Go To Section

Project: speedy math, adventure game, or personality test (if/else logic practice)

Project: mid-unit mini app (drawing application, flappy bird?, applying concepts up to lists)

Project: final project (create your own game applying cumulative concepts)

[L4 Computational Thinking](#u2ftmafwv91i)

[L5 Getting Started with Thumby, Computers, Coding, CS, Clean Code, Commenting](#qsvqmxg5azga)

[L6 Variables, Data Types, Boolean Expressions, Operators, Conditionals, Random Function](#44mdl7em3n9)

[L7 Input/Output](#cyfsym9wd7pc)

[L8 Collaborative Coding, Debugging](#i0uviaafu66)

[L9 Code Review, Playtesting](#7pncq17sh7tg)

[L10 Loops](#rxcmtt5cxuyk)

[L11 Lists](#exe95f8sva0u)

[L12 Functions](#4xd25cwq2le8)

[L13 Objects](#3b10obi1kht6)

[L14 Dictionaries](#qgra0si1gwns)

[L15 Libraries, Version Control, Open Source, Online Forums, Other Tools and Resources](#gheucdlhe964)

L4 Computational Thinking

# Introduction

This lesson covers the 4 steps of computational thinking in and outside a CS context by posing problems and having students solve them using a computational framework.

# Lesson Objectives

## Students will:

* Be able to rephrase tasks/actions using code and computational thinking
* be able to apply computational thinking to problems outside the context of CS to better understand problems and the steps needed to reach a solution

## Standards covered:

**CSTA 3A-IC-26**

Demonstrate ways a given algorithm applies to problems across disciplines.

## Higher-order thinking skills:

**Computational Thinking**

Students apply the steps of computational thinking to various problems, learning how to solve any problem computationally.

# Instructional Strategies

**Small group discussions**

# Time

**40 mins:** I want to bake a cake, Siren Beat in pseudocode, Dress up your sim in pseudocode, Choose a community issue and design a solution using computational thinking

**Additional activities:** continue working out steps to solving the community issue

# Materials

* Pen and paper

# Pre-lesson Homework

* Read how does a computer think
* Read computational thinking: what is it? Why is it important? Difference between computational thinking, coding, and CS
* Read BBC computational thinking article

# In Class

## Warm-up Activity

**I want to bake a cake**

Students are prompted with a problem: the teacher wants to bake a cake. Students need to help the teacher successfully bake a cake.

Students form groups. They are led down the 4 steps of computational thinking to break down and come up with a solution for this problem (solution: write a cake recipe).

Regroup as a class, share recipes, and identify emerging patterns. Briefly recap computational thinking.

Next, students use the computational thinking framework again to think about how a computer would write this recipe. Provide students with an example list of pseudocode that demonstrates the capabilities of coding and how it represents real life actions. Ask students to recreate their cake recipe in pseudocode.

## Slides

**NA**

## Activity

**Siren Beat in pseudocode**

**Dress up your Sim in pseudocode**

**Choose a community issue and design a solution using computational thinking**

Students are provided a list of issues relevant to their community and guided through computational thinking to plan next steps to solve the problem. Activity guidelines will touch upon various components of public problem solving and ask students to use those components to build an action plan to get closer to a solution.

# Homework

* Watch video on Thumby capabilities
* Watch video on how to use the IDE
* Read hardware to software
* Read what is coding
* Read commenting framework
* Read coding best practices
* Read journaling framework
* Read code review framework

# Examples

# Resources

L5 Getting Started with Thumby, Computers, Coding, CS, Clean Code, Commenting

# Introduction

This lesson covers definitions of coding/programming and CS, getting started with the Thumby IDE, a walk through tutorial of how to use all its features, and pseudocode best practices.

# Lesson Objectives

## Students will:

* Be able to create a new project in the Thumby IDE
* Understand how to use and apply all Thumby IDE features to their future coding projects
* Understand what a computer is
* Understand how basic hardware connects to software
* Understand what coding is
* Understand what a program is
* Understand what CS is
* Be able to write clean code
* Be able to write clear pseudocode

## Standards covered:

**CSTA 3A-CS-01**

Explain how abstractions hide the underlying implementation details of computing systems embedded in everyday objects.

**CSTA 3A-CS-02**

Compare levels of abstraction and interactions between application software, system software, and hardware layers.

