GRADE 8 | PROGRAMMING

LESSON 1: EXPLORING THE WORLD OF CODE

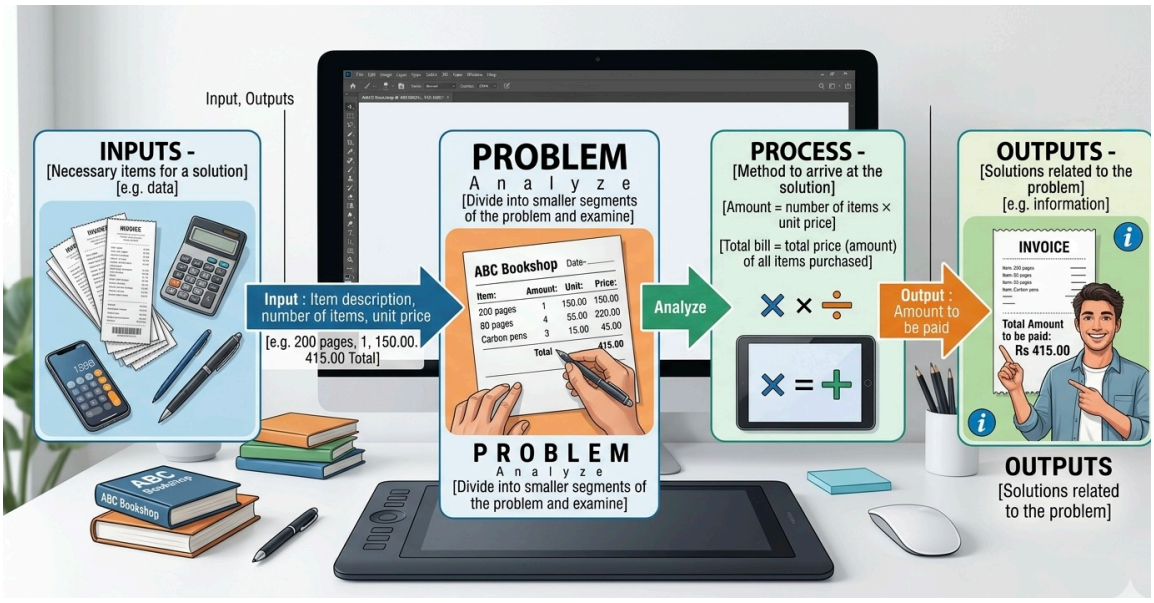
4.Problem Solving, Programming & Smart Devices

To become a good programmer, you must first become a good problem solver. Computers cannot think for themselves; they rely on humans to break down big problems into tiny, logical steps. This lesson will teach you how to analyze those problems, direct the flow of a program, and understand the smart devices that run these programs every day.

1. The Core of Computer Science: Problem Analysis

Before writing a single line of code, programmers use the **IPO (Input-Process-Output) Cycle** to understand exactly what a program needs to do.

- **Input (The Ingredients):** The raw data, variables, or items necessary to solve the problem.
- **Process (The Recipe):** The mathematical calculations, comparisons, or methods used to transform the input into the final answer.
- **Output (The Result):** The final information, solution, or display presented to the user.



Example 1: Calculating a Student's Term Average Imagine you are building a gradebook app for a teacher.

- **Inputs:** Marks for Math, Science, and English.
- **Process:** Add the three marks together to find the total, then divide that total by 3.
- **Output:** The student's average score for the term.

Example 2: ATM Cash Withdrawal

- **Inputs:** The user's ATM card details, their secret PIN, and the amount they want to withdraw.
- **Process:** Check if the PIN matches the bank's records. Check if the account balance is higher than the requested amount. Subtract the requested amount from the balance.
- **Output:** The physical cash dispensed from the machine and a printed receipt.

Problem 1: Making Toast

You put a slice of bread into a toaster and push the lever down. After a minute, it pops up.

- **Input:** * **Process:** * **Output:**

- **Input:** Slice of bread, the toaster, electricity.
- **Process:** Heating the bread.
- **Output:** Toast.

