# Introduction to Computer Science (CS) Syllabus

Supported by the Microsoft TEALS Program

**Period/Hour/Section Number:**

**Period/Hour/Section Times:**

**Classroom Number:**

## Instructors

Classroom Teacher:

Volunteer Teacher:

Volunteer Teacher:

Volunteer Teacher Assistant:

Volunteer Teacher Assistant:

## Classroom Contact Information

Phone:

Email:

Remote/Virtual Classroom Information:

## School Mission Statement

We strive to………

## Course Description

The TEALS intro course uses Microsoft MakeCode Arcade, starting in a blocked-based programming environment and transitioning to a text-based programming language (either JavaScript or Python).

Computing has changed the world in profound ways: it has opened wonderful new ways for people to connect, design, research, play, create, and express themselves. However, using the computer is just a small part. This course is an introductory programming course that helps prepare students for more advanced programming courses. The TEALS Intro to Computer Science uses Microsoft MakeCode Arcade, an approachable and visual programming environment with a robust tool set, perfect for introducing students to code for the first time. MakeCode Arcade is taught with its blocks-based environment as a single semester course or during the first semester of the full year course. In the year-long option, students will benefit from exposure to MakeCode Arcade’s built-in JavaScript or Python programming environment. The course is A-G approved for University of California credit for high school students in California.

## Prerequisites

Designed for students with algebra readiness skills. No prior programming experience is required.

## Technology Requirements

A desktop or laptop computer must be made available to each student for use during class. Students also will need access to the MakeCode Arcade environment, available at https://arcade.makecode.com. Some of MakeCode Arcade’s functionality requires access to additional web-based resources. Refer to the MakeCode Arcade FAQ list for additional information.

## Required Materials

* Notebook
* Pencils / Pens
* Headphones or Earbuds

## Curriculum Plan: Semester 1

### Unit 0: Beginnings

|  |  |
| --- | --- |
| Lesson | Objectives |
| 0.1: The First Day | Identify the class they are taking. List the high-level goals of the course. Describe classroom procedures, rules, and norms. |
| 0.2: Algorithms | Define algorithm. Construct algorithms for performing simple tasks. |
| 0.3: Programming Languages | Complete an Hour of Code activity in MakeCode Arcade. Explain why computer programs are written in specialized languages. |
| 0.4: MakeCode Arcade Exploration | Identify the major areas of the MakeCode Arcade programming environment. |
| 0.5: MakeCode Arcade Coordinate System | Learn about the coordinate system used in MakeCode Arcade. |
| 0.6: Getting to Know You | Create a simple program in MakeCode Arcade to describe themselves. |

### Unit 1: Sprites

|  |  |
| --- | --- |
| Lesson | Objectives |
| 1.1: Introduction to sprites | Explain sprites and their roles in MakeCode Arcade. Create sprites in the sprite editor. |
| 1.2: Sprites, sprites, and more sprites! | Place sprites on the screen. Work with different kinds of sprites. |
| 1.3: Sprite movement | Move hero sprite with the d-pad. Explain sprite velocity (*vx* and *vy*). Make sprites follow each other. |
| 1.4: Collisions | Use collision event handlers. Use built-in variables for life/health and score. Randomize sprite placement. |
| 1.5: Collisions continued | Detect collisions between different kinds of sprites in the game. Destroy a sprite with effects. Set and update player lives. |
| 1.6: Unit project | Build your first project! |
| Culture Day | Connect unit topics with current events |

### Unit 2: Event Handlers and Variables

|  |  |
| --- | --- |
| Lesson | Objectives |
| 2.1: Events | Create event handlers |
| 2.2: Variables | Create and use variables in programs, accept numeric input from the player |
| 2.3: Variable math | Use arithmetic operators and combine them into complex formulae |
| 2.4: Text variables | Use variable arithmetic to display sprites in a pattern, accept string input from the player, join multiple text values |
| 2.5: Conditionals | Use conditional statements |
| 2.6: Conditionals continued | Use conditional statements |
| 2.7: Unit project | Build a complex game |
| Culture Day | Connect unit topics with current events |

