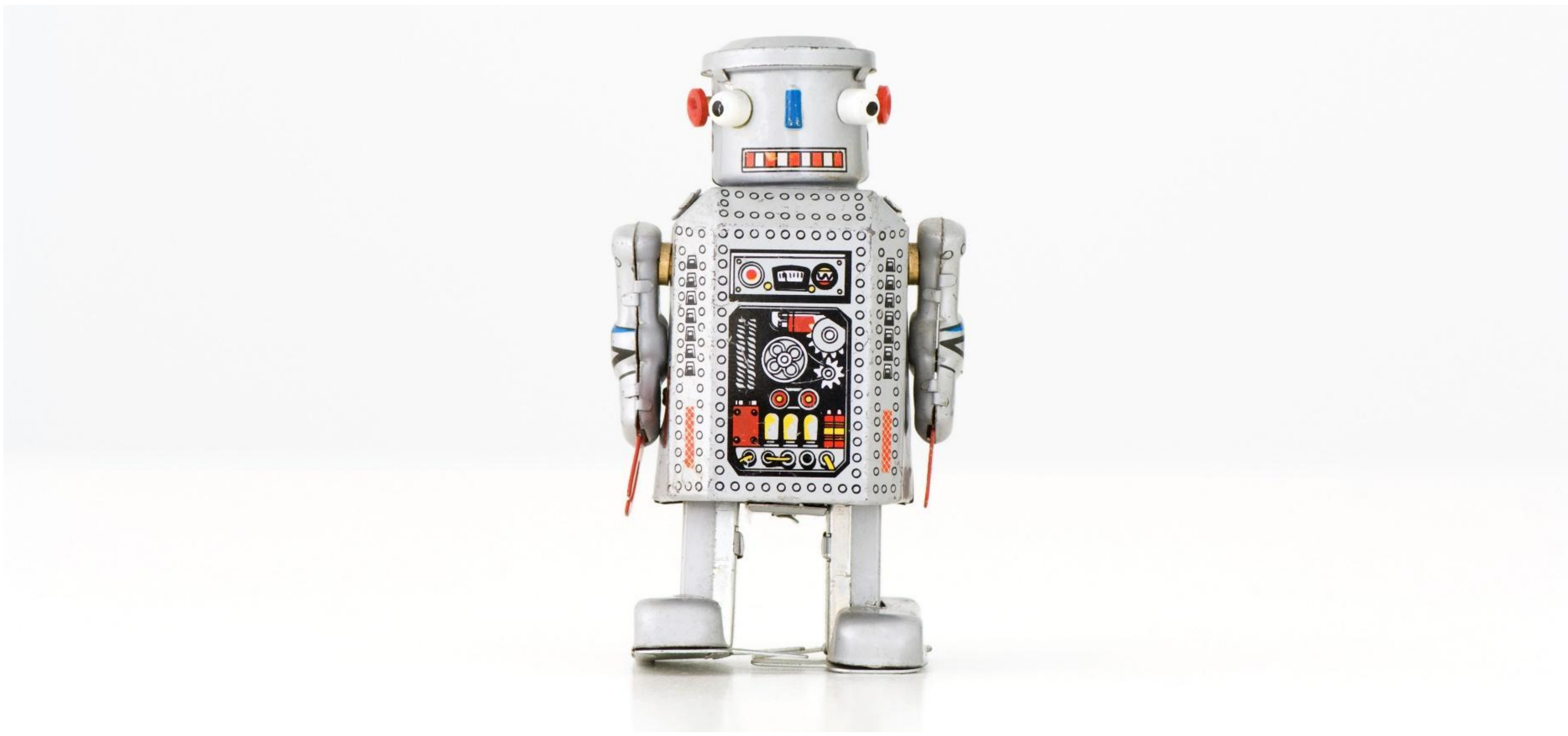


GRADE 6

Coding and Beyond

Intermediate Programming and Emerging Tech

Curriculum Overview

Lesson 1		Advancing with Scratch	
	1.	Transition to more advanced block-based programming tools like Scratch.	4
	2.	Explore complex coding concepts, including loops and conditionals.	9
	3.	Begin creating interactive and dynamic projects using Scratch.	13
Lesson 2		Problem-Solving Puzzles	
	●	Develop problem-solving skills through coding projects.	
	●	Engage in coding challenges and puzzles.	
	●	Learn to break down complex problems into manageable steps.	
Lesson 3		Text-Based Coding Basics	
	●	Introduction to text-based coding languages like Python or JavaScript.	
	●	Understand basic coding syntax and commands.	
	●	Write and execute simple text-based code.	
Lesson 4		Text-Based Coding Adventures	
	●	Dive deeper into text-based coding concepts.	
	●	Create more complex programs using text-based coding languages.	
	●	Understand how code can be used to solve real-world problems.	
Lesson 5		Exploring Algorithms	
	●	Understand the concept of algorithms and their importance in coding.	
	●	Learn how to design and implement algorithms.	
	●	Apply algorithms to coding challenges and projects.	
Lesson 6		Coding for a Cause	
	●	Explore how coding can be used for social and environmental causes.	
	●	Collaborate on coding projects with a purpose.	
	●	Understand the positive impact of coding on the community.	
Lesson 7		Beyond the Screen: Hardware Programming	
	●	Explore hardware programming concepts.	
	●	Learn to program microcontrollers and simple electronic devices.	
	●	Understand the relationship between software and hardware.	
Lesson 8		The World of Robotics Continues	
	●	Continue exploring robotics and automation.	

