CodeHS

Texas Computer Science I in Python Course Syllabus
1 Year for High School (165 contact hours)

Course Overview and Goals

The CodeHS Texas Computer Science I in Python curriculum fosters students' creativity and innovation by presenting opportunities to design, implement, and present meaningful programs. Students use computational thinking to identify task requirements, plan search strategies, and use computer science concepts to solve problems.

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 unit of the course is broken down into lessons. Lessons consist of video tutorials, short quizzes, example programs to explore, and written programming exercises, adding up to over 100 hours of hands-on programming practice in total. Each unit ends with a comprehensive unit test that assesses student's mastery of the material from that unit as well as challenge problems where students can display their understanding of the material.

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

More information: Browse the content of this course at https://codehs.com/course/27153

Prerequisites: The Computer Science I course is designed for complete beginners with no previous background in computer science. The course is highly visual, dynamic, and interactive, making it engaging for new coders.

Course Breakdown

Unit 1: What is Computing? (5 weeks/25 hours)

Students learn about the history of computing, and about the various parts that make up modern computers. Students also consider the impact computing has had on today's world, and the impacts computing could potentially have in the future.

Objectives / Topics Covered	• Digital information • Number systems • What is a computer? • What is software? • What is hardware? • Software licenses • Future of computing

Example Assignments / Labs	• Encoding data
	  ○ Create your own encoding scheme
	  ○ Encode images using binary
	  ○ Example Activity:

	■ Write a message by encoding the characters in binary, using the ASCII codes. ● Using different number systems ○ Convert numbers between decimal, binary, and hexadecimal ● What is a computer? ○ What parts do modern computers have? ○ What are input devices? ○ What are output devices? ○ Example Activity: ■ Draw a computer and label all of its parts, including the input devices and output devices ● Software/Hardware ○ What's the difference? ○ What hardware components make up a computer? ○ What is software used for? ○ Example Activity: ■ Label the parts of your computer ● Future of Computing ○ Research uses of Artificial Intelligence in use now ○ Research new ways of storing data ○ Example Class Activity: ■ In what ways can we use technology that we couldn't 10 years ago. Are these technological advances helpful or harmful overall?

Unit 2: Introduction to Programming in Python with Karel the Dog (3 weeks/15 hours)

Students learn the basics of programming by giving Karel the Dog commands in a grid world.

Objectives / Topics Covered	● Commands ● Defining vs. calling methods ● Designing methods ● Program entry points ● Control flow ● Looping ● Conditionals ● Commenting code ● Top down design ● Debugging strategies
Example Assignments / Labs	● Program-specific tasks for Karel the Dog ○ Example Exercise: Pyramid of Karel Write a program to have Karel build a pyramid. There should be three balls on the first row, two in the second row, and one in the third row. ● Teach Karel new commands like `turn_right()` or `make_pancakes()` ○ Example Exercise: Pancakes Karel is the waiter. they need to deliver a stack of pancakes to the guests on the 2nd, 4th, and 6th avenue. Each stack of pancakes should have three pancakes. Create a method called `makePancakes()` to help Karel solve this problem. ● Solve large Karel problems by breaking them down into smaller, more

	manageable problems using Top Down Design ○ Example Exercise: The Two Towers In this program, Karel should build two towers of tennis balls. Each tower should be 3 tennis balls high. At the end, Karel should end up on top of the second tower, facing East. ● Using control structures and conditionals to solve general problems ○ Example Exercise: Random Hurdles Write a program that has Karel run to the other side of first street, jumping over all of the hurdles. However, the hurdles can be in random locations. The world is fourteen avenues long.

Unit 3: Digital Citizenship and Cyber Hygiene (7 weeks, 35 hours)

Students learn about Internet etiquette and how to stay safe on the world wide web. They also look at the potential effects of their digital footprints, how to protect information from online risks, and the implications of cyberbullying. Finally, students learn how to find and cite quality resources online.

Objectives / Topics Covered	● Digital Footprint and Reputation ● Privacy and Security ● Information Literacy ● Creative Credit and Copyright
Example Example Assignments / Labs	● Digital Footprint and Reputation ○ Example activities: ■ What is your digital footprint? ■ Are you going to make any changes in what you post on social media? ● Keeping data private and secure ○ Example activities: ■ Test out various passwords on a site ■ Explore Google's privacy policy: What do they know about you? ● Information Literacy ○ Example activities: ■ Create and test search queries ■ Explore evidence for using sources ● Different types of copyright licenses ○ Example activities: ■ Create citations for sources ■ Explore image search tools

Unit 4: Basic Python and Console Interaction (3 weeks/15 hours)

Students learn the basics of programming by writing programs that interact with users through the keyboard.