### Unit 3: Loops and Arrays

|  |  |
| --- | --- |
| Lesson | Objectives |
| 3.1: Definite loops + debugging | Use definite loops, use debugging strategies |
| 3.2: Indefinite loops | Use indefinite loops, validate user input |
| 3.3: Arrays and lists | Use arrays and lists |
| 3.4: Animations | Use animations, describe frame-based animation |
| 3.5: Unit project | Create a complex game that uses loops, arrays, and/or animations |
| Culture Day | Document learning journey |

### Unit 4: Functions

|  |  |
| --- | --- |
| Lesson | Objectives |
| 4.1: Functions | Define and create functions |
| 4.2: Functions with parameters | Create functions that use parameters; use functions to create game levels |
| 4.3: Functions with return values | Create functions that return values; create code that represents a deck of cards |
| 4.4: Searching | Write a sequential search algorithm; define *binary search*. |
| 4.5: Unit project | Create a complex game that uses functions |
| Culture Day | Identify the different types of roles and skills needed both within and outside the technology industry; reflect on how technology could be part of their future careers. |

### Unit 5: Tile Maps and Platform Games

|  |  |
| --- | --- |
| Lesson | Objectives |
| 5.1: Tile Maps | Create and use tiles and tile maps. Use placeholders to add animated sprites, food, and enemies to a map. |
| 5.2: More Tile Maps | Use events with tile maps. Work with MakeCode Arcade extensions. Switch among different tile maps. |
| 5.3: Platformers | Use tile maps to create platform games. |
| 5.4: Advanced Platformers | Implement additional jumping and attack mechanisms in platform games. |
| 5.5: Unit Project | Create a complex game that uses tile maps. |
| Culture Day |  |

### Unit 6: Capstone Project

|  |  |
| --- | --- |
| Lesson | Objectives |
| 6.1: Introduction and brainstorming | Work with a team to generate ideas; explore brainstorming tools for future use |
| 6.2: Project planning | Create project specifications and implementation plan |
| 6.3: Project implementation | Implement a complex project; review and refine implementation plan |
| 6.4: Project sharing | Prepare and present a marketing pitch; critically evaluate the design process |

## Curriculum Plan: Semester 2

### Unit 0: A Return to Blocks

|  |  |
| --- | --- |
| Lesson | Objectives |
| 0.1: Revisiting sprites | Identify the class they are taking. List the high-level goals of the course. Describe classroom procedures, rules, and norms.  Explain sprites and velocity. Use built-in variables. |
| 0.2: Revisiting conditional statements | Create event handlers. Create and manipulate variables. Accept input from player. Use conditional statements. |
| 0.3: Revisiting loops and arrays | Create iteration structures. Explain and use arrays. Create pictographs. |
| 0.4: Revisiting functions | Create and use functions. Create a sequential search algorithm |
| 0.5: Revisiting tile maps | Create and use tile maps |

### Unit 1: Introducing JavaScript and Python

|  |  |
| --- | --- |
| Lesson | Objectives |
| 1.1: Introduction to JavaScript / Python | Switch between Blocks and a typing language in MakeCode Arcade. Add code from the toolbox. Create new sprites with a typing language. Identify syntax errors. |
| 1.2: Strings | Create string variables and string literals. |
| 1.3: Unit project | Present a silly story. |
| Culture day: Binary day | Describe different representations of data |

### Unit 2: Core Programming Structures

|  |  |
| --- | --- |
| Lesson | Objectives |
| 2.1: Event handlers in depth | Create event handlers |
| 2.2: Conditionals | Use logical operators and decision structures |
| 2.3: Loops | Use definite (for and foreach) and indefinite (while) loops |
| 2.4: Lists | Use arrays with loops |
| 2.5: Revisiting animations | Implementing animated sprites with a typing language |
| 2.6: Unit project | Create a complex project |
| Culture day |  |

### Unit 3: Functions

|  |  |
| --- | --- |
| Lesson | Objectives |
| 3.1: Revisiting functions | Create functions with parameters and return values |
| 3.2: Variable scope | Create local variables; compare with global variables |
| 3.3: Searching and sorting | Use basic search and sort mechanics |
| 3.4: Melodic diversion | Create simple melodies |
| 3.5: Unit project | Create a complex project |
| Culture day: Sorting algorithms | Investigate sorting algorithms |