	● Engage in more advanced coding projects with robots.	
	● Understand how robots are used in industry and research.	
Lesson 9	Emerging Technologies 2.0	
	● Explore the latest advancements in technology, including AI and IoT.	
	● Understand how AI is changing industries like healthcare and finance.	
	● Discuss ethical considerations related to emerging tech.	
Lesson 10	Coding and the Future	
	● Reflect on the coding and programming skills gained throughout the course.	
	● Discuss the importance of coding in future careers and industries.	
	● Encourage students to continue exploring coding and emerging technologies.	

Lesson 1: Advancing with Scratch

Transition to more advanced block-based programming tools like Scratch

- Review the basic features and functions of Scratch
- Compare and contrast Scratch with other block-based tools, such as Blockly or Snap!
- Learn how to use the Scratch editor, stage, sprites, costumes, sounds, and blocks
- Explore the different categories of blocks, such as motion, looks, sound, events, control, sensing, operators, and variables
- Practice creating and modifying simple scripts using different blocks

What exactly is Scratch?

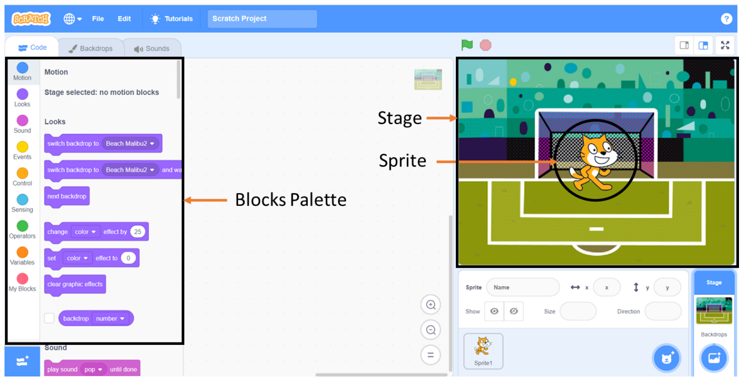
Scratch is a fun and creative way to make your own interactive stories, games, and animations. You can also share your projects with other people online. In this lesson, you will learn more about Scratch and how it is different from other block-based programming tools. You will also practice using the Scratch editor and the different types of blocks to create and modify simple scripts.

Review the basic features and functions of Scratch

Scratch is a **block-based programming** tool that lets you create programs by snapping together colorful blocks. Each block has a shape and a color that tells you what it does and how it fits with other blocks. For example, a green flag block is an event block that starts your program when you click on it. A blue move block is a motion block that makes your sprite move on the stage.

The **stage** is the area where you can see your sprites and the background of your project. A sprite is an object that you can control with blocks. You can choose from many different sprites in the sprite library, or draw your own using the paint editor. You can also change how your sprite looks by adding costumes and sounds.

The **blocks palette** is the area where you can find all the blocks that you can use in your project. You can drag and drop blocks from the palette to the scripts area, where you can snap them together to make scripts. A script is a set of instructions that tells your sprite what to do.



Compare and contrast Scratch with other block-based tools, such as Blockly or Snap!

Scratch is not the only block-based programming tool that you can use. There are many others, such as Blockly or Snap!, that have similar features and functions.

However, there are also some differences between them. For example:

Blockly	Snap!	Scratch
The library lets you create block-based programming interfaces for different applications, such as games, puzzles, or robots.	An extension of Scratch that lets you create more advanced projects, such as simulations, interactive art, or music.	Scratch is designed for beginners who want to learn the basics of programming and express their creativity.
Blockly uses JavaScript as its underlying language.	Snap! uses Scheme as its underlying language.	Scratch uses its own language.
Blocks can be converted to JavaScript code and run it in a web browser.	You can define your own blocks and data types, and use higher-order functions and recursion.	Blocks cannot be converted to other languages.

- Blockly is a library that lets you create block-based programming interfaces for different applications, such as games, puzzles, or robots. Blockly uses JavaScript as its underlying language, which means that you can convert your blocks to JavaScript code and run it in a web browser.

- Snap! is an extension of Scratch that lets you create more advanced projects, such as simulations, interactive art, or music. Snap! uses Scheme as its underlying language, which means that you can define your own blocks and data types, and use higher-order functions and recursion.

- Scratch is designed for beginners who want to learn the basics of programming and express their creativity. Scratch uses its own language, which means that you cannot convert your blocks to other languages or run them outside of Scratch. However, Scratch has a large and supportive community of users who share their projects and ideas online.

