



INTRODUCTION TO CODING

GRADE VI

Student Handbook

Version 1.0



INTRODUCTION TO CODING

GRADE VI

Student Handbook



ACKNOWLEDGEMENT

Patrons

- Sh. Ramesh Pokhriyal 'Nishank', Minister of Human Resource Development, Government of India
- Sh. Dhorte Sanjay Shamrao, Minister of State for Human Resource Development, Government of India
- Ms. Anita Karwal, IAS, Secretary, Department of School Education and Literacy, Ministry Human Resource Development, Government of India Advisory

Editorial and Creative Inputs

- Mr. Manuj Ahuja, IAS, Chairperson, Central Board of Secondary Education

Guidance and Support

- Dr. Biswajit Saha, Director (Skill Education & Training), Central Board of Secondary Education
- Dr. Joseph Emmanuel, Director (Academics), Central Board of Secondary Education
- Sh. Navtez Bal, Executive Director, Public Sector, Microsoft Corporation India Pvt. Ltd.
- Sh. Omjiwan Gupta, Director Education, Microsoft Corporation India Pvt. Ltd
- Dr. Vinnie Jauhari, Director Education Advocacy, Microsoft Corporation India Pvt. Ltd.
- Ms. Navdeep Kaur Kular, Education Program Manager, Allegis Services India

Value adder, Curator and Co-Ordinator

- Sh. Ravinder Pal Singh, Joint Secretary, Department of Skill Education, Central Board of Secondary Education



ABOUT THE HANDBOOK

Coding is a creative activity that students from any discipline can engage in. It helps to build computational thinking, develop problem solving skills, improve critical thinking and exposure to real life situations to solve problems in various realms.

Therefore, CBSE is introducing 'Coding' as a skill module of 12 hours duration in classes VI-VIII from the Session 2021-2022 onwards. The idea is also to simplify the coding learning experience by nurturing design thinking, logical flow of ideas and apply this across the disciplines. The foundations laid in the early years will help the students to build the competencies in the area of AI, data sciences and other disciplines.

CBSE acknowledges the initiative by Microsoft India in developing this coding handbook for class VI students. This handbook introduces concepts of coding and computational thinking using real life examples and block coding with open source MakeCode platform. It uses gamified learning approach to make learning experience more engaging. The book is intuitive with practical examples of theoretical concepts and applied exercises. There are mini projects that students can work on. Additionally, the handbook also focuses on creating exposure to ethics of coding and promotes empathy among students by activities curated to demonstrate empathy and sensitivity.

The purpose of the book is to enable the future workforce to acquire coding skills early in their educational phase and build a solid foundation to be industry ready.



RESOURCES FOR STUDENTS

Minecraft education edition

Minecraft education edition is a game-based learning platform that promotes creativity, collaboration, and problem-solving in an immersive digital environment. This platform provides a fun way of learning coding and design thinking concepts. Visit <https://education.minecraft.net/> for more details.

MakeCode

Microsoft MakeCode is a free, open source platform for creating engaging computer science learning experiences that support a progression path into real-world programming. It brings programming to life for all students with fun projects, immediate results, and includes both block and text editors for learners at different levels. Visit <https://www.microsoft.com/en-us/makecode> for more details.

GitHub

GitHub is a storehouse where you can manage and collaborate on your code. It helps to maintain different versions of the code easily. GitHub Student Developer Pack gives students free access to the best developer, web development, gaming and many other tools at no cost enabling practical learning.

Sign up for the GitHub Student developer pack here

https://education.github.com/discount_requests/student_application?utm_source=2021-06-11-cbse



TABLE OF CONTENTS

Table of Contents	I
Ethical practices in coding.....	1
Introduction to Coding.....	2
1.1 What will you learn in this chapter?	2
1.2 How do traffic lights work?.....	2
1.3 Where else do we see applications of coding?.....	2
1.4 What exactly is coding?.....	2
1.5 What is a programming language?.....	3
1.6 Quiz Time.....	4
1.7 What did you learn in this chapter?.....	5
Algorithms With Block Coding.....	6
2.1 What will you learn in this chapter?	6
2.2 Searching for a word in the dictionary.....	6
2.3 Flowcharts.....	7
2.3.1 What is a Flowchart?.....	7
2.3.2 Symbols used in a flowchart.....	7
2.3.3 Benefits of using a flowchart.....	8
2.4 Activity: Profit and loss with flowcharts	8
2.5 Pseudocode.....	8
2.5.1 What is a Pseudocode?	8
2.6 Activity: Profit and loss with pseudocode	9
2.7 Getting started with block coding.....	9
2.8 Quiz time.....	13
2.9 What did you learn in this chapter?.....	18
Variables using block coding.....	19
3.1 What will you learn in this chapter?	19
3.2 What are variables?.....	19



3.3	Naming variables	19
3.4	Data types in variables.....	20
3.5	Performing Operations on Variables.....	23
3.6	Activity: Addition operation using block coding.....	24
3.7	Quiz time.....	34
3.8	What did you learn in this chapter?.....	36
	Control with conditionals.....	37
4.1	What will you learn in this chapter?	37
4.2	Arranging blocks.....	37
4.3	And operator.....	37
4.4	OR Operator.....	38
4.5	NOT Operator.....	39
4.6	Combining logical operators.....	39
4.7	Quiz time.....	40
4.8	Relational operators.....	40
4.9	Activity: Are you a teen?.....	40
4.10	Activity: Dynamic backgrounds.....	41
4.11	Nested Conditional Statements	46
4.12	Activity: The remainder problem.....	46
4.13	Quiz time.....	50
4.14	What did you learn in this chapter?.....	53
	Loops using block coding.....	54
5.1	What will you learn in this chapter?	54
5.2	Introduction to loops	54
5.3	Increment Loops	54
5.4	Different types of loops.....	55
5.5	Activity: Building a music player.....	58
5.6	Entry Criteria	61
5.7	Exit Criteria	61
5.8	Break Statement.....	62
5.9	Continue Statement.....	63
5.10	Activity: A tale of two villages.....	63



5.11	Quiz time.....	68
5.12	What did you learn in this chapter?.....	72
	References	73



ETHICAL PRACTICES IN CODING

As you build capabilities around coding, you will be equipped to build software on your own, which will have an impact on society in general. So, it is very important to adhere to ethical practices while building your own code. Below are some practices you must remember as you keep learning to code.

Contribute to society and human wellbeing

- You must limit negative results of software, including dangers to safety, health, personal security, and privacy
- Do consider the aftereffects of the software. Ensure your Code respects diversity and is utilized responsibly with social issues in mind
- In addition to this, promote environmental sustainability both locally and globally

Avoid harm to others

- Your code should not cause physical or mental injury, unjustified destruction to property or information
- Avoid unjustified damage to reputation and environment



Chapter
1

INTRODUCTION TO CODING

1.1 What will you learn in this chapter?

Do you want to know what code is? How is code applied in real life, and how does it impact our day-to-day activities? Welcome to the introductory section on coding.

Here you will learn:

- Real world application of coding
- How coding impacts our daily lives
- What exactly is coding in context of computer science

1.2 How do traffic lights work?



Have you ever wondered how traffic signals function? The lights cycle through green, yellow, and red at regular intervals to control road intersections' traffic flow. They prevent accidents and help to avoid congestion on the roads.

However, how do the traffic lights change automatically?

Few lines of code running in the background drive the traffic lights. At regular intervals, the code changes the

traffic signals to show different colors. Sometimes it is even smarter, where the code detects congestion based on sensors and maximizes efficiency by only functioning when traffic is present.

1.3 Where else do we see applications of coding?

Most of us knowingly or unknowingly engage with programming, be it inside our homes or outside. Coding, in the modern world, can be seen on the streets, at the schools, at the local grocery stores, etc.

Some of the practical examples of coding in the real world are:

- Interaction with bar-code scanners at shopping store
- Automatic control of traffic using traffic lights
- Booking movie, bus, train, flight tickets online
- Printers
- Computer software we use like web browser, Word etc.
- Video games and animations for entertainment

1.4 What exactly is coding?

Coding, also referred to as programming, is creating instructions that can be executed on a computer to



perform a specific task or achieve a particular result.

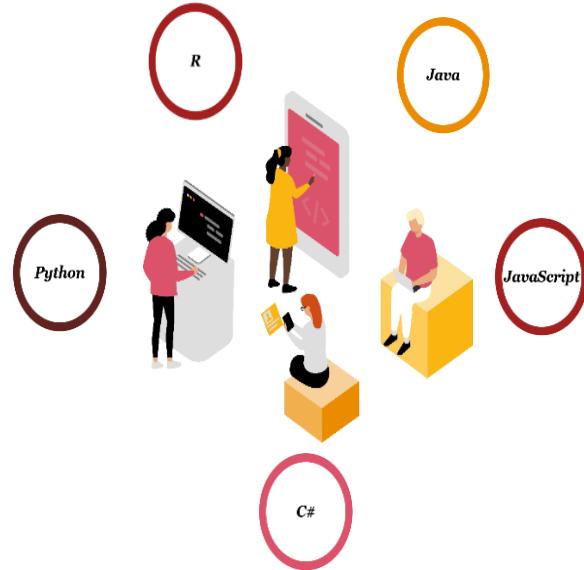
Coding is just like solving a math problem. There may be many ways to solve a problem. Similarly, there could be more than one way to write

code for the same task. Just like solving any other problem, some coding approaches are more efficient than others.

Think you are playing a video on your smartphone. Your phone is like a computer that needs to be instructed on what to be done. The app playing the video provides this instruction. This video-playing app is an example of coding. But how does the app communicate the instructions to the phone? It does via a programming language. In the next section, we will get to know more about programming languages.

1.5 What is a programming language?

Think about how we communicate with people around us? Language is our primary means of communication for all human interactions. Similarly, we can interact with computers via a language that computers understand. This language is called a programming language. Using programming languages, we can provide instructions to a computer to perform a set of activities. These sets of instructions are also called programs. Like any other language that has grammar, programming languages have syntax.



Some of the most frequently used ones are:

- Python
- Java
- JavaScript
- C#
- R
- C++
- C
- F#
- COBOL

Syntax is a set of rules that we need to follow when we write a computer program.

There are hundreds of programming languages used around the world. And new ones are getting developed all the time. Every programming language has its own syntax. But all programming languages have one common thing: they are eventually converted into a language that the computer will understand.



Throughout this chapter, we will learn about different programming techniques and how to apply them. Now, that we

know a little bit about coding and its applications, let us try to do some exercises.

1.6 Quiz Time

Question 1	For a given problem, there is always only one way to write a program
Option 1	True
Option 2	False

Question 2	Pin authentication for ATM card transaction is an example of programming
Option 1	True
Option 2	False

Question 3	Code is a set of instructions that can be executed on a computer to perform a specific task
Option 1	True
Option 2	False

Question 4	Which among the below are examples of programming in real life?
Option 1	Robots
Option 2	Computer Games
Option 3	Self-drive cars
Option 4	All the above

Question 5	Which among the below is not an example of programming language?
Option 1	Python
Option 2	English
Option 3	JavaScript



1.7 What did you learn in this chapter?

So, we are at the end of the introductory chapter for coding. By now, you should understand

- How coding is being utilized in everyday life to perform complex tasks in an easy manner
- What are the real-life applications of coding?
- What is exactly meant by the term coding in context of computer science?
- Now you know names of some of the most popular programming languages



Chapter
2

ALGORITHMS WITH BLOCK CODING

2.1 What will you learn in this chapter?

Now that we already have a fair understanding of coding applications in real life, let us understand how to solve a coding problem in detail. By the end of this chapter, you will learn:

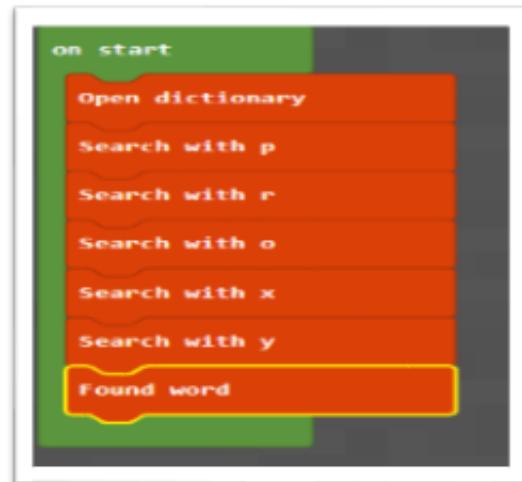
- What does the term algorithm mean?
- What is a flowchart?
- Applications of flowchart
- Get introduced to pseudocode

2.2 Searching for a word in the dictionary

While reading a book in a school library, Mukesh comes across a word say 'proxy' whose meaning he does not know. So how does he find out the meaning of this word?

The simple answer to this is that Mukesh searches for the meaning of the word 'proxy' in a dictionary. However, there are many words in a dictionary. So

how does he find that word 'proxy' in the



dictionary?

To achieve this, he first needs to find the dictionary section with the first letter of the word, which in this case is 'p.' Then, within the list of words starting the first letter 'p', he needs to find the section having the second letter of the word 'r'. He needs to do this operation again with the third, fourth & fifth letters until he finally reaches the word 'proxy' in the dictionary & then finds its meaning.

In other words, Mukesh needs to follow a set of steps to complete the task of finding the meaning of a word.

Similarly, before writing a program code for a given problem in computer science,



it is essential to devise a set of steps to be followed to solve the problem successfully. This set of steps is called an algorithm.

Thus, in computer science, an algorithm is defined as the step-by-step plan to solve the problem for a given problem statement.