**CSTA 3B-AP-23**

Evaluate key qualities of a program through a process such as a code review.

**k12CS 9–12.Computing Systems.Devices**

Devices. Recognizing computing devices and assessing the design of these devices for usability, dependability, security, and accessibility.

**k12CS 9–12.Computing Systems.Hardware and Software**

Hardware and Software. Understanding that there are levels of interaction between hardware, software, and the user. We most commonly use system software and applications. System software controls the flow of information between hardware components used for input, output, storage, and processing.

**AP CSP 1.2 Program Function and Purpose**

## Higher-order thinking skills:

**Communication**

Students share ideas about hardware, software, computers, coding, and computer science in the warm up discussion, learning to communicate coding concepts to peers in a clear, understandable way, and learning to reinforce concepts by listening to others’ perspectives.

Students read and internalize the commenting framework and coding best practices, learning to communicate their coding thought processes to code collaborators.

# Instructional Strategies

**Small group discussion**

**Follow along tutorial**

**Reflection journaling**

# Time

**40 mins:** warm-up discussion, IDE walkthrough, reviewing commenting framework, reviewing coding best practices

**Additional activities:** set up reflection journal, begin homework, work on practice exercises in homework in pairs

# Materials

* Laptops with access to the internet for each student
* Thumby for each student
* MicroUSB cable
* Projector/TV/Display

# Pre-lesson Homework

* Watch video on Thumby capabilities
* Watch video on how to use the IDE
* Read hardware to software
* Read what is coding
* Read commenting framework
* Read coding best practices
* Read journaling framework
* Read code review framework

# In Class

## Warm-up Activity

**What is coding warm-up discussion**

Discuss what are computers/coding/programming/CS, why we use it, its capabilities, and how you can apply those capabilities (coding applications).

## Slides

**Thumby IDE step by step tutorial using sample project**

Walkthrough how to use the Thumby IDE with a sample project. Pause between slides to check for understanding. Walk around and have peers help each other troubleshoot problems. By the end of the presentation, students should be able to independently use the IDE.

## Activity

**NA**

# Homework

* Create reflection journal on Medium
* Read variables/data types lesson
* Read boolean expressions lesson
* Read operators lesson
* Read statements/conditionals lesson
* Read function random lesson
* Read Thumby Link lesson

# Examples

* Getting started with Thumby IDE sample project

# Resources

* Hardware to software
* What is coding
* Commenting framework
* Coding best practices
* Journaling/documentation framework
* Code review framework
* Thumby IDE tutorial slides
* Key coding vocabulary sheet

L6 Variables, Data Types, Boolean Expressions, Operators, Conditionals, Random Function

# Introduction

This lesson covers variables, data types, boolean expressions, operators, conditionals, and the random function by having students design a buzzfeed quiz (if/else game logic).

# Lesson Objectives

## Students will:

* Understand how to use variables, data types, boolean expressions, operators, conditionals, and the random function to create Thumby games

## Standards covered:

**CSTA 3A-AP-15**

Justify the selection of specific control structures when tradeoffs involve implementation, readability, and program performance, and explain the benefits and drawbacks of choices made.

**CSTA 3A-AP-22**

Design and develop computational artifacts working in team roles using collaborative tools.

**CSTA 3A-IC-27**

Use tools and methods for collaboration on a project to increase connectivity of people in different cultures and career fields.

**k12CS 9–12.Algorithms and Programming.Control**

Control. (conditional statements, loops, event handlers, recursion) Programmers consider tradeoffs related to implementation, readability, and program performance when selecting and combining control structures.

**AP CSP 1.1 Collaboration**

**AP CSP 3.1 Variables and Assignments**

**AP CSP 3.3 Mathematical Expressions**

**AP CSP 3.4 Strings**

**AP CSP 3.5 Boolean Expressions**

**AP CSP 3.6 Conditionals**

**AP CSP 3.7 Nested Conditionals**

**AP CSP 3.15 Random Values**

## Higher-order thinking skills:

**Computational thinking**

Students analyze and find a solution for the buzzfeed quiz problem by following the steps of computational thinking, learning to apply computational thinking to solving problems.

Students diagram a chart before coding their buzzfeed quiz, applying decomposition, pattern recognition, abstraction, and algorithmic thinking.