Learn how to use the Scratch editor, stage, sprites, costumes, sounds, and blocks

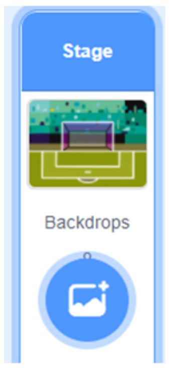
To start using Scratch, you need to open the Scratch editor on your computer or on the web. The Scratch editor has four main parts: the stage, the sprite list, the blocks palette, and the scripts area.

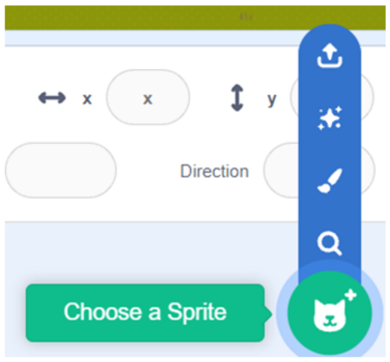
The **stage** is where you can see your project in action. You can change the background of the stage by choosing from different backdrops in the backdrop library, or drawing your own using the paint editor. You can also switch between different backdrops using blocks.

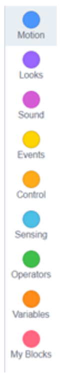
The **sprite list** is where you can see all the sprites that are in your project. You can add new sprites by clicking on the choose sprite button, which lets you pick from different categories of sprites in the sprite library, such as animals, people, or fantasy. You can also click on the paint sprite button, which lets you draw your own sprite using the paint editor. You can also click on the surprise sprite button, which gives you a random sprite.

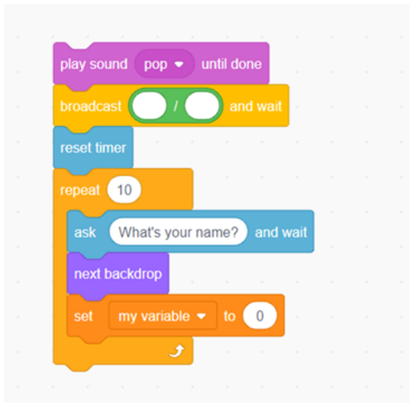
The **blocks palette** is where you can find all the blocks that you can use in your project. The blocks are organized into different categories, such as motion, looks, sound, events, control, sensing, operators, and variables. Each category has a different color and shape that helps you identify what kind of block it is. You can click on a category to see all the blocks in it.

The **scripts area** is where you can build your scripts by dragging and dropping blocks from the palette. You can snap blocks together to make them work as a unit. You can also separate blocks by pulling them apart. You can move your scripts around by dragging them with your mouse. You can also copy and paste your scripts by right-clicking on them and choosing copy or paste.









```
play sound pop until done
broadcast ( / ) and wait
reset timer
repeat 10
  ask What's your name? and wait
  next backdrop
  set my variable to 0
```

Explore the different categories of blocks, such as motion, looks, sound, events, control, sensing, operators, and variables

Each category of blocks has a different function and purpose in your project. Here are some examples of what each category does:

- **Motion blocks** let you control how your **sprite moves** on the stage. You can make your sprite move forward or backward by a certain number of steps, turn left or right by a certain angle, go to a specific position or direction, glide smoothly from one place to another, bounce off the edge of the stage, and change its x and y coordinates.

- **Looks blocks** let you control how your **sprite looks** on the stage. You can make your sprite say or think something, change its costume or backdrop, change its size or color, show or hide itself, and switch between different effects, such as ghost, pixelate, or mosaic.

- **Sound blocks** let you control how your **sprite sounds** on the stage. You can make your sprite play a sound from the sound library, or record your own sound using the sound editor. You can also change the volume or pitch of the sound, stop all sounds, and play a note or a drum for a certain duration.

- **Events blocks** let you control when your **script starts or stops**. You can make your script start when you click on the green flag, when you press a certain key on the keyboard, when you click on your sprite, when your sprite touches another sprite or the edge of the stage, when a certain message is broadcasted, or when a certain condition is true.

- **Control blocks** let you control the **flow** of your script. You can make your script repeat a certain number of times, or forever, using loops. You can also make your script wait for a certain amount of time, or until a certain condition is true, using delays. You can also make your script choose between different options, or do different things depending on a condition, using if-then-else statements. You can also make your script stop or start other scripts using stop and start blocks.

- **Sensing blocks** let you get **information** about your sprite or the stage. You can ask your sprite a question and get an answer from the user, using ask and answer blocks. You can also get the value of different properties of your sprite or the stage, such as its position, direction, size, costume, backdrop, color, loudness, timer, mouse position, mouse button status, key pressed status, distance to another sprite, touching status, and answer status.

- **Operators blocks** let you perform **mathematical and logical operations** on numbers, strings, or Booleans. You can add, subtract, multiply, divide, modulo, round, pick random numbers, join strings, letter of strings, length of strings, compare numbers, strings, or Booleans, and use logical operators such as and, or not.

- **Variables blocks** let you **store and change data** in your project. You can create variables that hold numbers, strings, or Booleans, and use them in your scripts. You can also change the value of your variables by setting them to a new value, or by adding or subtracting a value from them. You can also show or hide your variables on the stage.

