Introduction to Computer Science with Scratch

BEAUTY AND JOY OF COMPUTING

- BJC GROUP

Course Roadmap

- In this course, we will begin by introducing Computer Science, big ideas of computing, define the computational terms and get the gears rolling by going through projects using Scratch, a visual programming language.

Computer Science

- Computer Science is the study of computers and computational systems, focusing on data processing, software, hardware, security and algorithms.

- Some of the areas in computer science: artificial intelligence, system design, data analysis, software engineering and cybersecurity.

- It's integral in today society as it creates tools to simplify complex real-world problems, automate repetitive tasks, provide entertainment and connect people across the globe. It impacts various fields such as (education, healthcare, entertainment and a lot more..

Fields in Computer Science

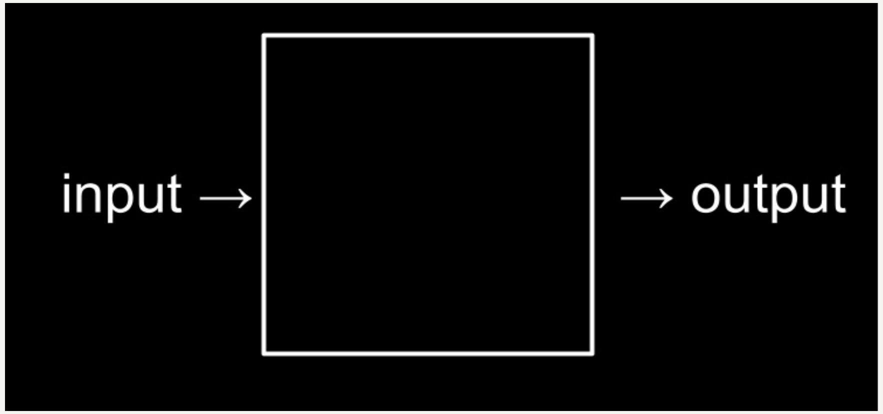
- Human-Computer Interaction (HCL): Study of the interaction between users and computers, ensuring usability and accessibility.
- Artificial Intelligence: Development of tools and technologies that able to perform tasks traditionally related to humans.
- Data Science: Applying tools and techniques of extracting insightful information from an organization's data.
- Software Engineering: Practice of building and maintaining software systems, focusing on efficiency, optimization and robustness.
- Embedded Systems: Programming hardware systems that operate in real-time environments such as IoT and microcontrollers.

Abstraction in Computing

- Abstraction is one of the big ideas in computing and computational thinking.
- Applying abstraction is simplifying complex problems by focusing on the essential elements of the problems and hiding the extra details.
- Two parts of abstraction:
  - Generalization: Formulating general ideas by abstracting common properties of an instance
  - Detail Removal: Removing or hiding irrelevant details from a system to focus on essential aspects that make the problem easier to understand and solve.

Computational Thinking

- Computer Science is all about problem solving!
- Programming is the act of giving computers a set of instructions to execute.
- We apply abstraction in programming by creating a black box program.
- A black box is like a function which takes in a set of inputs, evaluates and returns the expected output.



