Four-Week Summer Program: Scratch, Makey Makey, Microbit, Math Critical Thinking, and Robotics

Fatma :D

Overview

This four-week program is designed to teach young learners (ages 8 and 9) the basics of Scratch programming, Makey Makey, Microbit, integrating Makey Makey and Microbit with Scratch projects, critical thinking with math, and an introduction to robotics. The program emphasizes hands-on projects that are fun and creative, providing a comprehensive introduction to these technologies. By the end of the program, participants will have a solid foundation in each area and will be able to combine their skills to create integrated projects.

Expected Outcomes

By the end of the program, participants will:

- Understand the basics of Scratch programming, including sprites, coding blocks, loops, and conditionals.
- Be able to create simple Scratch applications and games.
- Have a good understanding of Makey Makey basics, including setting up circuits and using various components.
- Be capable of building and programming simple Makey Makey projects.
- Learn the fundamentals of Microbit, including programming and using built-in sensors.
- Be able to create and program simple Microbit projects.
- Integrate Makey Makey and Microbit with Scratch for more complex projects.
- Develop critical thinking and problem-solving skills through math activities.
- Gain an introduction to robotics, including building and programming simple robots.

Resources Needed

- **Scratch:** Computer with internet access.
- **Makey Makey:** Makey Makey kit (includes Makey Makey board, alligator clips, USB cable, etc.).
- **Microbit:** Microbit Starter Kit.
- **Robotics:** Simple robotics kit (includes motors, sensors, and controller board).
- **Math Activities:** Math puzzles, logic games, and problem-solving worksheets.

Schedule

Week	Day	Topic	Activities/Projects
1	1	Scratch Basics	Introduction to Scratch, setting up the environment, basic syntax, creating a simple project (animated name or a basic game).
	2	Scratch Motion and Events	Explore motion blocks and events, create a simple interactive story.
	3	Scratch Looks and Sound	Learn about looks and sound blocks, create a project with sound effects and changing costumes.
	4	Scratch Loops and Conditionals	Introduce loops and conditionals, create a simple game (e.g., maze game).
2	1-2	Makey Makey Basics	Introduction to Makey Makey, basic projects like banana piano, interactive storybook, game controller.
	3	Makey Makey Advanced Projects	Create a collaborative project using Makey Makey (e.g., an interactive board game).
	4	Microbit Basics	Introduction to Microbit, explore LEDs and buttons, create a simple project (e.g., digital name tag).
3	1	Microbit Sensors and Outputs	Learn about sensors and outputs, create a reaction game using buttons and LEDs.
	2	Microbit Wireless Communication	Explore wireless communication with Microbit, create a communication project.
	3	Combined Projects with Scratch	Plan and build a project integrating Makey Makey and Microbit with Scratch.
	4	Critical Thinking with Math	Math puzzles, logic puzzles, real-world math problems, math-based games.
4	1-2	Introduction to Robotics	Basic concepts, build a simple robot, program the robot with Scratch or Microbit, test and refine.
	3	Robotics Advanced Projects	Enhance the robot with additional features (e.g., sensors, advanced movements).
	4	Final Projects	Finalize all projects, presentation of projects, feedback, and celebration.

Table 1: Four-Week Educational Program Schedule

Day 1: Scratch Basics

Introduction to Scratch

Scratch is a visual programming language developed by MIT that allows young learners to create their own interactive stories, games, and animations. It is designed to be easy and fun to use, with a drag-and-drop interface that helps children understand programming concepts without the need for complex syntax.

Setting Up the Environment

1. Open a web browser and go to the Scratch website: https://scratch.mit.edu. 2. Click on "Join Scratch" to create a free account if you don't already have one. Follow the prompts to set up your account. 3. Once logged in, click on "Create" to start a new project.

Basic Syntax and Interface

The Scratch interface is divided into several sections:

- **Stage:** The area where your project runs. It displays your sprites and their actions.
- **Sprites:** Characters or objects that perform actions in your project. You can add, delete, and edit sprites.
- **Script Area:** The area where you drag and drop blocks to create scripts for your sprites.
- **Blocks Palette:** Contains all the blocks you can use to create your scripts. The blocks are categorized into Motion, Looks, Sound, Events, Control, Sensing, Operators, Variables, and My Blocks.

Creating a Simple Project

Let's create a simple project where a sprite says "Hello!" and moves around the stage.

Step 1: Adding a Sprite

1. Click on the "Choose a Sprite" button located at the bottom right of the stage. 2. Select a sprite from the library or upload your own.

Step 2: Adding Scripts

1. Go to the **Events** category in the Blocks Palette. 2. Drag the `when green flag clicked` block to the Script Area. 3. Go to the **Looks** category and drag the `say [Hello!] for [2] seconds` block and connect it to the `when green flag clicked` block. 4. Go to the **Motion** category and drag the `move [10] steps` block and connect it below the `say [Hello!] for [2] seconds` block.

Step 3: Running the Project

1. Click on the green flag above the stage to run your project. 2. You should see your sprite say "Hello!" and then move 10 steps.

Exercises and Activities

Exercise 1: Customizing the Sprite

1. Change the sprite to a different one from the library. 2. Customize the appearance of your sprite by changing its costume. Go to the **Costumes** tab and select a different costume or draw your own.

Exercise 2: Adding More Actions

1. Add more actions to your script to make the sprite perform additional movements or say different messages. 2. Explore other blocks in the **Motion** and **Looks** categories to see what else you can make your sprite do.

Activity: Creating an Animated Name

1. Create a new project and add a sprite for each letter of your name. 2. Use the **Looks** and **Motion** blocks to animate each letter. 3. Make the letters move, rotate, change colors, and say something when clicked.

Additional Resources

For more tutorials and examples, visit the Scratch website's `Ideas` section: `https://scratch.mit.edu/ideas`.

Day 2: Scratch Motion and Events

Introduction to Motion Blocks

Motion blocks in Scratch control the movement of sprites. These blocks allow you to move sprites in different directions, rotate them, and change their position on the stage.