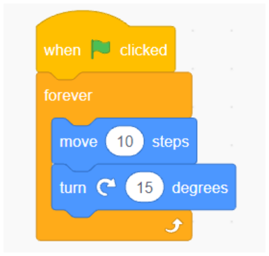
Practice creating and modifying simple scripts using different blocks

To practice using different blocks in Scratch, you can try creating and modifying some simple scripts that make your sprite do different things on the stage.

Here are some examples of what you can do:

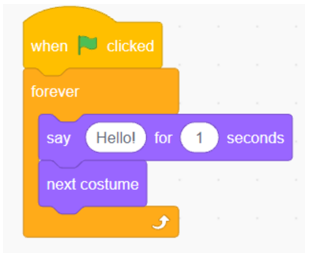
Make your sprite move around the stage using motion blocks. For example, you can make your sprite move forward by 10 steps and turn right by 15 degrees forever using this script:

~~~
scratch
when green flag clicked
forever
move (10) steps
turn right (15) degrees
end
```
~~~



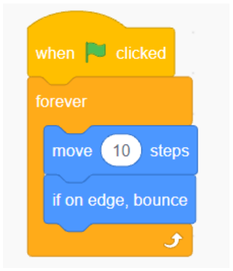
Make your sprite change its appearance using looks and sound blocks. For example,
you can make your sprite say "Hello!" and switch to a different costume every second using this script:

~~~
```scratch
when green flag clicked
forever
say [Hello!] for (1) seconds
next costume
end
```
~~~



Make your sprite react to different events using events and control blocks. For example,
you can make your sprite bounce off the edge of the stage when it touches it using this script:

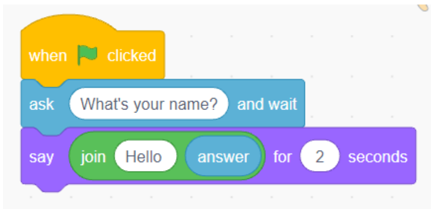
~~~
```scratch
when green flag clicked
forever
move (10) steps
if on edge, bounce
end
```
~~~



Think: How would the sprite move in this case?

Make your sprite interact with other sprites or the user using sensing and operators blocks. For example, you can make your sprite ask the user their name and greet them using this script:

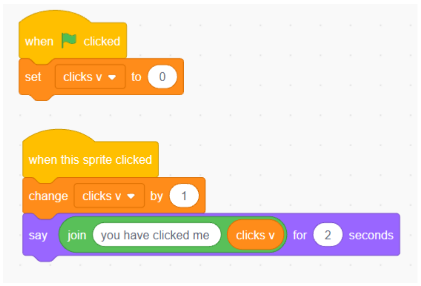
~~~
```scratch
when green flag clicked
ask [What is your name?] and wait
say (join [Hello ] (answer)) for (2) seconds
```
~~~



```
when clicked
ask What's your name? and wait
say join Hello answer for 2 seconds
```

Make your sprite use variables to store and change data using variables blocks. For example, you can make your sprite keep track of how many times it has been clicked using this script:

~~~
```scratch
when green flag clicked
set [clicks v] to [0]
when this sprite clicked
change [clicks v] by (1)
say (join [You have clicked me ] (clicks)) for (2) seconds
~~~
```
```



```
when clicked
set clicks v to 0

when this sprite clicked
change clicks v by 1
say join you have clicked me clicks v for 2 seconds
```

To Do: Create your own scripts by dragging and dropping blocks and analyze the sprite changes.

Explore complex coding concepts, including loops and conditionals

- Understand what loops and conditionals are and why they are useful in programming

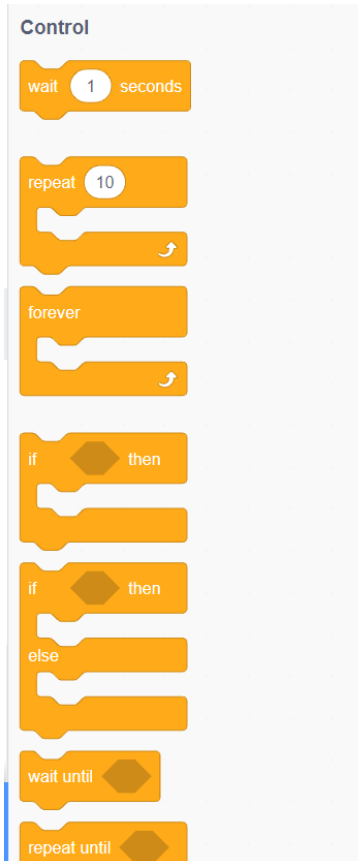
- Learn how to use the repeat, forever, and repeat until blocks to create loops

- Learn how to use the if, if else, and wait until blocks to create conditionals

- Experiment with different types of operators, such as arithmetic, logical, and comparison operators

- Apply loops and conditionals to create animations, games, and simulations

Loops and **conditionals** are two powerful concepts that can make your programs more efficient and interactive. Loops allow you to repeat a set of instructions multiple times, while conditionals allow you to execute different instructions depending on certain conditions. In this lesson, you will learn how to use loops and conditionals in Scratch, a visual programming language that lets you create animations, games, and simulations.