**Collaboration**

Students engage in pair programming on the buzzfeed quiz in-class activity, learning to code simultaneously with a peer.

# Instructional Strategies

**Flipped classroom learning**

**Pair programming**

**Project based learning**

# Time

**40 mins:** pair programming

**Additional activities:** start programming buzzfeed quiz or start homework

# Materials

* Laptops with access to the internet for each student
* Thumby for each student
* Pen and Paper
* MicroUSB cable
* Thumby Link cable

# Pre-lesson Homework

* Create reflection journal on Medium
* Read variables/data types lesson
* Read boolean expressions lesson
* Read operators lesson
* Read statements/conditionals lesson
* Read function random lesson
* Read Thumby Link lesson

# In Class

## Warm-up Activity

**NA**

## Slides

**NA**

## Activity

**Pair programming on buzzfeed quiz part 1**

Students pair up on an unplugged exercise to analyze their buzzfeed quiz step by step using computational thinking. Students further apply computational thinking by diagramming their if/else logic for their buzzfeed quiz.

# Homework

* Read input/output lesson

# Examples

* Text
* Import image
* Sound
* Simple choose your path game (using computational thinking to frame the problem, with if/else statements charted)

# Resources

* Pair programming guidelines
* MicroPython cheat sheet

L7 Input/Output

# Introduction

This lesson covers input and output and applies those concepts in part 2 of the buzzfeed quiz pair programming in-class activity.

# Lesson Objectives

## Students will:

* Understand how to receive user input on Thumby and output text, sound, and images.
* Be able to apply input/output to creating their own Thumby games

## Standards covered:

**CSTA 3A-AP-16**

Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue by using events to initiate instructions.

**CSTA 3A-AP-19**

Systematically design and develop programs for broad audiences by incorporating feedback from users.

**CSTA 3A-AP-21**

Evaluate and refine computational artifacts to make them more usable and accessible.

**CSTA 3A-AP-22**

Design and develop computational artifacts working in team roles using collaborative tools.

**CSTA 3A-AP-23**

Document design decisions using text, graphics, presentations, and/or demonstrations in the development of complex programs.

**CSTA 3A-IC-27**

Use tools and methods for collaboration on a project to increase connectivity of people in different cultures and career fields.

**CSTA 3B-AP-21**

Develop and use a series of test cases to verify that a program performs according to its design specifications.

**AP CSP 1.1 Collaboration**

**AP CSP 1.2 Program Function and Purpose**

**AP CSP 1.3 Program Design and Development**

**AP CSP 3.3 Mathematical Expressions**

**AP CSP 3.6 Conditionals**

**AP CSP 3.7 Nested Conditionals**

## Higher-order thinking skills:

**Collaboration**

Students engage in pair programming on the buzzfeed quiz in-class activity, learning to code simultaneously with a peer.

# Instructional Strategies

**Flipped classroom learning**

**Pair programming**

**Project based learning**

# Time

**40 mins:** pair programming

**Additional activities:** start homework

# Materials

* Laptops with access to the internet for each student
* Thumby for each student
* MicroUSB cable
* Thumby Link cable

# Pre-lesson Homework

* Read input/output lesson

# In Class

## Warm-up Activity

**NA**

## Slides

**NA**

## Activity

**Pair programming on buzzfeed quiz part 2**

Students code their buzzfeed quiz, applying variables, data types, boolean expressions, operators, conditionals, and input/output.

# Homework

* Project: speedy math, adventure game, or personality game.
* Journal entry

# Examples

* Line
* Triangle
* Rectangle
* Circle
* Planet
* Trapezoid
* Simple Draw
* Text
* Import Image
* Sound
* Simple choose your path game
* A/B input example
* Dpad example
* Speedy math game initial scaffolding (using computational thinking to frame the problem and charting out the if/else logic)

# Resources

* Pair programming guidelines
* Code review framework
* Intro to 2D graphics

L8 Collaborative Coding, Debugging

# Introduction

This lesson covers easier and more productive ways to code collaboratively, including providing workflows for combining code from separate programmers.

# Lesson Objectives

## Students will:

* be able to plan and execute projects that are coded collaboratively
* understand how to debug their code

## Standards covered:

**CSTA 3A-AP-21**

Evaluate and refine computational artifacts to make them more usable and accessible.