Basic Motion Blocks

Here are some basic motion blocks and their functions:

- `move [10] steps`: Moves the sprite forward by the specified number of steps.
- `turn clockwise [15] degrees`: Rotates the sprite clockwise by the specified number of degrees.
- `turn counterclockwise [15] degrees`: Rotates the sprite counterclockwise by the specified number of degrees.
- `go to x: [0] y: [0]`: Moves the sprite to the specified x and y coordinates on the stage.
- `glide [1] secs to x: [0] y: [0]`: Moves the sprite to the specified x and y coordinates over the specified number of seconds.

Creating an Interactive Story

Let's create an interactive story where a sprite moves and interacts with other sprites based on user input.

Step 1: Setting Up the Stage

1. Open a new Scratch project. 2. Choose a background for your story by clicking on the "Choose a Backdrop" button at the bottom right of the stage. Select a backdrop from the library or upload your own.

Step 2: Adding Sprites

1. Add multiple sprites that will be part of your story. Click on the "Choose a Sprite" button and select sprites from the library or upload your own. 2. Position the sprites on the stage as desired.

Step 3: Adding Scripts

1. Select the main sprite that will start the story. 2. Go to the **Events** category and drag the `when green flag clicked` block to the Script Area. 3. Go to the **Motion** category and add blocks to make the sprite move and interact with other sprites. For example, you can use the `glide` block to make the sprite move smoothly across the stage. 4. To add interaction, go to the **Control** category and drag the `wait [1] seconds` block to pause the script for a specified amount of time. 5. You can also use the **Looks** category to make the sprite say something or change costumes during the story.

Introduction to Event Blocks

Event blocks in Scratch are used to trigger actions based on different events, such as when the green flag is clicked, when a key is pressed, or when a sprite is clicked.

Basic Event Blocks

Here are some basic event blocks and their functions:

- `when green flag clicked`: Starts the script when the green flag is clicked.
- `when [space] key pressed`: Starts the script when the specified key is pressed.
- `when this sprite clicked`: Starts the script when the sprite is clicked.
- `when backdrop switches to [backdrop1]`: Starts the script when the backdrop changes to the specified backdrop.

Adding Interactivity with Event Blocks

Let's add interactivity to our story by using event blocks.

Step 1: Adding Key Press Events

1. Select a sprite that will respond to key presses. 2. Go to the **Events** category and drag the `when [space] key pressed` block to the Script Area. 3. Go to the **Motion** category and add blocks to make the sprite move when the key is pressed. For example, you can use the `move [10] steps` block to make the sprite move forward.

Step 2: Adding Click Events

1. Select a sprite that will respond to clicks. 2. Go to the **Events** category and drag the `when this sprite clicked` block to the Script Area. 3. Add blocks from the **Looks** and **Sound** categories to make the sprite say something or play a sound when clicked.

Exercises and Activities

Exercise 1: Moving the Sprite with Arrow Keys

1. Create a script to move the sprite up, down, left, and right using the arrow keys. 2. Use the `when [key] pressed` event block for each direction.

Exercise 2: Creating a Simple Animation

1. Create a script to animate the sprite by changing its costume and moving it across the stage. 2. Use the `when green flag clicked` event block to start the animation.

Activity: Creating an Interactive Story

1. Create a new project with a background and multiple sprites. 2. Write a script for each sprite to create an interactive story. Use motion blocks to move the sprites and event blocks to add interactivity. 3. Test your story by running the project and interacting with the sprites.

Additional Resources

For more tutorials and examples, visit the Scratch website's `Ideas` section: `https://scratch.mit.edu/ideas`.

Day 3: Scratch Looks and Sound

Introduction to Looks Blocks

Looks blocks in Scratch control the appearance of sprites. These blocks allow you to change costumes, colors, sizes, and other visual attributes of sprites.

Basic Looks Blocks

Here are some basic looks blocks and their functions:

- `say [Hello!] for [2] seconds`: Makes the sprite display a speech bubble with the specified text for the specified duration.
- `switch costume to [costume1]`: Changes the sprite's costume to the specified costume.
- `change color effect by [25]`: Changes the sprite's color effect by the specified amount.
- `set size to [100]%`: Sets the sprite's size to the specified percentage.
- `show`: Makes the sprite visible on the stage.
- `hide`: Makes the sprite invisible on the stage.

Creating a Project with Looks Blocks

Let's create a project where a sprite changes costumes, colors, and sizes.

Step 1: Adding a Sprite and Costumes

1. Open a new Scratch project. 2. Choose a sprite and add multiple costumes for the sprite. Click on the "Costumes" tab and select different costumes or draw your own.

Step 2: Adding Scripts

1. Select the sprite and go to the **Events** category. Drag the `when green flag clicked` block to the Script Area. 2. Go to the **Looks** category and add blocks to change the sprite's appearance. For example, you can use the `switch costume to [costume1]` block to change the sprite's costume. 3. Add blocks to change the sprite's color and size. For example, you can use the `change color effect by [25]` block and the `set size to [150]%` block.

Step 3: Running the Project

1. Click on the green flag above the stage to run your project. 2. You should see your sprite change costumes, colors, and sizes.

Introduction to Sound Blocks

Sound blocks in Scratch control the sounds that sprites make. These blocks allow you to play sounds, change the volume, and add sound effects.

Basic Sound Blocks

Here are some basic sound blocks and their functions:

- `play sound [meow] until done`: Plays the specified sound and waits until it finishes.
- `start sound [meow]`: Starts playing the specified sound without waiting for it to finish.
- `stop all sounds`: Stops all sounds that are currently playing.
- `change volume by [10]`: Changes the volume by the specified amount.
- `set volume to [100]%`: Sets the volume to the specified percentage.

Creating a Project with Sound Blocks

Let's create a project where a sprite plays different sounds and changes the volume.

Step 1: Adding a Sprite and Sounds

1. Open a new Scratch project. 2. Choose a sprite and go to the "Sounds" tab. 3. Add different sounds for the sprite by selecting sounds from the library or recording your own.

Step 2: Adding Scripts

1. Select the sprite and go to the **Events** category. Drag the `when green flag clicked` block to the Script Area. 2. Go to the **Sound** category and add blocks to play sounds. For example, you can use the `play sound [meow] until done` block to play a sound. 3. Add blocks to change the volume. For example, you can use the `change volume by [10]` block.

