



# **MODULE 1:**

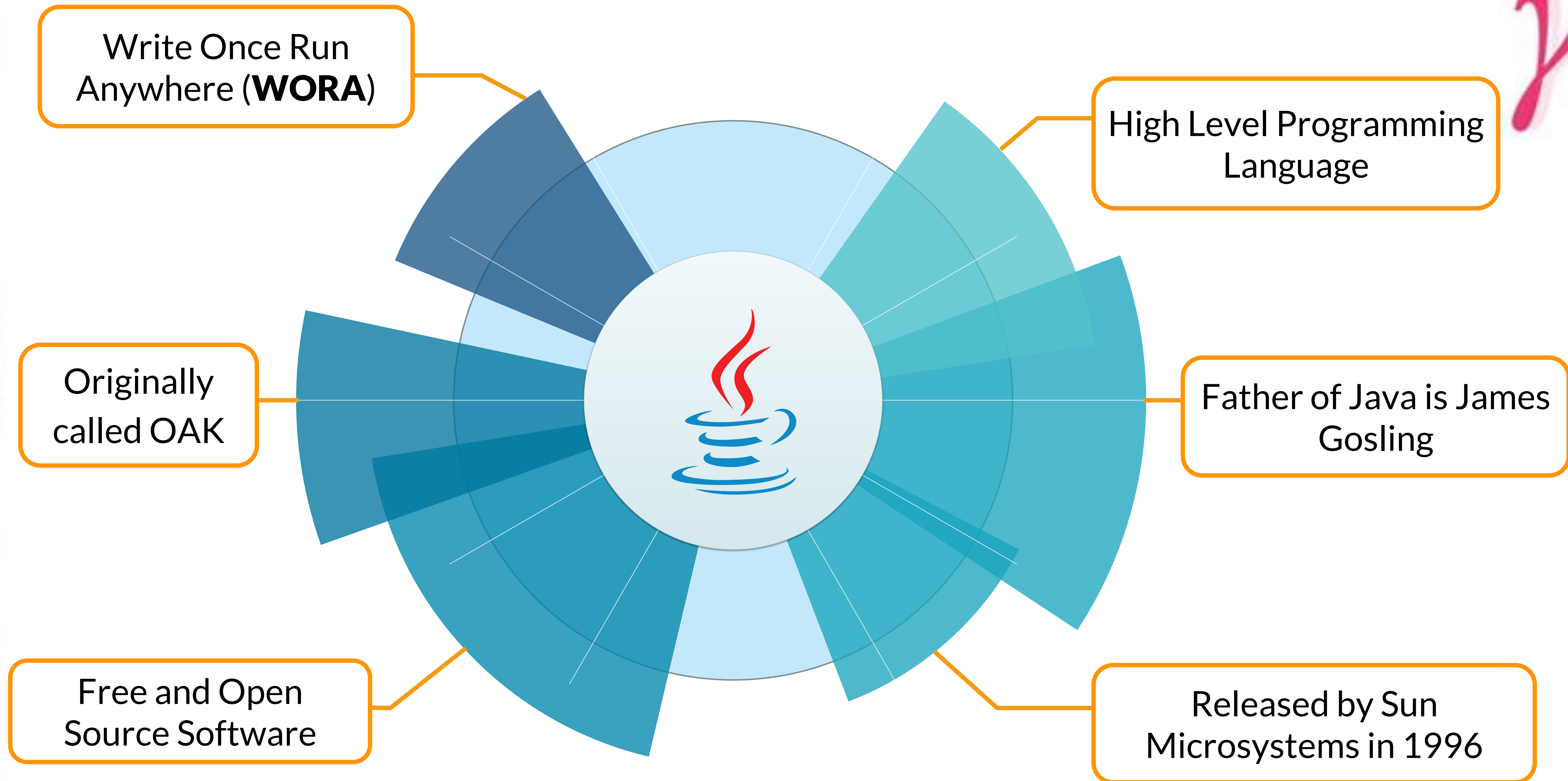
# **OBJECT ORIENTED ANALYSIS & DESIGN**

# **DATA STRUCTURES & ALGORITHMS**

Overview of Programming in Java



# Introduction to Java





# Features of Java



Easy

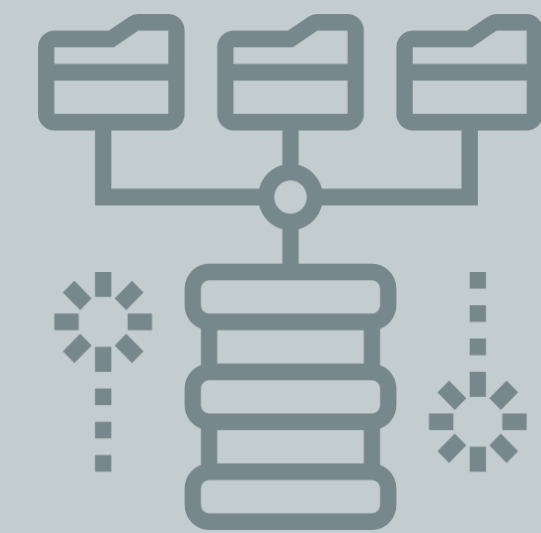


Robust



Portable

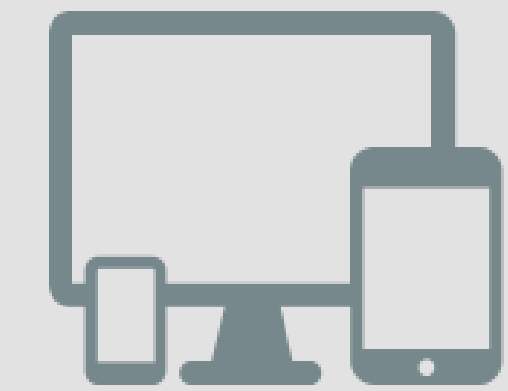
Distributed



Object  
Oriented

{OOP}

Platform  
Independent



Secure



Interpreted



Multithreaded



# JVM vs JRE vs JDK

## Java Virtual Machine

- ✓ JVM is an abstract machine that doesn't exist physically
- ✓ Provides runtime environment to drive the Java Code or applications
- ✓ It compiles the Java code into bytecode
- ✓ It is platform dependent

