Core Apprenticeship Library
Apprenticeship Sector: *Coding & Gaming*
Unit Guide: *Pencil Code*
Lesson #4: *Functions*

Essential Question

- Do you need to be creative to be good at computer programming?
- How can I use computer coding to design new ideas/ products?

Lesson Overview

Students will build a scene today using functions. By starting with either a world background or an underwater background and using the function Pencil Code Cards, students will be able to build a cityscape or a full underwater scene. Students will be able to use their knowledge of functions in future projects or to innovate further on past projects. Additionally, students will continue thinking about innovation by more explicitly outlining the computer programmer's design process. Students will use this process to create a few unique functions for their scenes! These scenes will be a part of their WOW! Portfolio.

Lesson Objectives

Standard #2: Citizen Schools students will use a design process to create ideas or products.
Lesson Objectives
- SWBAT use the programming design process to create a unique scene (underwater or city).

Standard #4: Implement problem solutions using a programming language, including: looping behavior, conditional statements, logic, expressions, variables, and functions.
Lesson Objectives
- SWBAT use Pencil Code to create one-three unique functions in his/ her scene.

Lesson Agenda

5 Minutes	Hook: Feature Project
10 Minutes	Introduction to New Material: Functions
25 Minutes	Activity 1: Cards
15 Minutes	Activity 2: Game OR Share Projects
30 Minutes	Activity 3: Innovations
5 Minutes	Assessment: Work Product

Lesson Preparation

- Preparation:
  - Practice the Pencil Code Cards for this week. Samples of all of the Pencil Code Cards are available at citizenschools.pencilcode.net. Username: citizenschools, Password: Citizen. Each card is saved in accordance with the file-saving procedure taught in this apprenticeship (week#/cardname). Innovation ideas are included for some cards.
  - Prepare words to add to the Word Wall.
  - Review student assessment work from the previous week. Write a note to the student appreciating some aspect of their work, making a recommendation for further innovation, and recognizing their

work and learning.
    - Pick a project from Week 3 to feature for the Hook. Ideally, you feature a different student's work each week. Select a student whose work is exemplary and might push other students' innovations, thinking, and creativity further.
    - Prepare to show students how to create functions in Pencil Code. Plan to use a teacher computer connected to a projector to demonstrate the programming language.
    - Watch the video "Chris Bosh explains Functions" by code.org. Cue it up ahead of class to play during Intro to New Material.
    - In Activity 3, you'll use a poster titled "Programming Design Process." While you can write this poster during class, you can also create it ahead of time and simply reveal each step as you go over them. You will want to display this visual in every class from here on out.
    - If you have students that are off-task, plan ahead to use this lesson to reinvest them in the work at hand. Make programming relevant to the individual student by targeting the student's interests.
        - For example, if the student loves basketball, perhaps encourage them to create a basketball court scene instead of an underwater or city scene and help them create functions for creating the basketballs, hoops, and people. Maybe give them an "advanced" lesson and show them how to change the turtle into a basketball. Then the student can also program the basketball to move around the screen. Tap into something that the student is already invested in to help invest them in learning what you want them to learn.

- Co-teaching plan:
    - If you had a co-teacher teach a modified lesson to the small group of new students combining Week 1 and Week 2 last week, then plan to have this same teacher teach a combination of Week 3 and 4 this week to catch these students up.
    - When you have students who miss days for being absent, consider having a co-teacher pull them aside to give them a small mini-lesson on topics covered in the previous week at the beginning of Activity 1. After the mini-lesson, the student can begin with a card from the previous week. The student can then progress at his/ her own pace by working through the cards and trying some from both the missed week and the current week. Encourage this student to "catch up" by completing more cards rather than diving deep to create numerous innovations on a few cards.

Materials

Every Class:
- Computer with internet access (one per student)
- Computer with projector (one per class)
- Student folders (one per student)
    - Student Guide*
    - Pencil Code Cards completed by each student (from previous weeks)
- Previous weeks' cards organized in a binder or file box
- Roadmap to WOW!
- Expectations poster
- Word Wall
- Stickers (to indicate earning badges)

This Class:
- Week 4 Pencil Code Cards (one per student per card)*
    - (Background: World, Ocean; Functions: House, Tree, Tower, Cloud, Flower, Butterfly, Grass, Mouse, Fish, Snake, Jellyfish, Seaweed)

- Word Wall words and definitions (Algorithm, Function, Coordinates, Iteration)
- Programming Design Process visual or poster
- Handout - Programming Design Process - Lesson #4

*These materials are located in the lesson folder, not in the Lesson Resources at the end of this lesson.