Step 3: Running the Project

1. Click on the green flag above the stage to run your project. 2. You should hear the sprite play different sounds and change the volume.

Exercises and Activities

Exercise 1: Creating a Talking Sprite

1. Create a script to make the sprite say different messages using the `say` block. 2. Add sounds to the script to make the sprite play sounds while speaking.

Exercise 2: Animating a Sprite with Costumes

1. Create a script to animate the sprite by changing its costumes. 2. Use the `switch costume to` block and the `wait` block to create the animation.

Activity: Creating an Interactive Animation

1. Create a new project with a sprite and multiple costumes. 2. Write a script to animate the sprite by changing costumes, colors, and sizes. 3. Add sounds to the animation to make it more interactive. 4. Test your animation by running the project and making adjustments as needed.

Additional Resources

For more tutorials and examples, visit the Scratch website's `Ideas` section: `https://scratch.mit.edu/ideas`.

Day 4: Scratch Loops and Conditionals

Introduction to Loops

Loops in Scratch allow you to repeat actions multiple times. There are different types of loops that you can use to control how many times an action is repeated.

Basic Loop Blocks

Here are some basic loop blocks and their functions:

- `repeat [10]`: Repeats the enclosed actions the specified number of times.
- `forever`: Repeats the enclosed actions indefinitely.
- `repeat until [condition]`: Repeats the enclosed actions until the specified condition is met.

Creating a Project with Loops

Let's create a project where a sprite moves in a loop.

Step 1: Adding a Sprite

1. Open a new Scratch project. 2. Choose a sprite from the library or upload your own.

Step 2: Adding Scripts with Loops

1. Select the sprite and go to the **Events** category. Drag the `when green flag clicked` block to the Script Area. 2. Go to the **Control** category and drag the `repeat [10]` block to the Script Area. 3. Go to the **Motion** category and add blocks to move the sprite. For example, you can use the `move [10] steps` block inside the `repeat` block.

Step 3: Running the Project

1. Click on the green flag above the stage to run your project. 2. You should see your sprite move in a loop according to the script.

Introduction to Conditionals

Conditionals in Scratch allow you to make decisions in your scripts based on certain conditions. These blocks enable your sprites to perform different actions depending on whether a condition is true or false.

Basic Conditional Blocks

Here are some basic conditional blocks and their functions:

- `if [condition] then`: Executes the enclosed actions only if the specified condition is true.
- `if [condition] then else`: Executes the first set of enclosed actions if the specified condition is true, otherwise executes the second set of enclosed actions.

Creating a Project with Conditionals

Let's create a project where a sprite moves based on user input.

Step 1: Adding a Sprite

1. Open a new Scratch project. 2. Choose a sprite from the library or upload your own.

Step 2: Adding Scripts with Conditionals

1. Select the sprite and go to the **Events** category. Drag the `when green flag clicked` block to the Script Area. 2. Go to the **Control** category and drag the `forever` block to the Script Area. 3. Inside the `forever` block, add an `if [condition] then` block. 4. Go to the **Sensing** category and drag the `key [space] pressed?` block to the `if` block. 5. Go to the **Motion** category and add blocks to move the sprite inside the `if` block. For example, you can use the `move [10] steps` block.

Step 3: Running the Project

1. Click on the green flag above the stage to run your project. 2. Press the specified key (e.g., space bar) to see the sprite move based on the condition.

Combining Loops and Conditionals

Let's create a project that combines loops and conditionals to make a sprite move in different directions based on user input.

Step 1: Adding a Sprite

1. Open a new Scratch project. 2. Choose a sprite from the library or upload your own.

Step 2: Adding Scripts with Loops and Conditionals

1. Select the sprite and go to the **Events** category. Drag the `when green flag clicked` block to the Script Area. 2. Go to the **Control** category and drag the `forever` block to the Script Area. 3. Inside the `forever` block, add multiple `if [condition] then` blocks to check for different key presses. 4. Go to the **Sensing** category and drag the `key [up arrow] pressed?`, `key [down arrow] pressed?`, `key [left arrow] pressed?`, and `key [right arrow] pressed?` blocks to the `if` blocks. 5. Go to the **Motion** category and add blocks to move the sprite in different directions based on the key presses.

Step 3: Running the Project

1. Click on the green flag above the stage to run your project. 2. Use the arrow keys to move the sprite in different directions based on the conditions.

Exercises and Activities

Exercise 1: Moving the Sprite in a Square

1. Create a script to move the sprite in a square pattern using the `repeat` block and motion blocks. 2. Use the `move` and `turn` blocks to create the square pattern.

Exercise 2: Creating a Simple Game

1. Create a script to move the sprite based on key presses using loops and conditionals. 2. Add other sprites and create interactions between them to form a simple game.

Activity: Creating an Interactive Maze Game

1. Create a new project with a sprite and a maze background. 2. Write a script to move the sprite through the maze using the arrow keys. 3. Use loops and conditionals to check for collisions with the maze walls and guide the sprite to the goal. 4. Test your maze game by running the project and making adjustments as needed.

Additional Resources

For more tutorials and examples, visit the Scratch website's `Ideas` section: `https://scratch.mit.edu/ideas`.

Week 2: Makey Makey Basics

Day 1-2: Introduction to Makey Makey

What is Makey Makey?

Makey Makey is an invention kit that turns everyday objects into touchpads and interfaces for the computer. It allows you to connect objects to the computer and use them as input devices, such as keyboards or game controllers.

How Does Makey Makey Work?

Makey Makey uses a circuit board with alligator clips and a USB cable. The circuit board connects to the computer via the USB cable, and the alligator clips are used to connect the circuit board to conductive objects. When you touch the conductive object, it completes the circuit and sends a signal to the computer, which interprets it as a keyboard or mouse input.

Basic Components of Makey Makey

1. **Circuit Board:** The main component that connects to the computer and interprets the signals. 2. **Alligator Clips:** Used to connect the circuit board to conductive objects. 3. **USB Cable:** Connects the circuit board to the computer. 4. **Conductive Objects:** Any object that can conduct electricity, such as fruits, vegetables, foil, or graphite.