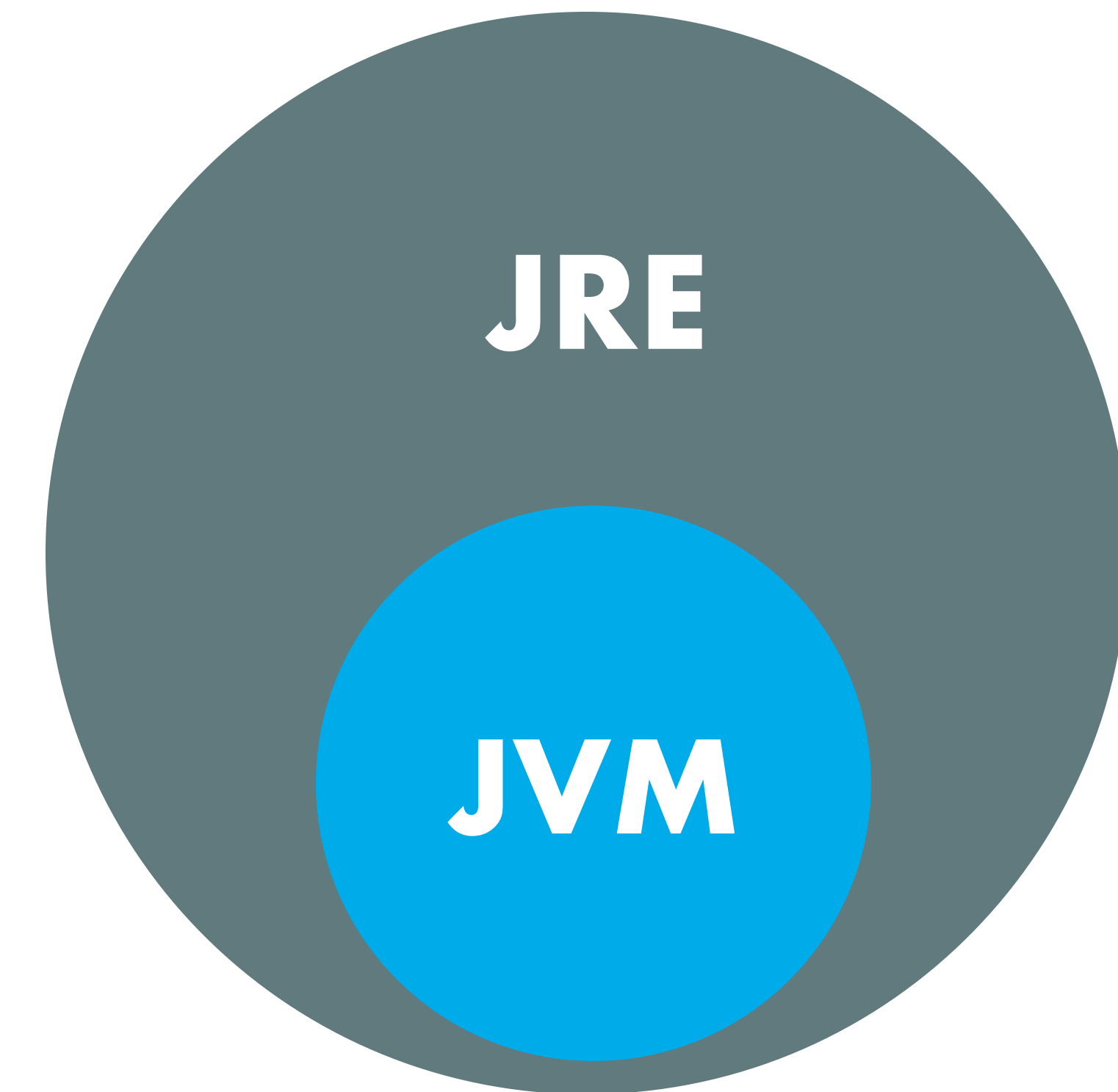
**JVM**



# JVM vs JRE vs JDK

## Java Runtime Environment

- ✓ JRE is the environment within which the JVM runs
- ✓ It contains a set of libraries + other files that JVM uses at runtime
- ✓ Also called Java RTE