### Unit 4: Introduction to Object-Oriented Programming

|  |  |
| --- | --- |
| Lesson | Objectives |
| 4.1: Introduction to OOP and inheritance | Create child classes of the Sprite class |
| 4.2: Class methods | Add custom methods to Sprite subclasses |
| 4.3: Unit project | Create a complex project using Sprite subclasses |
| Culture day |  |

### Unit 5: Tilemaps and Platformers (might be optional)

|  |  |
| --- | --- |
| Lesson | Objectives |
| 5.1: Revisiting tilemaps | Create and use tilemaps |
| 5.2: Revisiting platformers | Use physics with tilemaps to implement platform games |
| 5.3: Unit project | Create a complex project using tilemaps |
| Culture day |  |

### Unit 6: Advanced graphics (optional)

|  |  |
| --- | --- |
| Lesson | Objectives |
| 6.1: Backgrounds and parallax | Add depth to backgrounds with parallax |
| 6.2: Additional advanced graphics concepts | Single-point perspective? Other ways to simulate depth? |
| 6.3: Unit project | Create a complex project using advanced graphics techniques |
| Culture day |  |

### Unit 7: Capstone Project

|  |  |
| --- | --- |
| Lesson | Objectives |
| 7.1: Introduction and brainstorming | Work with a team to generate ideas; explore brainstorming tools for future use |
| 7.2: Project planning | Create project specifications and implementation plan |
| 7.3: Project implementation | Implement a complex project; review and refine implementation plan |
| 7.4: Project sharing | Prepare and present a marketing pitch; critically evaluate the design process |

## Grading Policy

* 96-100%=A+
* 93-95%= A
* 90-92%= A
* 86-89%= B+
* 83-85%= B
* 80-82%= B
* 76-79%= C+
* 73-75%= C
* 70-72%= C
* 66-69%= D+
* 63-65%= D
* 60-62%= D
* 59-lower%= E
* Discuss the established grading policy and classroom procedures provided by the classroom teacher in the School & Class Overview section to make sure everyone on the team understands them.
* If school policy allows for flexibility, discuss how the team could apply some of the strategies they learned in summer training to the Grading Policy that the team will follow.
* For each classroom action on that list where the teacher has not yet documented a procedure, use what they have learned in summer training to collaboratively develop one. Remember that classroom procedures need to be taught and rehearsed by students for them to save time and increase focus.

## Classroom Procedures

**Entering the room**: Enter quietly and politely; remove your hat if you're wearing one; don't interrupt other students; follow the appropriate procedures for each time of day (e.g., morning, after lunch, after a special class).

**Leaving the room**: Tell me where you are going; take the correct hall pass; do not run or play in the hallways or restrooms.

**Ending the day**: Clean off your desk; leave out your work notebook; pick up any trash within three feet of your desk; stack your chair; collect your mail; wait quietly to be dismissed.

**Obtaining help with assignments**: Quietly ask the students at your table for help with directions if you need it; if you are working alone, raise your hand to get help from me; if you are working with a group, ask them for help in understanding how you complete the assignment.

**Handing in finished work/homework**: [Enter your submission procedure for assignments]

**What to do with unfinished work**: If I ask for work to be turned in, let me know if it isn't finished; if I ask you to keep an unfinished project, put it in your class work notebook.

**When and how to use the school restroom**: If I am not teaching the whole group, stand by the classroom door with your hand raised; if I say "no," wait for a better class time to go; if I nod, leave the room quietly; do not play in the restroom; return to class before two minutes have passed (promptly).

**When and how to use the drinking fountain or sink**: When I am not teaching the whole group, you may get a drink; take only a three-second drink; you may bring a water bottle to keep on your desk; if you need to wash your hands, use only a little soap; wipe up any water you spill

**Getting into work groups**: Take all the materials you will need; greet each other; complete the task doing your personal best; make sure each person signs the project; thank the others in your group.

**Signals for attention**: When I need your attention, I will ring the chimes (or sound the rain stick, open the music box, etc.); as soon as you hear the signal, stop what you are doing, look at me, and listen for directions.