Principles of Conductivity

Conductivity is the ability of a material to conduct electricity. Conductive materials allow electricity to flow through them easily, while non-conductive materials do not. Some common conductive materials include: - Metals (e.g., aluminum foil, copper wire) - Fruits and vegetables (e.g., bananas, potatoes) - Water and human skin

Non-conductive materials include plastic, wood, and rubber.

Creating Your First Makey Makey Project

Project: Banana Piano

Let's create a simple project where bananas are used as piano keys.

Step 1: Setting Up the Circuit Board

1. Connect the Makey Makey circuit board to your computer using the USB cable. 2. Ensure the circuit board is recognized by the computer. It should light up and be ready to use.

Step 2: Connecting the Alligator Clips

1. Connect one end of an alligator clip to the "Earth" (ground) section of the Makey Makey board. 2. Attach the other end to a piece of conductive material that you will touch to complete the circuit (e.g., a piece of foil). 3. Connect additional alligator clips to the arrow keys or space bar sections of the Makey Makey board. 4. Attach the other ends of these clips to bananas.

Step 3: Programming the Piano in Scratch

1. Open a new Scratch project on your computer. 2. Add a sprite for each banana. You can use the same sprite with different costumes to represent different notes. 3. Go to the **Events** category and drag the `when [space key] pressed` block to the Script Area. 4. Go to the **Sound** category and drag the `play sound [meow] until done` block to the Script Area. Choose a sound to represent the piano note. 5. Repeat steps 3 and 4 for each arrow key (up, down, left, right), using different sounds for each key.

Step 4: Playing the Banana Piano

1. Touch the conductive material connected to the "Earth" section of the Makey Makey board to ground yourself. 2. Tap the bananas to play different notes and create music.

Exercises and Activities

Exercise 1: Creating a Fruit Drum Set

1. Use different fruits and connect them to different keys on the Makey Makey board. 2. Program different drum sounds in Scratch for each key. 3. Create a fruit drum set and play rhythms by tapping the fruits.

Exercise 2: Interactive Storybook

1. Create an interactive storybook where each page is represented by a different object. 2. Use Makey Makey to change pages by touching different objects. 3. Program the story in Scratch, where touching an object changes the backdrop or triggers a new scene.

Activity: Human Circuit Game

1. Create a game where players have to complete a circuit by holding hands and touching different objects connected to the Makey Makey board. 2. Use Scratch to create a program that responds to different key presses. 3. Make the game interactive and fun by adding challenges and obstacles.

Additional Resources

For more tutorials and examples, visit the Makey Makey website: https://makeymakey.com.

Day 3: Advanced Makey Makey Projects

Project: Interactive Storybook

Let's create an interactive storybook where different pages are represented by different objects connected to the Makey Makey board.

Step 1: Planning the Story

1. Decide on the story you want to tell. Outline the main events and the sequence of the story. 2. Choose objects that will represent each page or event in the story. For example, a toy car for a car chase scene, a leaf for a forest scene, etc.

Step 2: Setting Up the Circuit Board

1. Connect the Makey Makey circuit board to your computer using the USB cable. 2. Connect one end of an alligator clip to the "Earth" (ground) section of the Makey Makey board. 3. Attach the other end to a piece of conductive material that you will touch to complete the circuit (e.g., a piece of foil). 4. Connect additional alligator clips to the arrow keys or space bar sections of the Makey Makey board. 5. Attach the other ends of these clips to the objects you chose for your story.

Step 3: Programming the Story in Scratch

1. Open a new Scratch project on your computer. 2. Create a backdrop for each scene in your story. Go to the "Backdrops" tab and add different backdrops for each page or event. 3. Add a sprite that will act out the story. You can use multiple sprites for different characters. 4. Go to the **Events** category and drag the `when [space key] pressed` block to the Script Area. 5. Go to the **Looks** category and drag the `switch backdrop to [backdrop1]` block to the Script Area. Select the backdrop for the first scene. 6. Add blocks to make the sprite perform actions in the scene. For example, use the `say [Hello!]` block to make the sprite speak. 7. Repeat steps 4-6 for each arrow key (up, down, left, right) or space bar, using different backdrops and actions for each key.

Step 4: Running the Interactive Storybook

1. Touch the conductive material connected to the "Earth" section of the Makey Makey board to ground yourself. 2. Tap the objects connected to the arrow keys or space bar to switch scenes and progress through the story.

Project: Interactive Board Game

Let's create an interactive board game using Makey Makey. The board game will have different conductive sections that trigger different actions in the game.

Step 1: Designing the Board Game

1. Design a board game layout on a piece of cardboard or paper. Draw paths, obstacles, and interactive sections. 2. Choose conductive materials (e.g., foil, conductive paint) to mark the interactive sections on the board.

Step 2: Setting Up the Circuit Board

1. Connect the Makey Makey circuit board to your computer using the USB cable. 2. Connect one end of an alligator clip to the "Earth" (ground) section of the Makey Makey board. 3. Attach the other end to a piece of conductive material that players will touch to complete the circuit (e.g., a piece of foil). 4. Connect additional alligator clips to the arrow keys or space bar sections of the Makey Makey board. 5. Attach the other ends of these clips to the conductive sections on the board.

Step 3: Programming the Board Game in Scratch

1. Open a new Scratch project on your computer. 2. Create a backdrop for the game board. Go to the "Backdrops" tab and design the game board layout. 3. Add a sprite that will act as the game piece. You can use a custom sprite to represent the game piece. 4. Go to the **Events** category and drag the `when [space key] pressed` block to the Script Area. 5. Go to the **Motion** category and drag the `move [10] steps` block to the Script Area. 6. Add blocks to make the game piece perform actions when it lands on interactive sections. For example, use the `say [You found a treasure!]` block to display messages. 7. Repeat steps 4-6 for each arrow key (up, down, left, right) or space bar, using different actions for each key.

Step 4: Playing the Interactive Board Game

1. Touch the conductive material connected to the "Earth" section of the Makey Makey board to ground yourself. 2. Move the game piece by tapping the conductive sections on the board to trigger different actions and progress through the game.

Exercises and Activities

Exercise 1: Creating a Musical Instrument

1. Use different conductive materials (e.g., fruits, foil) to create different keys for a musical instrument. 2. Connect the materials to different keys on the Makey Makey board. 3. Program different sounds in Scratch for each key. 4. Play music by tapping the materials to trigger the sounds.