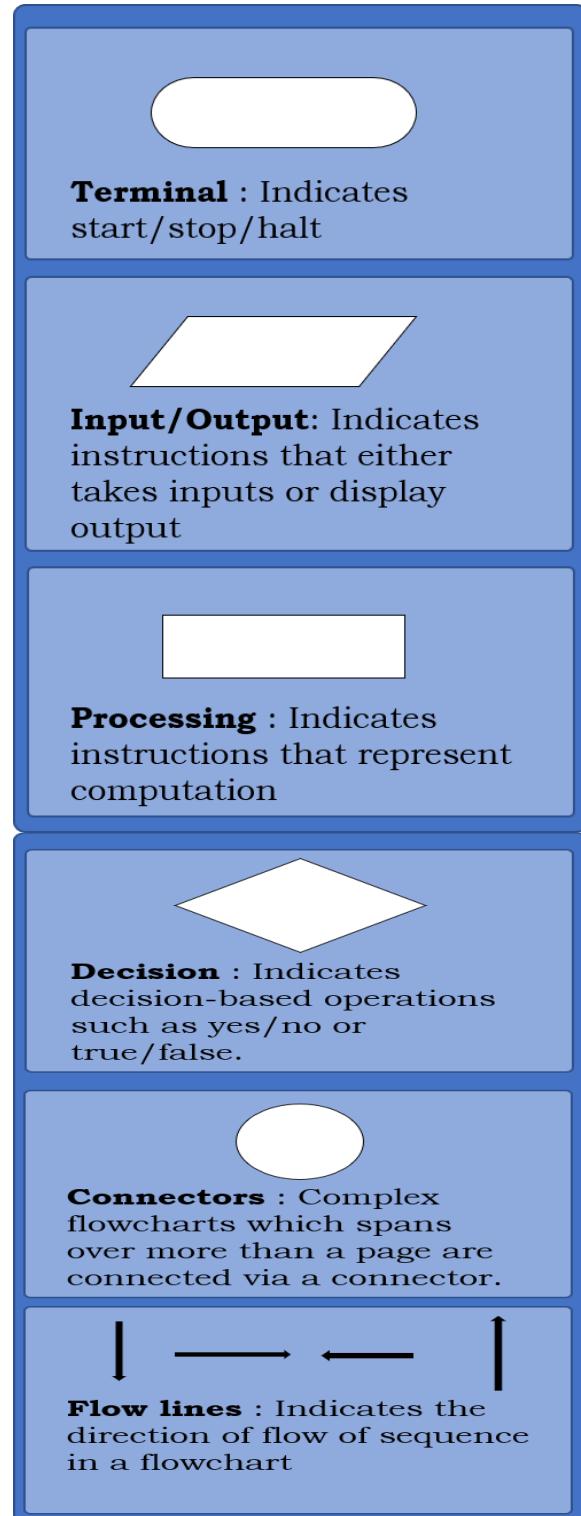
2.3 Flowcharts

2.3.1 What is a Flowchart?

A flowchart is a diagrammatic representation of the step-by-step plan to be followed for solving a task/problem statement.

This diagrammatic representation is made up of shapes like boxes, diamonds, parallelograms, circles, ellipses connected by arrows. Each shape acts as a step in the solution, and the arrows represent the direction of flow among the steps.

2.3.2 Symbols used in a flowchart





2.3.3 Benefits of using a flowchart

Some of the benefits of using a flowchart are:

1. It helps to explain your approach towards solving a problem
2. The flowchart helps in bringing in visual clarity to a problem, so it helps in practical problem solving
3. Once you build a flowchart, this remains as documentation of the code you are about to build. If you need to come back and understand the code, you can refer to the flowchart.

2.4 Activity: Profit and loss with flowcharts

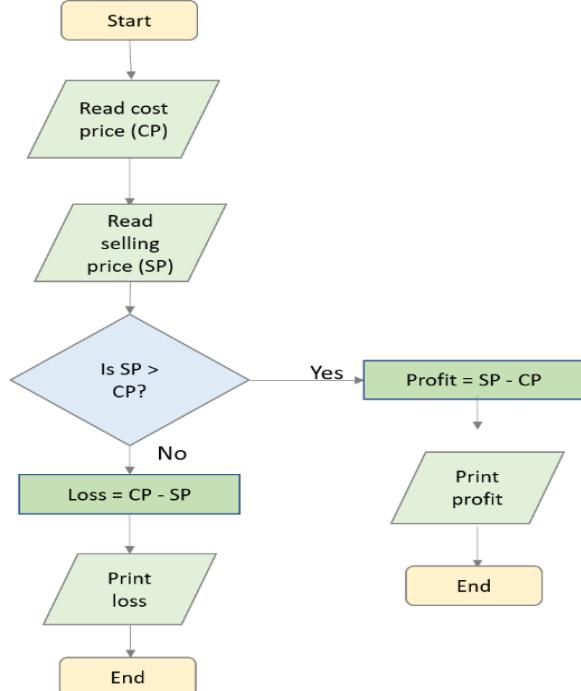
Shown in the image is a flowchart that takes two inputs, namely cost price and selling price. It then calculates profit or loss depending on the values and prints the same.

2.5 Pseudocode

2.5.1 What is a Pseudocode?

In computer science, pseudocode is used to describe the steps followed in an algorithm via simple human-comprehensible language. Thus, it has no syntax of any programming language and can be understood by a layman. The advantages of representing the solution as pseudocode are multifold:

- The focus is mainly on including all



the essential steps to solve the problem. Thus, the solution tends to be comprehensive

- Reviewers can quickly review the pseudocode & verify if the steps will generate the desired outcome
- While writing pseudocode, you can focus on all possible scenarios. So, this helps you understand the potential problems that might come up later
- Since you are not worried about coding syntax, you can concentrate on the actual problem
- Writing pseudocode will help writing your code much easier
- Works as documentation of the code. So even a layman with no coding knowledge can refer to the pseudocode



2.6 Activity: Profit and loss with pseudocode

Now that we have a fair understanding of flowcharts and pseudocode, let us try to implement the same profit and loss problem using pseudocode.

Program starts

Read Cost Price (CP)

Read Selling Price (SP)

If (SP > CP) then

 Profit = SP – CP

 Print Profit

Else

 Loss = CP – SP

 Print Loss

Program ends

programming and learn coding concepts easily.

MakeCode Arcade is one of the platforms to perform block-based programming.

Using arcade, solution to a problem statement can be implemented using various categories of blocks and the results can be seen side by side on the same screen.

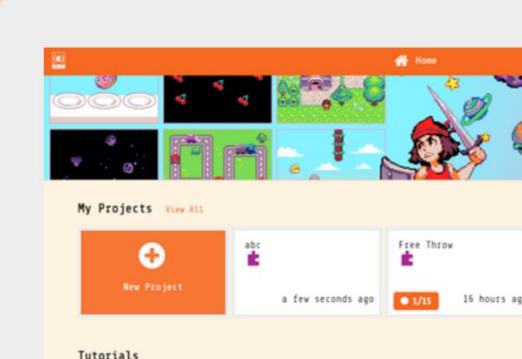
The different categories of blocks will be explored in the upcoming chapters.

Note: Minecraft is just one of the platforms for block coding. You can use many similar platforms available online for block coding like – Scratch (<https://scratch.mit.edu/>), Code.org (<https://code.org/>) etc.

We will now start a simple block coding exercise on Minecraft platform.

Open the URL

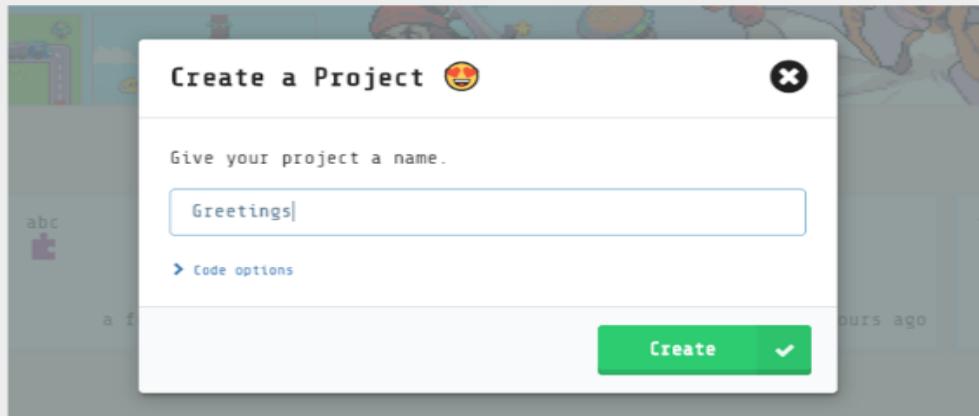
<https://arcade.makecode.com> in your favorite web browser.



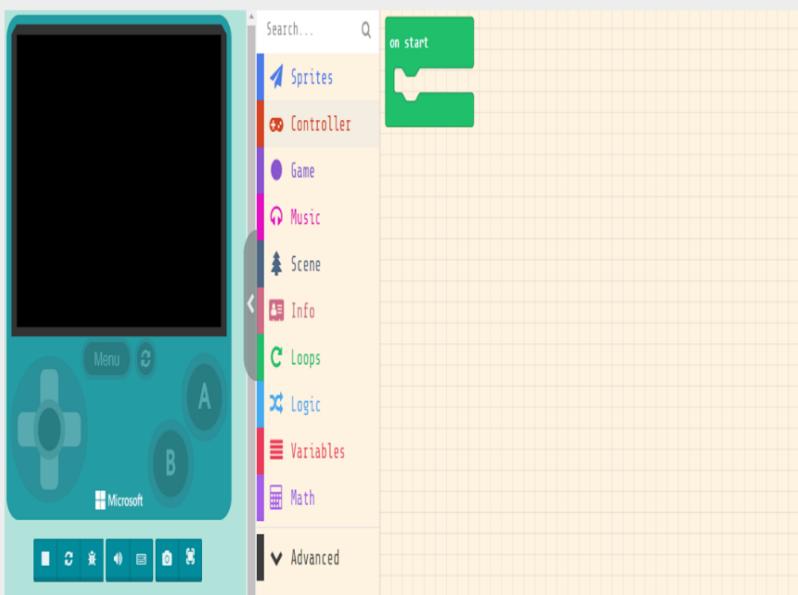
2.7 Getting started with block coding

Microsoft MakeCode is a framework for creating interactive and engaging programming experiences for those new to the world of programming.

The main objective of MakeCode is to bring programming to the forefront in a way that is more alluring and friendly. To achieve this, MakeCode utilizes the blocks programming model to let those who are new to the world of



Step 2: Click on new project and name it “Greetings”



Step 3: A new screen is opened.
The screen like structure on the left, is the place where results from a program output gets **displayed**.
The list in the middle represents the different **block types** that are available.
The space on the right having a block named “on start” is the space where we join different blocks to **execute** a set of steps.

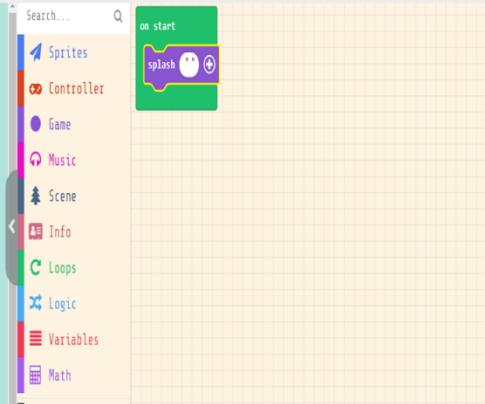


The Scratch interface is shown with the Game category selected in the blocks palette. A red box highlights the 'Game' category in the sidebar. In the script area, a green 'on start' hat block and a purple 'splash' block are present. The 'Game' category also contains blocks for 'ask', 'ask for number', 'ask for string', and 'set dialog text color to'.

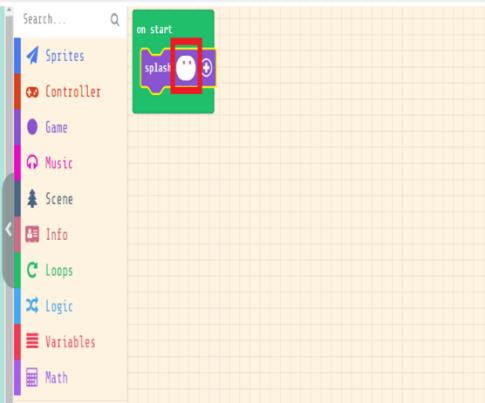
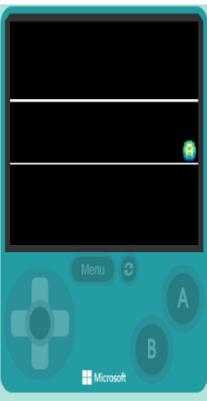
Step 4: Go to block type “Game” and select the block as highlighted in the diagram

The Scratch interface is shown with the Game category selected. A red box highlights a newly added green 'splash' block in the script area, which is a duplicate of the original 'splash' block from Step 4.

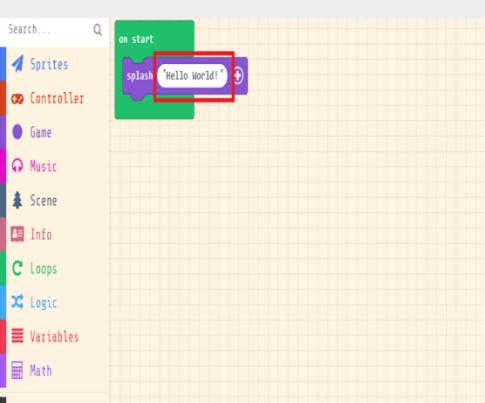
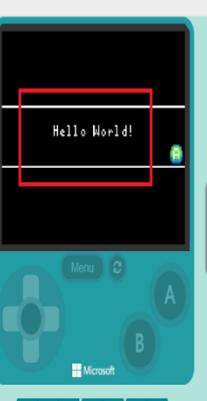
Step 5: A separate block gets created in the right-hand side as highlighted inside red lines shown below



Step 6: Drag and drop the above block inside the “**on start**” block. As soon as the splash block is placed inside “**on start**” block, the display on the left-hand screen changes.



Step 7: Left click the mouse inside the area highlighted using red lines.



Step 8: Type “Hello World!” inside the box where cursor is placed. The screen on the left automatically gets updated with the message - **Hello World!**



Congratulations, you have created your first block code program! Now let us do a quiz.