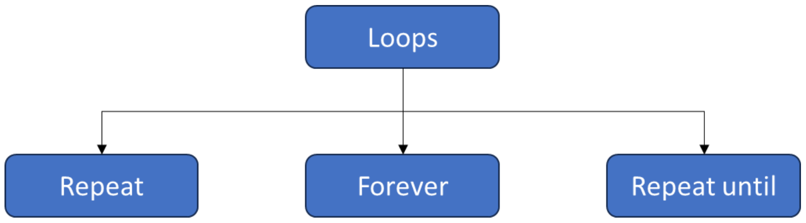
Understand what loops and conditionals are and why they are useful in programming

A **loop** block can be found in the control category. It is a way of repeating a set of instructions as many times as you want, or until a certain condition is met. For example, if you want to make a sprite move across the screen, you can use a loop to repeat the instruction of changing its x-position by a certain amount. Loops can save you time and make your code more concise and readable.

A **conditional** block can be found in the control category. It is a way of executing different instructions depending on whether a certain condition is true or false. For example, if you want to make a sprite say something when it touches another sprite, you can use a conditional to check if the touching condition is true, and then execute the instruction of saying something. Conditionals can make your programs more interactive and responsive to user input or events.

Learn how to use the repeat, forever, and repeat until blocks to create loops

Scratch has three types of blocks that can create loops: repeat, forever, and repeat until.



The **repeat block** allows you to specify **how many times** you want to repeat a set of instructions. For example, if you want to make a sprite spin 10 times, you can use a repeat block with 10 as the input, and put the instruction of turning 36 degrees inside the block.

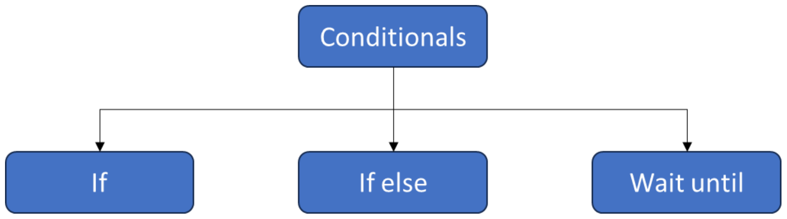
The **forever block** allows you to repeat a set of instructions **indefinitely**, or until you stop the program. For example, if you want to make a sprite bounce up and down continuously, you can use a forever block and put the instructions of changing its y-position and switching costumes inside the block.

The **repeat until block** allows you to repeat a set of instructions **until** a certain condition becomes true. For example, if you want to make a sprite move across the screen until it reaches the edge, you can use a repeat until block with the condition of touching the edge as the input, and put the instruction of changing its x-position inside the block.

Try yourself: Create scripts using repeat, forever, and repeat until and analyse the sprite outcomes.

Learn how to use the if, if else, and wait until blocks to create conditionals

Scratch has three types of blocks that can create conditionals: if, if else, and wait until.



The **if block** allows you to execute a set of instructions only if a certain **condition is true**. For example, if you want to make a sprite say "Hello" when it is clicked, you can use an if block with the condition of mouse down as the input, and put the instruction of saying "Hello" inside the block.

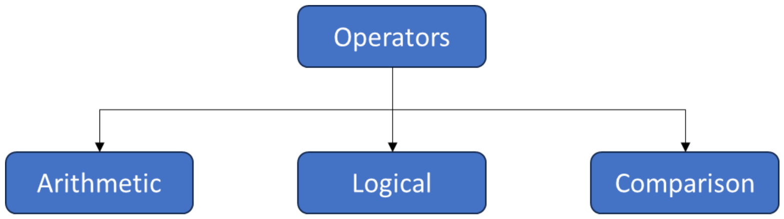
The **if else block** allows you to execute one set of instructions if a certain **condition is true**, and another set of instructions if the **condition is false**. For example, if you want to make a sprite change its color when it is clicked, and change it back when it is not clicked, you can use an if else block with the condition of mouse down as the input, and put the instructions of changing color effect inside the block.

The **wait until block** allows you to pause the execution of your program **until a certain condition becomes true**. For example, if you want to make a sprite wait for 3 seconds before moving, you can use a wait until block with the condition of timer > 3 as the input, and put the instruction of moving steps inside the block.

Try yourself: Create scripts using if, if else, and wait until and analyse the sprite outcomes.

Experiment with different types of operators, such as arithmetic, logical, and comparison operators

Operators are symbols or words that perform calculations or comparisons on values. Scratch has four categories of operators: arithmetic, logical, comparison, and string operators. In this lesson, we will focus on arithmetic, logical, and comparison operators.



Arithmetic operators are used to perform basic **mathematical operations** on numbers, such as addition (+), subtraction (-), multiplication (*), division (/), modulo (%), and exponentiation (^). For example, if you want to calculate the area of a rectangle with length 10 and width 5, you can use an arithmetic operator block with 10 * 5 as the input.

Logical operators are used to combine or negate **boolean values** (true or false), such as and (&), or (|), not (!), xor (^). For example, if you want to check if two sprites are touching each other or the edge of the screen, you can use a logical operator block with touching [sprite] or touching [edge] as the input.

