Computational Thinking Module - Scratch

Computational Thinking Team 2024/2025

Notes

1. This module is designed to guide Computational Thinking programming, then some materials may not exist because they are out of Computational Thinking scope.
2. You may open this module while you are doing the module exercise in practical class.
3. It is recommended to run all programs listed in this module in your computer, so you know the output of the programs.
4. It is better to experimenting on the programs listed in this module to grasp better understanding about what your programs really do.
5. You are strongly recommended to learn programming from other sources or materials and explore your programming language.

Contents

1 Introduction to Scratch

1.1 Introduction

Welcome to the first chapter of this practical module! In this chapter, we will explore the brief history of Scratch, a fun and easy-to-learn programming platform.

1.2 Brief History of Scratch

Scratch is a programming environment specifically designed to teach the basics of programming. It was developed by the Lifelong Kindergarten Group at MIT (Massachusetts Institute of Technology). The main goal of Scratch is to help people, especially children and beginners, learn how to think creatively, solve problems, and collaborate through programming.

Over time, Scratch has continued to evolve. Scratch 2.0 was released in 2013, bringing significant improvements to the user interface and new features. Scratch 2.0 allowed users to create more complex projects with a wider range of coding blocks.

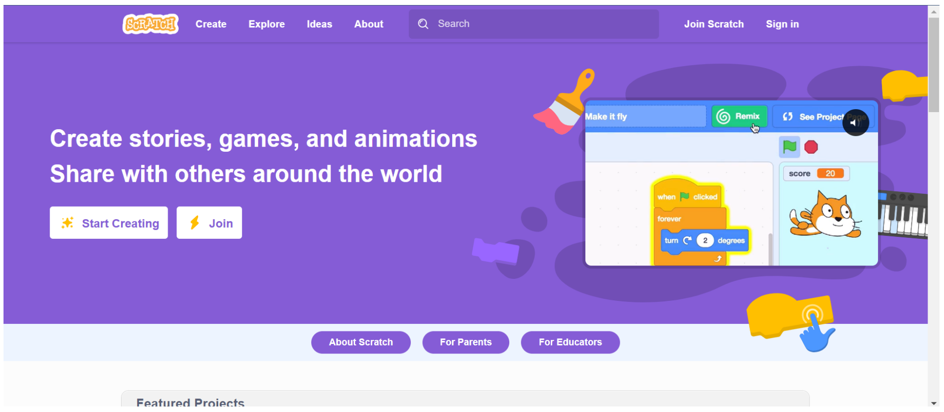
In 2019, Scratch 3.0 was launched, featuring a more modern design and additional new features. This made Scratch easier to use and more appealing to new users.

Scratch is used by a wide audience, including children at school and home, teachers, and even adults who want to learn programming. With its user-friendly interface and intuitive features, Scratch enables anyone to create their own creative projects through programming.

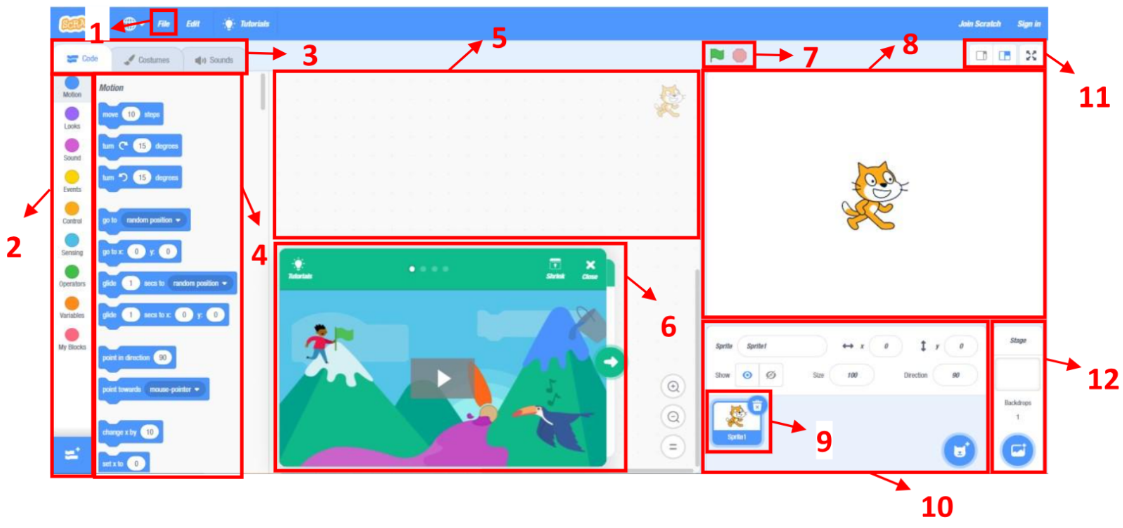
1.3 Introduction to the Scratch Workspace and Features