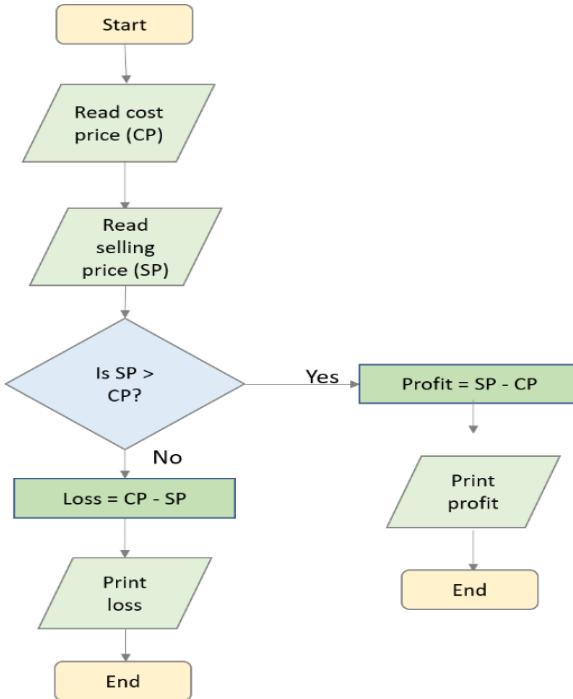
2.8 Quiz time

Objective type questions

1. Select a word from column A and place it in a cell on column B such that its definition matches in cells of column C

A	B	C
Algorithm		Writing steps involved to solve a problem in a human-understandable language
Flowchart		Block based coding platform
Pseudo Code		Define step by step plan to solve a problem statement
Arcade		Solution to a problem presented as a diagram

Below is a flowchart to calculate profit and loss. Answer questions 2 to 4 based on the flowchart.



Question 2	Jyoti owns a toy shop. She bought a toy for Rs 325 and sold it for Rs. 375. Which one from the below is correct?
Option 1	She made a loss of Rs. 50
Option 2	She made a profit of Rs. 50

Question 3	Raju buys a pen for Rs 15, and after some time, he sells it for Rs 10. Which one from the below is correct?
Option 1	He made a loss of Rs. 5
Option 2	He made a profit of Rs. 5

Question 4	Ram buys a table for Rs 500, he sells it for Rs 550. Which one from the below is correct?
Option 1	He made a loss of Rs. 50
Option 2	He made a profit of Rs. 50



Question 5	Which of the following is not an advantage of a flowchart?
Option 1	Efficient coding
Option 2	Systematic testing
Option 3	Improper documentation
Option 4	Better communication

Question 6	The following box denotes. 
Option 1	Initialization
Option 2	Decision
Option 3	Input / Output
Option 4	Connectors

Question 7	What is a flowchart?
Option 1	A specific programming language
Option 2	A text-based way of designing an algorithm
Option 3	A bullet point list of instructions
Option 4	A diagram that represents a set of instructions

Question 8	What shape represents the start and end of a flowchart?
Option 1	Square
Option 2	Diamond
Option 3	Oval
Option 4	Circle

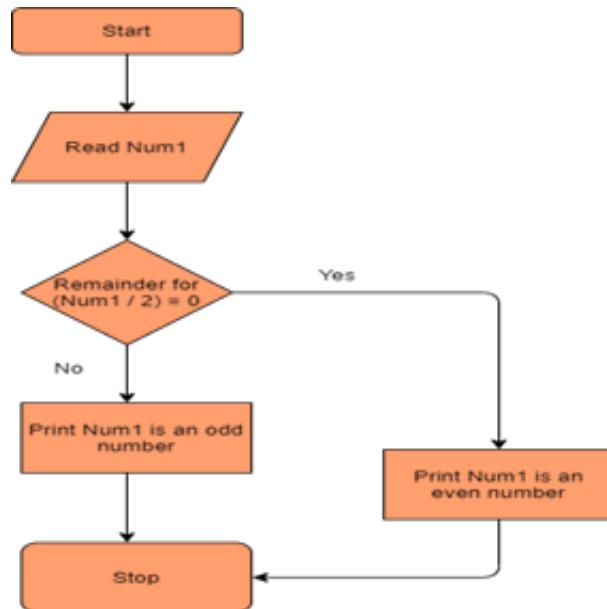
Short Answer questions

1. What is a pseudocode?
2. What are the benefits of using flowcharts?
3. What is a flowchart?



Higher Order Thinking Skills (HOTS)

- Below is a flowchart on how to calculate if a number is even or odd. Write the corresponding pseudocode for the problem.



Note: Modulus operator (%) is used to find the remainder of two number.

- The below pseudocode prints "Above average marks" if the average marks in three subjects are greater than 75. If average marks are less than or equal to 75, then it prints "Below average marks". Draw the corresponding flowchart for the problem.



Program starts

```
Read marks for subject SubA  
Read marks for subject SubB  
Read marks for subject SubC  
Calculate AverageMarks = (SubA + SubB + SubC)/3  
Print AverageMarks  
If (AverageMarks > 75)  
    Print ("Above Average Marks")  
Else  
    Print ("Below Average Marks")
```

Program ends

Applied Project

Create a flowchart based on your normal school day. Here are some guidelines:

Getting ready for school

- Look at your timetable and pack your school bag
- If PE is there packing your PE uniform and shoes

At school

- Attend the morning session with subjects Math, English and science
- A decision on what to play during recess

After reaching home

- Have snacks
- Depending on the day, choose the class you have to attend.
- Monday-Karate, Tuesday-Math Class, Wednesday and Thursday Free day, Friday-Karate



2.9 What did you learn in this chapter?

- By now you should have a basic understanding about algorithms, flowchart & pseudocode
- You have also practiced following a step by step approach to solve a problem using block-based programming
- You got oriented to MakeCode platform which we will use throughout the course



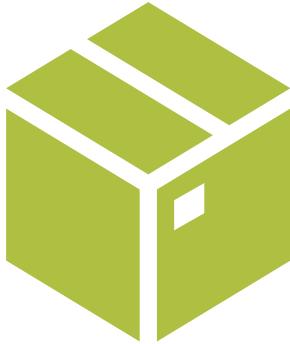
VARIABLES USING BLOCK CODING

3.1 What will you learn in this chapter?

- What are variables?
- How to name variables?
- Commonly used data types
- Performing operations on variables

3.2 What are variables?

In programming, variable is a packet in which we can store data. These packets



can be named and referenced and can be used to perform various operations. To perform a mathematical operation, you can declare two variables and perform the operation on them.

Scope of a variable refers to the part of the code where the variable can be used. The scope or accessibility of the variables defined in a program depends on where you have declared it in each

program. Any defined variable cannot be accessed beyond its scope.

3.3 Naming variables



As we have understood till now, variables are basically like nouns in a programming language. Every variable in a program is unique. To identify these variables uniquely, user needs to allocate them a unique name. This name acts as an identifier for that variable. In programming, a user is not allowed to use the same name of a variable more than once.

Naming variables make it easier to call them while performing operations. The name of a variable also suggests what information the variable contains.

You can refer to the example before for better understanding about variables:

If variable named as "a" is equal to 2 and variable named as "b" is equal to 2,



performing add operation on "a" and "b" is going to result into an output as "4".

3.4 Data types in variables

Variables are the values that are acted upon. Every value needs to be assigned to a specific data type to make the variable more readable by a computer. Data type identifies what the type of data that the declared variable can hold is. Thus, it indirectly helps the computer to understand what operations need to be performed on those variables. The declaration of a variable in a program contains two components – the name of the variable and its type.

Let us now understand what are the common data types that we can use in programming:

- Integer
- Floating-point number
- Character
- String
- Boolean

Integer Data Type

Integer data type variables store integer values only. They store whole numbers which have zero, positive and negative values but not decimal values. Multiple programming languages support different syntax to declare an Integer variable. If a user tries to create an

Example of declaring an Integer variable:

```
int a = 2;
```

integer variable and assign it a non-integer value, the program returns an error.

Variables of the integer data type are only capable of holding single values. These variables are not capable of holding a long list of values.

Floating Point Number Data Type

Floating-point numbers are used to store decimal values. They hold real numbers with decimal values. Depending on the programming language, the syntax to declare floating-point variable changes.

Example of declaring a floating-point number variable:

```
float a = 1.1;
```

There is another type of floating-point number known as a "double" data type, which is used to store even bigger values.

Example of a double value:

```
double a = 8.99999999 *  
7.66666666666;
```

Character Data Type

Character type variables are used to store character values. Syntax of declaring a character variable is specific to the programming language that you



are using. If a user tries to create a character variable and assign it with a non-character value, the program will throw an error. The character data type is the smallest data type in programming.

Any character values can be declared as a char variable.

Example of declaring a character variable:

```
char a = 'w';
```

String Data Type

To extend the character data type, a user may have a requirement to store and perform an operation on a sequence of characters. In such cases, the String data type is present to fit the gap. The String data type stores value in a sequence of characters i.e. in String format.

Example of declaring a String variable:

```
String a = "I am a String Variable";
```

Any string value can be declared as a string variable.

Boolean Data Type

There is a subtype of Integer Data Type called "Boolean Data Type", which stores values in Boolean type only i.e. "true" or "false". Users can choose between the

data types for variables as per program needs and assign variables an appropriate data type.

Boolean is a subtype of integer data type. It stores true and false where true means non-zero and false means zero.

Example of declaring a Boolean variable:

```
bool a = true;
```

Any Boolean variable holding Boolean value can be declared as Boolean.

Note: In some programming languages like python, there is no command to

Example of declaring variables in python

```
a = 5  
b = "Hello"
```

declare variables. A variable is created the moment you first assign a value to it.

If you want to specify the data type of variable this can be done using casting.

Example: y = str(7)

z = int(7)

a = float(7)

y will be saved as '7'

z will be saved as 7

a will be saved as 7.0



Let us now apply different data types in a pseudocode. We will first declare

```
Integer i = 1234567890  
Float f = 1.234  
Char c = c  
String s = string  
Boolean b = true
```

different types of variables, followed by assigning them to appropriate values. Finally, we will use the variables by outputting their values.

Function Main

Declare

```
Declare Integer i  
Declare Float f  
Declare Char c  
Declare String s  
Declare Boolean b
```

Assign

```
Assign i = 1234567890  
Assign f = 1.234  
Assign c = 'c'  
Assign s = "string"  
Assign b = true
```

Use

```
Output "Integer i = " & i  
Output "Float f = " & f  
Output "Char c = " & c  
Output "String s = " & s  
Output "Boolean b = " & b
```

End

The output from the above pseudocode will be like below.

Note: The syntax is different for different data types in other programming languages.



In python string variables can be declared either by single quotes or double quotes or triple quotes.

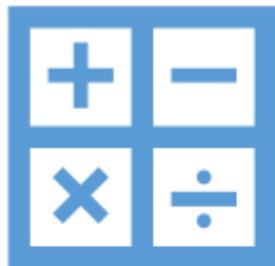
Below are some rules for naming a variable in python:

- A variable name cannot start with a number, it must start with an alphabet or the underscore (_) sign
- Variable name is case sensitive. **Sum** and **sum** are different variables
- A variable can only contain alpha numeric characters and underscore

3.5 Performing Operations on Variables

After declaring the data types in programming, now let us move ahead and understand what operations we can perform on the data types and how do we perform these operations:

Arithmetic Operations



An arithmetic operation combines two or more numeric expressions using the Arithmetic Operators to form a resulting

numeric expression. The basic operators for performing arithmetic are the same in many computer languages:

- Addition
- Subtraction
- Multiplication
- Division
- Modulus (Remainder)

Assignment operators

Assignment operators are used to assign values to variables

Different assignment operators are shown below:

- “=”: This operator is used to assign the value on the right to the left variable.
Example: $a=50$
- “+=”: This operator assigns the result to the variable on the left after adding the current value of the variable on left to the value on the right.
Example: $x += y$
It can also be written as $x = x + y$
- “-=”: This operator assigns the result to the variable on the left after subtracting the value of the variable on right from the current value on the left.
Example: $x -= y$
It can also be written as $x = x - y$
- “*=:” This operator assigns the result to the variable on the left after multiplying the current value of the variable on left to the value on the right.
Example: $x *= y$
It can also be written as $x = x * y$
- “/=”: This operator assigns the result to the variable on the left



after dividing the current value of the variable on left from the value on the right.

Example: $x /= y$

It can also be written as $x = x / y$

Increment operator

Increment operator adds one to the value.

Example: $A=8$

$B=A++$

The output of B will be 9 ($A+1$)

$A++$ has the same meaning as $A=A+1$

Decrement operator

Decrement operator subtracts one from the value.

Example: $A=8$

$B=A--$

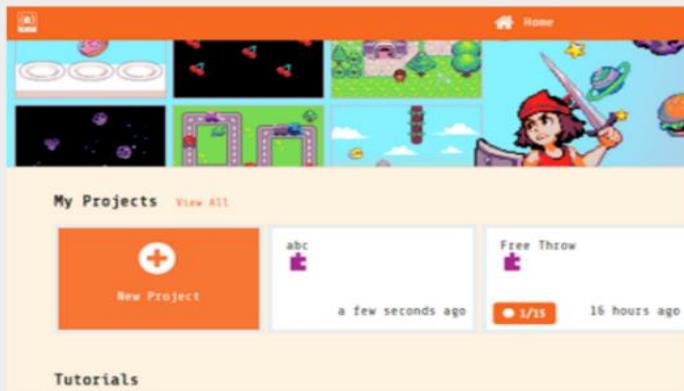
The output of B will be 7 ($A-1$)

$A--$ has the same meaning as $A=A-1$

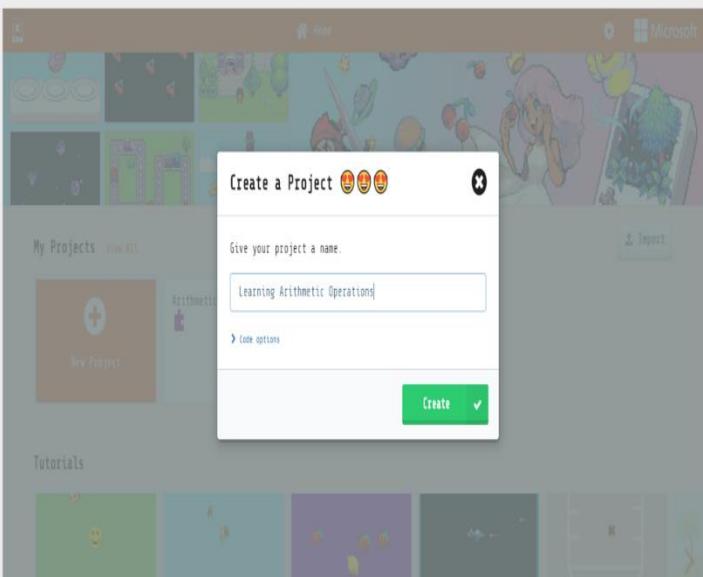
3.6 Activity: Addition operation using block coding

An addition arithmetic operation is used to add the values stored in two variables. Like the way we add values in mathematics, we can store values in different variables and perform an additional operation. The addition of these variables is displayed as an output of the program.