Objectives / Topics Covered	● Printing ● Variables ● Types ● User Input ● Converting Input Types ● Arithmetic Expressions ● String Operators ● Comments

	• Number Bases • Bitwise Operators

Example Assignments / Labs	• Printing
	  ○ Print messages to the console
	• Variables
	  ○ Create variables of different types, and print them to the console.
	• Types
	  ○ Investigate the types of different variables
	  ○ Convert between types
	• Arithmetic Expressions & Converting Input Types
	  ○ Age in One Year - Ask the user how old they are, and tell them how old they will be in one year
	  ○ Rectangle, part 1 - Make variables for length and width and compute area and perimeter
	  ○ Rectangle, part 2 - Ask the user for length and width and compute area and perimeter

Unit 5: Conditionals (2 weeks/10 hours)

Students teach their programs to make decisions based on the information it receives.

Objectives / Topics Covered	• If Statements • Boolean Values • Logical Operators • Comparison Operators • Floating Point Numbers and “Equality”

Example Assignments / Labs	• If statements and boolean values
	  ○ Is it raining? - Write a program that uses a boolean variable to determine whether or not it is raining
	• Boolean operators, and expressions
	  ○ Boolean variable - Take a variable and use it in an if statement
	  ○ Legally allowed to vote - User reports age and the program tells them whether or not they can vote in the US
	  ○ Transaction - The user reports balance and deposit/withdrawal, and the program prints a new balance or error
	  ○ Recipe - Ask the user for ingredients, amounts per serving, and number of servings, and report the total amount of each ingredient needed

Unit 6: Looping (2 weeks/10 hours)

Students learn how to write more efficient code by using loops as shortcuts.

Objectives / Topics Covered	• While Loops • For Loops • Break and Continue • Nested Control Structures • Else Clauses
Example Assignments / Labs	• While Loops ○ Divisibility - Ask the user to enter a numerator and denominator, and

	re-prompt until the denominator is non-zero ● For Loops ○ Average test score - Compute the average of several test scores ● Break and Continue ○ Higher/ Lower - Ask the user to guess a particular number between 1 and 100. If the user's guess was too high or too low, they should be notified ● Nested Control Structures ○ Rolling Dice - Print out all combinations that can be made when 2 dice are rolled

Unit 7: Functions and Exceptions (1-2 week/5-8 hours)

Students learn how to decompose problems into smaller pieces that work together to solve a problem.

Objectives / Topics Covered	● Functions ● Namespaces ● Parameters ● Return Values ● Recursion ● Exceptions
Example Assignments / Labs	● Example exercises: ○ Functions ■ Raining cats and dogs - Write functions to print text art of a cat and a dog ■ Temperature converter - write functions to convert from Fahrenheit to Celsius and vice versa ○ Exceptions ■ Temperature converter, part 2 - Add exception handling to your temperature conversion program ○ Putting it all together ■ Enter a positive number - Make a function to repeatedly ask the user to enter a number until they enter a positive number

Unit 8: Strings (1-2 weeks/5-8 hours)

Students learn more sophisticated strategies for manipulating text in their programs.

Objectives / Topics Covered	● Indexing and Slicing ● Math Operators on Strings ● For Loops Over a String ● String Methods
Example Assignments / Labs	● Example exercises: ○ Indexing ■ First character - write a function that takes a string and returns the first character ■ All but the first character - write a function that takes a string and returns everything but the first character ○ Math operators and strings ■ Full name - write a function that takes two strings (a first name and a last name) and returns a full name as a single

	string ■ Replace a letter - write a function that takes a string and returns a copy with the character at a particular index replaced with a dash ○ For loops on strings ■ Count occurrences - write a function that takes two strings and returns the number of times the second string appears in the first string ○ String methods ■ Add enthusiasm - write a function that takes a string and returns that string in all upper case ■ Remove all from string - write a function that takes two strings and returns a string that consists of the first string with all instances of the second string removed

Unit 9: Creating and Altering Data Structures (1-2 weeks/5-8 hours)

Students learn how tuples and lists are formed and the various methods that can alter them.

