Loops

* Why we need loops?
* where they are used in the industry?
* Example from industry or real life, assembly line, water pump
* Types of loops and difference between them
  → for
  → while
  → Do while (only in C and Java)
* Nested loops
* Trade-offs between for and while
* Examples of for loop
* pattern printing
* Multiplication table demo

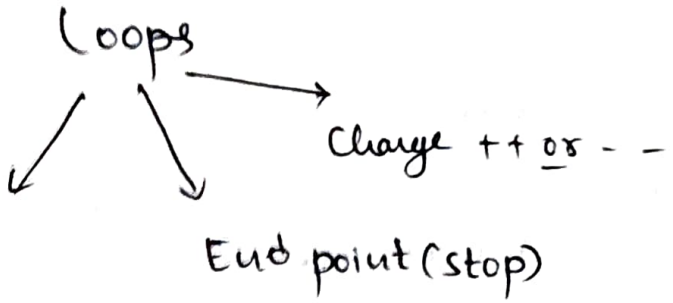
* why we need loops?

we need to find an item from the collection, so we are uses the loops.

Ex: data
Collection of students in that find the students name of 96% Grade

* loops can be used to generate 0 to 100 number or other number sequence and also used in arrays find the element from the index value

How to write loops



*Basically loops starts with the low value → Big value

1 ⟶ 100

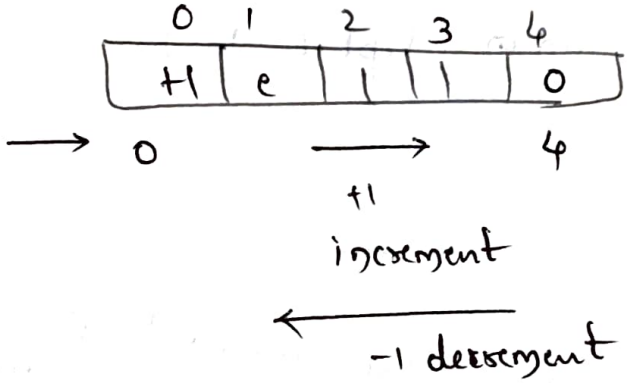
+1

increment

Big value ⟶ low value

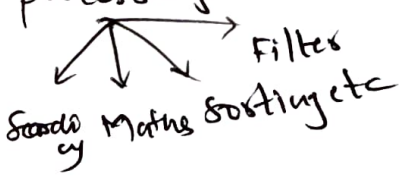
100 ⟶ 1

-1 decrement



① Set / collection of some items

② processing



*when ever we are using collection of data there is loops

③ Generation of sequence of number

1 → 100

2 → 200 Here also loop used

⑥ String "Raghu" → loop used
(↑ collection of characters)

① Generation of number

1 → 10

start → 1, stop = 10, +1 or -1 (Inc/Dec)

```
for(int i=0; i<=10; i++) i → index
        start   stop  incr   (or i++)
{
    printf("%d", index);
}
```

⑥ → we can call a function from here

for( ) → it also run if we kept it blank

for(int i=10; j=20; m=30)

index, count

Count=0
while (Condition)
{   Count<=10   * Condition is true while loop enters the iteration
___
___
___ → inc(++) | dec(--) write here
___
    Count ++;
}

---

do
{
[ ]
}
while (condition)

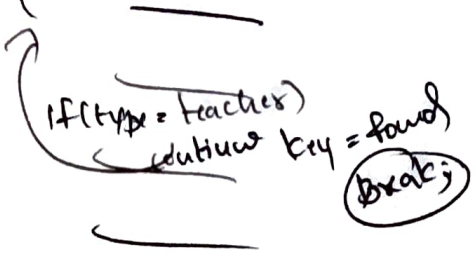
* instruction can be run atleast one time in dowhile loop.

break:

continue:

for (condition)

{



}

Example for continue statement:

student and teachess data all are in our we want go till student information and continue to condition

Loops in JAVA

Why We Need Loops

We use loops in programming to **automate repetitive tasks**. Instead of writing the same lines of code over and over, a loop allows you to execute a block of code multiple times. This makes your code more efficient, concise, and easier to maintain.

Where They Are Used in the Industry

Loops are used everywhere in software development. They are fundamental to processing data.

- **Data Processing:** Iterating through millions of records in a database to perform a calculation.
- **Web Development:** Displaying a list of products on an e-commerce website or a feed of social media posts.
- **Gaming:** Updating the position of every enemy character on the screen in each frame.
- **Robotics:** A robot arm repeatedly picking up and placing an item on an assembly line.
- **IoT:** A smart thermostat checking the room temperature every few minutes and adjusting the heating.

An excellent real-life example is a **water pump** that turns on and off to maintain a specific water level. The pump **loops** through a check: while (waterLevel < minLevel), it pumps water. Once the condition is false, the loop stops.

Types of Loops and Differences Between Them

1. For Loop

A **for loop** is used when you know exactly how many times you need to repeat a task. It's often used for iterating over a sequence of items or a specific range.

- **Syntax:** for (initialization; condition; increment)
- **Characteristics:** The loop's control variables (initializer, condition, and increment) are all defined in one line, making the loop's structure very clear.

Example: Printing numbers from 1 to 5

Java

```
public class ForLoopDemo {
   public static void main(String[] args) {
      for (int i = 1; i <= 5; i++) {
         System.out.println(i + " ");
      }
   }
```

```
}
```

Output: 1 2 3 4 5

2. While Loop

A **while loop** is used when you don't know the exact number of iterations. It continues to execute as long as a specified condition is true. The condition is checked at the beginning of each loop.