For example, performing add operation on a variable "a" holding value "3" and a variable "b" holding value "4" will result in an output "7". To understand this arithmetic operation better, let us understand how to implement it practically in programming. For this, we are going to take example of platform <https://arcade.makecode.com/>. Let us refer to the steps below understand more in details.



Step 1: Make code arcade home screen



Step 2: Click on “New Project”, give a name to your project and click on “Create” button.



The image shows the Scratch programming environment. On the left is a stage area with a teal background and a black center. On the right is a script editor with a light blue background. A vertical toolbar on the far left contains icons for Home, Share, Blocks, JavaScript, Help, Microsoft, and a gear icon. Below the stage is a control bar with Download, Stop, and Start buttons. The main menu bar at the top includes Home, Share, Blocks, JavaScript, Help, Microsoft, and a gear icon. The central workspace has a search bar and a list of categories: Sprites, Controller, Game, Music, Scene, Info, Loops, Logic, Variables (which is highlighted in red), Math, and Advanced. A sub-menu for Variables is open, showing a button labeled "Make a Variable...".

Step 3: From the list in the center of the page, click on “Variables” and then click on “Make a Variable”

The image shows the Scratch environment after creating a variable. The stage and control bar remain the same. The workspace now displays three blocks in the script editor: "set [a v] to [0]", "change [a v] by [1]", and "change [a v] by [-1]". The "Variables" sub-menu in the central list is still open, showing the "a" variable listed under "Variables".

Step 4: Name the new Variable as “a” and click on “Ok” button



Step 5: Click on the value of “**a**” and change to the desired value. In this case, we are taking it as “**10**”

Step 6: Drag and drop “**a**” to the green block



Step 7: Similarly, make another variable “**b**” and assign it a value “**20**” and drag and drop it on the green block.

Step 8: Now click on “**Math**” link from the center of the page and click on addition operation.



The screenshot shows the Scratch workspace. In the script area, there are two green 'on start' blocks. The first block sets variable 'a' to 10, and the second block sets variable 'b' to 20. The stage area is currently blank. On the left, the sprite palette shows a single sprite named 'Microsoft'. Below the stage, the script editor shows the count of blocks as 0.

Step 9: Now click on “**Variables**” link from the center of the page and drag and drop variables “**a**” and “**b**” in the addition box as seen in the below screenshot

The screenshot shows the Scratch workspace after Step 9. A new green 'on start' block has been added to the script area. This block sets variable 'c' to 0. The stage area is still blank. The sprite palette shows the 'Microsoft' sprite. The script editor now shows a count of 1 block.

Step 10: Like the way we had created variables “**a**” and “**b**”, now create variable “**c**”. Drag and drop block of “Set “**c**” to” in the play area.



Step 11: Now drag and drop addition block in the “set “c” to” block as seen in the screenshot.

Step 12: Drag the block of “Set “c” to “a” + “b”” in the green block



The screenshot shows the Scratch workspace. A script is attached to a sprite. The script starts with 'on start' and contains the following blocks:

- set **a** to **10**
- length of "Hello"
- join "Hello" "World" **c**
- parse to number "123"
- split "this" at **a**
- "this" includes **b**
- "this" find index of **a**
- "this" is empty

The Variables palette on the left shows three variables: **a**, **b**, and **c**. The **c** variable has a value of 10.

Step 13: You have now created the variables and assigned them their respective values. You have also created a third variable “c” which will have output of “a” and “b”. Let us now move ahead and see how we can execute this program. For that, first you must click on “Advanced” link from the center of the page and then click on “Text” Link. Select “Join” operation from the list

The screenshot shows the Scratch workspace. A script is attached to a sprite. The script starts with 'on start' and contains the following blocks:

- set **a** to **10**
- set **b** to **20**
- set **c** to **a** + **b**

Below the script, there is a text block: **join "Sum of "a" and "b" equals to" **c****.

Step 14: In joint operation, you can rename the first block as “Sum equals to”. In the second block, you need to drag and drop variable “c” from Variables



The screenshot shows the Scratch workspace. On the left, there's a green stage with a blue background featuring a Microsoft logo and two circular buttons labeled 'A' and 'B'. In the center, the script editor shows an 'on start' script with three 'set [variable] to [value]' blocks: 'set [a v] to [10]', 'set [b v] to [20]', and 'set [c v] to [a v + b v]'. Below the stage, a yellow 'join' block is connected to a 'splash' block, which contains the text 'Sum = [c v]'. A green 'script' block is also visible.

Step 15: Now, click on “Game” link from the center of the page and click on “Splash” block. Once the splash block is visible, drag and drop “Join” block into “Splash” block

The screenshot shows the Scratch workspace after Step 16. The stage now displays the text 'Sum equals to 30' in white on a black background. The script editor shows the same 'on start' script with the same three 'set [variable] to [value]' blocks. The yellow 'join' block is now connected directly to the 'script' block, and the 'join' block no longer contains the variable 'c'.

Step 16: Drag and drop the splash block into the green block, below the set “c” block



The screenshot shows the Scratch programming environment. On the left, there's a preview window displaying a black screen with the text "Sum equals to 30". Below the preview are two circular buttons labeled "A" and "B". On the right, the script editor shows a script starting with "on start" which sets variables "a" to 10 and "b" to 20, then sets variable "c" to the sum of "a" and "b". A "splash" costume is selected, with the script "Sum equals to [c v]" attached to it.

Step 17: Once these steps and completed and you click on Play button on the left-hand side of the page, you will see that the program displays the output as "30" which is the addition of "a" and "b"

Subtraction

Subtraction arithmetic operation is used to subtract the values stored in one variable from another variable. Like the way we subtract values in mathematics, we can store values in different variables and perform subtraction operations. Subtraction of these variables is displayed as an output of the program.

For example, performing subtraction operation on a variable "a" holding value "10" and a variable "b" holding value "8" will result in an output "2".

Multiplication

Multiplication arithmetic operation is used to multiply the values stored in two variables. Like the way we multiply values in mathematics, we can store values in different variables and perform multiplication operations. Multiplication of these variables is displayed as an output of the program.

For example, performing multiply operation on a variable "a" holding value "2" and a variable "b" holding value "2" will result in an output "4"

Division

Division arithmetic operation is used to divide the value stored in one variable by the value stored in another variable. Like the way we divide values in mathematics, we can store values in different variables and perform division operations. Division of these variables is displayed as an output of the program.

For example, performing division operation on a variable "a" holding value "2" and a variable "b" holding value "2" will result in an output "1".

Modulus (Remainder)

Modulus operator (%) calculates the remainder when two variables are divided. Please note that this operation can only be performed on integer and float variables in Python.

For example, performing modulus operation on a variable "a" holding value "9" and variable "b" holding value "3" will result in an output "0" as there is no remainder in this operation.



3.7 Quiz time

Objective Type Questions

Question 1	An integer data type can hold decimal values.
Option 1	True
Option 2	False

Question 2	Variables must be defined with a name and a data type before they can be used
Option 1	True
Option 2	False

Question 3	Which of the following is not a valid variable name in python?
Option 1	_test
Option 2	11test
Option 3	Test13
Option 4	Test_2

Question 4	Fill in the blanks to declare sum equal to a + b (int _ = a __b)
Option 1	Sum,+
Option 2	Var,-
Option 3	Bool,+
Option 4	Add,+

Question 5	Which of the following data type is used to store decimal values?
Option 1	Integer
Option 2	Float
Option 3	Boolean
Option 4	String

Question 6	How many times should a data type be mentioned for a variable
Option 1	Everywhere the variable is used



Option 2	When entering variable's value
Option 3	When printing a variable's value
Option 4	Only once; When declaring the variable

Question 7	Which of the following symbol is used to multiply variables?
Option 1	*
Option 2	+
Option 3	x
Option 4	%

Question 8	What is the alternative of $y=y+9$
Option 1	$y=x+9$
Option 2	$y+=9$
Option 3	$y-=9$
Option 4	$x=+9$

Question 9	$Y++$ has the same meaning as
Option 1	$Y=Y+1$
Option 2	$X==2$
Option 3	$y-=6$
Option 4	$X=x-5$

Question 10	Which of the following symbol is used to find the remainder?
Option 1	*
Option 2	+
Option 3	x
Option 4	%

Short Answer Questions

1. Define variables in programming
2. Can we declare two variables in a program with the same name?
3. What are the common Data Types in programming?
4. Name a data type that can store exponential values
5. Write the pseudocode to perform an addition operation on two variables in a program



Higher Order Thinking Skills (HOTS)

1. Create a flowchart to perform different mathematical operation (Multiplication, Subtraction, Addition, Division) on two or more variables.
2. Create a project in <https://arcade.makecode.com/> to perform Modulus operation on two variables in a program.

Applied Project

Using block coding, create your normal school day (Monday to Friday).

Here are some guidelines:

Getting ready for school

- Look at your timetable and pack your school bag
- If PE is there packing your PE uniform and shoes

At school

- Attend the morning session with subjects Math, English and science
- A decision on what to play during recess
- Attend the afternoon session subjects
- Lunch Break
- PE Class

After reaching home

- Have snacks
- Depending on the day choose the class you must attend.
Monday-Karate, Tuesday-Math Class, Wednesday-Swimming and Thursday-Chess, Friday-Dance (You can use variables to define the class you have to attend)

3.8 What did you learn in this chapter?

- What are variables and how they are used in programming
- Ways of naming variables.
- Different data types in programming and its usage
- Various operations that we can perform of different data types in programming



CONTROL WITH CONDITIONALS

4.1 What will you learn in this chapter?

- What are conditions and how to apply them in real life?
- What are the different types of operators?
- How to combine multiple operators?
- Apply logical operations in block coding

4.2 Arranging blocks

In the image below, we see several blocks arranged in a specific order. Every time we place a new block, we apply logic to build a diagonal line with blocks marked with arrows. This logic in coding terms are called conditions.

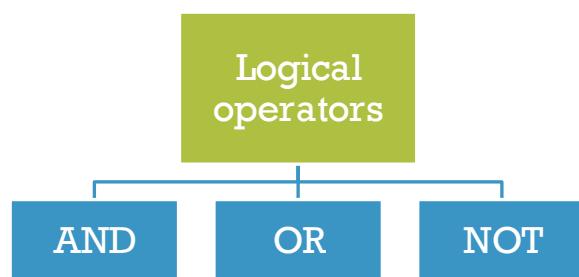


Similarly, every day we take many decisions depending on our situation. For instance, when it is cold outside, we wear warm clothes, otherwise, we don't.

Logical operators are fundamental blocks that can be used to build a decision-making capability in your code. In the earlier chapters, we discussed how to handle decisions in a flow chart. Now we shall see how to implement it in our code.

We can do things conditionally in our programs using if statements or if/else statements combined with logical operators. Logical operators work like Boolean variables and return either TRUE or FALSE.

The three most important logical operators are AND, OR and NOT.



4.3 And operator



AND operator is used to determine if two or more conditions are true. If all the conditions are true, the AND operator returns TRUE. If any one of the conditions fail, the AND operator returns FALSE. In some programming languages AND operator is denoted by “**&&**” symbol.

For example - you should go to bed only after you have completed your homework and the time is past 8 PM.

Here, if we want to derive the logical operation from this scenario, we have the following conditions:

Condition 1: Have you completed homework?

Condition 2: Is the time past 8 PM?

And the decision we are deriving is:

Decision: Should you go to bed?

Based on this we can write the below pseudo code:

```
IF (Homework completed) AND (Time is past  
8 PM)  
THEN  
    Go to bed  
ELSE  
    Do not go to bed  
END
```

Let us now try to see the different combinations possible with the above pseudo code.

Condition 1	Condition 2	Decision
Have you completed homework?	Is the time past 8 PM?	Should you go to bed?
Yes	Yes	Yes
No	Yes	No

Yes	No	No
No	No	No

Having this example in mind, let us now see how this is different from OR operator.

4.4 OR Operator

The OR operator is used to determine if either one of two or more conditions is TRUE. If any of the condition is true, the OR operator returns TRUE. If all the conditions fail, the OR operator simply returns FALSE. In some programming languages OR operator is denoted by “**||**” symbol.

For example - We should carry an umbrella when either it is sunny, or it is raining. Otherwise, we should not carry it. Like the previous example, if we want to derive the logical operation from this scenario, we have the following conditions:

Condition 1: Is it sunny outside?

Condition 2: Is it raining outside?

And the decision we are deriving is:

Decision: Should we carry an umbrella?

The pseudocode for this will look like below:

```
IF (It is sunny outside) OR (It is raining  
outside)  
THEN  
    Carry an umbrella  
ELSE  
    Do not carry an umbrella  
END
```



Shown below are the different possible combinations for the above example.

Condition 1	Condition 2	Decision
Is sunny?	Is raining?	Carry umbrella?
Yes	Yes	Yes
Yes	No	Yes
No	Yes	Yes
No	No	No

Let us now look at the NOT operator.

4.5 NOT Operator

We use the NOT operator to reverse or negate a condition. If the condition is true, NOT will return false and vice-versa. In some programming languages NOT operator is denoted by “!” symbol.

For example, you can go to play only if it is not raining, otherwise, you must stay indoors.

Unlike the previous examples, here we have only one condition.

Condition: Is it raining?

Decision: Go out to play?

The pseudocode for this will look like below:

```

IF NOT (It is raining)
THEN
    Go out to play
ELSE
    Stay indoors
END

```

And the corresponding table for this is below.