- **Problem 2: Squeezing Orange Juice**

You cut oranges in half and press them onto a manual juicer to make a drink.

- **Input:** * **Process:** * **Output:**

Problem 2: Squeezing Orange Juice

- **Input:** Oranges, the juicer.
- **Process:** Squeezing the oranges.
- **Output:** Orange juice.

- **Problem 3: Brushing Your Teeth**

You put toothpaste on your toothbrush and clean your teeth for two minutes.

- **Input:** * **Process:** * **Output:**

Problem 3: Brushing Your Teeth

- **Input:** Toothbrush, toothpaste, dirty teeth.
- **Process:** Brushing and scrubbing the teeth.
- **Output:** Clean teeth.

Simple Computing Scenarios

Instructions for Students: Think about how computers and machines work. Identify the Input, Process, and Output.

Problem 4: Printing a Document

You finish typing an essay and click "Print" on your computer so you can hand it in to your teacher.

- **Input:** * **Process:** * **Output:** printed document
- **Problem 5: Using a TV Remote**

You are watching TV and want to make it louder, so you press the "Volume Up" button on your remote.

- **Input:** * **Process:** * **Output:**
- **Problem 6: A Simple Addition Program**

A computer program asks you to type in two numbers, and then it tells you the total sum of those numbers.

- **Input:** * **Process:** * **Output:** ---

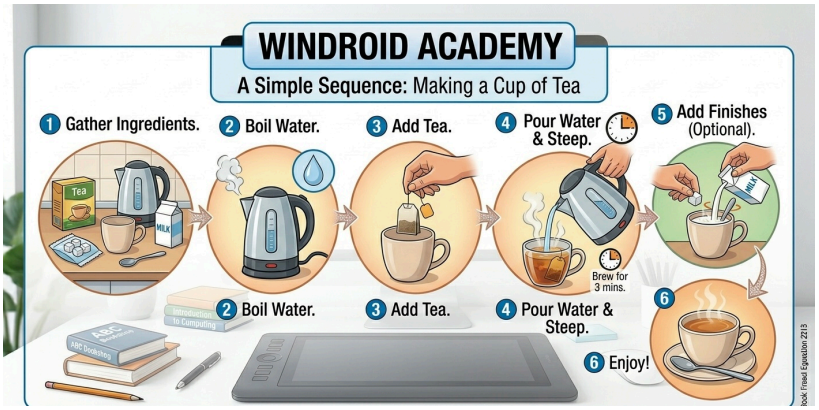
2. Control Structures: Steering the Program

A **control structure** is a block of programming that analyzes variables and dictates the direction the program will flow. Think of them as traffic cops for your code.