Exercise 2: Interactive Quiz Game

1. Create a quiz game where each question is represented by a different object. 2. Use Makey Makey to select answers by touching the objects. 3. Program the quiz in Scratch to display questions and check answers. 4. Make the game interactive and educational by adding feedback for correct and incorrect answers.

Activity: Human Circuit Maze

1. Create a maze on a large piece of cardboard or paper. 2. Use conductive tape or foil to create the walls of the maze. 3. Connect the walls to different keys on the Makey Makey board. 4. Create a program in Scratch that responds to different key presses. 5. Guide a conductive object (e.g., a ball of foil) through the maze by completing the circuit with your touch.

Day 4: Introduction to Microbit

What is Microbit?

Microbit is a small, programmable microcontroller designed to make learning and teaching easy and fun. It can be used to create a variety of projects, from simple games to complex experiments.

Components of Microbit

1. **LED Display:** 25 individually-programmable LEDs to display text, numbers, and images. 2. **Buttons:** Two programmable buttons (A and B) that can be used as input controls. 3. **Pins:** Input/output rings that can be connected to sensors, motors, and other devices. 4. **Sensors:** Built-in accelerometer, magnetometer, and temperature sensor. 5. **Communication:** Bluetooth and radio for wireless communication. 6. **Power:** Can be powered by a USB cable, battery pack, or other external power sources.

Setting Up Microbit

1. Connect the Microbit to your computer using a USB cable. 2. Go to the Microbit website: `https://microbit.org`. 3. Click on "Let's Code" and select "MakeCode Editor." 4. Create a new project to start programming your Microbit.

Programming Microbit with MakeCode

MakeCode is a block-based coding environment for Microbit. It allows you to create programs by dragging and dropping blocks.

Basic Blocks

Here are some basic blocks and their functions: 1. **On Start:** Executes the blocks inside it when the Microbit is powered on. 2. **Forever:** Repeats the blocks inside it indefinitely. 3. **Show Icon:** Displays a specified icon on the LED display. 4. **Show String:** Scrolls a string of text across the LED display. 5. **Pause:** Pauses the program for a specified amount of time.

Creating Your First Microbit Project

Project: Digital Name Tag

Let's create a digital name tag that scrolls your name across the LED display.

Step 1: Setting Up the Program

1. Open a new project in the MakeCode editor. 2. Drag the `on start` block to the workspace.

Step 2: Adding Blocks

1. Go to the **Basic** category and drag the `show string [Hello!]` block to the `on start` block. 2. Change the text in the `show string` block to your name.

Step 3: Downloading the Program

1. Click on the "Download" button to download the program to your computer. 2. Drag and drop the downloaded file onto the Microbit drive that appears on your computer.

Step 4: Running the Program

1. The Microbit will automatically start running the program. 2. You should see your name scroll across the LED display.

Introduction to Buttons

The Microbit has two programmable buttons (A and B) that can be used as input controls.

Programming Buttons

Let's create a project where pressing the buttons displays different messages.

Step 1: Setting Up the Program

1. Open a new project in the MakeCode editor. 2. Drag the `on button A pressed` block to the workspace. 3. Drag the `on button B pressed` block to the workspace.

Step 2: Adding Blocks

1. Go to the **Basic** category and drag the `show string [A]` block to the `on button A pressed` block. 2. Change the text in the `show string` block to a message, such as "Hello!". 3. Repeat step 1 for the `on button B pressed` block and change the text to a different message, such as "Goodbye!".

Step 3: Downloading and Running the Program

1. Click on the "Download" button to download the program to your computer. 2. Drag and drop the downloaded file onto the Microbit drive. 3. Press button A on the Microbit to display the first message. 4. Press button B on the Microbit to display the second message.

Exercises and Activities

Exercise 1: Temperature Sensor

1. Create a program that uses the built-in temperature sensor to display the temperature on the LED display. 2. Use the **Input** category to find the temperature sensor block.

Exercise 2: Shake Detector

1. Create a program that detects when the Microbit is shaken and displays a message. 2. Use the **Input** category to find the shake detector block.

Activity: Reaction Game

1. Create a reaction game where the LED display shows a countdown, and players have to press button A as quickly as possible when a symbol appears. 2. Use the **Control** category to create the countdown. 3. Use the **Input** category to detect button presses and measure the reaction time.

Additional Resources

For more tutorials and examples, visit the Microbit website: `https://microbit.org`.

Week 3: Microbit Sensors and Outputs

Day 1: Exploring Microbit Sensors and Outputs

Introduction to Microbit Sensors

Microbit comes equipped with several built-in sensors that allow it to interact with its environment. These sensors include an accelerometer, magnetometer (compass), light sensor, and temperature sensor.

Accelerometer

The accelerometer detects the orientation of the Microbit and measures acceleration. It can be used to detect movements such as shaking, tilting, and free fall.

Magnetometer (Compass)

The magnetometer measures magnetic fields and can be used as a compass to detect the direction the Microbit is facing.

Light Sensor

The LED display on the Microbit also functions as a light sensor, detecting the ambient light level.

Temperature Sensor

The temperature sensor measures the ambient temperature in degrees Celsius.

Programming the Accelerometer

Project: Shake Detector

Let's create a project that detects when the Microbit is shaken and displays a message.

Step 1: Setting Up the Program

1. Open a new project in the MakeCode editor. 2. Drag the `on shake` block from the **Input** category to the workspace.

Step 2: Adding Blocks

1. Go to the **Basic** category and drag the `show string [Hello!]` block to the `on shake` block. 2. Change the text in the `show string` block to a message, such as "Shaken!".

Step 3: Downloading and Running the Program

1. Click on the "Download" button to download the program to your computer. 2. Drag and drop the downloaded file onto the Microbit drive. 3. Shake the Microbit to see the message displayed on the LED screen.

Programming the Compass

Project: Digital Compass

Let's create a project that uses the magnetometer to display the direction the Microbit is facing.

Step 1: Setting Up the Program

1. Open a new project in the MakeCode editor. 2. Drag the `on start` block to the workspace.

Step 2: Calibrating the Compass

1. Go to the **Input** category and drag the `calibrate compass` block to the `on start` block. 2. The Microbit will prompt you to move it in a circular motion to calibrate the compass.