Is it raining?	Go out to play?
Yes	No
No	Yes

4.6 Combining logical operators

Sometimes, we need to combine different logical operators to create a complex expression. Imagine your library is open on Monday between 10 AM to 12 PM OR on Wednesday between 3 PM to 5 PM.

```

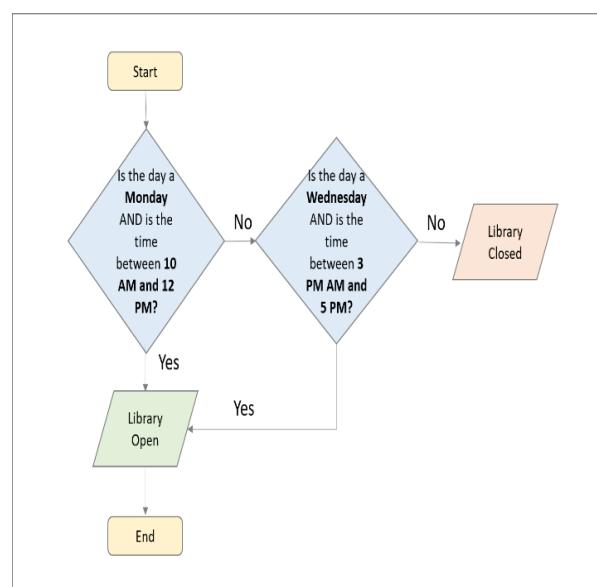
IF (Day == Monday AND (Time >= 10 AM
AND Time <= 12 PM)) OR
    (Day == Wednesday AND (Time >= 3
PM AND Time <= 5 PM))
THEN
    Library Open
ELSE
    Library Closed

```

Let's see how this looks on a flowchart.

Let us write a pseudocode for this.

The corresponding flowchart for the pseudocode will be like below.





4.7 Quiz time

Let us now answer some questions based on the above example.

Question 1

Question 2

Situation	Today is Monday and the time being 10.30 AM. Is the library open now?
Option 1	Yes
Option 2	No

Question 3

Situation	Today is Friday, and the time is 12 PM. Is the library open now?
Option 1	Yes
Option 2	No

Question 4

Situation	Today is Wednesday, and the time is 10.30 AM. Is the library open now?
Option 1	Yes
Option 2	No

Question 5

Situation	Today is Wednesday and the time is 3.30 PM. Is the library open now?
Option 1	Yes
Option 2	No

Situation	Today is Tuesday, and the time is 4 PM. Is the library open now?
Option 1	Yes
Option 2	No

4.8 Relational operators

In our previous example, we got introduced to some relational operators like greater than equals (\geq), equals ($=$), and less than equals (\leq).

Let us now look at the full list of relational operators.

Operator	Symbol	Example	Meaning
Greater than	$>$	$x > y$	x greater than y
Equal to	$=$	$x = y$	x is equal to y
Less than	$<$	$x < y$	x is less than y
Greater than or equal to	\geq	$x \geq y$	x is either greater than or equal to y
Less than or equal to	\leq	$x \leq y$	x is either less than or equal to y
Not equal to	\neq	$x \neq y$	x is not equal to y

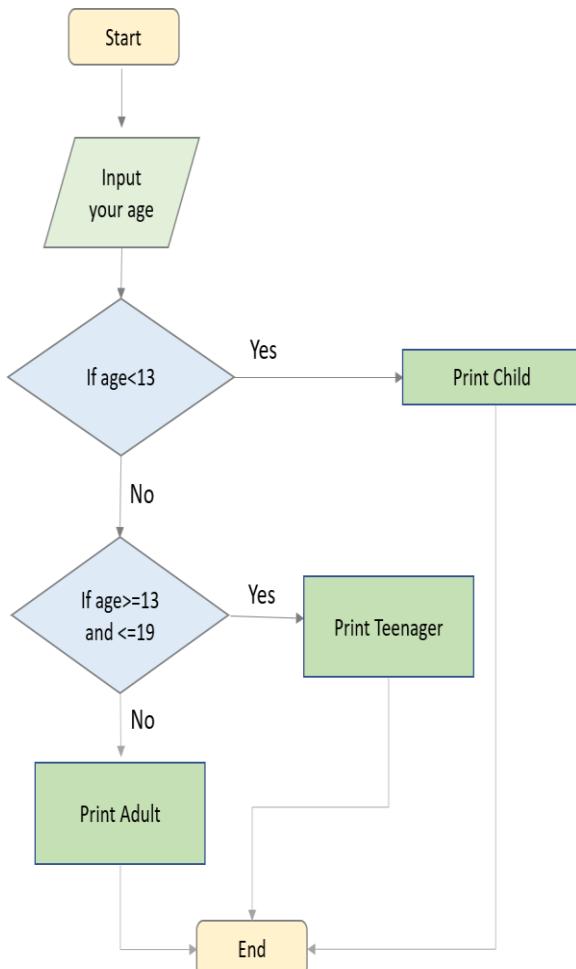
4.9 Activity: Are you a teen?

In this activity we will check if you are a child, teenager or an adult.

Steps to create the flowchart:



- Input your age
- If $age < 13$ then child, else,
If $age \geq 13$ and $<= 19$ then
teenager, else,
If $age > 19$ then adult
- Print “child or teenager or adult”
depending on the condition satisfied

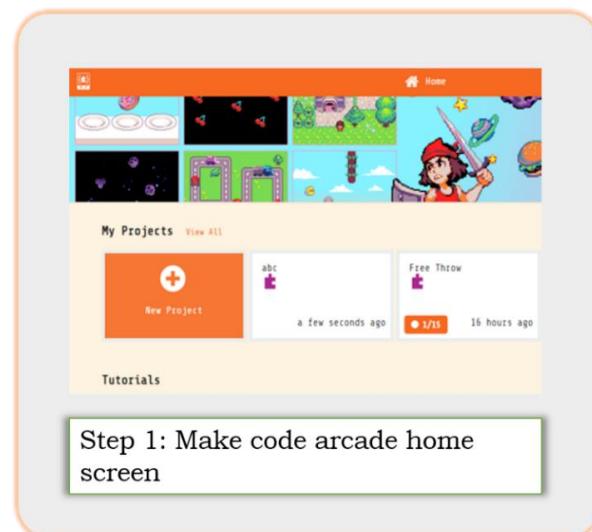


4.10 Activity: Dynamic backgrounds

Let us now do an activity to test our understanding of conditions and operators.

You should try this exercise on the MakeCode platform.

To access the MakeCode editor, open your favorite browser and go to <https://arcade.makecode.com/>. Create a new project as shown below.





The screenshot shows the 'Create a Project' dialog box. At the top, there are three emoji icons: a smiling face, a thinking face, and a happy face. Below them is a text input field containing the text 'Logical Operators'. To the right of the input field is a 'Create' button with a dropdown arrow. A small link labeled 'Code options' is visible above the 'Create' button. The background of the dialog has a light grey gradient.

Once you create you should see an editor like below.

The screenshot shows the Scratch project editor. On the left, there's a preview window showing a green character (Sprite) with a blue cross-shaped body and two circular buttons labeled 'A' and 'B'. Below the preview are buttons for 'Download' and '...'. The main workspace is a grid where a green script block is placed. The script starts with a 'on start' hat block followed by a 'say [] for []' block. To the right of the workspace is a 'JavaScript' dropdown menu. On the far right, there are Microsoft branding elements including a 'Microsoft' logo and a 'Microsoft' watermark. The bottom of the screen features a toolbar with various icons.

Follow the following steps to complete the exercise.

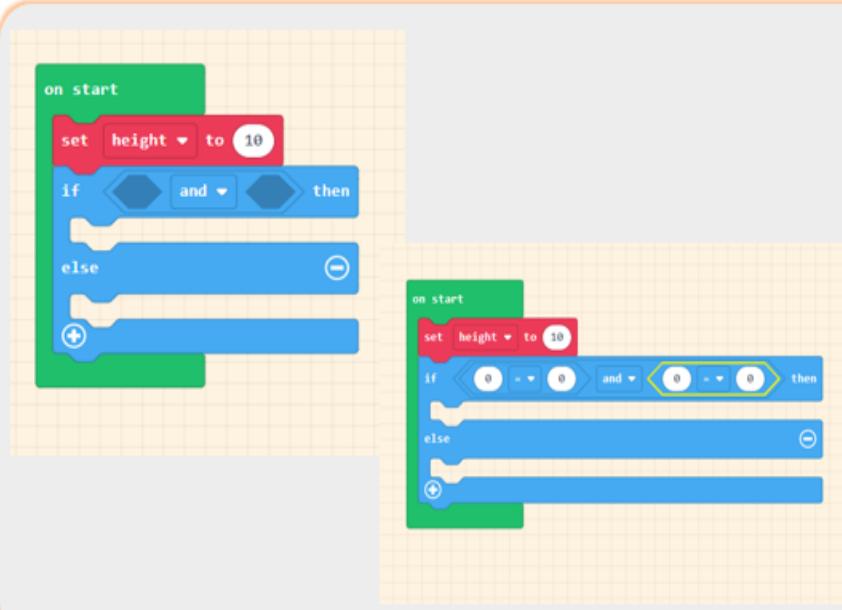


The image shows the Scratch interface. On the left, the script editor pane displays a script starting with the 'on start' hat block followed by a 'set height to 10' block. The script editor has a sidebar with categories: Sprites, Controller, Game, Music, Scene, Info, Loops, Logic, Variables (which is selected), Math, and Advanced. A floating window titled 'Variables' is open, showing a 'Make a Variable...' button. Another floating window titled 'New variable name:' shows the text 'height' entered into a text input field, with a green checkmark button at the bottom right.

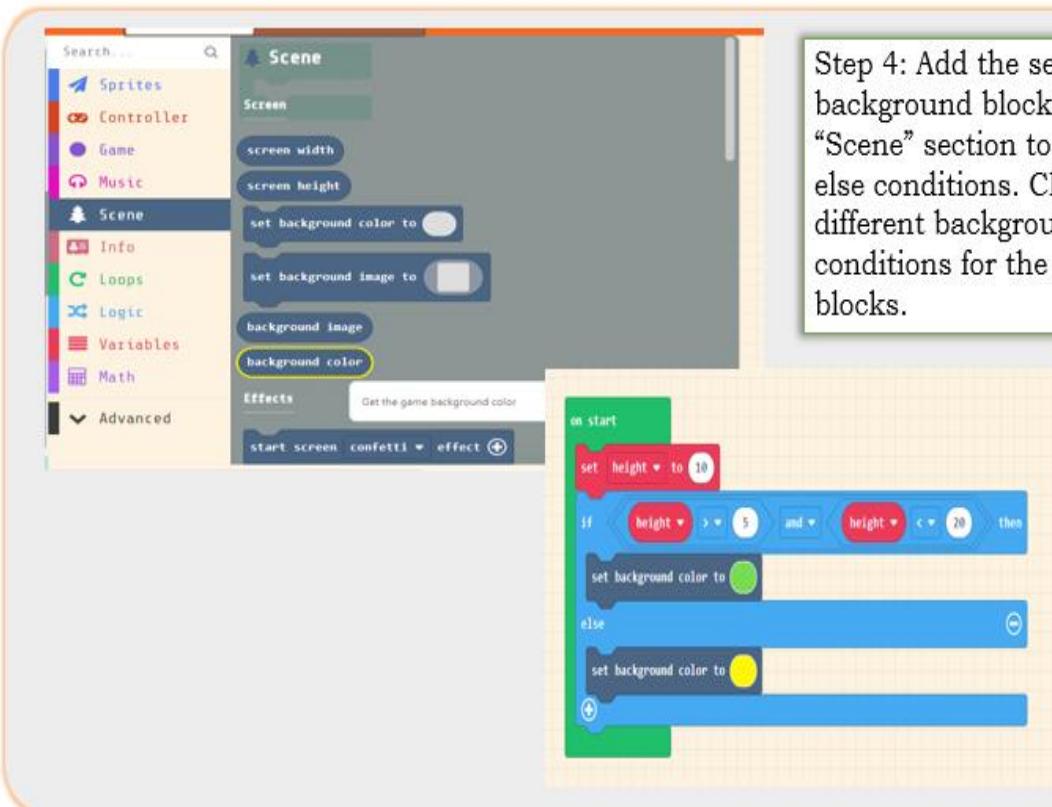
Step 1: Click on the variable menu on the left pane. Click on 'Make a Variable' and create a variable called "height".

The image shows the Scratch interface with two scripts. The top script starts with the 'on start' hat block followed by an 'if true then' block. Inside the 'if true then' block is a single green 'else' block. The bottom script starts with the 'on start' hat block followed by a 'set height to 10' block. This is followed by an 'if true then' block which contains a green '+' block (for addition) and a blue '-' block (for subtraction). The 'if true then' block is closed by a matching blue '-' block. The script editor sidebar shows the 'Logic' category selected.

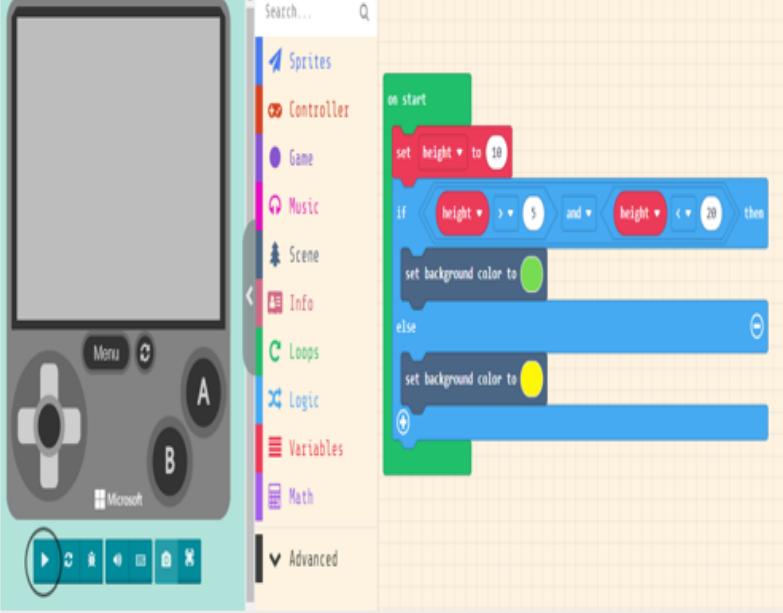
Step 2: Set "height" to 10. Choose an if-else block from the "Logic" section.