**CSTA 3A-AP-22**

Design and develop computational artifacts working in team roles using collaborative tools.

**CSTA 3A-CS-03**

Develop guidelines that convey systematic troubleshooting strategies that others can use to identify and fix errors.

**CSTA 3A-IC-27**

Use tools and methods for collaboration on a project to increase connectivity of people in different cultures and career fields.

**9–12.Computing Systems.Troubleshooting**

Troubleshooting. Using multiple sources when researching, evaluating, and implementing potential solutions. Using experience, recognizing a similar problem, and adapt solutions that have worked previously.

**9–12.Algorithms and Programming.Program Development**

Program Development. (libraries, IDEs, debugging tools, program performance and functionality testing, end-user testing) Diverse teams can develop programs with a broad impact through careful review and by drawing on the strengths of members in different roles. Design decisions often involve tradeoffs (ie. Fast/Good/Cheap choose 2). The development of complex programs is aided by resources such as libraries and tools to edit and manage parts of the program. Systematic analysis is critical for identifying the effects of lingering bugs.

**AP CSP 1.1 Collaboration**

**AP CSP 1.3 Program Design and Development**

**AP CSP 1.4 Identifying and Correcting Errors**

## Higher-order thinking skills:

**Communication**

Students read and internalize the collaborative coding process, commenting, debugging, test cases, code review, giving feedback, and journaling frameworks, learning to standardize their coding processes to sync group communication and understanding.

Students read and internalize the lesson frameworks, learning to easily identify and resolve gaps in understanding between code collaborators.

**Collaboration**

Students read and internalize the lesson frameworks, learning strategies to code more effectively and efficiently with collaborators.

**Troubleshooting**

Students read and internalize the debugging framework, learning strategies to identify and resolve bugs fast.

# Instructional Strategies

**Pair programming**

**Class discussion**

**Frameworks**

# Time

**40 mins:** warmup activity combining codes, slides, debugging activity

**Additional activities:** continue working on Project: speedy math, adventure game, or personality test

# Materials

* Laptops with access to the internet for each student
* Pen and Paper
* Projector/TV/Display

# Pre-lesson Homework

* Continue Project: speedy math, adventure game, or personality test
* Continue journaling

# In Class

## Warm-up Activity

**Combining separate codes**

Students first work individually on a program, then pair up and try to combine programs. Class debrief on the issues students encountered when trying to combine programs.

## Slides

**Collaboration processes and frameworks**

Present the common issues faced when coding collaboratively, then provide solutions to those issues. Present the various collaboration processes and frameworks, asking students to guess the pros and cons of each, and when/what problems to apply each framework to. By the end of the presentation, students should be able to identify the capabilities of each framework and when and how to apply each framework.

## Activity

**Debugging activity**

Students take a few minutes to identify as many errors as possible. Then, regroup and have students share their debugging strategies. Present a debugging framework.

# Homework

* Project: speedy math, adventure game, or personality test.

# Examples

* Example TinyCircuits story of how 2 programmers code together

# Resources

* Collaborative coding processes
* Commenting framework
* Debugging framework
* Documentation framework
* Giving feedback framework
* Code review framework
* Test cases framework

L9 Code Review, Playtesting

# Introduction

This lesson lets students practice code reviews, playtesting, and giving/receiving feedback, steps in prototyping games.

# Lesson Objectives

## Students will:

* Be able to apply code reviews and playtesting to projects
* Be able to give and receive feedback

## Standards covered:

**CSTA 3A-AP-21**

Evaluate and refine computational artifacts to make them more usable and accessible.

**CSTA 3B-AP-23**

Evaluate key qualities of a program through a process such as a code review.

## Higher-order thinking skills:

**Communication**

Students practice code reviews, playtesting, and giving/receiving feedback, learning to elicit and provide constructive feedback.

# Instructional Strategies

**Practical application**

**Social learning**

# Time

**40 mins:** code review and playtesting Project: speedy math, adventure game, or personality test

**Additional activities:** continue refining Project: speedy math, adventure game, or personality test

# Materials

* Laptops with access to the internet for each student
* Thumby for each student
* MicroUSB cable
* Thumby Link cable

# Pre-lesson Homework

* Continue Project: speedy math, adventure game, or personality test
* Continue journaling

# In Class

## Warm-up Activity

**NA**

## Slides

**NA**

## Activity

**Code review and playtesting**

Students get 3 other students to give them feedback on Project: speedy math, adventure game, or personality test. Students document their feedback for Project: speedy math, adventure game, or personality test using the template provided in project guidelines.