Hook: Feature Project (5 Minutes)

Teacher's Note: *This Hook repeats each week and is an opportunity for you to highlight each student's work. Ideally, you feature a different student's work each week. Select a student whose work from the previous week is exemplary and might push other students' innovations, thinking, and creativity further.*

- **Say:** Welcome back to Pencil Code! Let's begin this week by taking a look at some of the awesome work you completed last week!
  - **Play** the program from the featured student of the week.
- **Point out** why you selected this project to share. Highlight an innovation, a unique piece of code, or way the student was thinking creatively.
- **Ask** the student who is featured if he or she wants to share anything about the project or ask the group for any feedback.
  - **For example,** the student could share a future innovation that he or she thought would be cool, but isn't sure how to do or the student could ask for other students' ideas of what to try next.
- **Transition:** Today we're going to learn how to create functions in our code. Let's get started!

Introduction to New Material: Functions (10 Minutes)

Teacher's Note: *Plan ahead for how you will review the benefits of creating functions. Plan to use a teacher computer with a projector to show how buttons and events are created and used.*

- Briefly **review** commands (button, key down, click) and concepts (events, user/ programmer) covered in Week 3.
  - **Note:** If you had a co-teacher teach a modified lesson to the small group of new students combining Week 1 and Week 2 last week, then plan to have this same teacher teach a combination of Week 3 and 4 this week to catch these students up.
- **Show** the video "Chris Bosh explains Functions" by code.org. Cue it up ahead of class to play during Intro to New Material.
- **Present** the overarching activity to the students. They will create a city scene or an underwater scene
- **Teach** how to create a function and **review** the benefits of creating functions.
  - Key Messages:
    - Functions allow you to create a single command to carry out a set of instructions. Then you do not have to write the set of code over and over.
    - **Word Wall Definition: Function =** A portion of code or set of instructions that may be called and executed anywhere in a program by using a single command.

        - **Use** the word that you want to name your function, then an "=" and follow with the code to create the function.
        - **Show** an example from the sample scene.
- Show a sample scene at scene.pencilcode.net. Review the functions being used in this sample and why they make programming easier.
    - house tree cloud tower flower butterfly grass snake mouse fish tallfish jellyfish seaweed wave road car
- **Say:** When you're creating your scene, you'll want to strategically place your functions around the screen. To set this up properly, you need to be able to identify the coordinates of where you want the turtle to start the function. Then, use the "jumpto" command followed by the coordinates to tell the turtle where to start.
    - **Word Wall Definition: Coordinates =** A set of numbers that indicate a point on the screen.
    - **Highlight:** In the bottom right corner, the coordinates of the mouse are highlighted in yellow. Use this feature to know what coordinates to have the turtle jump to.
- **Explain** that unlike previous weeks, they **must** create a scene using functions today rather than work on previous week's projects. If they finish their scene early, they can go back to previous programs as well.
    - **Explain** why working on previous week's projects is not optional today. This is the only way they will master functions and the scene will be included in their WOW! Portfolios. Additionally, highlight the ways the students will be able to use functions in future projects. Using functions is a key skill to being able to efficiently program a game or other complex interactive program.
- **Review** the basic agenda for the day. Note that this is generally the schedule that we will use in this class each week. The class is very much run workshop-style and students will guide their own work.
    - **Note:** Remember to review, reteach and practice all procedures you taught in Week 1, if needed. Students will naturally forget from week to week and need reinforcement.
    - Agenda:
        - Work time
        - Pause for sharing projects
        - Think about innovations
        - More work time
        - Save an assessment for the week
- **Transition:** Cue the procedure to get folders, turn on computers, and get to work!

Activity One: Cards (25 Minutes)

Teacher's Note: *When you have students who return after being absent, consider having a co-teacher pull them aside to give them a small mini-lesson on topics covered in the previous week at the beginning of Activity 1. After the mini-lesson, the student can begin with a card from the previous week. The student can then progress at his/her own pace by working through the cards and trying some from both the missed week and the current week. Encourage this student to "catch up" by completing more cards rather than diving deep to create numerous innovations on a few cards.*