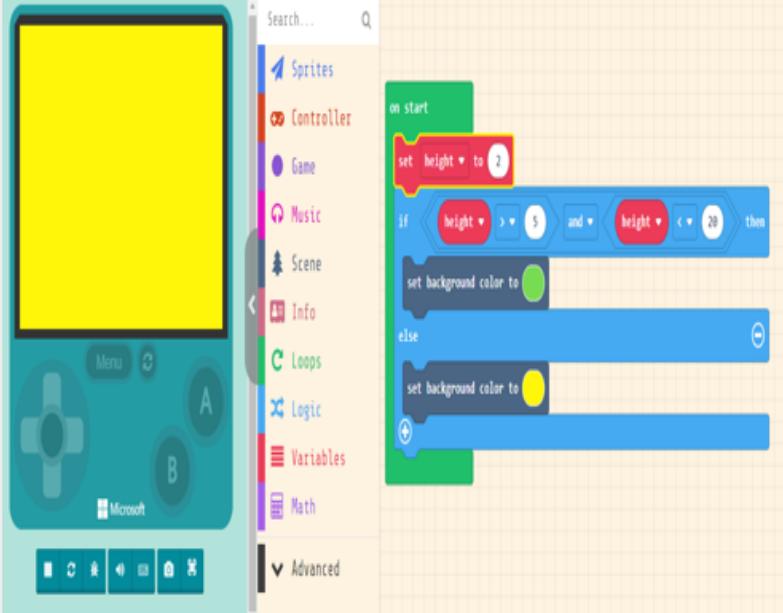
Step 3: Choose a Boolean AND operator from the logic section and add it to the if-else block as shown below. Add two comparison operators to both sides of the AND condition.



Step 4: Add the set background block from the "Scene" section to both if and else conditions. Choose different background conditions for the if and else blocks.



Step 5: Add two comparison operators to both sides of the AND condition.



Step 6: Change the value of height to 2 and see if the background changes!



The screenshot shows the Scratch programming environment. On the left, there's a green stage area with a blue border. Below the stage are control buttons: a grey cross for movement, two circular buttons labeled 'A' and 'B', and other standard Scratch controls. At the bottom, there's a Microsoft logo. The right side contains the script editor. A script is attached to the 'on start' event of a sprite. The script consists of the following blocks:

- A red "set height to 10" block.
- An orange "if" block with three conditions: "height > 5", "and", "height < 20".
- Inside the "if" block:
 - A blue "set background color to green" block.
 - A blue "else" block.
 - Inside the "else" block:
 - A blue "set background color to yellow" block.

Step 7: Change the value of height to 10 and see to which color does the background change

When you change the height to 10, the loop will go inside the if condition as the height is greater than 5 AND less than 20. The background color will change to green.

So, what did we learn from this activity?

You have now learnt how to create conditional statements. What do you think will the output of the code be if you change the height to 15?

That is correct! The background color change to green as height is greater than 5 AND less than 20.

where using a single if-else loop might not be enough.

Suppose we want to check if a number is divisible by 2 or 3 or both 2 and 3.

In this case, we first need an IF condition to check if the number is divisible by 2. Within that condition, we can implement another IF condition to check if the number is divisible by 3 or not. By doing so, we can check the divisibility of the number.

4.11 Nested Conditional Statements

Under certain circumstances, we might have to deal with complex scenarios

4.12 Activity: The remainder problem

Let us now run through a practice exercise to understand the problem



stated above. You should try this exercise on the MakeCode platform.

First, create a new project for this exercise using the steps shown at the beginning of exercise 3.1.6 and name

the project 'Nested Conditional Statements'

Follow the below steps to complete the exercise.

The screenshot shows the MakeCode interface. On the left, there's a sidebar with categories: Sprites, Controller, Game, Music, Scene, Info, Loops, Logic, **Variables** (which is highlighted in red), Math, and Advanced. In the center, there's a 'Variables' panel with a 'Make a Variable...' button. A small window titled 'New variable name:' is open, showing the text 'nested' in the input field, with a green checkmark button at the bottom right.

Step 1: Click on the variable menu on the left pane. Click on '**Make a Variable**' and create a variable called "**number**".

The screenshot shows two Scratch-style scripts. The first script, labeled 'on start', has a green 'set [number v] to [10 v]' block. The second script, also labeled 'on start', has a green 'set [number v] to [10 v]' block followed by a blue 'if [true v] then' control block. Inside the loop, there is a green '+' operator block.

Step 2: Choose the set block and set the number to 10. Choose an if block from the **logic** section.



Step 3: Choose a comparison block from the logic section.

The Scratch script consists of the following blocks:

- on start
- set number to 10
- if $0 = 0$ then
- + []

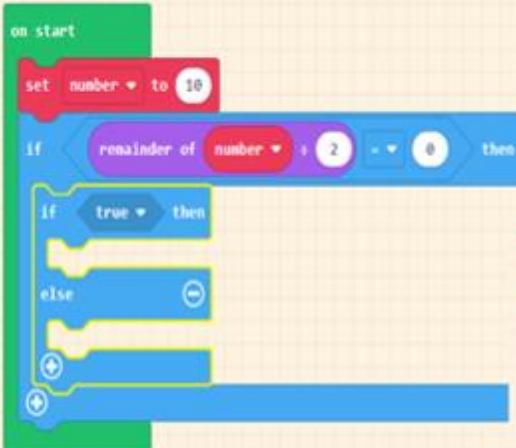
The "Comparison" section of the logic category is highlighted, showing the $=$ block selected.

Step 4: Choose a remainder block from the math section and add it to the if block

The Scratch script consists of the following blocks:

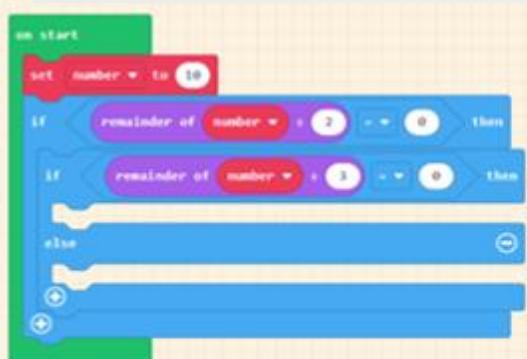
- on start
- set number to 10
- if $\text{remainder of } \text{number} \div 2 = 0$ then
- + []

The "Math" section of the category list is highlighted, showing the $\text{remainder of } \text{number} \div 2$ block selected.

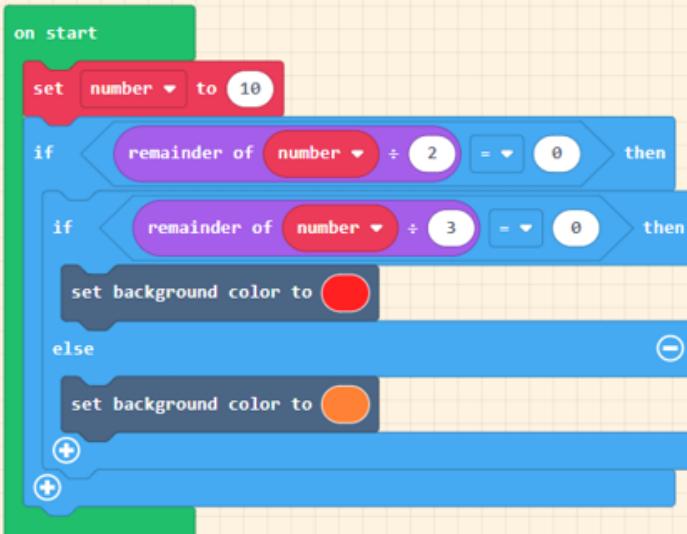


```
on start
  set [number v] to [10]
  if (remainder of [number v] ÷ [2]) = [0] then
    [set background color to [red v]]
  else
    [set background color to [orange v]]
  end
  +1
  +1
```

Step 5: Now add an if-else block and another remainder block to check if the number is divisible by both 2 and 3.



```
on start
  set [number v] to [10]
  if (remainder of [number v] ÷ [2]) = [0] then
    [set background color to [orange v]]
  if (remainder of [number v] ÷ [3]) = [0] then
    [set background color to [orange v]]
  else
    +1
  end
```



```
on start
  set [number v] to [10]
  if (remainder of [number v] ÷ [2]) = [0] then
    [set background color to [red v]]
  if (remainder of [number v] ÷ [3]) = [0] then
    [set background color to [red v]]
  else
    [set background color to [orange v]]
  end
  +1
  +1
```

Step 6: Add the set background label to both the if and else blocks.



The Scratch script starts with an 'on start' event. It sets the variable 'number' to 10. A nested if-else loop checks if the number is divisible by both 2 and 3. If so, it sets the background color to red. If not, it checks if the number is divisible by 2 only. If so, it sets the background color to orange. Otherwise, it remains red.

Background color changes to orange when the input number is 10

Result

You can now change the value of the number to 6 and check if the background of the controller changes.

The background should change to red for numbers that are divisible both by 2 and 3. It should change to orange for numbers that are divisible by 2 but not by 3.

4.13 Quiz time

Objective Type Questions

Using the below pseudocode answer the following questions (Question 1-4):

```
IF (Day == Saturday OR Day == Sunday  
    IF (Time >= 12 AM AND Time <= 8 PM))  
THEN  
    Holiday  
ELSE  
    School Day  
ELSE  
    School Day  
END
```



Question 1	Today is Monday and the time being 11.30 AM. Is today a school day?
Option 1	True
Option 2	False

Question 2	Today is Saturday and the time being 1.30 AM. Is today a Holiday?
Option 1	True
Option 2	False

Question 3	Today is Wednesday and the time is 5.30 AM. Is today a school day?
Option 1	True
Option 2	False

Question 4	Today is Sunday and the time being 09.30 PM. Is today a Holiday?
Option 1	True
Option 2	False

Question 5	Logical operators can be used to make decisions in our code
Option 1	True
Option 2	False

Question 6	Which operator is used if the statement evaluates true only if both the expressions are true
Option 1	And
Option 2	Or
Option 3	Not
Option 4	None of the above

Question 7	Which operator is used if the statement evaluates true only if only one of the expressions is true
Option 1	And
Option 2	Or
Option 3	Not
Option 4	None of the above



Question 8	Which of the following operator is used to reverse or negate a condition
Option 1	And
Option 2	Or
Option 3	Not
Option 4	None of the above

Based on the flowchart Are you a teen, let us now try to answer the below questions.

Question 9	Your father's age is 35. In which category will he fall?
Option 1	Child
Option 2	Teenager
Option 3	Adult
Option 4	None of the above

Question 10	Your sister's age is 19. In which category will she fall?
Option 1	Child
Option 2	Teenager
Option 3	Adult
Option 4	None of the above

Short Answer Questions

1. What are the different types of logical operators? Explain with examples.
2. Explain with example on how to combine different logical operators.
3. Write the pseudocode using logical operators to decide if today is a school day or not.

Monday-Friday: School day;

Sunday: Holiday;

1st and 3rd Saturday: Holiday;

2nd and 4th Saturday: School day)



Higher Order Thinking Skills (HOTS)

1. Create an if-else block with a NOT condition and set two different background colors on the controller.
2. Create an if-else block with an OR condition and set two different background images on the controller.
3. Create a nested if-else block to check if a number is divisible by 3 or 5 or both.
4. Create a nested if-else block using NOT to check if a number is a power of 2 or 3 or both.

Applied Project

Create a program using conditional statements for a sports event. If an event occurs, THEN what happens?

Here are some guidelines:

- If a referee blows the whistle
- If the time limit is reached
- If a player crosses the finish line first
- If a player touches a friend during tag
- If a ball goes outside the boundary

4.14 What did you learn in this chapter?

- You should now have an understanding on the AND, OR, NOT logical operators.
- You can now combine different logical operators.
- You have understood how to apply logical operators in block coding.
- Understand nested conditional statement in block coding.



Chapter
5

LOOPS USING BLOCK CODING

5.1 What will you learn in this chapter?

- What are loops?
- How to increment loops?
- Different type of loops
- Concept of nested loops

5.2 Introduction to loops

There are many tasks in our day to day life which we repeat at specific intervals, like eating meals, taking a bath, going to school etc. There is a very similar concept to this in programming where we need to repeat certain lines of code at specific interval or till specified condition is met.

In programming, repetition of a line or a block of code is also known as iteration. A loop is an algorithm which executes a block of code multiple times till the time a specified condition is met. Therefore, we can say that a loop iterates a block of code multiple times till the time mentioned condition is satisfied.

For example, consider that you want to print alphabets a to c on the screen. We can do so by printing the values a, b and

c by writing 3 lines of code. Let us now look at the following pseudocode:

This was easy. Now consider a requirement where you we need to print numbers in incremental order from 1 to 1000. Although it is possible to print it following the above pseudocode, it will get a very tedious and lengthy process. This is where loops come into the picture to make this task easier. You can use the concept of loops and get the desired output by writing just a few lines of code.

5.3 Increment Loops

Loops provide the facility to execute a block of code repetitively, based on a condition. This block of code is executed repeatedly till the time a specified condition remains true. This condition is checked based on loop's control

Start

...This program demonstrates printing alphabets a to c

Print a

Print b

Print c

End

Output:

A

B

C



mind while programming that the condition should result false at a certain point in time. Otherwise, this block of code will enter an infinite loop.

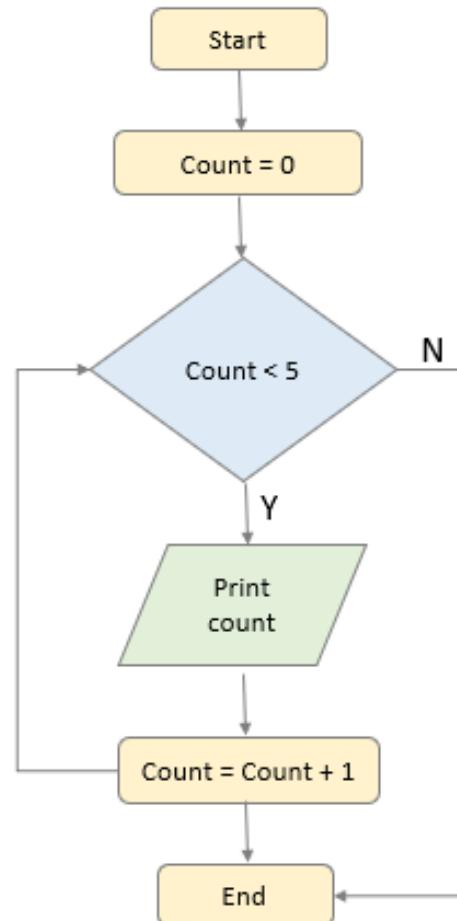
Execution of loops is based on iterations. To run a block of code in a loop, one needs to set a condition and set its number of iterations. Each time the condition is true, and the block of code executes once, it is counted to be one iteration. Before moving to the next iteration, one needs to increase the count of iteration to two. This is called as incrementing a loop.