Scratch offers various features that allow users to create animations, games, and other creative projects easily.

To start a project, you must first visit http://scratch.mit.edu/, then click the Start Creating button.



After that, a new page will open, and the features of the new page will look something like this:

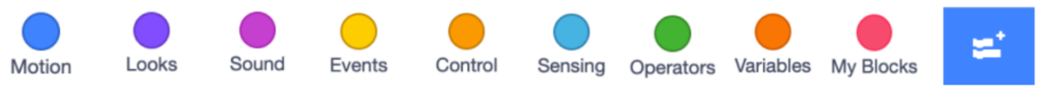


Descriptions:

1. **File**: used to save the user's project and upload previously worked projects.
2. **Code tab**: contains interactive code categories that can be used.
3. **Code – costumes – sounds tab**: the code tab displays code categories, the costumes tab is used to edit the sprite's costumes, and the sounds tab is used to edit the sprite's sounds.
4. **Code puzzle tab**: contains code blocks that can be drag-and-dropped and are color-coded according to their category.
5. **Editor area**: this is where code sequences for the project are placed. The sprite being worked on appears at the top right of the editor area.
6. **Tutorial tab**: contains Scratch tutorials that help users work on projects. This tab can be moved, shrunk, or closed.
7. **Start and stop buttons**: the green button is used to start the program, while the red button is used to stop all commands.
8. **Results area**: this is where the program from the editor area is visually executed.
9. **Sprite**: characters or objects used in the user's project. The blue outline around the sprite box indicates the sprite currently selected.
10. **Sprite configuration**: used to edit the settings for the sprite, such as name, position, size, direction, visibility, and a blue button to add a sprite.
11. **Deploy area configuration**: used to minimize, maximize, and make the deploy area full screen.
12. **Stage tab**: used for setting the background in the program. The blue button is used to add or change the background in the program.

1.4 Introduction to Code Categories

Code categories can be seen on the far left of the workspace or in number 2 (Code tab) in the workspace introduction. These code blocks are divided into several categories:



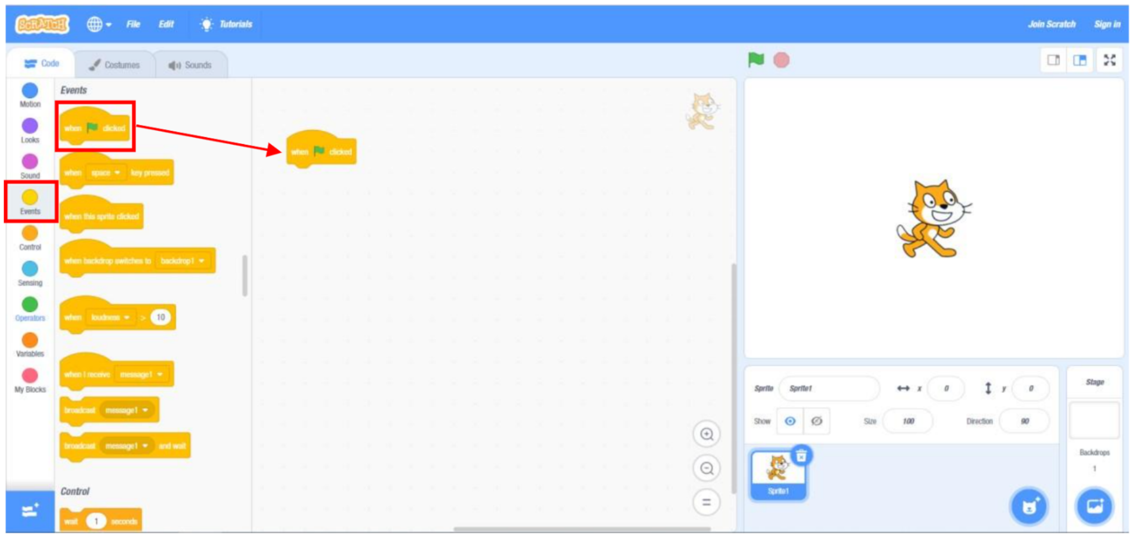
Descriptions:

