Utah Computer Programming 2 Syllabus

High School - One Semester (90 Contact Hours)

Course Overview and Goals

Utah Computer Programming 2 introduces students to more advanced programming concepts. Students will learn to create more powerful programs using functions, strings, data structures, file i/o operations, and objects. Students apply these skills in individual and team projects, then explore cyber ethics and security, artificial intelligence and prompt engineering, and computer science careers.

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. In the Intro to AI module, students also work with HTML, CSS, and JavaScript while building AI-assisted projects.

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]

Prerequisites

Utah Computer Programming 2 is designed for high school students who have completed Utah Computer Programming 1 or an equivalent introductory programming course. Students should be comfortable writing Python programs that use variables, user input, conditionals, and loops before enrolling. No prior experience with functions, data structures, or object-oriented programming is required.

More Information

Browse the content of this course at https://codehs.com/course/28680/explore

Course Breakdown

Module 1: Functions and Exceptions (1 week / 5 hours)

In this module, students write their own functions. They pass parameters, work with default values, reason about namespaces and scope, return values to the caller, and handle runtime errors with exceptions.

Objectives / Topics Covered	• Functions • Functions and Parameters • Namespaces in Functions • Functions and Return Values • Exceptions

Example Assignments / Labs	• **Area of a Square with Default Parameters**
	  ○ Students write a function with a default parameter value and call it with and without an argument.
	• **Fix the Scorekeeper**
	  ○ Students debug a scoring program whose variables are used outside the scope where they were defined.
	• **Temperature Converter, Part 2**
	  ○ Students write a function that converts Celsius to Fahrenheit and use it on a value the user supplies.
	• **Enter a Positive Number**
	  ○ Students combine functions, loops, and exception handling to keep prompting until the user enters a valid positive number.

Module 2: Strings (1 week / 5 hours)

In this module, students manipulate text. They index and slice strings, discover that strings are immutable, traverse strings with for loops, search with the in keyword, and apply built-in string methods.

Objectives / Topics Covered	• Indexing • Slicing • Immutability • Strings and For Loops • The in Keyword • String Methods

Example Assignments / Labs	• **Initials**
	  ○ Students write a function that returns a person's initials, formatted with periods and no extra spaces.
	• **Replace a Letter**
	  ○ Students write a function that returns a new string with the character at a given index replaced, working around string immutability.
	• **Keeping Count**
	  ○ Students write a function that counts how many times a character occurs in a string.
	• **Remove All From String**
	  ○ Students write and then apply a function that removes every instance of one string from another.

Module 3: Project: The Game of Pig (1 week / 5 hours)

In this module, students plan and build the dice game Pig, then extend it with a computer opponent that follows its own strategy.

Objectives / Topics Covered	• Game Logic • Randomness • Loops and Conditionals • Program Extension

Example Assignments / Labs	• **The Game of Pig**
	  ○ Students plan the rules, scoring, and banking strategy of the game before writing any code.
	• **Build the Basic Program**
	  ○ Students implement die rolls, round scores, banking, and a win condition at 100 points.
	• **Adding a Computer Opponent**
	  ○ Students add an automated opponent that takes turns against the player using its own banking strategy.

Module 4: Creating and Altering Data Structures (1 week / 5 hours)

In this module, students store collections of values. They create and unpack tuples, build and index lists, traverse lists with for loops, and modify lists with built-in list methods.

Objectives / Topics Covered	• Tuples • Lists • For Loops and Lists • List Methods

Example Assignments / Labs	• **Coordinate Pairs**
	  ○ Students complete a `distance` function that takes two coordinate tuples and returns the distance between them.
	• **Max In List**
	  ○ Students write a function that traverses a list of integers and returns the largest value.
	• **Librarian**
	  ○ Students collect five author last names from the user and print them in sorted order.
	• **Word Ladder**
	  ○ Students build a list of words in which each word differs from the previous one by a single letter.

Module 5: Extending Data Structures (1 week / 5 hours)

In this module, students work with more advanced structures. They build and traverse 2D lists, condense loops into list comprehensions, pack and unpack values, and store key-value pairs in dictionaries.

Objectives / Topics Covered	• 2D Lists • List Comprehensions

	• Packing and Unpacking • Dictionaries

Example Assignments / Labs	• **Checkerboard, v3**
	  ○ Students combine two earlier programs to generate a full checkerboard grid stored in a 2D list.
	• **Tic Tac Toe**
	  ○ Students complete the missing pieces of a tic tac toe simulation built on a 2D list.
	• **Slopes**
	  ○ Students collect five coordinate pairs from the user, store them as tuples in a list, and compute the slopes between them.
	• **Word Counts**
	  ○ Students split user text into words and use a dictionary to count how many times each word appears.

Module 6: Project: Guess the Word (2 weeks / 10 hours)

In this module, students build a word-guessing game as a team, in four incremental parts. They agree on ownership and naming up front, keep a development log, and present the finished program to an audience.