For example, if you need to print numbers 0 to 4, you will execute a block of code with Print statement in five iterations. With each passing iteration, you will increment the count by one.

Below are the two important benefits of loops:

1. Reduces lines of code
2. Code becomes easier to understand

Let us understand loops with a flowchart. The following flowchart prints the numbers 1 to 5. Here every time the condition ($\text{Count} < 5$) is true, "Print count" gets executed. So, we do not have to write the "Print" statement multiple times. The loop takes care of that. What is important to note is every loop must have an exit condition. In our example the exit condition is ($\text{Count} < 5$). The loop will exit when the condition becomes false. Also, most loops will have a variable which in programming terms is called counter variable. The counter variable keeps track of how many times



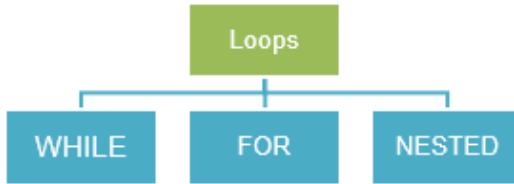
the loop executed. In this example, the "count" variable is our counter.

Often, counter variables are incremented within the loop.

5.4 Different types of loops

Loops make our code more manageable and organized. Let us now see what the different types of loops are:

1. While Loop
2. For Loop
3. Nested Loop



The While Loop

The While loop can execute a set of commands till the condition is true. While Loops are also called conditional loops.

Once the condition is met then the loop is finished. Let's now see a few examples:

Example 1 - Print number from 1 to 15

Here, if we want to derive the loop from this scenario, we have the following conditions:

Condition: Write from 1 to 15

And the decision we are deriving is:

Decision: Have we reached 15

Based on this we can write the below pseudocode:

```
y = 0
while y < 15
    y+=1
    print(y)
```

Example 2 - Print a statement 3 times using while loop.

Condition: Till the time the test expression remains true we will run the body of while loop

And the decision we are deriving is:

Decision: Have we printed the statement 3 times

Based on this we can write the below pseudocode:

```
y = 0
while y < 3
    y+=1
    print("I will eat healthy food every day.")
```

This is the result for the code

```
I will eat healthy food every day.
I will eat healthy food every day.
I will eat healthy food every day.
```

Example 3 - Print a statement 3 times after that print different statement.

Condition: Till the time the test expression remains true we will run the body of while loop printing statement 1 and after the condition is false, we will print statement 2.

And the decision we are deriving is:

Decision: Have we printed the statement 1 three times, if yes then print the statement 2

Based on this we can write the below pseudocode:

```
y = 0
while y < 3
    y+=1
    print("Please,")
else
    print("I want to play.")
```

This is the result for the code



Please,
Please,
Please,
I want to play.

Example 4 - Print number from 10 to 1

Condition: Till the time the test expression remains true we will run the body of while loop printing statement and after the condition is false the loop will stop

And the decision we are deriving is:

Decision: Have we printed 1

Based on this we can write the below pseudocode:

```
x = 10
while x==1
print(x)
x-=1
```

This is the result for the code

```
10
9
8
7
6
5
4
3
2
1
```

The For Loop

For loop is needed for iterating over a sequence. A for loop executes for a specific number of times.

Now let us see a few examples:

Example 1 – We need to print a statement five times using the for loop. The pseudo code for this will look like below:

```
for i in range(5)
    print("I will study every day.")
    i++
```

range(5) used in the above snippet returns a sequence of numbers from 0 to 5. This is the result for the code

```
I will study every day.
```

Example 2 – Use for loop to print all the odd numbers from a list.

The pseudo code for this will look like below:

```
list1 = [13, 24, 7, 49, 74, 29]
for x in list1:
    if x % 2!= 0:
        print(x, end = " ")
```

This is the result for the code

```
13 7 49 29
```

Example 3 – Use for loop to print all the even numbers from a list.

The pseudo code for this will look like below:



```
list2 = [13, 24, 7, 49, 74, 29]
for x in list2:
    if x % 2 == 0:
        print(x, end = " ")
```

This is the result for the code

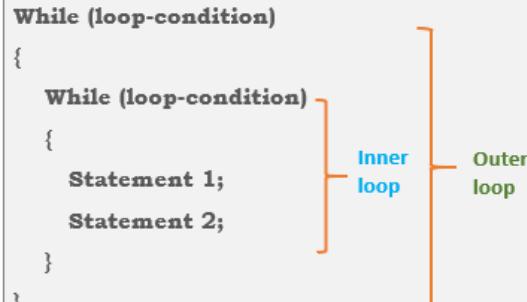
```
24 74
```

The Nested Loop

Any loop in program may contain another loop inside it. When there is a loop inside another loop, it is called a nested loop. How it works is that first success condition of outer loop triggers the inner loop which runs and reaches completion. This combination of loops inside loop is helpful while working with requirements where user wants to work of multiple conditions simultaneously. There is no restriction on how many loops can be nested inside a loop.

To understand the concept of Nested loops better, consider an example of an Analogue clock. An analogue clock has one hand as the nested loop and every full rotation knocks the minute hand on by one etc. We can take this even further to say that clocks are just a form of the counting system. This is how nested loops work in real life.

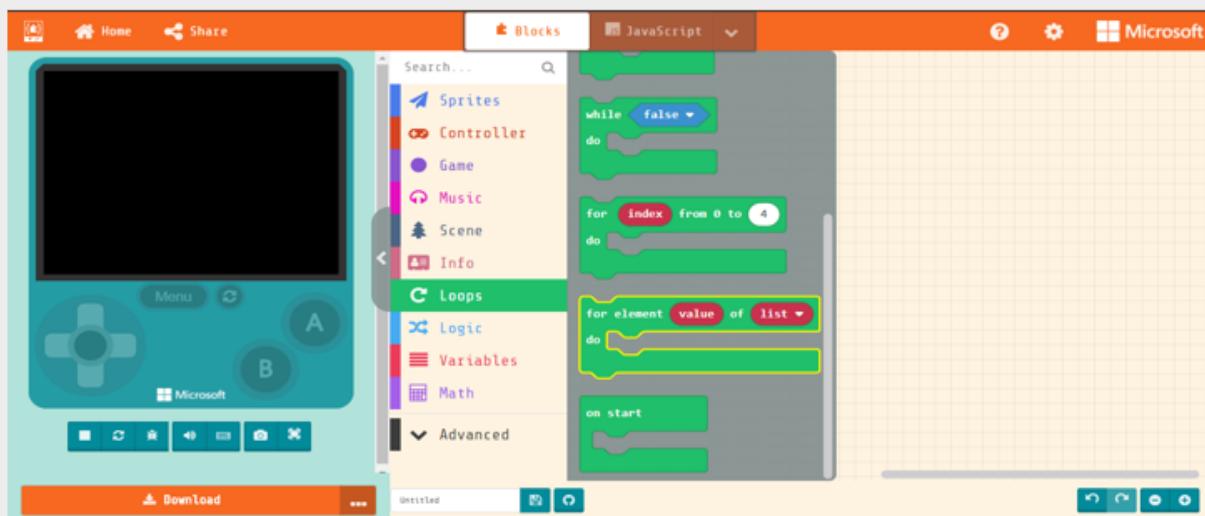
Below is an example of nested loop.



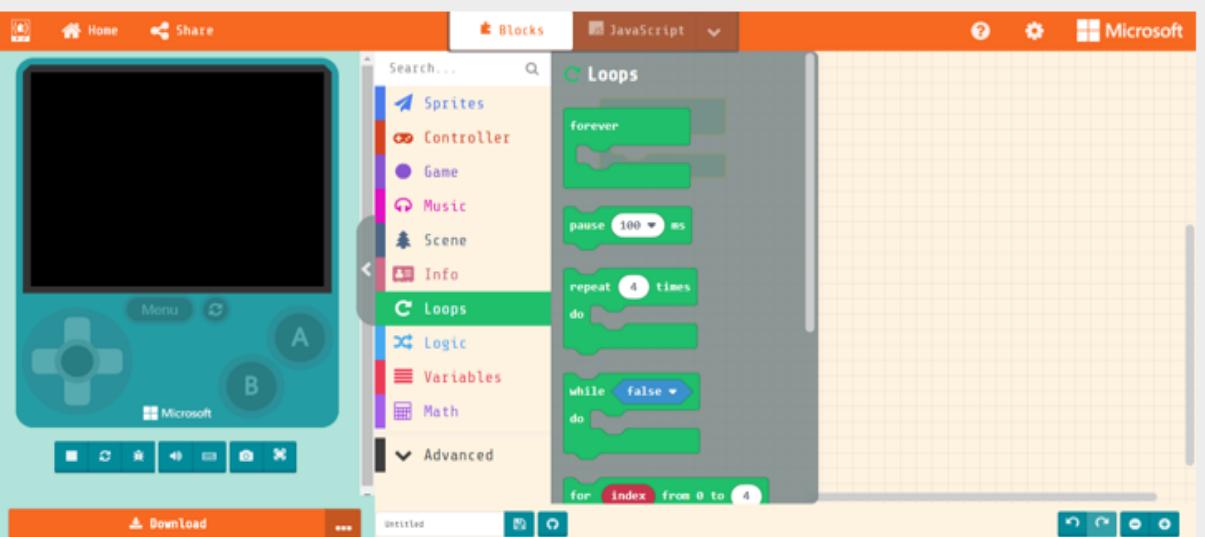
Now that we know the concept of loops let us try this out with block coding. We will build a very simple music player using the concepts of loops.

5.5 Activity: Building a music player

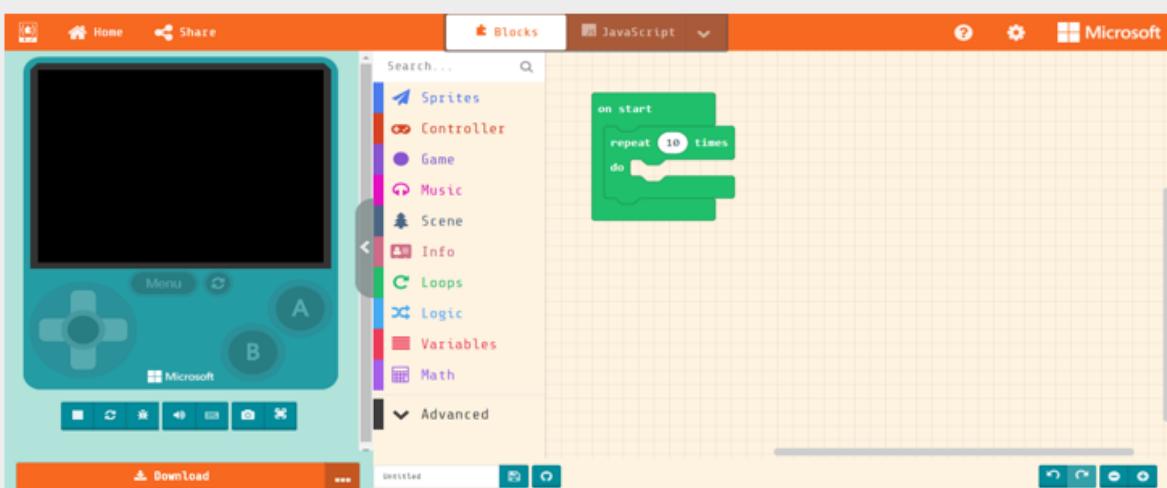
We will implement the concept of incrementing loops with the help of Arcade platform. Let us go through the below exercise step by step.



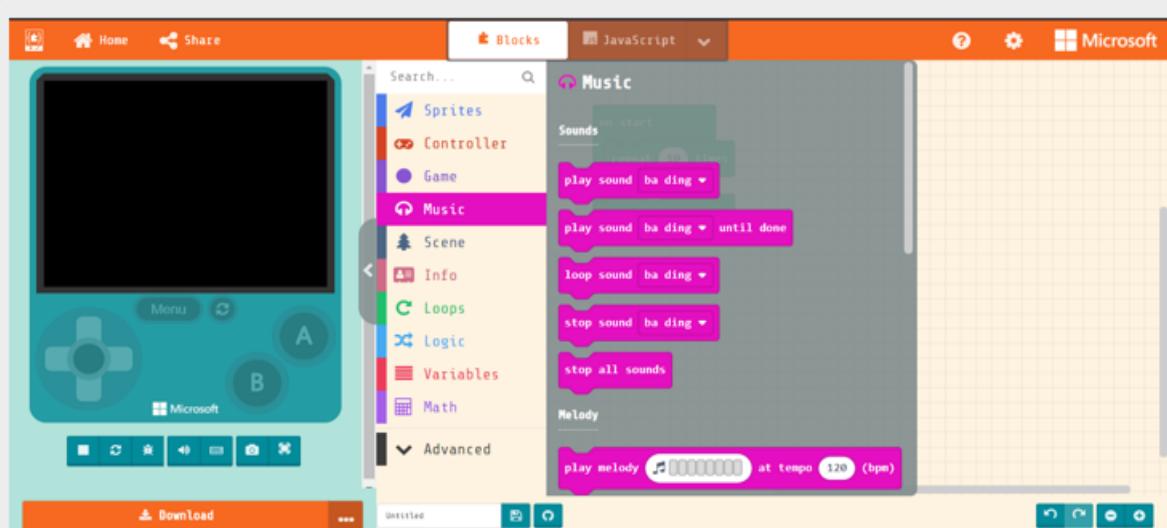
Step 1: In the arcade MakeCode editor, click on “Loops” link in the toolbox and then drag and drop “on start” block to the play area.



