details of computing systems embedded in everyday objects.
- 3A-DA-09 Translate between different bit representations of real-world phenomena, such as characters, numbers, and images.
- 3A-DA-10 Evaluate the tradeoffs in how data elements are organized and where data is stored.
- 3A-DA-11 Create interactive data visualizations using software tools to help others better understand real-world phenomena.
- 3A-AP-13 Create prototypes that use algorithms to solve computational problems by leveraging prior student knowledge and personal interests.
- 3A-AP-13 Use lists to simplify solutions, generalizing computational problems instead of repeatedly using simple variables.

- 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.
- 3A-AP-17 Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.
- 3A-AP-18 Create artifacts by using procedures within a program, combinations of data and procedures, or independent but interrelated programs.
- 3A-AP-19 Systematically design and develop programs for broad audiences by incorporating feedback from users.
- 3A-AP-20 Evaluate licenses that limit or restrict use of computational artifacts when using resources such as libraries.
- 3A-AP-21 Evaluate and refine computational artifacts to make them more usable and accessible.
- 3B-AP-10 Use and adapt classic algorithms to solve computational problems.
- 3B-AP-14 Construct solutions to problems using student-created components, such as procedures, modules and/or objects.
- 3A-IC-24 Evaluate the ways computing impacts personal, ethical, social, economic, and cultural practices.
- 3A-IC-28 Explain the beneficial and harmful effects that intellectual property laws can have on innovation.
- 3A-IC-30 Evaluate the social and economic implications of privacy in the context of safety, law, or ethics.
- 3B-NI-03 Describe the issues that impact network functionality (e.g., bandwidth, load, delay, topology).
- 3B-IC-25 Evaluate computational artifacts to maximize their beneficial effects and minimize harmful effects on society.
- 3B-IC-26 Evaluate the impact of equity, access, and influence on the distribution of computing resources in a global society.