Comparison operators are used to **compare two values** and return a boolean value (true or false), such as equal (=), not equal (!=), less than (<), greater than (>), less than or equal (<=), greater than or equal (>=). For example, if you want to check if a sprite's x-position is between -100 and 100, you can use a comparison operator block with -100 < x-position and x-position < 100 as the input.

Did You Know!

A **string** represents alphanumeric data. This means that a string can contain many different characters, but they are all considered as if they were text, even if the characters are numbers. A string can also contain spaces.

Try yourself: Create scripts using arithmetic, logical, and comparison and analyse the sprite outcomes.

Apply loops and conditionals to create animations, games, and simulations

Now that you have learned the basics of loops and conditionals, you can use them to create more complex and interesting programs in Scratch. For example, you can use loops and conditionals to make a sprite move in different directions, change its appearance, react to user input or events, interact with other sprites, and create various effects. You can also use loops and conditionals to create animations, games, and simulations that showcase your creativity and skills.

Here are some examples of projects that use loops and conditionals:

- A bouncing ball animation that uses a forever loop and an if else block to make a ball bounce off the edges of the screen.
- A guessing game that uses a repeat until loop and an if else block to make the user guess a random number between 1 and 10.
- A simulation of the solar system that uses a repeat block and an arithmetic operator block to make the planets orbit around the sun.

Begin creating interactive and dynamic projects using Scratch

- Learn how to use the broadcast and receive blocks to communicate between sprites
- Learn how to use the ask and answer blocks to get user input
- Learn how to use the variables and lists blocks to store and manipulate data
- Learn how to use the sound and pen blocks to add sound effects and drawings
- Design and develop your own creative projects using Scratch

`

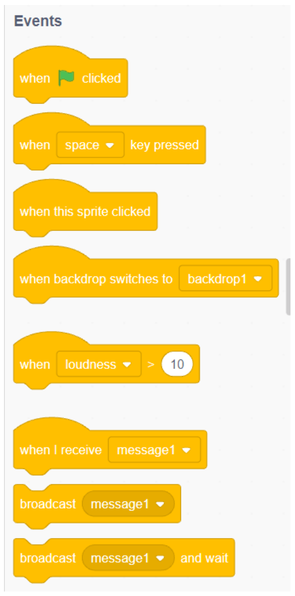
Scratch is a fun and easy way to create your own interactive and dynamic projects. In this section, you will learn how to use some advanced features of Scratch to make your projects more exciting and engaging. Here are some of the topics you will cover in this lesson:

Broadcast and receive blocks:

Sometimes, you may want to make different sprites do something at the same time, or in response to an event. For example, you may want to make a sprite say "Hello" when another sprite touches it, or make all the sprites change color when the green flag is clicked. To do this, you can use the broadcast and receive blocks. These blocks allow you to send and receive messages between sprites.

The **broadcast block** can be found in the Events category. It has a dropdown menu where you can **choose a message** to send, or type your own message. When you use this block, it will send the message to all the sprites in your project.

The **receive block** can also be found in the Events category. It has a hat shape, which means it starts a script **when something happens**. In this case, it starts a script when it receives a certain message. You can choose the message from the dropdown menu, or type your own message. When you use this block, it will run the script below it when it receives the matching message.



For example, if you want to make a sprite say "Hello" when another sprite touches it, you can use these blocks:

- In the first sprite, add a script that broadcasts a message (such as "greeting") when it is touched by another sprite. You can use the touching block from the Sensing category to check if the sprite is touching another sprite.
- In the second sprite, add a script that receives the message (such as "greeting") and says "Hello". You can use the say block from the Looks category to make the sprite say something.

Try it out and see what happens when you run your project.

Interesting Fact

- Not only the sprites but also the user can collaborate with other users on Scratch projects in real-time.

Ask and answer blocks:

Sometimes, you may want to get some input from the user who is playing your project. For example, you may want to ask them their name, their favorite color, or their opinion on something. To do this, you can use the ask and answer blocks. These blocks allow you to ask a question and get an answer from the user.

The **ask block** can be found in the Sensing category. It has a text input where you can **type** your question. When you use this block, it will show your question on the stage and wait for the user to type an answer.

The **answer block** can also be found in the Sensing category. It has a reporter shape, which means it returns a value that can be used in other blocks. In this case, it **returns the answer** that the user typed after using the ask block.

For example, if you want to ask the user their name and greet them, you can use these blocks:

- Add a script that asks "What is your name?" and waits for an answer.
- Add another script that says "Hello [answer]!" using the join block from the Operators category to combine "Hello" and the answer.

Try it out and see what happens when you run your project.

To Do: Create your own scripts and execute using ask and answer blocks.