# Homework

* Read loops lesson
* Find a picture of your favorite meme, icon, or character (Note that picture must have defined outlines/be easily traceable). Convert the picture to size, then convert the picture to black and white.

# Examples

* Playtesting video example

# Resources

* User feedback framework
* Giving feedback framework
* Code review framework
* Playtesting framework

L10 Loops

# Introduction

This lesson covers loops, animating a sprite, and simple game actions.

# Lesson Objectives

## Students will:

* Be able to apply loops to animating a sprite
* Be able to apply loops to coding simple game actions

## Standards covered:

**CSTA 3A-AP-15**

Justify the selection of specific control structures when tradeoffs involve implementation, readability, and program performance, and explain the benefits and drawbacks of choices made.

**k12CS 9–12.Algorithms and Programming.Control**

Control. (conditional statements, loops, event handlers, recursion) Programmers consider tradeoffs related to implementation, readability, and program performance when selecting and combining control structures.

**AP CSP 3.8 Iteration**

## Higher-order thinking skills:

**Computational Thinking**

Students apply loops to animate a sprite, learning another type of algorithmic thinking to automate repetitive steps.

# Instructional Strategies

**Flipped classroom learning**

**Practical application**

# Time

**40 mins:** animate a sprite

**Additional activities:** practice additional exercises or start homework

# Materials

* Laptops with access to the internet for each student
* Thumby for each student
* MicroUSB cable

# Pre-lesson Homework

* Read loops lesson
* Prepare image to use in class

# In Class

## Warm-up Activity

**NA**

Brief description of the activity, how the activity works, and what concepts the activity will cover.

**Ie. What is coding warm-up discussion**

Discuss what is coding, why we use it, its capabilities, and how you can apply those capabilities (coding applications)

## Slides

**NA**

## Activity

**Animate a sprite**

Practice loops by animating a simple image in various ways (up and down, side to side, expand and contract, random locations on the screen, move, turn, jump, shoot, walk).

Additional exercises: text loop, scrolling background, simple draw, additive draw, scoreboard

# Homework

* Read lists lesson

# Examples

# Resources

* What is animation? Why can computers animate?

L11 Lists

# Introduction

This lesson covers lists and a saur run pair programming activity.

# Lesson Objectives

## Students will:

* understand how to use and apply lists

## Standards covered:

**CSTA 3A-AP-13**

Create prototypes that use algorithms to solve computational problems by leveraging prior student knowledge and personal interests.

**CSTA 3A-AP-14**

Use lists to simplify solutions, generalizing computational problems instead of repeatedly using simple variables.

**CSTA 3A-AP-16**

Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue by using events to initiate instructions.

**CSTA 3A-AP-19**

Systematically design and develop programs for broad audiences by incorporating feedback from users.

**CSTA 3A-AP-21**

Evaluate and refine computational artifacts to make them more usable and accessible.

**CSTA 3A-AP-22**

Design and develop computational artifacts working in team roles using collaborative tools.

**CSTA 3A-AP-23**

Document design decisions using text, graphics, presentations, and/or demonstrations in the development of complex programs.

**CSTA 3A-IC-27**

Use tools and methods for collaboration on a project to increase connectivity of people in different cultures and career fields.

**CSTA 3B-AP-10**

Use and adapt classic algorithms to solve computational problems.

**CSTA 3B-AP-21**

Develop and use a series of test cases to verify that a program performs according to its design specifications.

**CSTA 3B-AP-23**

Evaluate key qualities of a program through a process such as a code review.

**k12CS 9–12.Algorithms and Programming.Algorithms**

Algorithms. Evaluating algorithms on performance, reusability, and ease of implementation. Knowledge of common algorithms to improve software development, data security, and storing of information.

**k12CS 9–12.Algorithms and Programming.Variables**

Variables. (lists, stacks, queues, trees, hash tables, user defined types, OOP) Data structures are used to manage program complexity. Programmers choose data structures based on functionality, storage, and performance tradeoffs.