Step 3: Displaying the Direction

1. Go to the **Forever** category and drag the `forever` block to the workspace. 2. Inside the `forever` block, add the `show string` block from the **Basic** category. 3. Go to the **Input** category and drag the `compass heading` block to the `show string` block. 4. The `compass heading` block will display the direction the Microbit is facing in degrees.

Step 4: Downloading and Running the Program

1. Click on the "Download" button to download the program to your computer. 2. Drag and drop the downloaded file onto the Microbit drive. 3. Move the Microbit to see the direction displayed on the LED screen.

Programming the Light Sensor

Project: Light Level Indicator

Let's create a project that uses the light sensor to measure the ambient light level and display it on the LED screen.

Step 1: Setting Up the Program

1. Open a new project in the MakeCode editor. 2. Drag the `on start` block to the workspace.

Step 2: Measuring the Light Level

1. Go to the **Forever** category and drag the `forever` block to the workspace. 2. Inside the `forever` block, add the `show number` block from the **Basic** category. 3. Go to the **Input** category and drag the `light level` block to the `show number` block. 4. The `light level` block will measure and display the ambient light level on the LED screen.

Step 3: Downloading and Running the Program

1. Click on the "Download" button to download the program to your computer. 2. Drag and drop the downloaded file onto the Microbit drive. 3. Cover and uncover the LED display to see the light level change on the screen.

Programming the Temperature Sensor

Project: Temperature Display

Let's create a project that uses the temperature sensor to measure and display the ambient temperature.

Step 1: Setting Up the Program

1. Open a new project in the MakeCode editor. 2. Drag the `on start` block to the workspace.

Step 2: Measuring the Temperature

1. Go to the **Forever** category and drag the `forever` block to the workspace. 2. Inside the `forever` block, add the `show number` block from the **Basic** category. 3. Go to the **Input** category and drag the `temperature` block to the `show number` block. 4. The `temperature` block will measure and display the ambient temperature on the LED screen.

Step 3: Downloading and Running the Program

1. Click on the "Download" button to download the program to your computer. 2. Drag and drop the downloaded file onto the Microbit drive. 3. Observe the temperature displayed on the LED screen.

Exercises and Activities

Exercise 1: Tilt Detector

1. Create a program that detects when the Microbit is tilted in different directions. 2. Use the accelerometer blocks from the **Input** category to detect tilts and display corresponding messages.

Exercise 2: Ambient Light Alarm

1. Create a program that turns on an LED or plays a sound when the ambient light level drops below a certain threshold. 2. Use the light sensor block from the **Input** category and add conditionals to create the alarm.

Activity: Weather Station

1. Create a program that uses the temperature and light sensors to measure and display the current weather conditions. 2. Use the `show string` block to display messages like "Sunny" or "Cold" based on the sensor readings. 3. Enhance the weather station by adding more sensors or features.

Day 2: Exploring Wireless Communication with Microbit

Introduction to Microbit Wireless Communication

Microbit has built-in capabilities for wireless communication using radio and Bluetooth. These features allow Microbits to communicate with each other and with other devices, enabling the creation of interactive and collaborative projects.

Radio Communication

Radio communication allows Microbits to send and receive messages over a short distance. It is useful for projects where Microbits need to interact with each other directly.

Bluetooth Communication

Bluetooth communication enables Microbits to connect to Bluetooth-enabled devices, such as smartphones, tablets, and computers. It is useful for projects that require integration with external devices.

Programming Radio Communication

Project: Simple Messaging System

Let's create a project where two Microbits send and receive messages using radio communication.

Step 1: Setting Up the Program for Microbit A

1. Open a new project in the MakeCode editor. 2. Drag the `on button A pressed` block from the **Input** category to the workspace.

Step 2: Adding Blocks for Sending Messages

1. Go to the **Radio** category and drag the `radio send string [hello]` block to the `on button A pressed` block. 2. Change the string in the `radio send string` block to a message, such as "Hi!".

Step 3: Setting Up the Program for Microbit B

1. Open a new project in the MakeCode editor for the second Microbit. 2. Drag the `on radio received receivedString` block from the **Radio** category to the workspace.

Step 4: Adding Blocks for Receiving Messages

1. Go to the **Basic** category and drag the `show string [Hello!]` block to the `on radio received receivedString` block. 2. Change the text in the `show string` block to `receivedString` to display the received message.

Step 5: Downloading and Running the Program

1. Click on the "Download" button to download the program to your computer. 2. Drag and drop the downloaded files onto the respective Microbit drives. 3. Press button A on Microbit A to send a message. 4. The message should be displayed on the LED screen of Microbit B.

Programming Bluetooth Communication

Project: Controlling a Microbit with a Smartphone

Let's create a project where a Microbit is controlled by a smartphone using Bluetooth communication.

Step 1: Enabling Bluetooth on the Microbit

1. Open a new project in the MakeCode editor. 2. Go to the **Bluetooth** category and drag the `bluetooth start accelerometer service` block to the `on start` block. 3. This block enables the Bluetooth service for the accelerometer, allowing the smartphone to read the accelerometer data.

Step 2: Adding Blocks for Control

1. Go to the **Input** category and drag the `on button A pressed` block to the workspace. 2. Inside the `on button A pressed` block, add the `show icon` block from the **Basic** category and choose an icon to display.

Step 3: Setting Up the Smartphone App

1. Download and install the Microbit app on your smartphone from the app store. 2. Open the app and pair it with your Microbit via Bluetooth. 3. Use the app to send commands to the Microbit, such as pressing a virtual button A.

Step 4: Downloading and Running the Program

1. Click on the "Download" button to download the program to your computer.
2. Drag and drop the downloaded file onto the Microbit drive.
3. Use the smartphone app to control the Microbit and see the icon display when button A is pressed.

Exercises and Activities

Exercise 1: Radio Group Chat

1. Create a program that allows multiple Microbits to send and receive messages in a group chat using radio communication.
2. Use the `radio send string` and `on radio received` blocks to handle messaging.

Exercise 2: Remote Control Car

1. Create a program to control a toy car using Microbit and Bluetooth.
2. Use a second Microbit as a remote control and connect it to the toy car's motors.
3. Program the Microbits to communicate via Bluetooth, sending control commands from the remote to the car.