Building Blocks in Programming

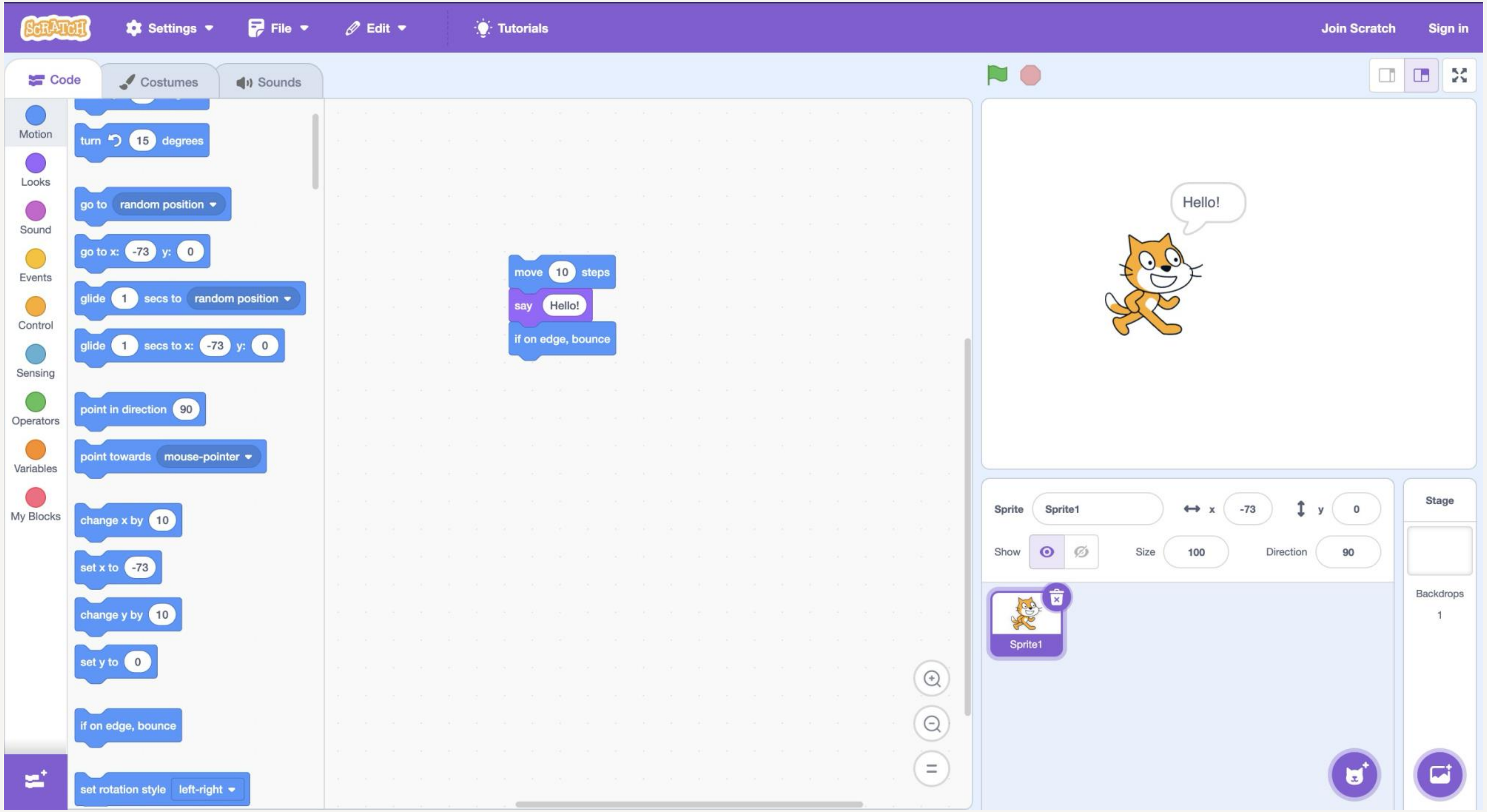
- Variables: Containers for storing data values.
- Conditions: Logical statements that control the flow of a program based on True or False.
- Loops: Used to repeat a block of code multiple times.
- Functions: Blocks of reusable code that perform a specific task.
- Data Structures: Ways to organize and store data efficiently.

Visual Programming

- Visual Programming (aka Block Coding) is a programming paradigm that allows us to create programs by manipulating programmed blocks instead of having to specify through text.
- Visual programming languages come packaged with a GUI (Graphic User Interface) and Drag-and-Drop features making it easier to start developing.
- In this course, we will be building visual programs using Scratch, a visual programming language developed by MIT lab.
- Scratch allows you to apply the building blocks we covered without worrying about the programming language syntaxes like curly braces. With this abstraction, you can get an immediately get a feel of programming.

Introduction to Scratch

- Scratch's IDE(Integrated Develompent Environment) looks like this!