- **Syntax:** while (condition)
- **Characteristics:** It's more flexible than a for loop for tasks where the number of repetitions is unknown. You must manually update a variable inside the loop to eventually make the condition false, or you'll create an **infinite loop**.

Example: Simulating a user login

Java

```
import java.util.Scanner;


public class WhileLoopDemo {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String password = "mysecretpassword";
        String userInput = "";


        System.out.println("Please enter the password:");
        userInput = scanner.nextLine();


        while (!userInput.equals(password)) {
            System.out.println("Incorrect password. Please try again:");
            userInput = scanner.nextLine();
        }
        System.out.println("Login successful!");
        scanner.close();
    }
}
```

Explanation: The loop continues as long as the user's input does not match the password.

3. Do-While Loop

A **do-while loop** is similar to a while loop, but it guarantees the code block inside the loop runs **at least once** before the condition is checked. The condition is evaluated at the end of each iteration.

- **Syntax:** do { ... } while (condition);
- **Characteristics:** This is perfect for situations like a menu-driven program where you want to display the menu at least once before checking the user's choice.

Example: Displaying a menu

Java

```
import java.util.Scanner;


public class DoWhileLoopDemo {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        char choice;


        do {
            System.out.println("Menu:");
            System.out.println("A. Play game");
            System.out.println("B. View score");
            System.out.println("C. Exit");
            System.out.print("Enter your choice: ");
            choice = scanner.next().charAt(0);
        } while (choice != 'C' && choice != 'c');


        System.out.println("Exiting the program. Goodbye!");
        scanner.close();
    }
}
```

Explanation: The menu is shown, and the user's choice is read. The loop only continues if the user didn't choose to exit.

Trade-offs Between for and while Loops

The choice between a for loop and a while loop often comes down to readability and clarity.

Feature	For Loop	While Loop
Use Case	Ideal for a **fixed number** of iterations (e.g., iterating through an array).	Best when the number of iterations is **unknown** (e.g., reading user input until a specific value is entered).
Readability	More concise and readable when you have a counter variable.	Can be less readable if the loop's termination condition is complex or not clearly visible.
Risk	Less prone to infinite loops if set up correctly.	Higher risk of creating an **infinite loop** if the loop variable isn't updated properly inside the body.

Nested Loops

A **nested loop** is a loop inside another loop. This is useful for working with two-dimensional data (like grids or matrices) or for creating patterns. The inner loop completes all its iterations for each single iteration of the outer loop.

Example: Printing a 5x5 square of asterisks

Java

```
public class NestedLoopsDemo {
    public static void main(String[] args) {
        int size = 5;
        for (int i = 0; i < size; i++) { // Outer loop for rows
            for (int j = 0; j < size; j++) { // Inner loop for columns
                System.out.print("* ");
            }
            System.out.println(); // Move to the next line
        }
    }
}
```

Output:

```
* * * * * * * * * * * * * * * * * * * * * * * * * ```***
```

Pattern Printing Demo

This example uses nested loops to create a right-angled triangle.

```java
public class TrianglePatternDemo {
    public static void main(String[] args) {
        int rows = 5;
        for (int i = 1; i <= rows; i++) { // Outer loop for rows
            for (int j = 1; j <= i; j++) { // Inner loop for columns (prints stars)
                System.out.print("* ");
            }
            System.out.println(); // New line after each row
        }
    }
}
```

Output:

```
* * * * * * * * * * * * * * *
```

***

Multiplication Table Demo

This program uses a single `for` loop to print the multiplication table for a given number.

```java
public class MultiplicationTableDemo {
    public static void main(String[] args) {
        int num = 7;
        System.out.println("Multiplication Table for " + num + ":");
        for (int i = 1; i <= 10; i++) {
            System.out.println(num + " * " + i + " = " + (num * i));
        }
```

```
    }
}
```

Output:

Multiplication Table for 7:

7 * 1 = 7

7 * 2 = 14

...

7 * 10 = 70

Interview Questions on Loops

1. Why do we need loops in programming?
2. Can you explain real-life or industry scenarios where loops are usefull?
   Example: Assembly line, water pump, elevators, etc.
3. What are the different types of loops available in most programming languages?
4. What is the difference between a **for loop** and a **while loop**?
5. How does a **do-while loop** differ from a while loop? (especially in C/Java)
6. In which situations would you prefer a for loop over a while loop?
7. Can a for loop always be converted to a while loop? Give an example.
8. Write a program to print the **multiplication table of a number** using a loop.
9. How would you print a simple **pattern** (like a triangle or square) using nested loops?
10. Can you write a for loop example to print numbers from 1 to 100, skipping multiples of 5?
11. Write a while loop to calculate the sum of digits of a number.
12. What are **nested loops**? Give an example.
13. What is the time complexity of nested loops?
14. Can you provide a real-life scenario where nested loops are required?
15. What are the **trade-offs between for and while loops**?
16. Which loop is better for scenarios where the number of iterations is:
    (a) Known beforehand?
    (b) Unknown beforehand?
17. What happens if the loop condition is never met?
18. Can you create an **infinite loop** using each type of loop (for, while, do-while)?
19. How can you break out of a nested loop in Java/C?
20. What is the difference between break and continue statements inside loops?