Teacher's Note: *If you have students that are off-task, plan ahead to use this lesson to reinvest them in the work at hand. Make programming relevant to the individual student by targeting the student's interests. For example, if the student loves basketball, perhaps encourage them to create a basketball court scene instead of an underwater or city scene, and help them create functions for creating the basketballs, hoops, and people. Maybe give them an "advanced" lesson and show them how to change the turtle into a basketball. Then the student can also program the basketball to move around the screen. Tap into something that the student is already invested in to help invest them in learning what you want them to learn.*

- **Hand out cards:** Each student should start with either Background: World or Background: Underwater Pencil Code Cards. These are starter cards for example backgrounds for the scenes that they are to create today. There is no significant difference between the cards other than the type of scene the background leads to. Selection should be based entirely on student preference.
    - After students have started creating their backgrounds, **Hand out** the Function Pencil Code Cards. Students can copy and integrate these into their scenes.
        - **Note:** If students are getting stuck, direct them to try scene.pencilcode.net where all the functions and backgrounds are compiled.
- **Say:** Use this time to work on creating functions from the cards and creating your own functions for your scenes. Create a scene, either an underwater scene or a city scene! Make sure your scene has at least 3 functions, one of which you've modified from the cards provided.
- **Give** students 25 minutes of work time.
    - Once students complete a Background or Function card, they should place it in their "Completed Cards" section in their folder and follow the appropriate procedure to select another card.
    - **Remind** students to NAME and SAVE their projects regularly. If needed, set the timer to cue you and the students to remember to save their work.
    - During this work time, CTs should be rotating around the room, checking in with students, assisting as needed, and encouraging students to try a new card or a new step in their programming.
        - Encourage students to switch back and forth between text code and block code as needed.
        - **Note:** To make this activity more efficient, consider giving students the link to scenes.pencilcode.net. Show them how to copy and paste the code for a function into their scene to create the function. This will save time and limit frustration over code details and focus energy on creating and repeating functions.
        - **Note:** If students are not excited about the scenes they are creating and are off task, set a clear expectation of what must be included in the scene (i.e. 3 functions, 1 of which is created by the student) and then allow the students to work on other programs or cards when they finish their scene.
- **Bring** the group back together
- **Say:** We are going to have more work time in a bit, but first I want to pause and highlight some great work that's being accomplished and think more about the innovations we can make in our projects.

Activity Two:
Game or Share Projects
(15 Minutes)

Teacher's Note: *Use this time to either share projects that students are working on or share a project that pushes their skills further.*

- **Say:** let's take a look at xx student's project this afternoon...
    - **Share** two-four different students projects. Highlight how they created unique or intriguing functions. Connect back to past skills learned and praise innovative thinking and perseverance.
    - **Share** feedback and ideas for further work.

Alternatively, if you'd rather play a game than share student work, consider playing a quick "Guess the Code" game.

- **Say:** I'm going to show you sets of written code and the same number of programs being run. The game is to guess which code goes with which program. These are a little trickier than last week! These all feature buttons and key downs!
- **Use** the following links to four rounds of "Guess the Code"
    - Intro: Interactivity
    - Day 4, Game 1

- Day 4, Game 2
- Day 4, Game 3
- **Connect to Lesson 3** by highlighting the use of buttons and key downs. Reinforce vocabulary: "Event" and "User." Think about which programs are the most user friendly.
- See all games available for this section at http://inventgame.pencilcode.net

Activity Three: Innovation (30 Minutes)

- **Say:** As you remember from Week 2, there are three stages of innovation. The three stages are:
  - 1. Generating Ideas--Brainstorm ideas of what you can do and pick the best idea
  - 2. Design Process--Create a plan for how you can create it
  - 3. Realizing the Idea--Do it!
- **Say:** The last few weeks we have focused in on the first step of innovation, brainstorming ideas and selecting one to pursue. This was the first standard of our 21st Century Skill, Innovation.
  - **Review:** Standard #1: Citizen Schools students will generate an original idea or product that suits a practical or artistic purpose.
- **Say:** Today we're going to move on to the second step--using a design process.
  - **Review:** Standard #2: Citizen Schools students will use a design process to create ideas or products.
  - **Ask:** Let's discuss. What is our design process? When you want to create an innovation, what do you do?
    - **Solicit answers.** Sample answers will likely include: I just try and do it. I think about what I want to do and the code language I know, and I try to put it together to work. If I can't get it to work, I ask the teacher.
- **Say:** Yes, exactly! In programming, our process can be very simple and repeats many, many times. Let's take your answers and create a list of steps for our design process as computer programmers.
  - **Handout "**Programming Design Process." **Ask** students to fill in the missing blanks as you go.
  - **Review** steps on a poster titled "Programming Design Process." If you did not prepare this visual before class, **write** these steps on a poster now.
    - Step 1: We have selected an idea we want to create.
    - Step 2: Write code to make the idea come to life.
    - Step 3: Press "play" and see what happens. If it works to create the image in our minds, great! Done! If it doesn't, continue to Step 4.
    - Step 4: Ask ourselves what went wrong. Is there an error in the code language (a missing comma, an extra parentheses, etc)? Is there a direction missing (a right turn, a backwards movement, etc)? Or is there something that I don't know how to code (animation, multiple turtles, etc)? Based on the answer, pick Step 5.
    - Step 5a: Check all the details in the code!
    - Step 5b: Add all missing directions!
    - Step 5c: Ask for help!
    - Step 6: Repeat steps 2-5 until the program works just like you imagined it to.
    - Step 7: Celebrate!
  - **Say:** We have all been there when we create code that is supposed to do one thing, but it doesn't work. We have to go back, look at all the details in the code, find the errors, and fix them. Or we realize that we skipped a step, so we go back and add in more and more details and instructions to the computer, until we get our desired result.