Objectives / Topics Covered	● Tuples ● Lists ● For Loops and Lists ● List Methods
Example Assignments / Labs	● Example exercises: ○ Tuples ■ Cookout Orders - Given a tuple of food orders, add up the number of burgers and hotdogs and print the total sums. ○ Lists ■ Listed Greeting - Ask a user to enter their name, age, and favorite sport, then split their response into list elements and use index values to greet them by name and respond that you enjoy that sport as well! ■ Exclamat!on Po!nts - Ask the user for a string and then print the same string with every lowercase i replaced with an exclamation point. ■ Librarian - Ask the user for the last names of the authors of the five books they are returning. Print a list of those names in sorted order.

Unit 10: Extending Data Structures (1-2 weeks/5-8 hours)

Students learn to build more complex programs that make use of grids and dictionaries.

Objectives / Topics Covered	● Dictionaries ● 2d and 3d Lists ● List Comprehensions ● Packing and Unpacking ● Mutable vs. Immutable
Example Assignments / Labs	● Example exercises: ○ Dictionaries ■ Phone book - user repeatedly enters their name, and the

	program either asks for the person's phone number or reports the phone number already provided ○ 2d lists ■ Checkerboard - write a program that prints the initial setup of a checkerboard, with a 1 where a piece would be and a 0 where a blank square would be

Unit 11: Python Graphics and Objects (1-2 weeks/8-10 hours)

Students learn about the canvas and creating graphics using Python. Students are also introduced to objects in Python.

Objectives / Topics Covered	● Creating lines ● Adding basic shapes ● Creating graphics with loops ● Using functions with graphics ● Classes and objects ● Object methods
Example Assignments / Labs	● Example exercises: ○ Lines ■ Tic Tac Toe Board - Draw two horizontal and two vertical lines on the canvas to create a Tic Tac Toe board. Create a canvas that is 250x250px and then center your board, leaving space around the outside. ○ Shapes ■ Flag of [your choice]! - Choose any country and design their flag using Python Graphics! ○ Classes and objects ■ The Rectangle Class, Part 1 - Make a class that represents a rectangle. The class should be called Rectangle, and it should have two instance variables that you set in the __init__ method - length and width.

Unit 12: File I/O (1-2 weeks/5-8 hours)

Students learn to read, write, and process information from text files.

Objectives / Topics Covered	● Reading from Files ● Writing to Files ● Processing File Data
Example Assignments / Labs	● Example exercises: ○ Reading from Files ■ Validating Tweet Length - Write a function called that reads the contents of a text file tweet.txt and determines whether the text represents a valid tweet. ○ Write to Files ■ Activity Tracker - Imagine you are building an activity tracker program. Your task is to write a program that logs a list of activities to a file.

Unit 13: Roles in a Software Development Team (1 week/5-6 hours)

Students learn the key roles and responsibilities of members of a software development team.

Objectives / Topics Covered	• Software Engineers • Quality Assurance Engineers • Designers • Project Managers

Example Assignments / Labs	• Create a Mood Board
	  ○ In this assignment, you will act as a designer and create a mood board for a store of your choosing. To visually represent the brand and theme of the store, your mood board must include the following:
	  ■ 1. A color palette that best represents the store's brand
	  ■ 2. One or two fonts that align with the store's identity
	  ■ 3. Images related to the store's products or target audience
	• Create a Task Board
	  ○ Imagine that you are a Project Manager. Before assigning work to members of the software development team, you need to create a list of tasks needed to create an application for the store you created a mood board for in the previous lesson.

Unit 14: Final Project (2 weeks/10 hours)

Students learn about what makes an engaging and accessible user interface, and will employ an iterative design process including rapid prototyping and user testing to design and develop their own engaging projects.

Objectives / Topics Covered	• Collaborative Programming • Project Planning • Pseudocode • Prototype • Testing

Example Assignments / Labs	• Collaborative open-ended final project which encourages creativity
	• Program Requirements:
	Your program:
	  ○ must utilize mouse interaction from the user
	  ○ must use at least one timer
	  ○ must break down the program into multiple functions
	  ○ must utilize control structures where applicable

Unit 15: Computer Science Careers (2 weeks/10 hours)

Students learn about a variety of computer science careers and organizations, and what the next steps could look like for them if interested.

Objectives / Topics Covered	• Careers and internships • CS career preparation • Legal and ethical responsibilities • Workplace readiness
Example Assignments / Labs	• Exploring computer science careers, internships, and organizations • Learning about CS resumes and certifications

	● Researching about a major ethical or legal topic in CS ● Reflecting on what it means to be a leader and the skills required to be successful in the workplace