Variables and lists blocks:

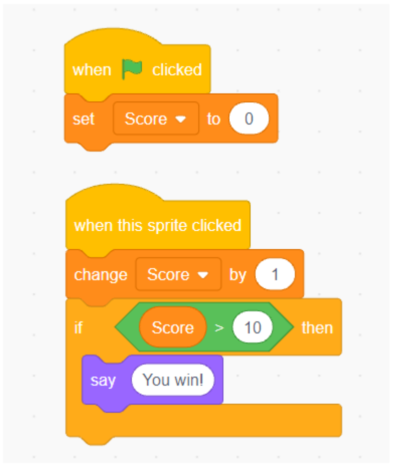
Sometimes, you may want to store some information or data in your project. For example, you may want to keep track of the score, the time, or some choices that the user made. To do this, you can use variables and lists. These are containers that can hold different types of values.

A **variable** can hold **one value** at a time, such as a number or a text. You can create your own variables by clicking on the Make a Variable button in the Data category. You can name your variable anything you want, and choose if it is for all sprites or for one sprite only. You can also choose if it is visible on the stage or not.

Once you create a variable, you will see some blocks related to it in the Data category. You can use these blocks to set the value of your variable, change it by adding or subtracting something, show or hide it on the stage, or get its value in other blocks.

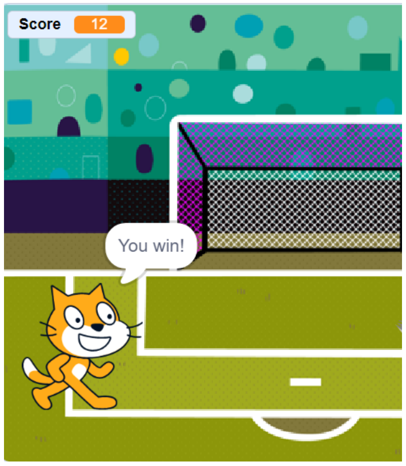
For example, if you want to keep track of the score in your project, you can create a variable called score and use these blocks:

- Set the score to 0 at the start of your project.
- Change the score by 1 when something happens (such as collecting an item or answering a question correctly).
- Show the score on the stage so that the user can see it.
- Use the score in other blocks (such as checking if it is greater than 10 or saying "You win!").



```
when [flag] clicked
set Score to 0

when this sprite clicked
change Score by 1
if Score > 10 then
    say You win!
```



A **list** can hold multiple values at a time, such as a **collection** of numbers or texts. You can create your own lists by clicking on the Make a List button in the Data category. You can name your list anything you want, and choose if it is for all sprites or for one sprite only. You can also choose if it is visible on the stage or not.

Once you create a list, you will see some blocks related to it in the Data category. You can use these blocks to add items to your list, delete items from your list, insert items at a certain position in your list, replace items in your list, show or hide your list on the stage, or get the value of an item or the length of your list in other blocks.

For example, if you want to store some choices that the user made in your project, you can create a list called choices and use these blocks:

- Add an item to choices when the user makes a choice (such as choosing a color or a character).
- Delete an item from choices when the user changes their mind (such as clicking on a different color or character).
- Show the choices on the stage so that the user can see them.
- Use the choices in other blocks (such as checking if they match a certain pattern or saying "You chose [choices]!").

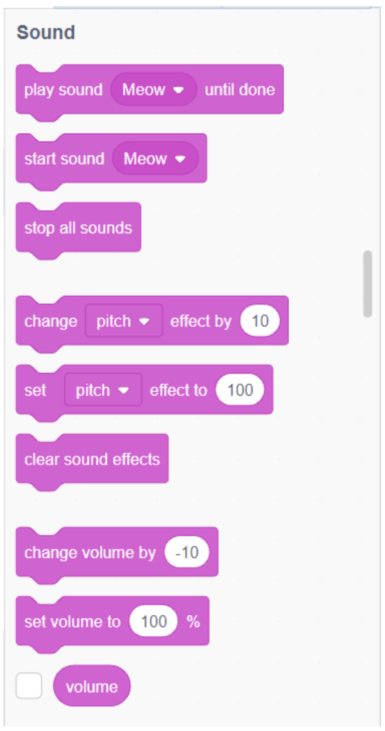
Try it out and see what happens when you run your project.

To Do: Create your own scripts and execute using variable and list blocks.

Sound and pen blocks:

Sometimes, you may want to add some sound effects or drawings to your project. For example, you may want to make a sprite play a sound when it does something, or draw a shape on the stage. To do this, you can use the sound and pen blocks. These blocks allow you to play sounds and draw lines with your sprites.

The **sound blocks** can be found in the Sound category. You can choose from some built-in **sounds**, or record your own sounds by clicking on the microphone icon. You can also import sounds from your computer by clicking on the folder icon.



Once you have some sounds in your sprite, you will see some blocks related to them in the Sound category. You can use these blocks to play a sound, stop all sounds, change the volume or pitch of a sound, or get the volume or loudness of a sound in other blocks.