        - **Say:** When we're going through this process of trying to run the program, finding an error, fixing it and then going to try the program again--over and over again--we call it trying many iterations.
        - **Say:** We also use the word "iteration" in programming to talk about a command that does an action repeatedly. For example, when you create a loop, you use the "for" command to complete many iterations of a piece of code.
        - **Word Wall Definition: Iteration =** The process of trying something over and over again. OR the process of running the same code over and over on multiple pieces of information.
    - **Say:** Sometimes, we realize that we don't know the coding language to get the computer to do what we want it to do. Then we bring in someone who knows more than us--in this case, the teachers and fellow programmers--to teach and give tips.
    - **Say:** All of this is part of our design process as computer programmers.
    - **Say and highlight:** This is the same design process that I go through in my career. I have an idea of something I want to create in a computer program and I try over and over to perfect the code to make it happen. Sometimes I need support from people who know more than I do, and sometimes I can do it on my own using trial and error.
        - **Share** a real story from your professional programming experience to bring this to life for the students.
- **Say:** Today we don't have a handout to fill out. Instead, practice using this process as you finish your scenes. Create one-three unique functions and include them in your scene. You do not need to create something totally new (although, you can), but instead, you can innovate off the Function cards that you have already created. Remember, innovation doesn't have to mean that you create something totally new. Instead, it can mean that you modify something that already exists to make it different and suit your purpose better.
    - **Ask for a teachback** of what students are supposed to do during this work time.
- **Give** students 10-15 minutes work time.
- **Remind** students to NAME and SAVE their projects regularly.
- During this work time, CTs should be rotating around to room, checking in with students, assisting as needed, and encouraging students to try a new card or a new step in their programming.
    - Be sure to reference the Programming Design Process as it is relevant.
    - CTs should also hand out stickers to award badges for the day. Students will likely earn the "function" challenge in Badge #4 and have the opportunity to catch up on challenges in previous badges that they haven't completed yet.
- **Bring** the group back together and **Transition** to the Assessment.

Assessment
Work Product
(5 Minutes)

- **Say:** Today you will submit a project--your scene--for your assessment. You probably already have it saved as your week4/scene. Go ahead and save that project now and save a copy as "week4/assessment" also. In your student folder, fill out the Week 4 Assessment section.
    - **Review** how to click the down arrow next to the "save" button and select "Copy and Save as" to create a copy and save it as another file name.
    - **Walk students** through how to fill out the Week 4 Assessment written portion, if necessary. (See Student Guide).
        - **Note:** Be sure to write down the innovations you made to create one-three unique functions in your scene.
        - I'm choosing this program because _____
        - I created ____ (#) functions in my _____ scene.
        - The unique functions I created are: (1 is required, more are bonus!)

_____

- **Say:** Before next week I will check out your work, and next week I'll give you some feedback!
- **Look ahead:** Also, next week we will get a chance to share some of our work with each other. We will learn new programming techniques and have new cards to create.
- **Collect** student folders.
- **Note:** If there is time available, share one or two student projects now.

- **Connect to WOW!:** These scenes will make an excellent addition to everyone's WOW! Portfolios.
- **Connect to Week 5**: Next week we will talk more about debugging our programs.
- **Ask** students to follow the procedure for shutting down the computers and cleaning up for the day.
- **Ensure** all students have received stickers for the badges they have earned!