**Helping other students**: In a cooperative classroom, it is good to help one another; if someone needs help with directions or reading an assignment, help him or her if you are able; if someone needs help with understanding the problem, tell him or her to ask me for help; never put down another student who asks for help.

## CS Culture Days

TEALS encourage classes to implement CS Culture Days, taking a break from normal lessons and activities to connect the academic course content to real world applications. TEALS provide several lesson plans to illustrate how to run a culture day. They can include “show and tell” by the instructors, or topics researched and presented by students.

* How often will the team hold culture days?
* What are some topics the team would like to share with the class?
* Based on what the team collectively knows about the students, what topics might the students want to learn about?
* How will the team tie current events in computer science into the classroom?

## Assessment & Grading

* 20% = Notebooks and Daily Participation
* 40% = Homework and Daily Labs
* 40% = Quizzes and Projects

## Semester Grade

* 40% Quarter 1
* 40% Quarter 2
* 20% Semester Exam (cumulative)

## Expectations

**Bell Ringer**: The Bell Ringer is to be completed within the first 3-5 minutes of class in your notebook or online (see instructions for that day). You should begin it immediately and work on it silently (unless instructed otherwise). Your notebooks and/or binders will be collected and graded toward your class participation grade.

**Notebooks**: Taking notes on paper results in better learning outcomes for students. In computer science, notebooks can be used to record definitions, syntax, programming patterns and idioms, examples, and diagrams. Students can also reflect on the work they are doing and use the notebook as a scratch space to plan out their approach to problems before implementation. Notebooks help make learning more explicit and help students to organize and process new information.

* When should students use their notebooks in the class?
* How often will the team check the notebooks? Who checks them and when? Is there a grade associated with notebook completeness?
* When can and should students refer to their notes (during lab? On quizzes? On tests?)
* What should students do about class notes when they miss class?

**Notebook Checks**: It is extremely important that you keep all your course materials for the entire semester. You will have a Binder and Notebook check from time to time during the semester. All in-class assignments are “fair game” as well as all Do Now that we have done (including Do Now that took place when you were absent). Stay organized and complete all work to receive full credit!

**Cheating**: Academic honesty will be emphasized in the computer science classroom. Cheating by supplying or copying answers will result in the following of the procedure laid out in the Student Handbook found in your Student Agendas. We will discuss when collaborative group work is encouraged and when it is not. Note, please follow this procedure when you want to ask an instructor to help you:

1. Ask a classmate for help.
2. Try something!
3. Raise your hand and we will ask, “Who have you asked for help? What have you tried?”

**Absences**: If you are absent, you are still responsible for the material that was covered that day - all notes, handouts, homework, quizzes, tests, etc. Ask other students or the teacher before or after school NOT during class! There will be an “Absent” folder in which you will find any handouts with your name already on them. All quizzes and tests must be made up within one week of your absence or they will become a zero. **Making up tests and quizzes must be done before or after school or during a free block. Please see us to set up an appointment to make up your missing test or quiz.**

**Late Work**: Any late work that is being turned in will only be accepted for that unit. Once a unit test or project is complete, late work from that unit will no longer be accepted. For every class period that an assignment is late (if not from an absence) 10% will be taken off the assignment. All assignments turned in after 5 class periods will receive half-credit.

**Our responsibilities to you**:

* We will treat you with respect.
* We will praise your successes and assist you when you are having difficulties.
* We will help you learn by providing thoughtful and meaningful instruction.
* We will make ourselves available after school if you require extra help.

# Intro to CS - Student and Parent Syllabus Contract & Contact Information (Please print neatly)

We, student and parent/guardian, have read, understand, and agree to all the policies, procedures, and expectations outlined in this Intro to Computer Science Syllabus.

Student Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Student Signature Date

Parent/Guardian Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Parent/Guardian Signature Date

## Parent/Guardian Contact Information

Cell Phone Number: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Work Phone Number: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

E-mail address: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Best time of day to reach parents/guardians: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Does your student have a computer and internet access at home? (Yes / No)

\*Not required but may be helpful for some assignments.