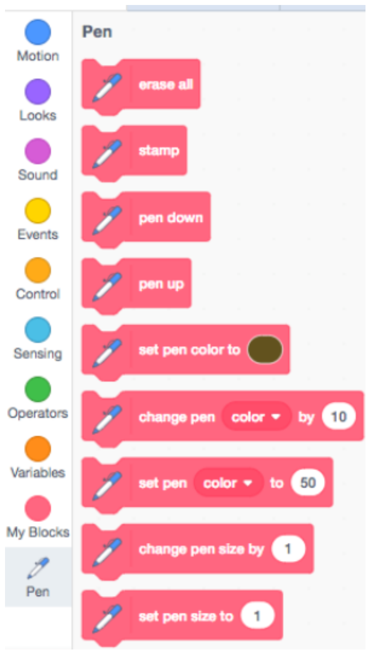
For example, if you want to make a sprite play a sound when it is clicked, you can use these blocks:

- Add a sound to your sprite (such as a meow or a beep).
- Add a script that plays the sound when the sprite is clicked. You can use the when this sprite clicked block from the Events category to start the script.

Try it out and see what happens when you run your project.

The **pen blocks** can be found in the Pen category. You can choose from different **colors and sizes for your pen**. You can also erase everything that you have drawn by clicking on the clear button.

Once you have chosen your pen settings, you will see some blocks related to them in the Pen category. You can use these blocks to put down or lift up your pen, draw a line with your pen, stamp your sprite on the stage, set the color or size of your pen, or change them by adding or subtracting something.



For example, if you want to make a sprite draw a square on the stage, you can use these blocks:

- Choose a color and size for your pen.
- Add a script that draws a square with your pen. You can use these steps:
    - - Put down pen
    - - Move 100 steps
    - - Turn 90 degrees
    - - Move 100 steps
    - - Turn 90 degrees

  - - Move 100 steps
  - - Turn 90 degrees
  - - Move 100 steps
  - - Turn 90 degrees
  - - Lift up pen

Try it out and see what happens when you run your project.

To Do: Create your own scripts and execute using sound and pen blocks.

Creative projects:

Now that you have learned how to use some advanced features of Scratch, you are ready to design and develop your own creative projects. You can use any combination of blocks from any category to make your projects more interactive and dynamic. You can also remix other projects from the Scratch community to get some inspiration and ideas.

Here are some suggestions for what you can create:

- A quiz game that asks questions and keeps score.
- A story that has different endings based on user choices.
- A drawing app that lets users create their own art.
- A music maker that plays different sounds and rhythms.
- A simulation that models something from real life (such as weather or gravity).

Have fun and be creative!

Test Your Learning:

1. Which category of blocks in Scratch allows you to control how your sprite moves on the stage?
   a) Looks
   b) Events
   c) Motion
   d) Sound

2. What does the "green flag" block in Scratch do?
   a) Changes the background color of the stage.
   b) Adds a new sprite to the project.

c) Starts the program when clicked.
d) Stops all sounds in the project.

3. What can be stored in a Scratch variable?
   a) Only numbers
   b) Only text
   c) Images
   d) Numbers, text, or Booleans

4. What type of operators are used to perform basic mathematical operations in Scratch?
   a) Arithmetic operators
   b) Logical operators
   c) Comparison operators
   d) String operators

5. Which block is used to check if a sprite is touching another sprite or the edge of the stage in Scratch?
   a) If
   b) Sensing
   c) Control
   d) Motion

6. What is the purpose of the "forever" block in Scratch?
   a) To repeat a set of instructions a specific number of times
   b) To pause program execution until a condition is met
   c) To execute a set of instructions continuously
   d) To create conditional statements

7. Which category of blocks in Scratch is used to control when a script starts or stops?
   a) Events
   b) Looks
   c) Control
   d) Sensing

8. In Scratch, what is the purpose of the "if else" block?
   a) To repeat a set of instructions indefinitely
   b) To execute different instructions based on a condition
   c) To create a loop that repeats a specific number of times
   d) To change the appearance of a sprite

9. What type of block is used to create a loop that repeats a certain number of times in Scratch?
   a) Forever
   b) Repeat
   c) Repeat until
   d) Wait until

10. What block is used to send and receive messages between sprites in Scratch?
    a) Motion
    b) Events
    c) Looks
    d) Broadcast

Answers:

1. c
   Motion blocks in Scratch are specifically designed to control the movement of sprites on the stage. They include commands like "move," "turn," and "go to."

2. c
   The "green flag" block is an event block that initiates the program when clicked, making it an essential starting point for Scratch projects.

3. d
   Scratch variables can store different types of data, including numbers, text, and Booleans, offering flexibility in programming.

4. a
   Arithmetic operators in Scratch are used for mathematical operations like addition, subtraction, multiplication, and division.

5. b
   The "sensing" category in Scratch includes blocks to check various conditions, including whether a sprite is touching another sprite or the stage's edge.

6. c
   The "forever" block in Scratch allows a set of instructions to be executed continuously, creating a looping behavior.

7. a
   The "Events" category in Scratch contains blocks for controlling script execution based on events like clicking the green flag or pressing keys.

8. b
   The "if else" block in Scratch allows you to execute different sets of instructions based on whether a condition is true or false.

9. b
   The "Repeat" block in Scratch is used to create a loop that repeats a specified number of times.

10. d
    The "broadcast" block is used to send messages to other sprites, and the "receive" block is used to react to those messages, enabling communication between sprites.