1. **Motion**: contains blocks used to move sprites, changing their position in the deploy area.
2. **Looks**: used to change or add visuals to the sprite and its surroundings, such as adding a ″hello″ bubble, changing the color, changing the costume, size, or backdrop.
3. **Sound**: used to add sound to the sprite, modify volume, and pitch.
4. **Events**: used to add triggers for starting a code block, such as ″when green flag clicked″ or ″when x key pressed.″ The trigger button can be customized.
5. **Control**: controls the sequence of code execution, such as wait x seconds, repeat x times, forever, if x then y else z, etc.
6. **Sensing**: enables interaction with the user, such as detecting mouse touches, asking for names, or detecting specific activities.
7. **Operators**: performs mathematical operations.
8. **Variables**: manages custom variables created by the user.
9. **My Blocks**: allows users to create custom blocks based on their code.
10. **Add Extension**: adds additional features beyond the default functions of Scratch.

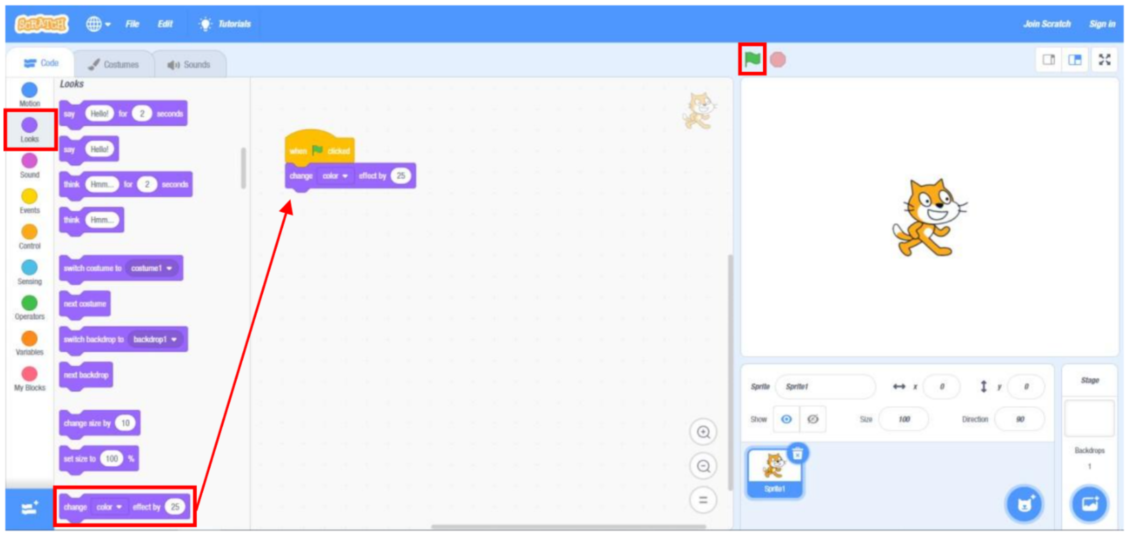
2 Starting a Program

To start a program, first, choose a sprite to use. By default, the Scratch cat sprite will be selected.