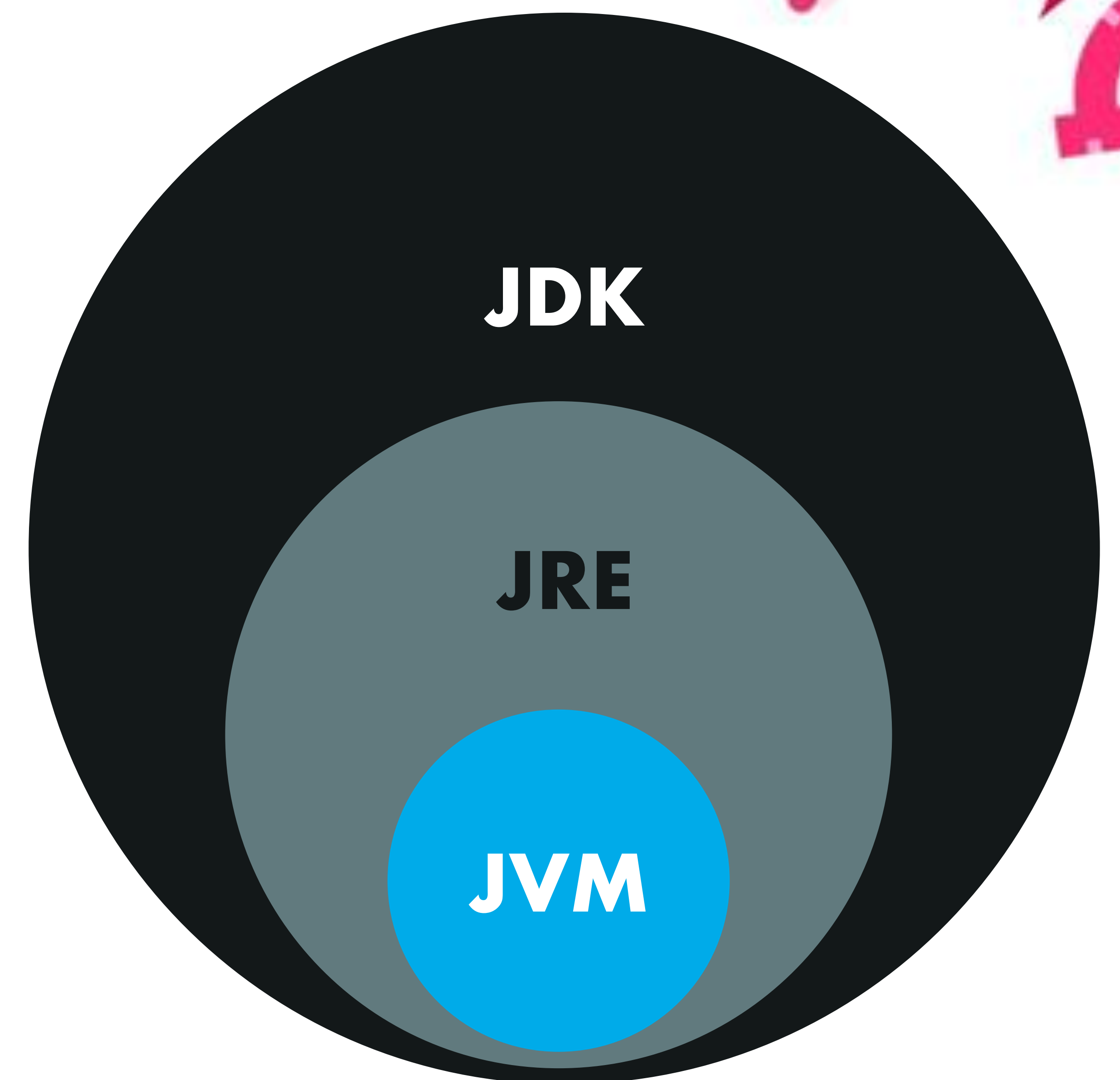




# JVM vs JRE vs JDK

## Java Development Kit

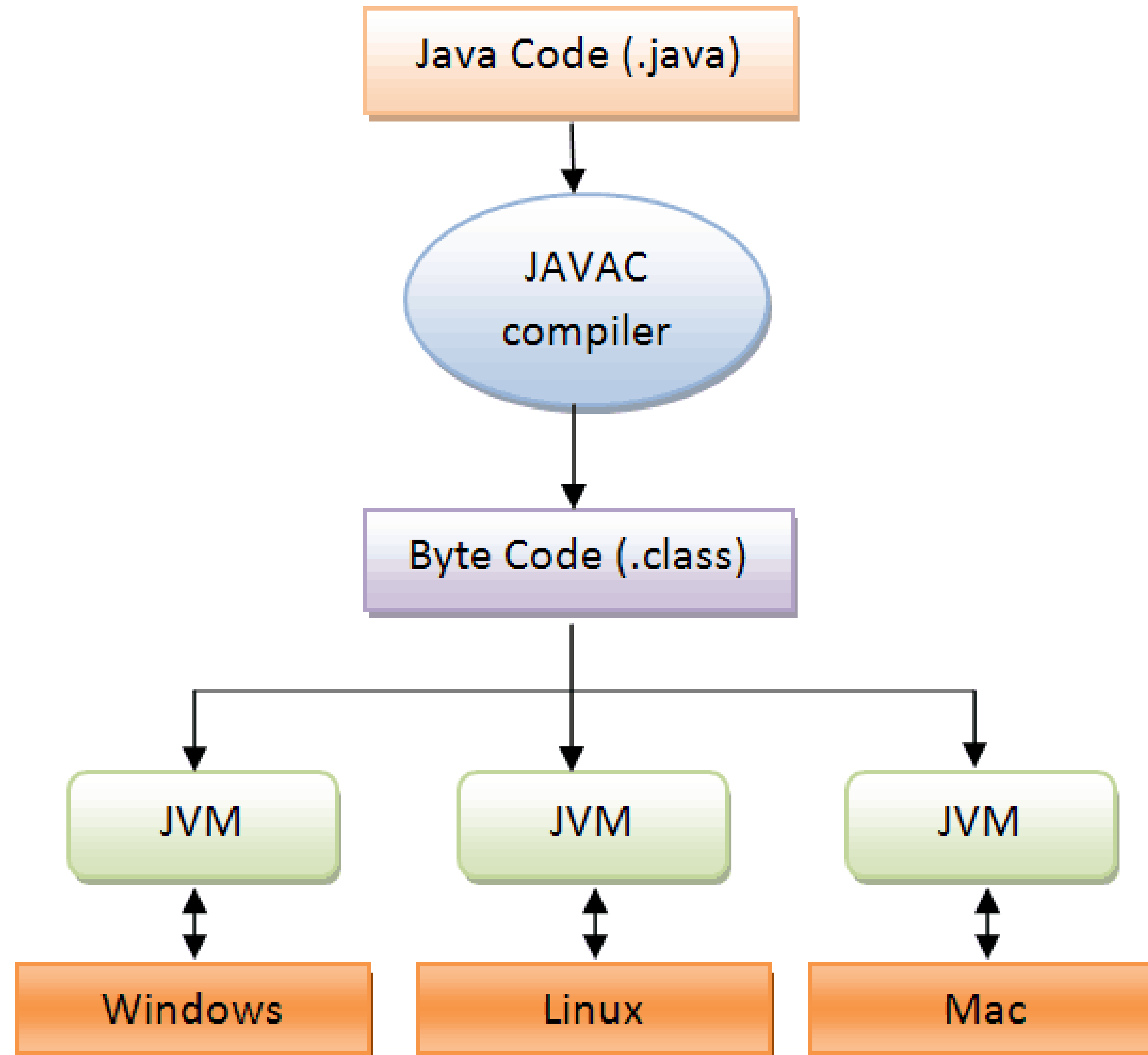
- ✓ JDK is a software development environment which is used to develop Java applications and applets
- ✓ It contains Development Tools and JRE