Activity: Wireless Sensor Network

1. Create a network of Microbits that communicate wirelessly to monitor environmental conditions (e.g., temperature, light levels).
2. Use radio communication to send sensor data from multiple Microbits to a central Microbit.
3. Program the central Microbit to display the collected data on its LED screen.

Day 3: Integrating Makey Makey and Microbit with Scratch

Introduction to Combined Projects

Combining Makey Makey and Microbit with Scratch allows you to create more interactive and complex projects. By integrating these technologies, you can use Makey Makey as an input device and Microbit for additional functionalities like sensing and displaying data.

Project: Interactive Game Controller

Let's create a project where Makey Makey is used as a game controller and Microbit is used to display scores and provide feedback.

Step 1: Setting Up the Makey Makey Game Controller

1. Connect the Makey Makey circuit board to your computer using the USB cable. 2. Connect alligator clips to the arrow keys and space bar sections of the Makey Makey board. 3. Attach the other ends of the alligator clips to conductive objects that will act as buttons (e.g., fruits, foil).

Step 2: Programming the Game in Scratch

1. Open a new Scratch project on your computer. 2. Add a sprite for the game character. You can use a pre-made sprite from the Scratch library or create your own. 3. Go to the **Events** category and drag the `when [space key] pressed` block to the Script Area. 4. Go to the **Motion** category and add blocks to move the sprite when the space key is pressed. 5. Repeat step 3 for the arrow keys (up, down, left, right), adding appropriate motion blocks for each key.

Step 3: Setting Up the Microbit for Score Display

1. Connect the Microbit to your computer using a USB cable. 2. Open a new project in the MakeCode editor. 3. Drag the `on start` block to the workspace. 4. Go to the **Variables** category and create a variable called "score". 5. Set the initial value of "score" to 0 in the `on start` block. 6. Go to the **Radio** category and drag the `radio set group [1]` block to the `on start` block. This ensures that the Microbit listens to messages from Scratch.

Step 4: Integrating Microbit with Scratch for Score Updates

1. In Scratch, go to the **Extensions** section and add the "Microbit" extension. 2. In the Scratch script, add a block to broadcast a message whenever the score changes. 3. In the MakeCode editor, add an `on radio received [score]` block to update the score variable and display it on the Microbit's LED screen.

Step 5: Downloading and Running the Programs

1. Download the Scratch project to your computer. 2. Download the MakeCode project to your Microbit by clicking on the "Download" button and transferring the file to the Microbit drive. 3. Run the Scratch project and play the game using the Makey Makey controller. The Microbit will display the score as it changes.

Exercises and Activities

Exercise 1: Advanced Game Features

1. Add more features to the game, such as obstacles, power-ups, and levels. 2. Use Makey Makey to control different aspects of the game and Microbit to provide feedback (e.g., displaying remaining lives).

Exercise 2: Interactive Quiz with Feedback

1. Create an interactive quiz game where Makey Makey is used to select answers. 2. Use Microbit to display feedback for correct and incorrect answers. 3. Program the Scratch project to ask questions and use Makey Makey inputs to select answers.

Activity: Collaborative Storytelling Project

1. Create a storytelling project where different scenes are controlled by Makey Makey inputs. 2. Use Microbit to display additional information or provide interactive elements (e.g., showing the current scene number). 3. Collaborate with others to create an engaging and interactive story.

Day 4: Developing Critical Thinking Skills with Math

Introduction to Critical Thinking

Critical thinking involves analyzing facts to form a judgment. It is an essential skill in mathematics, as it helps in solving problems, making decisions, and understanding concepts deeply.

Principles of Critical Thinking in Math

1. **Questioning:** Always ask questions to understand the problem better. 2. **Logic and Reasoning:** Use logical steps to solve problems. 3. **Pattern Recognition:** Identify patterns to make predictions and solve problems. 4. **Proof and Evidence:** Justify your solutions with evidence and logical reasoning. 5. **Reflection:** Reflect on your problem-solving process and identify areas for improvement.

Mathematical Puzzles and Challenges

Puzzle 1: Magic Squares

A magic square is a grid of numbers where the sum of each row, column, and diagonal is the same.

Activity: Creating a 3x3 Magic Square 1. Create a 3x3 grid. 2. Fill the grid with numbers from 1 to 9 so that the sum of each row, column, and diagonal equals 15.

```
|   |   |   |
|---|---|---|
|   |   |   |
|---|---|---|
|   |   |   |
```

Hint: Place the number 5 in the center, and use a logical pattern to fill in the remaining numbers.

Puzzle 2: Sudoku

Sudoku is a puzzle where you fill a 9x9 grid so that each column, each row, and each of the nine 3x3 grids contain all the digits from 1 to 9.

Activity: Solving a Sudoku Puzzle 1. Start with a partially filled 9x9 grid. 2. Use logic and reasoning to fill in the missing numbers, ensuring that each number from 1 to 9 appears only once in each row, column, and 3x3 grid.

```
|5|3| | |7| | | | |
|6| | |1|9|5| | | |
| |9|8| | | | |6| |
|8| | | |6| | | |3|
|4| | |8| |3| | |1|
|7| | | |2| | | |6|
| |6| | | | |2|8| |
| | | |4|1|9| | |5|
| | | | |8| | |7|9|
```

Hint: Start with the numbers that are already filled in and use logical elimination to find the missing numbers.

Puzzle 3: Logic Puzzles

Logic puzzles require you to use reasoning and deduction to solve problems.

Activity: Solving a Logic Puzzle 1. Read the clues carefully. 2. Use a process of elimination to find the solution.

Example Puzzle: Four friends (Alice, Bob, Carol, and Dave) are sitting in a row. Use the following clues to determine the order in which they are sitting: - Alice is not at either end. - Bob is sitting to the right of Alice. - Carol is not next to Bob. - Dave is sitting to the left of Carol.

Solution: - Alice is not at either end (so she is in the middle). - Bob is to the right of Alice. - Carol is not next to Bob, so she must be at one of the ends. - Dave is to the left of Carol.

The order is: Dave, Alice, Bob, Carol.