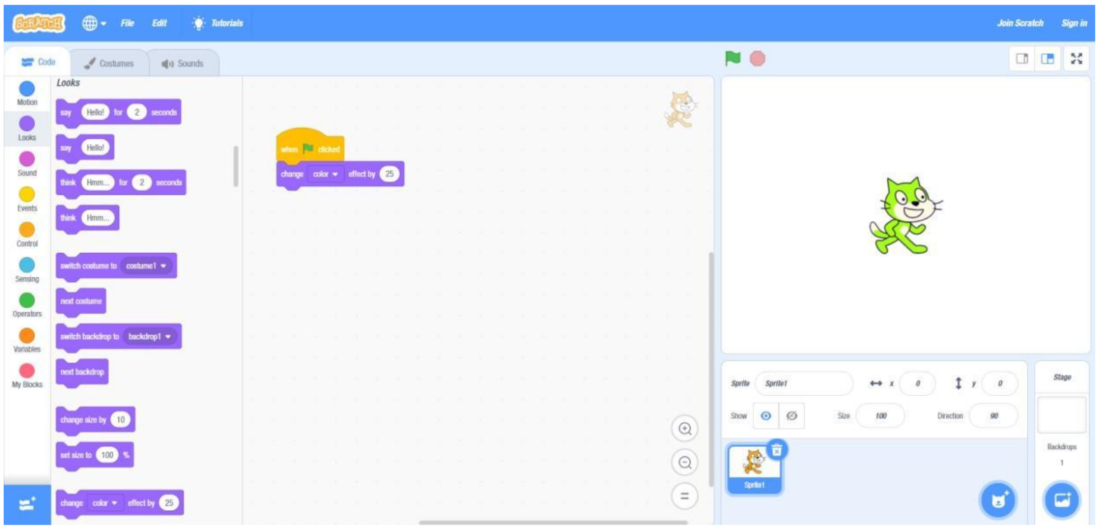
Next, drag and drop the desired code blocks from the code puzzle tab. Typically, the sequence starter block is placed first. In this case, we will use the “when green flag clicked” command. First, click the “Event” button, then drag-and-drop the block (see the figure below).



Once the sequence starter block is placed, you can add any block as the next step. As an example, we will use the “change color effect by 25” block. First, click the “Looks” button, then drag-and-drop the block below the first one. After placing the block, click the green flag to run the program (see the figure below).


Looks
say Hello! for 2 seconds
say Hello!
think Hmm... for 2 seconds
think Hmm...
switch costume to costume1
next costume
switch backdrop to backdrop1
next backdrop
change size by 10
set size to 100 %
change color effect by 25
when clicked
change color effect by 25

Here is the program after running.

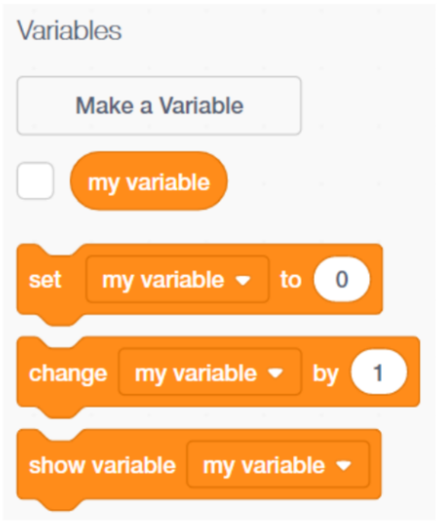


3 Input and Output

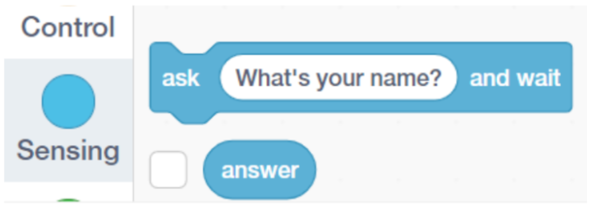
Input is the way to enter data or information into a program. In Scratch, there are several ways to get input from the user. Meanwhile, output is the result or response displayed by the program to the user.