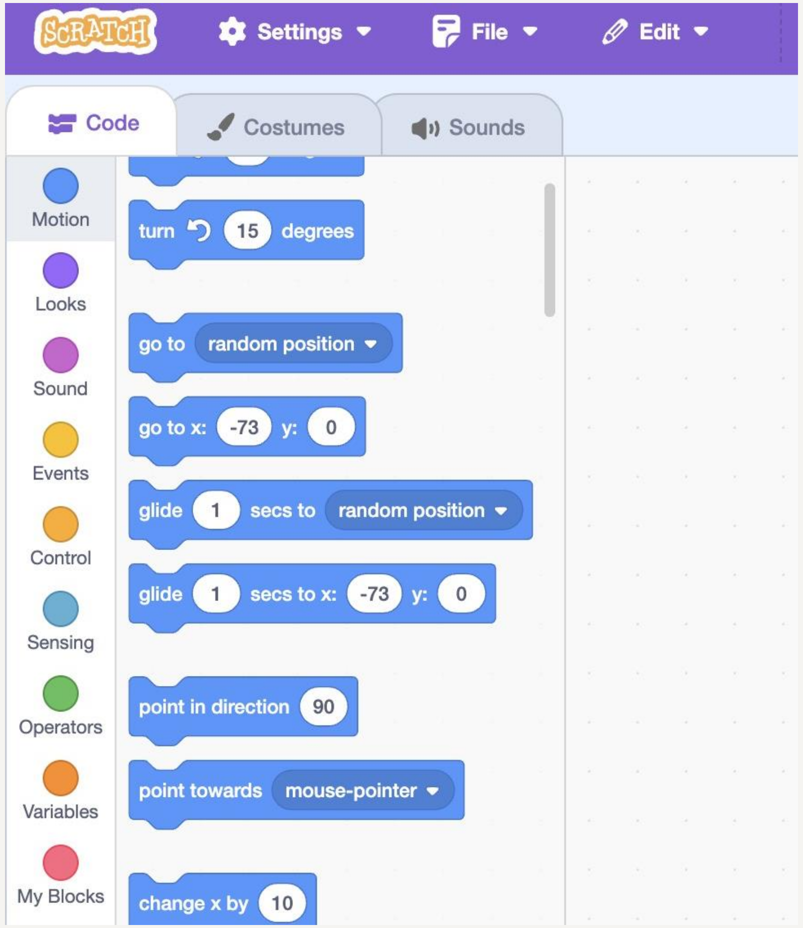


```
move 10 steps
say Hello!
if on edge, bounce
```

