Intro to Computational Thinking

Overview

This course introduces students to the basic ideas of computational thinking and its applications to problem solving in STEM fields. Students will use an open source, Web-based programming environment to create code for simple drawings, animations and simulations, through which they learn how to use abstraction, decomposition and pattern recognition to model problems and arrive to an algorithmic solution. Program code is presented with a dual purpose: as the main way to interact with a computer and as a proxy to organize ideas explicitly and communicate them to other people. Students taking Algebra I concurrently with this course will benefit the most, because many examples are drawn from Algebra I, so that students can visualize and manipulate the mathematical concepts in a more concrete form.

Objectives

- Describe applications of Computational Thinking to solve Math, Science and Engineering problems.
- Model objects made of multiple parts, as well as their behaviors and interconnections, using variables and functions, and construct virtual artifacts that simulate them.
- Use the program development process to create, debug, and redesign computing artifacts.
- Implement creative projects in which Computational Thinking and code is used to create artistic or technical renderings of diagrams, illustrations or graphs.
- Demonstrate the use of code as a medium to communicate ideas and designs precisely.
- Demonstrate effective communication skills, through team working, oral presentations, and good written communication.

Assessment

Formative assessment includes worksheets and several practice activities for each lesson, and unit quizzes. Summative assessment includes a programming project at the end of each unit.

Course Essentials

Equipment	Cost/Unit
Classroom set of computers	$0 if you already have some, $500-600 per computer if you need to purchase

Outline

Unit 1: The Software Development Cycle	Computers and networks. Basic Web design. Coding environments. Drawing shapes with code. Using functions and lists in code. Pseudocode and prototypes. Iterative planning, designing, implementation and testing of programs.
Unit 2: Abstraction and Decomposition	Function applications and syntax trees. Composing transformations: nesting and chaining. Scope, local definitions and data dependency. Expression evaluation and program execution.
Unit 3: Patterns and regularity	Function definitions. Introduction to Boolean logic. How to design functions. Repetition. Pseudo-random generators. Algorithms. Dealing with exceptions in patterns.
Unit 4: Data and calculations	Data structures for heterogeneous data. Text processing. Calculations of areas. Calculations with integers and dollar amounts. Creating charts.
Unit 5: Models in Space and Time	Functions as models. Encapsulation and generalization. Degrees of Freedom. Dependent and independent variables. Animations as functions of time.