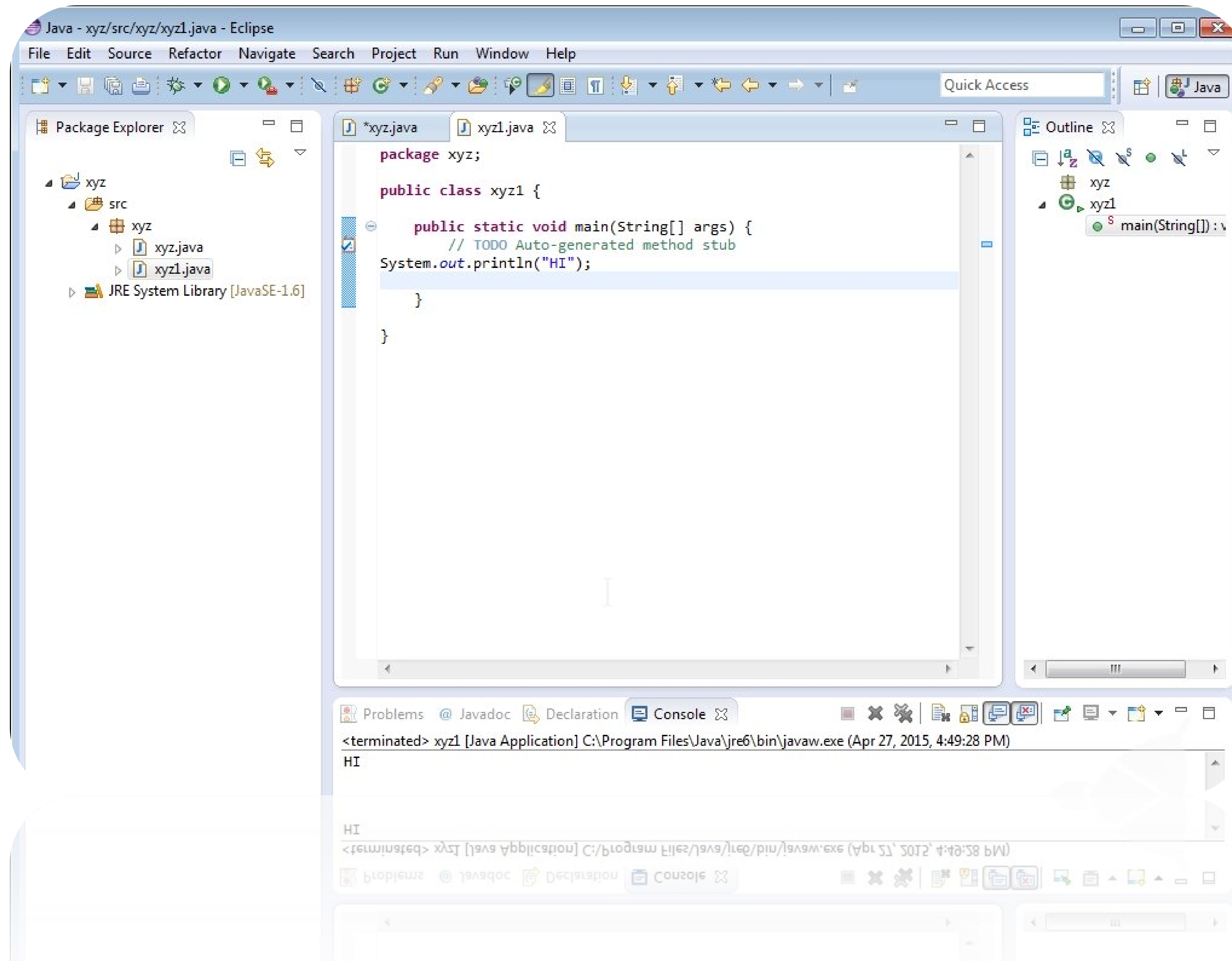
# Java Compilation and Execution







# Eclipse IDE





# My First Java Program

```
class Demo{  
    public static void main(String args[])  
        System.out.println("Hello World!!!");  
    }  
}
```