Word Wall - Lesson #4

Algorithm

A list of instructions, procedures, or formulas used to solve a problem. For example, you might write an algorithm to sort a set of words into alphabetical order. The code that does the sorting is called your alphabetical sorting algorithm.

Coordinates

A set of numbers that indicate a point on the screen.

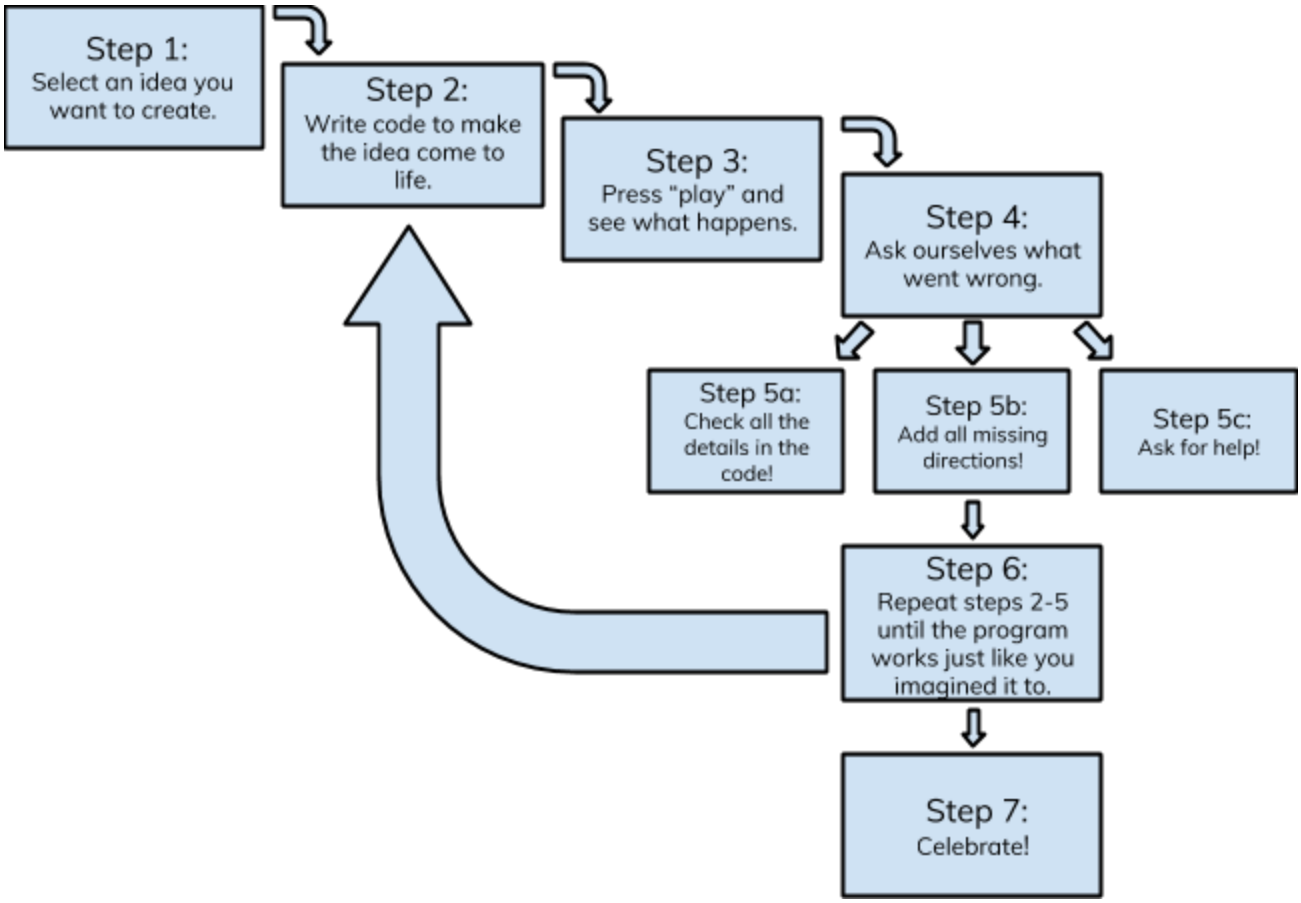
Function

A portion of code or set of instructions that may be called and executed anywhere in a program by using a single command.

Iteration

The process of trying something over and over again.
OR
The process of running the same code over and over on multiple pieces of information.

Sample Visual - Programming Design Process - Lesson #4



Handout - Programming Design Process - Lesson #4

Name - _____