Step 2: Again, click on the “Loops” link in Toolbox in the center of the page. Drag and drop “repeat 4 time do” block to the play area as shown in the image below



Step 3: Now, move the “repeat” block and place it inside “start” block. Also, reset the value in “repeat” block to “10”



Step 4: Click on the “Music” link from the toolbox and drag and drop “play sound until done” block in the play area



The image shows the Scratch programming environment. On the left, there's a preview window with a black screen, a control panel with a joystick and buttons labeled A and B, and a download button. The main workspace has a script on the stage:

```
on start
repeat (10)
  do [play sound [ba ding v] until done]
end
```

A sidebar on the left lists categories: Sprites, Controller, Game, Music, Scene, Info, Loops, Logic, Variables, Math, and Advanced. The 'Loops' category is currently selected. A search bar at the top says 'Blocks'.

Step 5: Fix the "play sound until done" block in the "repeat" block

When you click on the play for the above exercise that we created, you will hear the music 10 times. The loop repeats itself 10 times (counter that we had set).

The flow starts with 1 and in each iteration, it increments once. Each time loop completes once iteration, and you will hear music once.

5.6 Entry Criteria

Now that we have understood about different loops and its iterations, it is also important to understand when and where should one start iterating through these loops. When the looping condition is true then the code will enter a loop. It is important to define an entry criterion for the loop to ensure that the loop runs.

Entry criteria is defined as a condition that must be met before starting a specific task. It is a set of conditions that

must exist before you can start a task. These criteria differ from program to program as per the requirement.

For example, To start a car you need petrol/diesel. If your fuel tank is empty your car won't start. So, the entry criteria for the car to start is fuel tank should not be empty.

5.7 Exit Criteria

Now that we have understood about loops and its iterations, it is also important to understand when and where should one stop iterating through these loops. As mentioned in the previous topic, it is crucial to keep in mind that the looping condition should result false at a certain point of time while programming. Otherwise, the block of code will enter an infinite loop. To ensure that the loop does not enter



an infinite loop, it is important to define an exit criterion for the loop.

Exit criteria is defined as a condition that must be met before completing a specific task. It is a set of conditions that must exist before you can declare a program to be complete. Exit criteria is one of the most important components while defining a loop. As without an exit criterion, the program tends to enter in an infinite loop. These criteria differ from program to program as per the requirement.

For example, while creating a loop to print numbers from 1 to 1000, exit criteria is that loop should exit the block of code when the 1000th number is printed, else the program will enter an infinite loop.

5.8 Break Statement

The break statement modifies the normal flow of execution while it terminates the existing loop and continues execution of the statement following that loop. Break statement is required as sometimes you want to break out of a loop early when a condition is met.

Let us now understand Break Statement with help of below pseudocode:

Example – Use break statement to

While (loop-condition)

```
{  
    Statement 1;  
    Statement 2;  
    .....  
    Break;  
    .....  
}
```

Statement

Execution skips the lines after the break statement within the loop body

print number from 9 to 5 and another print statement when you encounter break

The pseudo code for this will look like below:

```
a = 9  
while a > 0:  
    print 'Current variable is:', a  
    a = a - 1  
    if a == 5:  
        break  
  
    print "Bye!"
```

This is the result for the code

```
Current variable is: 9  
Current variable is: 8  
Current variable is: 7  
Current variable is: 6  
Bye!
```

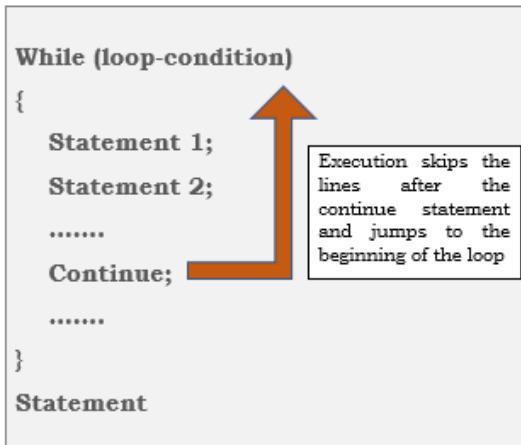
As we see the control skips the lines after the break statement and executes the first statement outside the loop.



5.9 Continue Statement

Whenever a program comes across a continue statement, the control skips the execution of remaining statements inside the loop for the current iteration and jumps to the beginning of the loop for the next iteration. If the loop's condition is still true, it enters the loop again, else the control will be moved to the statement immediately after the loop. This is somewhat similar to break statement and is used when we want to force the next iteration and skip some lines of code within the loop.

Let us now understand continue statements with the below pseudocode:



As you can see, as soon as the continue statement is encountered, the lines below the continue statement are skipped. But unlike the break statement, the loop is not terminated. Instead, the control jumps to the beginning of the loop.

Example – Use continue statement to print number from 9 to 0 and does not print when the value is 5

The pseudo code for this will look like below:

```
b = 9
while b > 0:
    b = b - 1
    if b == 5:
        continue
    print 'Current variable is:', a
    print "Bye!"
```

This is the result for the code

```
Current variable is: 9
Current variable is: 8
Current variable is: 7
Current variable is: 6
Current variable is: 4
Current variable is: 3
Current variable is: 2
Current variable is: 1
Current variable is: 0
Bye!
```

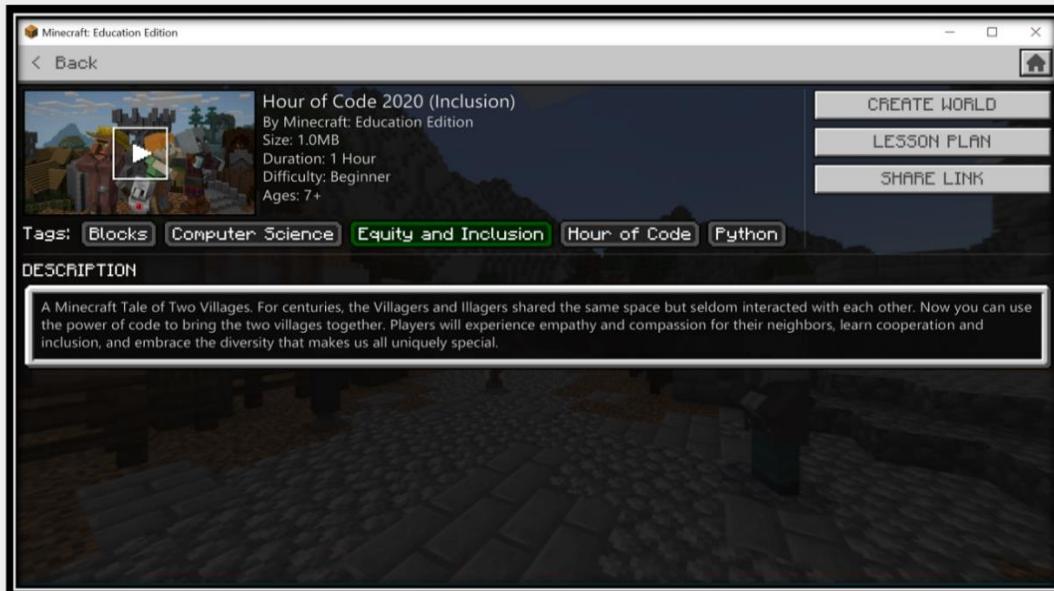
Now that we know the concepts of loops let us do an activity.

5.10 Activity: A tale of two villages

In this exercise, we will use basic coding concepts to bring two villages together in **Minecraft: Education Edition**. You setup Minecraft education edition from <https://education.minecraft.net/get-started>



Step 1: Click on try a demo lesson and accept the terms and conditions to start the activity



Step 2: Choose the 2020 lesson and click on “Create World”



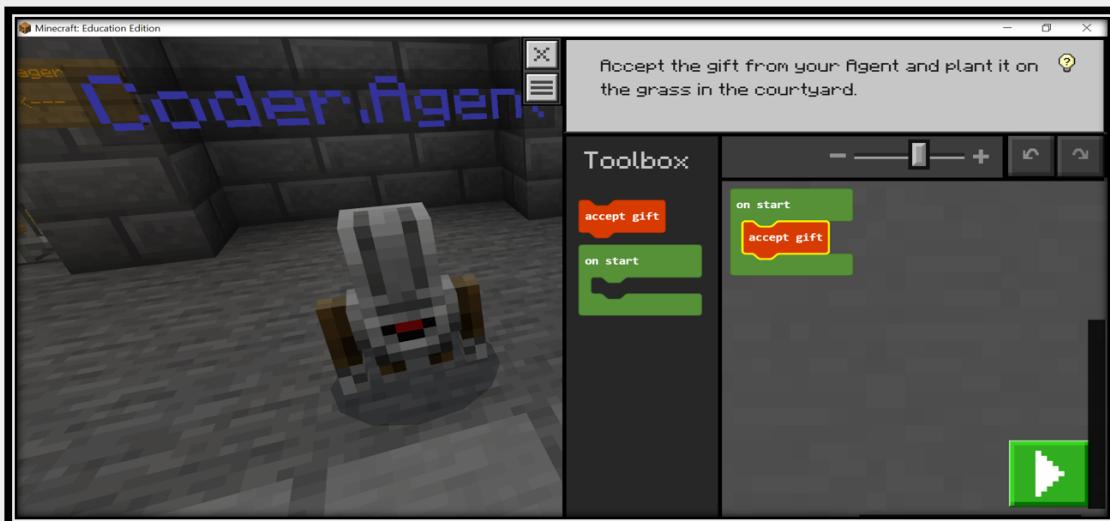
Step 3: In this world the villagers and the illagers share the same space but they seldom interact. Your job is to bring the two villages together to learn from each other and appreciate each other too. Start your journey at the castle by learning to move using the keyboard and mouse or touch commands.



Step 4: Press W to move forward. When you see a non-Player character, Right click to read what they have to say. Your onboarding activities will help you decide which coding language to use, either MakeCode Blocks or Python. For this activity we will use blocks.



Step 5: Move the agent forward just enough so that it can stand on the gold block. To start coding press C or tap the agent icon at the top of the screen if you're using touch. Remember to press play when ready to run your code.



Step 6: The agent has given you a gift. Plant your sapling with code, and it will grow as you complete the challenges in the villages. Come back and see your progress after each challenge.



There are many other challenges that can be done. Let us do one more to help villagers learn how to farm.



Step 7: Teach farming – In this activity the agent will show the illagers how to till two rows of land



Step 8: These are the following steps we are going to do:

- Till and move the agent forward by 9 times
- Turn the agent right
- Move agent forward by 1
- Turn the agent right
- Till and move forward the agent by 10 times
- Click on play



Result: Preparing field for farming activity is complete

Refer <https://education.minecraft.net/hour-of-code-2020> for the entire activity.

5.11 Quiz time

Objective Type Questions

Question 1	The if statement is used to execute some code if a statement is true.
Option 1	True
Option 2	False

Question 2	An else statement should always be after an if statement which executes when the code is false.
Option 1	True
Option 2	False



Question 3	A while loop statement repeatedly executes a statement as long as the condition remains true
Option 1	True
Option 2	False

Question 4	Without a statement that eventually evaluates the while loop condition to false, the loop will continue indefinitely
Option 1	True
Option 2	False

Question 5	A for loop executes for a specific number of times
Option 1	True
Option 2	False

Question 6	A continue statement is used to skip all the remaining statements in the loop and moves the control back to the top of the loop.
Option 1	True
Option 2	False

Question 7	When a break statement is encountered inside a loop, the loop is immediately terminated, and the program execution moves on to the next statement in the loop.
Option 1	True
Option 2	False

Question 8	Doing something over and over again or repeating code is called as
Option 1	Code
Option 2	loop
Option 3	Program
Option 4	Bug

Question 9	Which is the correct operator for equality testing?
Option 1	==
Option 2	=
Option 3	!=



Option 4	<code>+ =</code>
----------	------------------

Question 10	What is the output of the below pseudocode?
Option 1	5
Option 2	10
Option 3	15
Option 4	0

```
count = 0;  
sum = 0;  
while (count < 5)  
{  
    sum = sum + count;  
    count = count + 1;  
}  
print sum;
```

Question 11	Which letter won't print while running the below pseudocode?
Option 1	'd'
Option 2	'c'
Option 3	'n'
Option 4	'o'

```
for letter in "coding":  
    if letter == "i":  
        break  
    print(letter)  
print("End")
```



Question 12	Which letter won't print while running the below pseudocode?
Option 1	'g'
Option 2	'o'
Option 3	'd'
Option 4	'i'

```
for letter in "coding":  
    if letter == "i":  
        continue  
    print(letter)  
print("End")
```

Short Answer Questions

1. Define loops and nested loops in programming
2. What is an exit criterion?
3. How do we increment loops?
4. What is a break statement?
5. What is a continue statement?

Higher Order Thinking Skills (HOTS)

Now that you know the concept of Nested Loops. Using Arcade MakeCode platform create an example for nested loops.

Applied Project

Write a program using loops that ask the user to enter an even number. If the number entered does not display an appropriate message and asks them to enter a number again. Do not stop until an even number is entered. Print a congratulatory message at end.



5.12 What did you learn in this chapter?

- You should now have an idea of what are loops and how they are used in programming?
- Incrementing loops
- Different types of loops
- What is entry and exit criteria in loops and its usage?
- Break and continue statements
- Concept of nested loops



REFERENCES

Microsoft. 2021. Microsoft MakeCode Arcade. [Online]. [25 February 2021]. Available from: <https://arcade.makecode.com>

Microsoft. 2021. Microsoft MakeCode for Minecraft. [Online]. [25 February 2021]. Available from: <https://minecraft.makecode.com>

Microsoft. 2021. Computer Science Subject Kit | Minecraft: Education Edition. [Online]. [25 February 2021]. Available from: <https://education.minecraft.net/class-resources/computer-science-subject-kit>

Microsoft. 2020. Activity: We Built a Zoo. [Online]. [25 February 2021]. Available from: <https://education.minecraft.net/class-resources/computer-science-subject-kit>

ACM, Inc. 2021. Code of Ethics. [Online]. [25 February 2021]. Available from: <https://www.acm.org/code-of-ethics>

Microsoft. 2020. Hour Of Code 2020 | Minecraft: Education Edition. [Online]. [25 February 2021]. Available from: <https://education.minecraft.net/hour-of-code-2020>