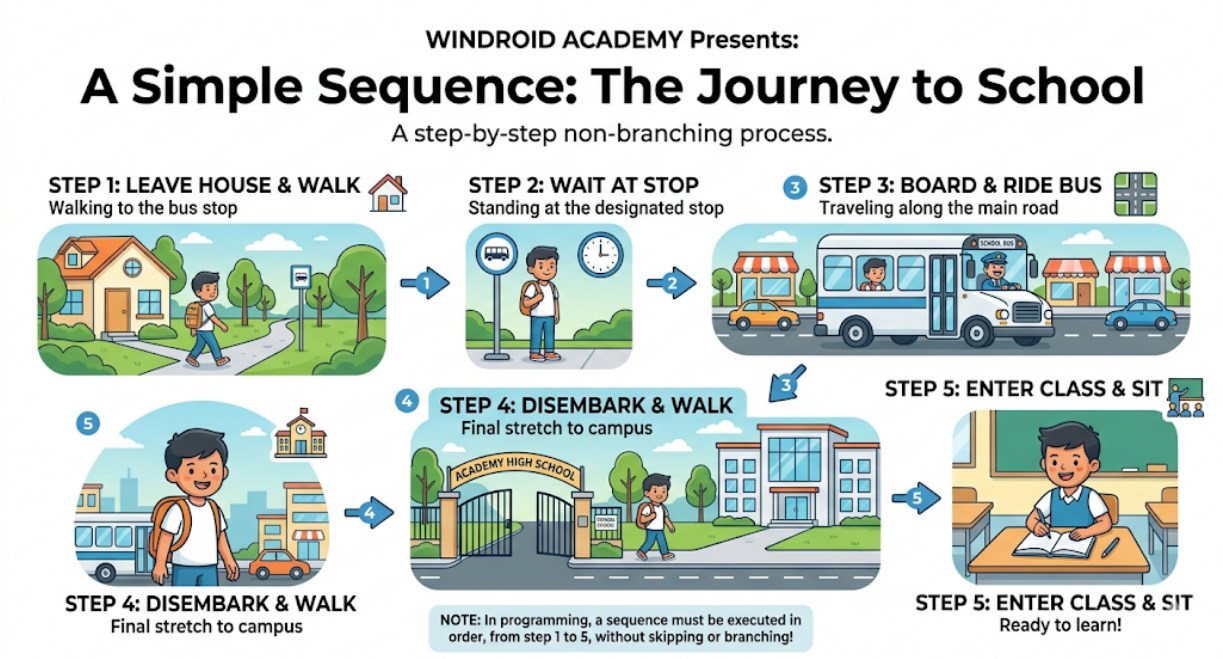
A. Sequence (Straight Line Logic)

This is when instructions are executed one after the other in a strict order, from top to bottom. The computer does not skip any steps.

- **New Scenario: A simple tea-making robot.**
    1. Input:Water, sugar, tea bag
    2. Processes:
        1. Let the water boil in the kettle
        2. Pour hot water into a cup with a teabag.
        3. Wait for 3 minutes,
        4. then add sugar.
    3. Output:Serve the tea.



- **Go to the school:**
    - **Get away from home and turn right**
    - **Walk straight for 200m**
    - **Then turn right and walk straight for 800m**
    - **Arival to the Waliveriya town**
    - **Cross the pedestrian crossing and wait for the bus to Yakkala**
    - **Wait 30 minutes until getting down from the bus**