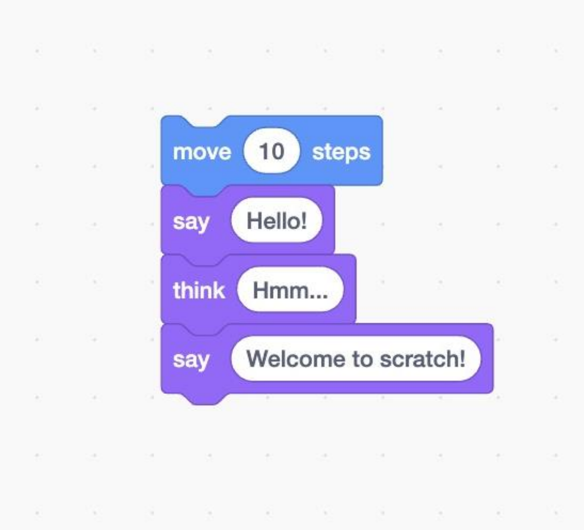
Sprite: Sprite1 x: -73 y: 0

Show Size: 100 Direction: 90

Introduction to Scratch

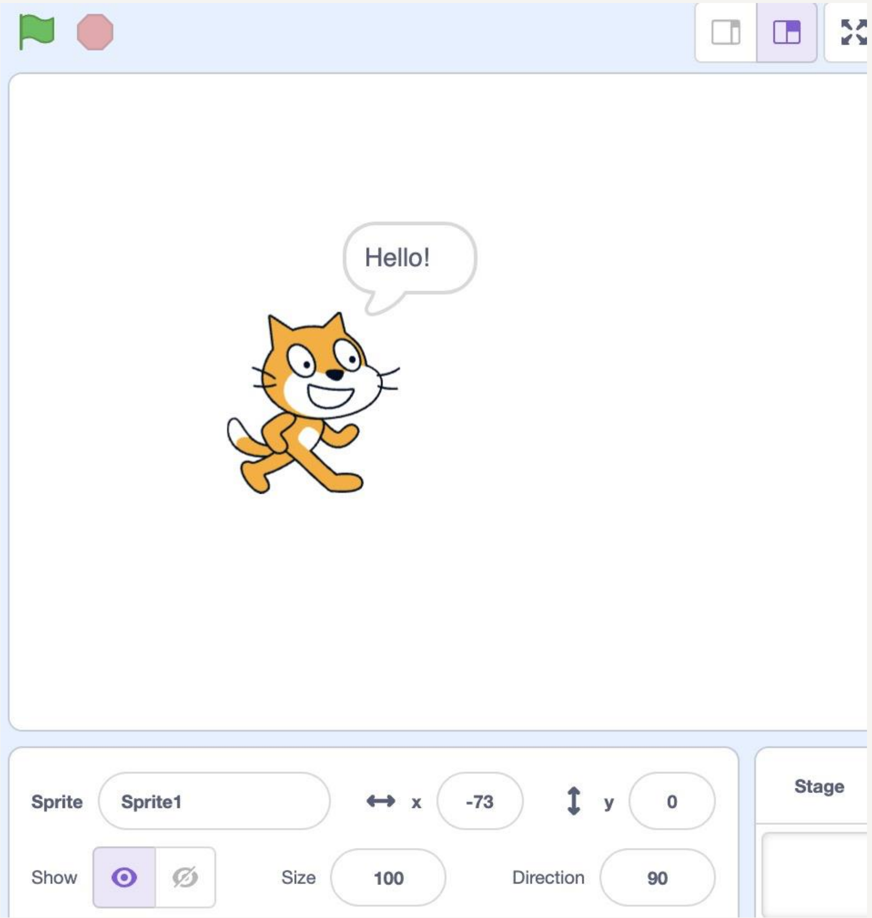


Drag-and-Drop menu with program blocks



```
move 10 steps
say Hello!
think Hmm...
say Welcome to scratch!
```

Interactive GUI for arranging blocks into intended functionality



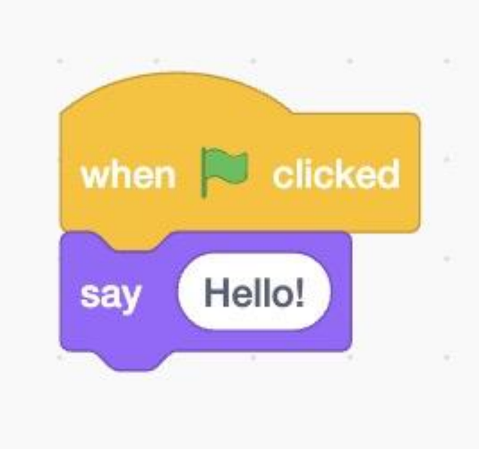
Sprite: Sprite1 x: -73 y: 0

Show Size: 100 Direction: 90

Display area for the sprite when the code is executed

Introduction to Scratch

- To get started, go to website: https://scratch.mit.edu/projects/editor/?tutorial=getStarted
- Drag the "When green flag is clicked" block from the programming area. Then, drag "Say Hello!" Block into the programming area and connect it with the "When green flag is clicked" just like you would connect Lego blocks.