Objectives / Topics Covered	• Team Collaboration • Program Decomposition • Incremental Development • Strings and Lists • Project Presentation • Peer Evaluation

Example Assignments / Labs	• **Team Charter**
	  ○ Before writing code, students agree on who owns each piece, how it will be named, and when it is due.
	• **Guess the Word, Parts 1-4**
	  ○ Students build the game in stages, adding letter guessing, dash display, guess limits, and a random secret word.
	• **Present Your Project**
	  ○ Students explain what their program does and how it works to an audience that did not build it.
	• **Peer Evaluation**
	  ○ Students evaluate their teammates' contributions the way a real software team reviews its members.

Module 7: File I/O (2 weeks / 10 hours)

In this module, students read and write external data. They read files by character, by line, and all at once, write and append to files, move the file pointer, and pull live data from an API.

Objectives / Topics Covered	• What is File I/O • Reading Characters from a File • Reading Lines from a File • Reading All Lines from a File • Writing to a File • Moving the File Pointer • File I/O With an API

Example Assignments / Labs	• **Validating Tweet Length**
	  ○ Students write a function that reads a text file and reports whether its contents form a valid tweet.
	• **Update a High Score**
	  ○ Students read a leaderboard file, update a player's score, and write the file back out.
	• **Exploration Inside An API Response**
	  ○ Students navigate a realistic API response made of nested dictionaries and lists to pull out the values they need.
	• **Challenge: Build With Live Data**
	  ○ Students write a program that works against a real API response instead of sample data.

Module 8: Classes and Objects (2 weeks / 10 hours)

In this module, students design their own types. They write classes with constructors and instance variables, add methods, override built-in methods, overload operators, and distinguish class variables from instance variables.

Objectives / Topics Covered	• Classes and Objects • Methods • Built-In Methods • Operator Overloading • Class Variables vs. Instance Variables

Example Assignments / Labs	• **The Rectangle Class, Parts 1-9**
	  ○ Students build one class across nine exercises, adding a constructor, area and perimeter methods, `__repr__`, `__eq__`, operator overloading, and a class variable.
	• **Names In a Hat**
	  ○ Students write a `NameHat` class that stores names and draws one at random.
	• **Contact Merge**
	  ○ Students implement `__add__` for a `ContactBook` class so two contact books can be combined.
	• **Cars, Part 2**
	  ○ Students convert an instance variable into a class variable and reason about which values belong to the class.

Module 9: Ethics and Cybersecurity (1 week / 5 hours)

In this module, students examine the responsibilities that come with writing software. They compare ethics and laws, study common cyber attacks and how to prevent them, protect personal data, evaluate software licenses, and weigh security against usability.

Objectives / Topics Covered	• Cyber Ethics and Laws • Common Cyber Attacks and Prevention • Personal Data Security • Software Licenses • Application Security

Example Assignments / Labs	• **Cyber Ethics Scenarios**
	  ○ Students read a set of scenarios and decide for each whether the action is ethical, and why.
	• **Complete Your Story**
	  ○ Students write a fictional cyber attack narrative using the attack techniques they studied.
	• **Do I Need a Software License?**
	  ○ Students decide what licensing their original phone app requires before selling it.
	• **Security vs. Usability**
	  ○ Students evaluate permission and security settings and argue which trade-offs are worth making.

Module 10: Intro to AI (5 weeks / 25 hours)

In this module, students study artificial intelligence and use it as a tool. They learn to use AI safely, compare generative and predictive AI, explore how large language models work, practice and refine prompts, examine who builds AI and whose biases it carries, and complete an AI-assisted coding project.

Objectives / Topics Covered	• How to Use AI Tools Safely • Human & Artificial Intelligence • Generative vs. Predictive AI • Large Language Models • Prompt Engineering • Prompt Practice & Refinement • Who Builds AI? • AI-Assisted Coding • Project: AI-Assisted Coding

Example Assignments / Labs	• **Explore & Reflect: Test the Intelligence of an AI**
	  ○ Students probe a public chatbot, then evaluate what its answers reveal about machine intelligence.
	• **Compare Your Prompts**
	  ○ Students submit their first and final refined prompts for a scenario and explain what changed and why.
	• **Review the AI's Code**
	  ○ Students inspect AI-written scoring logic for a word game and determine whether it is actually correct.
	• **Your AI Use Statement**

	○ Students write a professional disclosure statement describing exactly how they used AI on their project. ● **Final Draft: Create Your Own AI-Assisted Project** ○ Students plan, build, test, and present an original AI-assisted solution to a real-world problem.

Module 11: Exploring CS Careers (1 week / 5 hours)

In this module, students connect programming to the working world. They survey computer science careers, examine bias in technology, and profile themselves against a career they research and present.

Objectives / Topics Covered	● Computer Science Careers ● Career Research ● Bias in Technology ● Career Exploration Presentation
Example Assignments / Labs	● **CS Career Response** ○ Students research one computer science career using a legitimate source such as the U.S. Bureau of Labor Statistics. ● **Inclusive Coding Response** ○ Students respond to a TEDx Talk on algorithmic bias with examples of programs that show bias. ● **Do Your Research** ○ Students research three careers that interest them, comparing field, required education, and outlook. ● **Career Exploration Presentation** ○ Students present a career profile that ties their hobbies, talents, and personality to a specific career path.