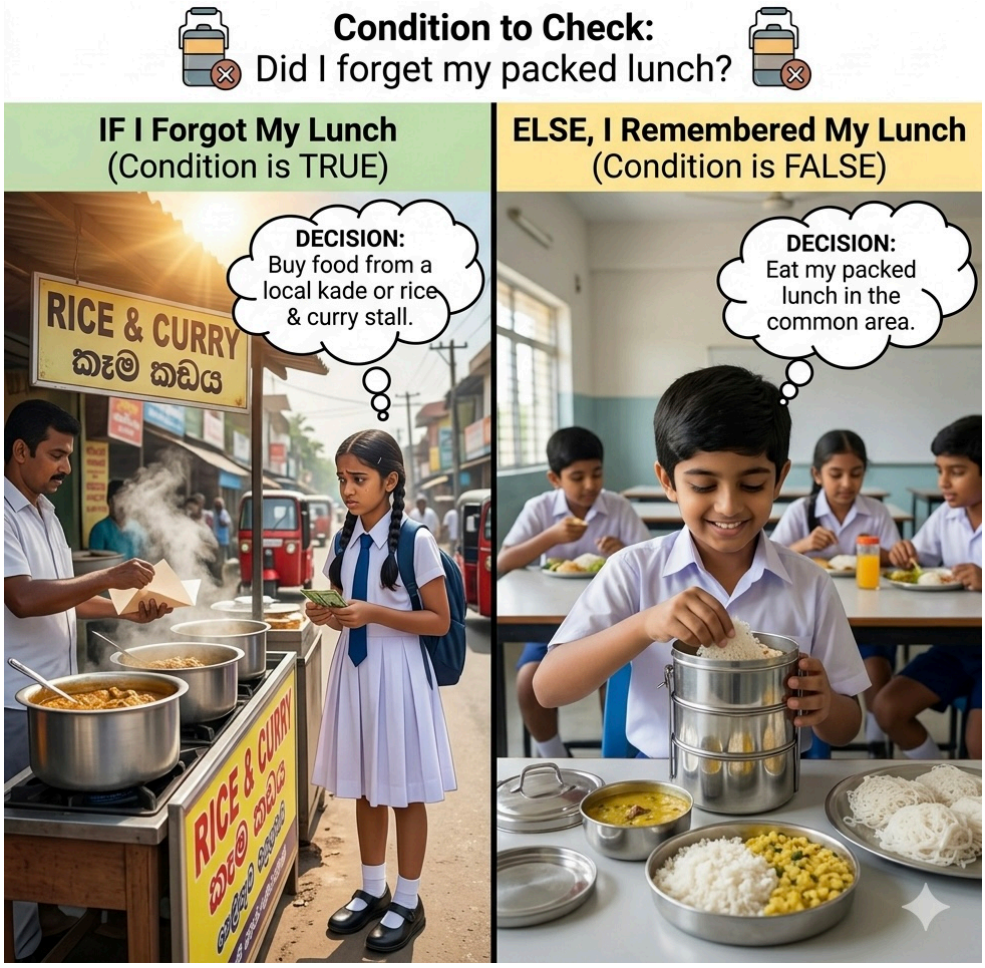


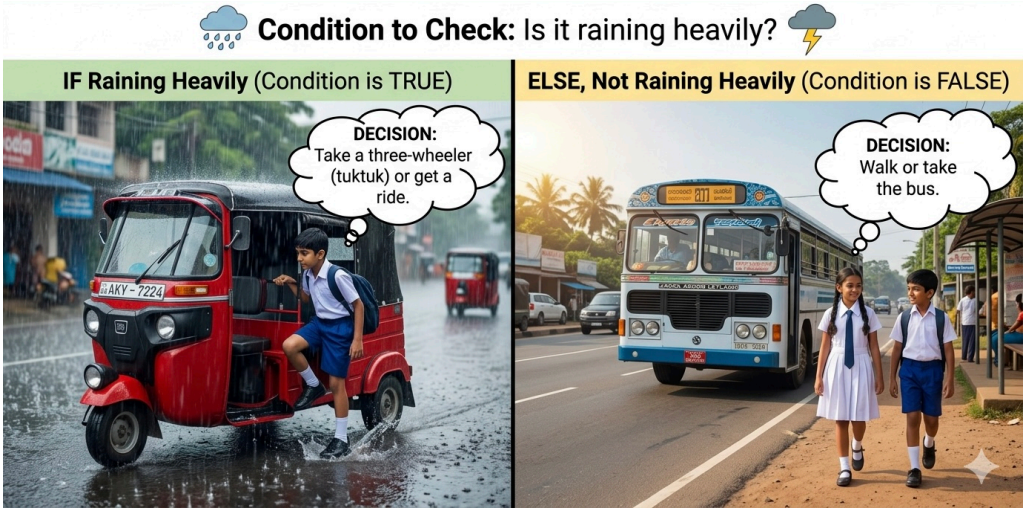
B. Selection (Decision Making)

Selection allows a program to branch off into different paths based on a **Condition**. If a condition is met (True), the program does one thing. If it is not met (False), it does something else.

- **New Scenario 1 (Traffic Lights):** *
    - Drive your car
    - See the color lights

- See whether the color light is red?
    - ***True (Yes):*** **Stop the car.**
    - ***False (No):*** **Keep driving.**





- **New Scenario 2 (Exam Results):**

- **Enter students marks**
- **Check whether the mark is equal to 50 or higher**
- **Check if marks>=50**
  - ***True (Yes):*** **Print "Congratulations, you passed!**
  - ***False (No):*** **Print "You failed, please try again."**

1. Enter the password for login to the computer

Turn on your computer

Wait until login screen

Enter the password

If password="Entered Password"

You can log in

Else

You can't login

2. Consider that you can vote for an election if your age is over 18 or equal

Enter your age

Check whether the age>=18

If yes

"You can vote"

Else

"No you can't vote"

3. If you have paid to the class you will receive the note

Enter the student details

Check for payment details

If the payment for the Month is done?