# My First Java Program

```
public static void main(String args[])
```



# My First Java Program

```
public static void main(String args[])
```

**public**

It is the access modifier of the main method

**static**

It is a keyword which identifies the class related thing

**void**

It is used to define the Return Type of the Method

**main**

It is the name of the method that is searched by JVM as a starting point for an application with a particular signature only

**String args[]**

It is the parameter to the main Method where the argument name could be anything

**NOTE:** *main() in Java is the most important method as it is the entry point of any java program*



# Java Fundamentals



# Java Fundamentals - Variables

Variable refers to the name of reserved memory area

1

Local  
Variables

2

Instance  
Variables

3

Static  
Variables



# Java Fundamentals - Constants

A constant is the one whose value cannot be modified

1

literals

2

final  
variables



# Java Fundamentals - Comments

Comments can be used to explain Java code, and to make it more readable.

- **Single Line**

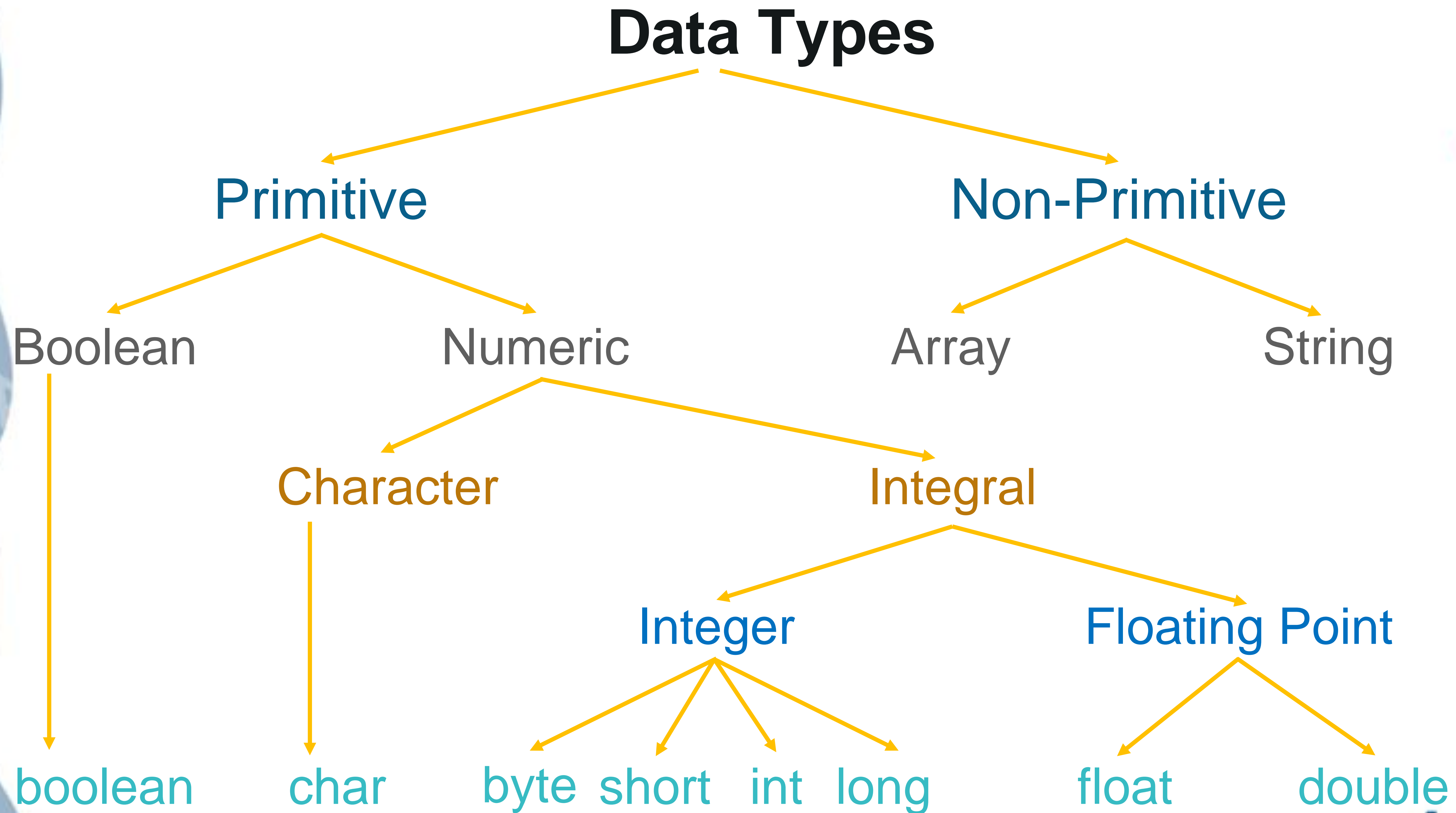
Single-line comments start with two forward slashes (//).