**AP CSP 1.1 Collaboration**

**AP CSP 1.2 Program Function and Purpose**

**AP CSP 1.3 Program Design and Development**

**AP CSP 3.2 Data Abstraction**

**AP CSP 3.10 Lists**

## Higher-order thinking skills:

**Prototyping**

Students write project proposals, play instructions, document progress, comment for readability, design test cases, and ask for/implement feedback, learning the processes of prototyping a game.

**Computational Thinking**

Students apply lists to code saur run, learning another type of algorithmic thinking to automate repetitive steps.

# Instructional Strategies

**Flipped classroom learning**

**Pair programming**

**Project based learning**

# Time

**40 mins:** saur run activity

**Additional activities:** start Project: mid-unit mini app

# Materials

* Laptops with access to the internet for each student
* Thumby for each student
* MicroUSB cable

# Pre-lesson Homework

* Read lists lesson

# In Class

## Warm-up Activity

**NA**

## Slides

**NA**

## Activity

**Saur run**

Students pair up to program saur run, applying cumulative concepts.

# Homework

* Project: mid-unit mini app
* Journal entry

# Examples

# Resources

* Code review framework

L12 Functions

# Introduction

This lesson covers functions.

# Lesson Objectives

## Students will:

* Understand how to use functions and successfully apply it to their final project

## Standards covered:

**CSTA 3A-AP-17**

Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.

**CSTA 3B-AP-14**

Construct solutions to problems using student-created components, such as procedures, modules and/or objects.

**k12CS 9–12.Algorithms and Programming.Modularity**

Modularity. (functions, OOP) Complex programs are designed as systems of interacting modules, each with a specific role, coordinating for a common overall purpose. These modules can be procedures within a program; combinations of data and procedures; or independent, but interrelated, programs. Modules allow for better management of complex tasks.

**AP CSP 3.12 Calling Procedures**

**AP CSP 3.13 Developing Procedures**

## Higher-order thinking skills:

**Computational Thinking**

Students learn functions, another type of algorithmic thinking to automate repetitive steps.

# Instructional Strategies

**Self paced learning**

# Time

**40 mins:** functions lesson

**Additional activities:** Project: mid-unit mini app

# Materials

* Laptops with access to the internet for each student
* Thumby for each student
* MicroUSB cable

# Pre-lesson Homework

* Continue Project: mid-unit mini app
* Continue journaling

# In Class

## Warm-up Activity

**NA**

## Slides

**NA**

## Activity

**Read functions lesson**

# Homework

* Continue Project: mid-unit mini app

# Examples

# Resources

L13 Objects

# Introduction

This lesson covers objects.

# Lesson Objectives

## Students will:

* Understand how to use objects and successfully implement it in their final project

## Standards covered:

**CSTA 3A-AP-17**

Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.

**CSTA 3B-AP-14**

Construct solutions to problems using student-created components, such as procedures, modules and/or objects.

**k12CS 9–12.Algorithms and Programming.Variables**

Variables. (lists, stacks, queues, trees, hash tables, user defined types, OOP) Data structures are used to manage program complexity. Programmers choose data structures based on functionality, storage, and performance tradeoffs.

**k12CS 9–12.Algorithms and Programming.Modularity**

Modularity. (functions, OOP) Complex programs are designed as systems of interacting modules, each with a specific role, coordinating for a common overall purpose. These modules can be procedures within a program; combinations of data and procedures; or independent, but interrelated, programs. Modules allow for better management of complex tasks.

## Higher-order thinking skills:

**Computational Thinking**

Students learn objects, another type of algorithmic thinking to automate repetitive steps.

# Instructional Strategies

**Self paced learning**

# Time

**40 mins:** objects lesson

**Additional activities:** Project: mid-unit mini app

# Materials

* Laptops with access to the internet for each student
* Thumby for each student
* MicroUSB cable

# Pre-lesson Homework

* Continue Project: mid-unit mini app
* Continue journaling

# In Class

## Warm-up Activity

**NA**

## Slides

**NA**

## Activity

**Read objects lesson**

# Homework

* Continue Project: mid-unit mini app

# Examples

# Resources

* When to use objects vs. functions
* What is OOP

L14 Dictionaries

# Introduction

This lesson covers dictionaries.

# Lesson Objectives

## Students will:

* Understand how to use dictionaries and can successfully implement them in their final project

## Standards covered:

**CSTA 3A-AP-13**

Create prototypes that use algorithms to solve computational problems by leveraging prior student knowledge and personal interests.