If yes

“Provide the note”

If Not

“Access without the note”

4. Take your umbrella by checking the weather condition(just see, check the weather app)

Check for the weather report using TV or the Mobile

Check is it raining ?

If yes

Bring Your umbrella

If Not

Go without umbrella

C. Repetition (Looping)

What is Repetition?

Imagine having to write "I will not talk in class" 100 times on a chalkboard. Doing it step-by-step takes a long time and gets boring. Computers, however, love doing the same thing over and over again!

Repetition (also called "looping") is a control structure that tells the computer to repeat a specific block of instructions multiple times without you having to write the instruction out every single time.

Normal Everyday Examples

We use repetition in our daily lives all the time without realizing it:

- **Eating a meal: Repeat** [take a bite] **Until** [the plate is empty].
- **Washing hands: Repeat** [scrub hands together] **20 times**.
- **Waiting for a bus: Repeat** [check the road] **Until** [the bus arrives].

Understanding the Everyday Examples

- **Eating a meal (Repeat Until):** You do not tell your brain, "I will take exactly 42 bites." Instead, your brain uses a "Repeat Until" loop. The action is taking a bite. You repeat this action until the condition (the plate is empty) becomes true.
- **Washing hands (Repeat Number):** When you wash your hands, you might count out 20 seconds. The action is scrubbing. The loop runs for a specific, set number of times (20 times) and then stops automatically.

- **Waiting for a bus (Repeat Until):** You stand at the bus stop. Your action is checking the road. You do not check the road 5 times and then walk away; you repeat the action of looking down the road until you see the bus arrive.

2. How to Build These Examples in Scratch

Here is how you can recreate these loops in Scratch. You will need to look for blocks by their color categories (like **Blue** for Motion, **Yellow** for Events, and **Orange** for Control).

Example A: The Square (Using the "Repeat" Block)

This program tells the Scratch Cat to draw a square. Because a square has 4 equal sides and 4 equal corners, we can repeat the same two movements 4 times.

Steps to build it:

1. Go to the **Events** (Yellow) category and drag out the When [Green Flag] clicked block.
2. Go to the **Control** (Orange) category and drag out a Repeat (10) block. Attach it below the yellow block.
3. Click on the number "10" inside the repeat block and type **4** (since a square has 4 sides).
4. Go to the **Motion** (Blue) category and drag out move (10) steps. Snap it inside the mouth of the repeat block. Change the "10" to **100** so the cat takes a big step.
5. STILL in the **Motion** category, drag out turn right (15) degrees. Snap it directly under the move block, but still inside the repeat block. Change "15" to **90** (for a 90-degree corner).

When you click the Green Flag, the cat will instantly trace the shape of a square!

Example B: The Spinning Star (Using the "Forever" Block)

This program will make a sprite spin continuously.

Steps to build it:

1. Add a new sprite from the Scratch library (like a Star or a Ball).
2. Go to the **Events** (Yellow) category and drag out the When [Green Flag] clicked block.
3. Go to the **Control** (Orange) category and drag out a Forever block. Attach it to the yellow block.
4. Go to the **Motion** (Blue) category and drag out turn right (15) degrees.
5. Snap the turn block inside the mouth of the Forever block.

When you click the Green Flag, the star will spin round and round and will not stop until you click the red Stop button.

Example C: Running to the Wall (Using the "Repeat Until" Block)

This program makes the cat walk forward, but forces it to stop the exact moment it bumps into the edge of the screen.[1]

Steps to build it:

1. Go to the **Events** (Yellow) category and drag out the When [Green Flag] clicked block.
2. Go to the **Control** (Orange) category and drag out a Repeat until < > block. Attach it to the yellow block.
3. Go to the **Sensing** (Light Blue) category. Find the touching [mouse-pointer] ? block.
4. Drag that light blue block and drop it inside the diamond-shaped hole < > of the Repeat until block. Click the little arrow next to "mouse-pointer" and change it to **edge**.

