Utah Computer Programming 1 Syllabus

High School - One Semester (60 Contact Hours)

Course Overview and Goals

Utah Computer Programming 1 introduces students to the fundamentals of computer programming, with an emphasis on helping students develop logical thinking and problem-solving skills. Students will learn to design, code, and test their programs while applying mathematical concepts. Students begin with Karel the Dog before moving into Python to work with variables and data types, user input, operators, conditionals, and loops, applying each in projects of their own design and exploring the roles that make up a professional software development team.

Learning Environment

The course utilizes a blended classroom approach. The content is fully web-based, with students writing and running code in the browser. Teachers utilize tools and resources provided by CodeHS to leverage time in the classroom and give focused 1-on-1 attention to students. Each module of the course is broken down into lessons. Lessons may be composed of short video tutorials, interactive learning pages, explorations, simulations, free-response prompts, coding exercises, and quizzes. Each module ends with a comprehensive quiz that assesses students' mastery of that module's material.

Programming Environment

Students write and run Python programs in the browser using the CodeHS editor.

Prerequisites

Utah Computer Programming 1 is designed for high school students with no prior programming experience. Students should be comfortable with basic algebra and typing on a computer. This course is the first semester of a year-long sequence and prepares students for Utah Computer Programming 2.

Technology Requirements

To complete all activities and exercises in this course, students must have access to the 3rd party sites and tools listed here: [insert course whitelist link]

More Information

Browse the content of this course at https://codehs.com/course/28679/explore

Course Breakdown

Module 1: Karel in Python (4 weeks / 20 hours)

In this module, students learn the fundamentals of programming by giving commands to Karel the Dog. They write functions, apply top-down design and decomposition, comment and abstract their code, use for loops, if/else statements, and while loops, and practice systematic debugging.

Objectives / Topics Covered	• Karel Commands • Functions in Karel • Top Down Design and Decomposition • Commenting Your Code • Abstraction • Super Karel and Ultra Karel • For Loops • If and If/Else Statements • While Loops • Debugging Strategies and Algorithms

Example Assignments / Labs	• **Fireman Karel**
	  ○ Students define and use a `turn_right()` function to get Karel down a fireman's pole.
	• **Pancakes**
	  ○ Students write a `make_pancakes()` function and call it repeatedly to deliver stacks of pancakes to guests on multiple avenues.
	• **Random Hurdles**
	  ○ Students combine while loops and conditionals so Karel jumps hurdles only when one blocks the way.
	• **Debug: Big Tower**
	  ○ Students find and fix the errors in a broken Big Tower program so it completes its task.
	• **Super Cleanup Karel**
	  ○ Students decompose an open-ended challenge to clear tennis balls scattered across an unknown world.

Module 2: Basic Python and Console Interaction (2 weeks / 10 hours)

In this module, students write their first Python programs. They print to the console, declare variables of different types, collect user input, build expressions with mathematical and string operators, document code with comments, and compare how programming languages are designed and executed.

Objectives / Topics Covered	• Printing in Python • Variables and Types • Constants and Data Types • User Input • Mathematical Operators • String Operators • Comments • Programming Languages

Example Assignments / Labs	• **Variable Scope Exploration**
	  ○ Students investigate a score-tracking program to see how variables behave inside and outside a function.
	• **Rectangle, Part 3**
	  ○ Students revise an earlier program so the user supplies the length and width, then compute and label the results.
	• **Recipe**
	  ○ Students build a program that collects salad ingredients and quantities from the user and prints a formatted recipe.
	• **Compiled vs. Interpreted Languages**
	  ○ Students compare the characteristics of compiled and interpreted languages and discuss the trade-offs of each.

Module 3: Project: Mad Libs (1 week / 5 hours)

In this module, students apply what they know about variables, input, and string operators to build an interactive Mad Libs game of their own design.

Objectives / Topics Covered	• Program Planning • User Input • String Concatenation • Console Output Formatting
Example Assignments / Labs	• **Project: Mad Libs** ○ Students design a story template with placeholders, prompt the user for different parts of speech, and print the completed story.

Module 4: Conditionals (1 week / 5 hours)

In this module, students add decision-making to their programs. They work with boolean values, if/elif/else statements, comparison and logical operators, and the rounding behavior of floating point numbers.

Objectives / Topics Covered	• Booleans • If Statements • Comparison Operators • Logical Operators • Floating Point Numbers and Rounding

Example Assignments / Labs	• **Positive, Zero, or Negative?**
	  ○ Students use an if-elif-else statement to classify a number the user enters.
	• **Presidential Eligibility**
	  ○ Students collect several facts from the user and use logical operators to report whether the person can run for president.
	• **Network Access**
	  ○ Students use the logical `not` operator to decide whether a device is allowed to join a school network.
	• **Transaction**

	○ Students write a program that processes a customer's deposit or withdrawal for a bank's automated system.

Module 5: Project: Quiz Game (1 week / 5 hours)

In this module, students apply conditional logic to build a multiple-choice quiz game that checks answers and reports a final score.

Objectives / Topics Covered	● Conditional Branching ● User Input Validation ● Score Tracking ● Program Testing
Example Assignments / Labs	● **Project: Quiz Game** ○ Students write a quiz with at least three multiple-choice questions, compare each answer using if statements, track the score, and give the user feedback.

Module 6: Looping (1 week / 5 hours)

In this module, students repeat work with loops. They write while loops and for loops, control loop flow with break and continue, and nest loops and conditionals inside one another.

Objectives / Topics Covered	● While Loops ● For Loops ● Break and Continue ● Nested Control Structures
Example Assignments / Labs	● **Higher/Lower** ○ Students build a guessing game that keeps prompting the user and reports whether each guess is too high or too low. ● **Divisibility** ○ Students extend a starter program so it never divides by zero, using a loop to re-prompt the user. ● **Rolling Dice** ○ Students use nested loops to print every combination of two dice rolls. ● **Categories** ○ Students use nested loops to collect three categories and three items in each, then print the results.

Module 7: Project: Password Authenticator (1 week / 5 hours)

In this module, students apply looping and input validation to build a program that authenticates a user's password.

Objectives / Topics Covered	• Loops and Input Validation • String Comparison • Attempt Limits • User Feedback
Example Assignments / Labs	• **Project: Password Authenticator** ○ Students prompt the user for a password, compare it against the stored value, and loop until the user succeeds or runs out of attempts.

Module 8: Roles in a Software Development Team (1 week / 5 hours)

In this module, students step into the roles that make up a professional software team. They act as software engineer, QA engineer, designer, and project manager, and examine how reliability standards and automation shape the industry.

Objectives / Topics Covered	• Software Engineer • QA Engineer • Designer • Project Manager • Software Reliability • Automation's Impact on Work

Example Assignments / Labs	• **Report the Bug**
	  ○ Acting as a QA engineer, students file a detailed bug report for a defect they found in a web application.
	• **Design a Mood Board**
	  ○ Acting as a designer, students build a mood board with a color palette, fonts, and imagery that represent a store's brand.
	• **Create a Task Board**
	  ○ Acting as a project manager, students break an app launch into tasks and map the dependencies between them.
	• **Reliable Software Ethics Reflection**
	  ○ Students analyze a startup scenario in which testing is cut to hit a deadline and argue what the team should do.