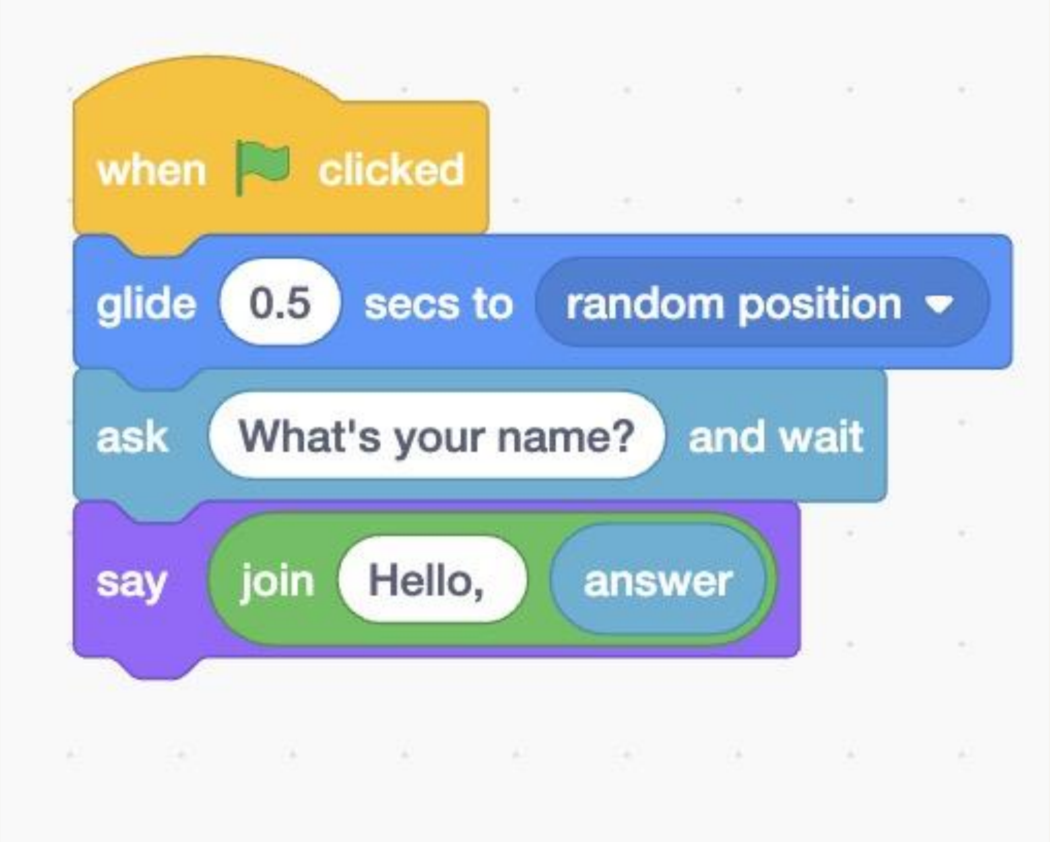


```
when 🏁 clicked
say Hello!
```

- When you click on the green flag, notice that the cat sprite would show a bubble saying "Hello!"

Introduction to Scratch

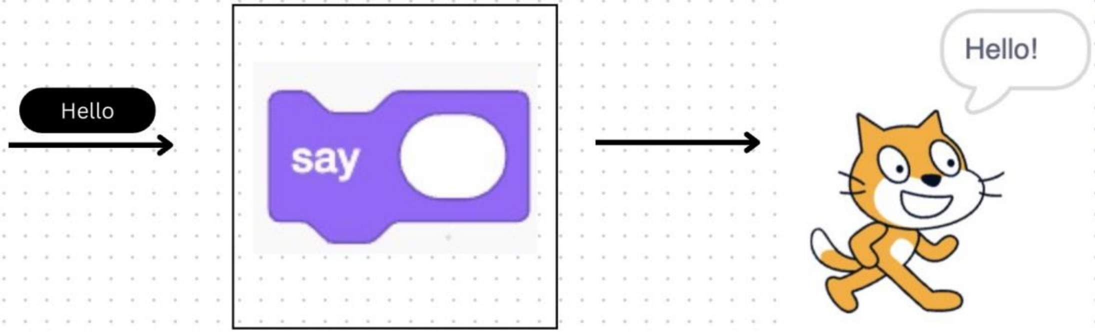
- You can make the program more interactive by having ask for name (get input) and greeting someone.
- The program asks, "What's your name" and waits for a response. It stores that response in a **variable** called answer.
- We then concatenate "Hello" with the answer by using the join block.