5. Go to the **Motion** (Blue) category and drag out move (10) steps. Snap it inside the mouth of the Repeat until block.

What is a Flowchart? A flowchart is a visual map of a computer program. Instead of reading lines of code, programmers use shapes and arrows to show exactly how a program thinks and moves from step to step. It is the blueprint of your logic.

The Universal Language of Shapes To make sure every programmer in the world understands the map, we use very specific shapes for different actions:

Symbol	Name	Function
	Start/end	An oval represents a start or end point
	Arrows	A line is a connector that shows relationships between the representative shapes
	Input/Output	A parallelogram represents input or output
	Process	A rectangle represents a process
	Decision	A diamond indicates a decision

- **The Oval (Terminal):** Every flowchart must have an Oval at the very top that says "Start" and one at the very bottom that says "End".
- **The Parallelogram (Input / Output):** Used when the program asks the user for information (Input) or shows the user an answer (Output).
- **The Rectangle (Process):** Used for actions, math calculations, or background work. This is the computer "doing" something.
- **The Diamond (Decision):** Used for **Selection** and **Repetition**. It asks a True/False question, and the path splits based on the answer.
- **The Arrows (Flow Lines):** These connect the shapes and show the strict direction the computer must read.

How the Control Structures Look Visually

- **Sequence:** A straight vertical line of rectangles and parallelograms pointing straight down.
- **Selection:** The path hits a Diamond, splits into two arrows ("Yes" and "No"), performs different actions, and then usually merges back together.
- **Repetition:** The path hits a Diamond, and if the condition isn't met yet, an arrow loops *backwards* up the flowchart to repeat previous steps.

Example 1: Voting Eligibility

The Rule: A person must be 18 years or older to vote. The computer checks their age to see if they are eligible.

The Flowchart:
[OVAL: Start]
⬇️
[PARALLELOGRAM: Input]
Enter your age
⬇️
[DIAMOND: Decision]
Is age greater than or equal to 18?

If YES ➡️ **[PARALLELOGRAM: Output]** Show "You can vote!" ➡️ **[OVAL: End]**

If NO ⬇️
[PARALLELOGRAM: Output]
Show "You are too young to vote."
⬇️
[OVAL: End]

Example 2: Fever Checker

The Rule: A digital thermometer checks your body temperature. If it is 38°C or higher, it warns you that you have a fever.

The Flowchart:
[OVAL: Start]
⬇️
[PARALLELOGRAM: Input]
Read body temperature
⬇️
[DIAMOND: Decision]
Is temperature 38 or higher?

If YES ➡️ **[PARALLELOGRAM: Output]** Show "Fever detected!" ➡️ **[OVAL: End]**

If NO ⬇️
[PARALLELOGRAM: Output]
Show "Temperature is normal."
⬇️
[OVAL: End]

Example 3: Low Battery Alert (Single Alternative)

The Rule: Your phone checks its battery. If the battery is below 20%, it shows a warning. If it is above 20%, it does nothing and just ends the check.

The Flowchart:
[OVAL: Start]
⬇️
[PARALLELOGRAM: Input]
Check battery percentage
⬇️
[DIAMOND: Decision]
Is battery less than 20%?

If YES ➡️ **[PARALLELOGRAM: Output]** Show "Battery Low, please charge!" ➡️
[OVAL: End]

If NO ⬇️
[OVAL: End]

Example 4: Positive or Negative Number

The Rule: A user types a number. The computer checks if the number is less than 0 to tell the user if it is negative or positive.

The Flowchart:
[OVAL: Start]
⬇️
[PARALLELOGRAM: Input]
Enter a number
⬇️
[DIAMOND: Decision]
Is the number less than 0?