Puzzle 4: Knights and Knaves

In Knights and Knaves puzzles, knights always tell the truth, and knaves always lie.

Activity: Solving a Knights and Knaves Puzzle 1. Read the statements made by the characters. 2. Determine who is a knight and who is a knave based on their statements.

Example Puzzle: There are two characters, Alice and Bob. - Alice says, "Bob is a knave." - Bob says, "We are both knaves."

Solution: - If Bob were a knight, his statement "We are both knaves" would be false, which is a contradiction. Therefore, Bob is a knave. - Since Bob is a knave, his statement is false, so they are not both knaves. Therefore, Alice must be a knight.

Real-World Math Problems

Problem 1: Shopping Budget

You have a budget of $50 to buy school supplies. The prices of the items are as follows: - Notebooks: $3 each - Pens: $1 each - Pencils: $0.50 each - Erasers: $0.25 each

Activity: Creating a Shopping List 1. Create a shopping list that stays within your budget. 2. Calculate the total cost and ensure it does not exceed $50.

Problem 2: Cooking Measurements

You are making a recipe that requires the following ingredients: - 2 cups of flour - 1.5 cups of sugar - 0.75 cups of butter

Activity: Adjusting the Recipe 1. Adjust the recipe to make twice as much. 2. Adjust the recipe to make half as much. 3. Calculate the new measurements for each ingredient.

Problem 3: Distance and Time

You are planning a road trip that covers a distance of 300 miles. Your car travels at an average speed of 60 miles per hour.

Activity: Calculating Travel Time 1. Calculate the total travel time for the trip. 2. If you take a 30-minute break every 100 miles, calculate the total travel time including breaks.

Summary

In Day 4 of Week 3, we focused on developing critical thinking skills with math. We explored mathematical puzzles and challenges, including magic squares, Sudoku, logic puzzles, and Knights and Knaves. We also solved real-world math problems involving shopping budgets, cooking measurements, and travel time calculations. Keep practicing critical thinking with different math problems and see how you can apply these skills in everyday life!

Additional Resources

For more math puzzles and challenges, visit: https://www.mathsisfun.com.

Day 5: Additional Critical Thinking Math Activities

Review of Critical Thinking Concepts

Before we dive into new activities, let's review the critical thinking concepts we have learned so far: 1. **Questioning:** Always ask questions to understand the problem better. 2. **Logic and Reasoning:** Use logical steps to solve problems. 3. **Pattern Recognition:** Identify patterns to make predictions and solve problems. 4. **Proof and Evidence:** Justify your solutions with evidence and logical reasoning. 5. **Reflection:** Reflect on your problem-solving process and identify areas for improvement.

Mathematical Puzzles and Challenges

Puzzle 5: The River Crossing Puzzle

In this classic puzzle, you need to help a farmer transport a wolf, a goat, and a cabbage across a river using a boat. The boat can only carry the farmer and one other item at a time. You cannot leave the wolf alone with the goat, or the goat alone with the cabbage.

Activity: Solving the River Crossing Puzzle 1. Use logic to determine the steps needed to get all three items across the river without any of them being eaten. 2. Write down each step and ensure the conditions are met.

Solution:

1. Take the goat across the river. 2. Return alone to the original side. 3. Take the wolf across the river. 4. Bring the goat back to the original side. 5. Take the cabbage across the river. 6. Return alone to the original side. 7. Take the goat across the river.

Puzzle 6: The Tower of Hanoi

The Tower of Hanoi is a mathematical puzzle consisting of three rods and a number of disks of different sizes. The objective is to move the entire stack to another rod, obeying the following rules: - Only one disk can be moved at a time. - Each move consists of taking the upper disk from one of the stacks and placing it on top of another stack or on an empty rod. - No disk may be placed on top of a smaller disk.

Activity: Solving the Tower of Hanoi 1. Use logic and planning to move the disks from the source rod to the destination rod. 2. Try solving it with 3 disks, then increase the number of disks to make it more challenging.

Solution for 3 Disks:

1. Move Disk 1 to Rod C. 2. Move Disk 2 to Rod B. 3. Move Disk 1 to Rod B. 4. Move Disk 3 to Rod C. 5. Move Disk 1 to Rod A. 6. Move Disk 2 to Rod C. 7. Move Disk 1 to Rod C.

Puzzle 7: The Monty Hall Problem

This is a probability puzzle based on a game show scenario. You are given the choice of three doors. Behind one door is a car; behind the other two doors are goats. You pick a door, say Door 1, and the host, who knows what's behind the doors, opens another door, say Door 3, which has a goat. The host then asks if you want to switch your choice to Door 2 or stay with Door 1.

Activity: Understanding the Monty Hall Problem 1. Analyze the probability of winning if you switch versus if you stay with your original choice. 2. Discuss why switching doors gives you a better chance of winning.

Solution:

- If you stay with your original choice, your probability of winning is 1/3. - If you switch doors, your probability of winning is 2/3. - This counterintuitive result occurs because the host's action of revealing a goat provides additional information that changes the probability distribution.

Real-World Math Problems

Problem 4: Sharing Pizza

You have 3 friends and a pizza cut into 8 slices. You want to share the pizza equally.

Activity: Dividing the Pizza 1. Calculate how many slices each person gets. 2. Determine how to fairly distribute the remaining slices.

Solution:

1. Each person gets 2 slices. 2. One slice remains. To distribute it fairly, you could cut the remaining slice into smaller pieces or save it for later.

Problem 5: Saving Money

You want to save $100 in 10 weeks. How much money do you need to save each week to reach your goal?

Activity: Creating a Savings Plan 1. Calculate the amount you need to save each week. 2. Make a plan to track your savings progress.

Solution:

1. You need to save $10 each week to reach your goal of 100 in 10 weeks.

Exercises and Activities

Exercise 1: Number Patterns

1. Identify and continue the following number pattern: 2, 4, 8, 16, $_{,,2.Explainthepatternanddeterminethenextthreenumbers.}$

Solution:

- The pattern is doubling each number. - The next three numbers are 32, 64, 128.

Exercise 2: Logical Deduction

1. Solve the following logical deduction puzzle: - A, B, and C are standing in a line. You can see that A is taller than B, but shorter than C. Who is the tallest?

Solution:

- C is the tallest.