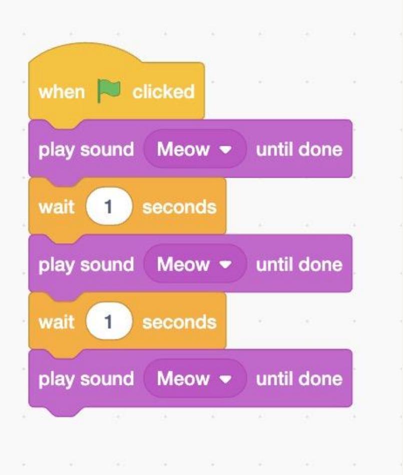
```
when 🏳 clicked
glide (0.5) secs to (random position ▾)
ask (What's your name?) and wait
say (join (Hello,) (answer))
```

Computational Abstraction in Scratch

- So far we have applied abstraction by having the say block do the functionality for displaying the bubble with the "Hello".



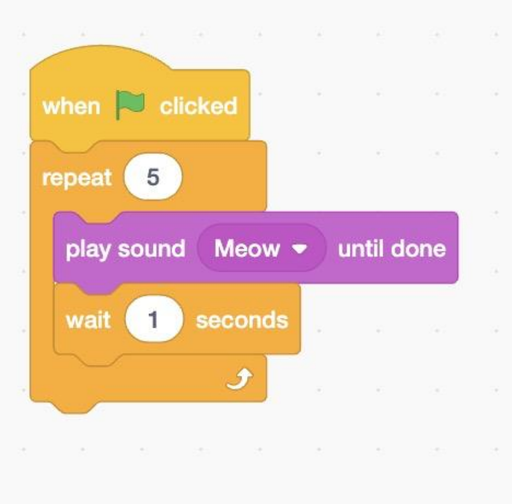
- Here is a program that repeats playing "Meow".



```
when 🏁 clicked
play sound Meow ▾ until done
wait 1 seconds
play sound Meow ▾ until done
wait 1 seconds
play sound Meow ▾ until done
```

- Notice it becomes tedius if we want to continue saying "Meow".

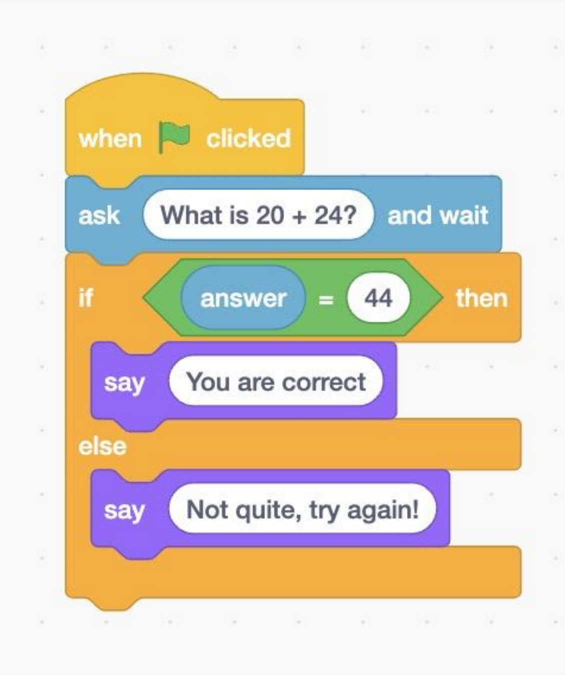
We can apply abstraction by having it loop or repeat the sequence of blocks.



```
when 🏁 clicked
repeat 5
  play sound Meow ▾ until done
  wait 1 seconds
↻
```

Computational Abstraction in Scratch

- We can also apply conditionals to our programs.
- If we want to ask a simple math question, we could use the if-else block.
- The if-else block takes in an equality statement and two conditions that are executed depending on the equalities result.



```
when 🏁 clicked
ask (What is 20 + 24?) and wait
if <(answer) = (44)> then
  say (You are correct)
else
  say (Not quite, try again!)
```

Project Challenge and Tips

- Using the building blocks we have covered, develop a small project of interest
- Requirements: Use loops and if-else statements.
- Tip: It might help to first write the high-level step by step logic explanation of the program using english (Pseudocode)

Checkout these cool projects built with Scratch

Ivy's Hardest Game - Harvard