If YES ➡️ **[PARALLELOGRAM: Output]** Show "This is a Negative number" ➡️ **[OVAL: End]**

If NO ⬇️
[PARALLELOGRAM: Output]
Show "This is a Positive number"
⬇️
[OVAL: End]

Example 5: Cinema Ticket Discount

The Rule: A cinema gives a half-price discount to children under 12. Everyone else pays the full price of Rs. 1000.

The Flowchart:
[OVAL: Start]
⬇️
[PARALLELOGRAM: Input]
Enter customer age
⬇️
[DIAMOND: Decision]
Is age less than 12?

If YES ➡️ **[RECTANGLE: Process]** Calculate Ticket = Rs. 500 ➡️
[PARALLELOGRAM: Output] Show Ticket Price ➡️ **[OVAL: End]**

If NO ⬇️

[RECTANGLE: Process]
Calculate Ticket = Rs. 1000
⬇️
[PARALLELOGRAM: Output]
Show Ticket Price
⬇️
[OVAL: End]

Example 6: Online Store Free Shipping

The Rule: If a customer's total bill is over Rs. 5000, they get free shipping. Otherwise, they must pay a Rs. 300 shipping fee.

The Flowchart:
[OVAL: Start]
⬇️
[PARALLELOGRAM: Input]
Get total bill amount
⬇️
[DIAMOND: Decision]
Is bill amount greater than 5000?

If YES ➡️ **[PARALLELOGRAM: Output]** Show "You get Free Shipping!" ➡️
[OVAL: End]

If NO ⬇️

[RECTANGLE: Process]
Add 300 to total bill
⬇️
[PARALLELOGRAM: Output]
Show "Shipping fee added. Show new total."
⬇️
[OVAL: End]

Example 7: Smart Plant Waterer

The Rule: A smart sensor checks the soil moisture of a plant. If the soil is dry, the water pump turns on. If it is wet, the pump stays off.

The Flowchart:
[OVAL: Start]
⬇️
[PARALLELOGRAM: Input]
Read soil moisture level
⬇️
[DIAMOND: Decision]
Is soil dry?

If YES ➡️ **[RECTANGLE: Process]** Turn on water pump ➡️ **[OVAL: End]**

If NO ⬇️
[RECTANGLE: Process]
Keep water pump off
⬇️
[OVAL: End]

Example 8: Speed Limit Camera

The Rule: A traffic camera measures a car's speed. If the speed is over 70 km/h, it takes a photo to issue a fine.

The Flowchart:
[OVAL: Start]
⬇️
[PARALLELOGRAM: Input]
Measure car speed
⬇️
[DIAMOND: Decision]
Is speed greater than 70?

If YES ➡️ **[RECTANGLE: Process]** Take a photo of the license plate ➡️ **[OVAL: End]**

If NO ⬇️
[OVAL: End]

Example 9: Library Overdue Fine

The Rule: When you return a library book, the computer checks if it is late. If it is late, it calculates a fine. If not, it just accepts the book.

The Flowchart:
[OVAL: Start]
⬇️
[PARALLELOGRAM: Input]
Scan returned book and get due date
⬇️
[DIAMOND: Decision]
Is today past the due date?

If YES ➡️ **[RECTANGLE: Process]** Calculate late fine ➡️ **[PARALLELOGRAM: Output]** Show "Please pay fine" ➡️ **[OVAL: End]**

If NO ⬇️
[PARALLELOGRAM: Output]
Show "Book returned successfully!"
⬇️
[OVAL: End]

Example 10: Game High Score Checker

The Rule: At the end of a video game, the computer checks if your current score is higher than the saved High Score. If it is, it updates the High Score.

The Flowchart:
[OVAL: Start]
⬇️
[PARALLELOGRAM: Input]
Get Player Score and Saved High Score
⬇️
[DIAMOND: Decision]
Is Player Score greater than (>) High Score?

If YES ➡️ **[RECTANGLE: Process]** Save Player Score as the new High Score ➡️ **[PARALLELOGRAM: Output]** Show "New High Score!" ➡️ **[OVAL: End]**

If NO ⬇️
[PARALLELOGRAM: Output]
Show "Game Over"
⬇️
[OVAL: End]