Input and Output are important concepts in programming, and Scratch provides various ways to interact with the user and generate interesting output. By using programming blocks, sensors, and variables, you can create interactive and responsive programs in your Scratch project.

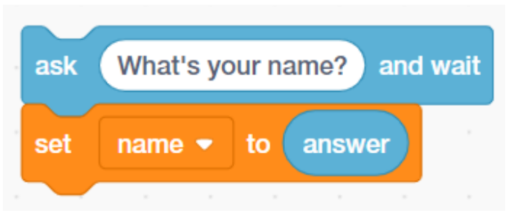
To receive input, we need a variable to store the input data. To create a variable, use the "Make a Variable" option. The created variable will appear below it.



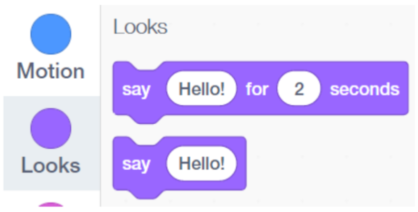
To receive input, use the "Ask (...) and wait" code. The provided input will be temporarily stored in the "answer" variable.



For example, below is a program that receives a name input and stores it in a variable called "name."



To display the output, use the "say (...)" code.

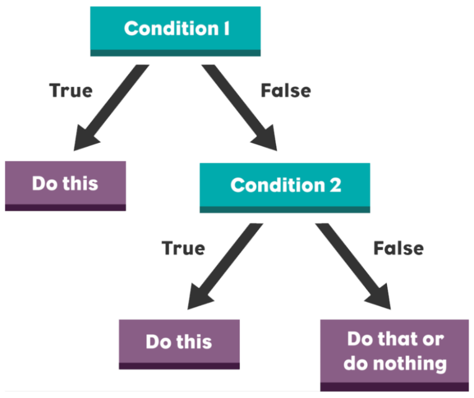


4 Branching in Scratch

Branching is one of the fundamental concepts in programming used to make decisions based on certain situations. In Scratch, we can use the "if" and "else" blocks to implement branching.

Branching allows our program to execute different commands depending on a condition. The condition can either be true or false.

We can visualize branching in a program with the image below.



5 Loops in Scratch

Loops are a concept used in programming to execute a set of instructions repeatedly. In Scratch, we can use two types of loops: ”Repeat” and ”Repeat until.”

1. **Repeat (...)**:
   The ”Repeat” block is used when we know how many times the repetition should occur. We can specify the number of repetitions by filling in the blank inside the ”Repeat” block. For example, if we want to execute a set of instructions 5 times, we can enter the number 5 inside the ”Repeat” block.
2. **Repeat until (logical expression)**:
   The ”Repeat until” block is used when we don't know how many times the repetition should occur, but we know the condition that will stop the repetition. The stopping condition is determined by a logical expression. The repetition will continue until the condition is met or becomes true.

In both types of loops, the action or instructions to be repeated can be placed inside the loop block. For example, we can move the sprite to a position, change the color, or perform other actions in each repetition.

Loops are very useful in various situations, such as for performing repeated mathematical calculations, controlling sprite movements in games, or repeating user interactions.

6 Learn More

To explore the topics covered in this module and develop a deeper understanding of Scratch, here are some recommendations:

1. **Scratch Documentation**: Explore the official Scratch documentation on their website to deepen your understanding of basic Scratch concepts, types of code blocks, and more complex project development.
2. **History and Origins**: Look for literature or articles to learn more about the history and origins of Scratch and its influence on educational programming.
3. **Tutorials and Videos**: Use Scratch tutorials and video tutorials to better understand Scratch features and the workspace.
4. **Practical Exercises**: Create small projects that combine different categories of code to master each category. You can also explore projects from the Scratch community.

By utilizing these resources, you will be able to explore Scratch and enhance your programming skills.