**CSTA 3A-AP-16**

Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue by using events to initiate instructions.

**CSTA 3A-AP-19**

Systematically design and develop programs for broad audiences by incorporating feedback from users.

**CSTA 3A-AP-21**

Evaluate and refine computational artifacts to make them more usable and accessible.

**CSTA 3A-AP-23**

Document design decisions using text, graphics, presentations, and/or demonstrations in the development of complex programs.

**CSTA 3B-AP-21**

Develop and use a series of test cases to verify that a program performs according to its design specifications.

**CSTA 3B-AP-23**

Evaluate key qualities of a program through a process such as a code review.

**AP CSP 1.2 Program Function and Purpose**

**AP CSP 1.3 Program Design and Development**

## Higher-order thinking skills:

**Prototyping**

Students write project proposals, play instructions, document progress, comment for readability, design test cases, and ask for/implement feedback, learning the processes of prototyping a game.

**Ie. Communication**

Students read and internalize the commenting framework and coding best practices, learning to communicate their coding thought processes to code collaborators.

# Instructional Strategies

**Self paced learning**

**Project based learning**

# Time

**40 mins:** dictionaries lesson

**Additional activities:** continue Project: mid-unit mini app

# Materials

* Laptops with access to the internet for each student
* Thumby for each student
* MicroUSB cable

# Pre-lesson Homework

* Continue Project: mid-unit mini app

# In Class

## Warm-up Activity

**NA**

## Slides

**NA**

## Activity

**Read dictionaries lesson**

# Homework

* Continue Project: mid-unit mini app
* Read Final Project guidelines

# Examples

# Resources

* When to use dictionaries vs. lists
* Code review framework

L15 Libraries, Version Control, Open Source, Online Forums, Other Tools and Resources

# Introduction

This lesson covers libraries, version control, open source, online forums, and provides other tools and resources for students to continue their coding journey.

# Lesson Objectives

## Students will:

* Be able to create their own libraries
* Be able to find and use external libraries (APIs)
* Understand how to use and implement version control in their final project
* Understand licensing restrictions and open source
* Be able to identify code they can use with the proper licensing
* Know other forums, tools, and resources they can use for help

## Standards covered:

**CSTA 3A-AP-18**

Create artifacts by using procedures within a program, combinations of data and procedures, or independent but interrelated programs.

**CSTA 3A-AP-20**

Evaluate licenses that limit or restrict use of computational artifacts when using resources such as libraries.

**CSTA 3A-AP-22**

Design and develop computational artifacts working in team roles using collaborative tools.

**CSTA 3A-IC-27**

Use tools and methods for collaboration on a project to increase connectivity of people in different cultures and career fields.

**CSTA 3B-AP-20**

Use version control systems, integrated development environments (IDEs), and collaborative tools and practices (code documentation) in a group software project.

**k12CS 9–12.Algorithms and Programming.Program Development**

Program Development. (libraries, IDEs, debugging tools, program performance and functionality testing, end-user testing) Diverse teams can develop programs with a broad impact through careful review and by drawing on the strengths of members in different roles. Design decisions often involve tradeoffs (ie. Fast/Good/Cheap choose 2). The development of complex programs is aided by resources such as libraries and tools to edit and manage parts of the program. Systematic analysis is critical for identifying the effects of lingering bugs.

**AP CSP 3.14 Libraries**

## Higher-order thinking skills:

**Collaboration**

Students use external libraries, tools, forums, and resources in their final project, learning to reference existing code to improve their own coding skills.

# Instructional Strategies

**Follow along tutorial**

# Time

**40 mins:** Libraries, version control, open source, online forums, other tools and resources lesson

**Additional activities:** continue working on Final Project

# Materials

* Laptops with access to the internet for each student
* Thumby for each student
* MicroUSB cable

# Pre-lesson Homework

* Continue Project: mid-unit mini app
* Read Final Project guidelines

# In Class

## Warm-up Activity

**NA**

## Slides

**Version control tutorial**

Go over all the features of Github. Follow along tutorial of setting up a sample project in Github.

## Activity

**NA**

# Homework

* Continue Final Project
* Continue journaling

# Examples

# Resources

* When to use libraries
* Github tutorial slides