- **Multi Line**

Multi-line comments start with `/*` and ends with `*/`.

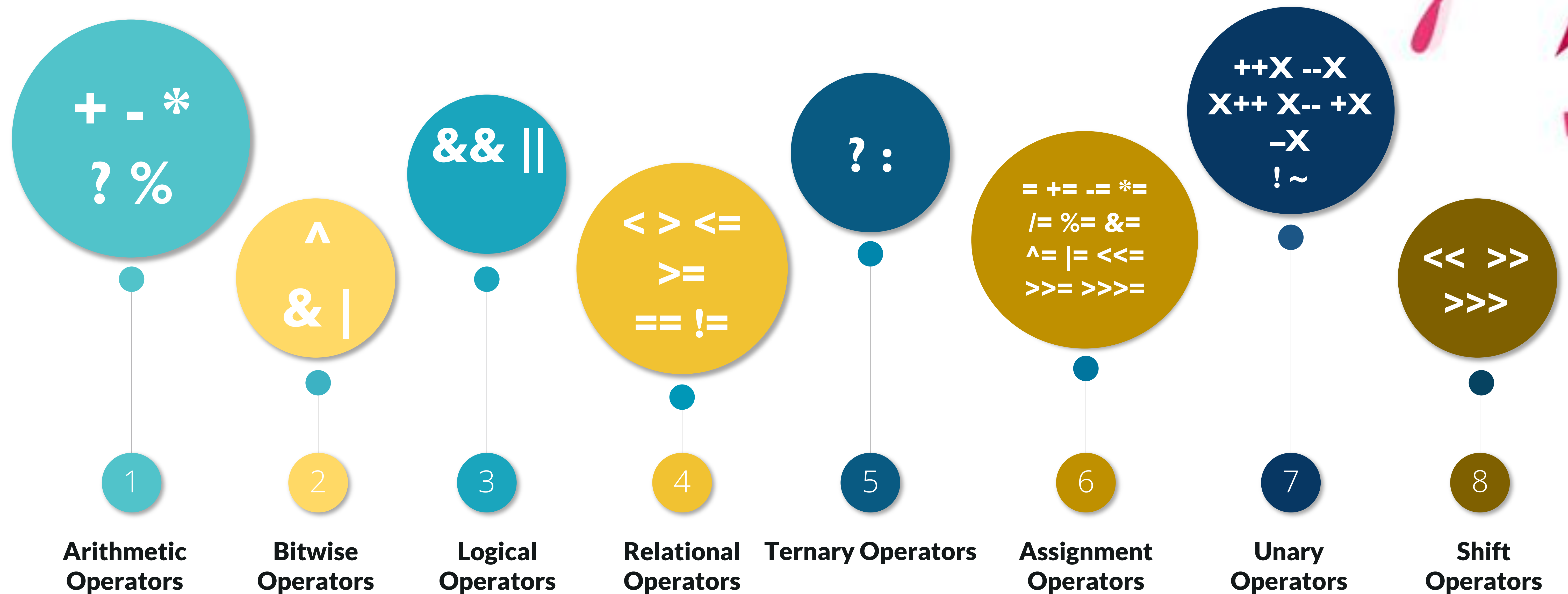


# Java Fundamentals – Data Types





# Java Fundamentals - Operators





# Conditional Statements



# Conditional Statements - If

It is the simplest selection statement in the Java language that checks the condition and executes the loop if condition is true

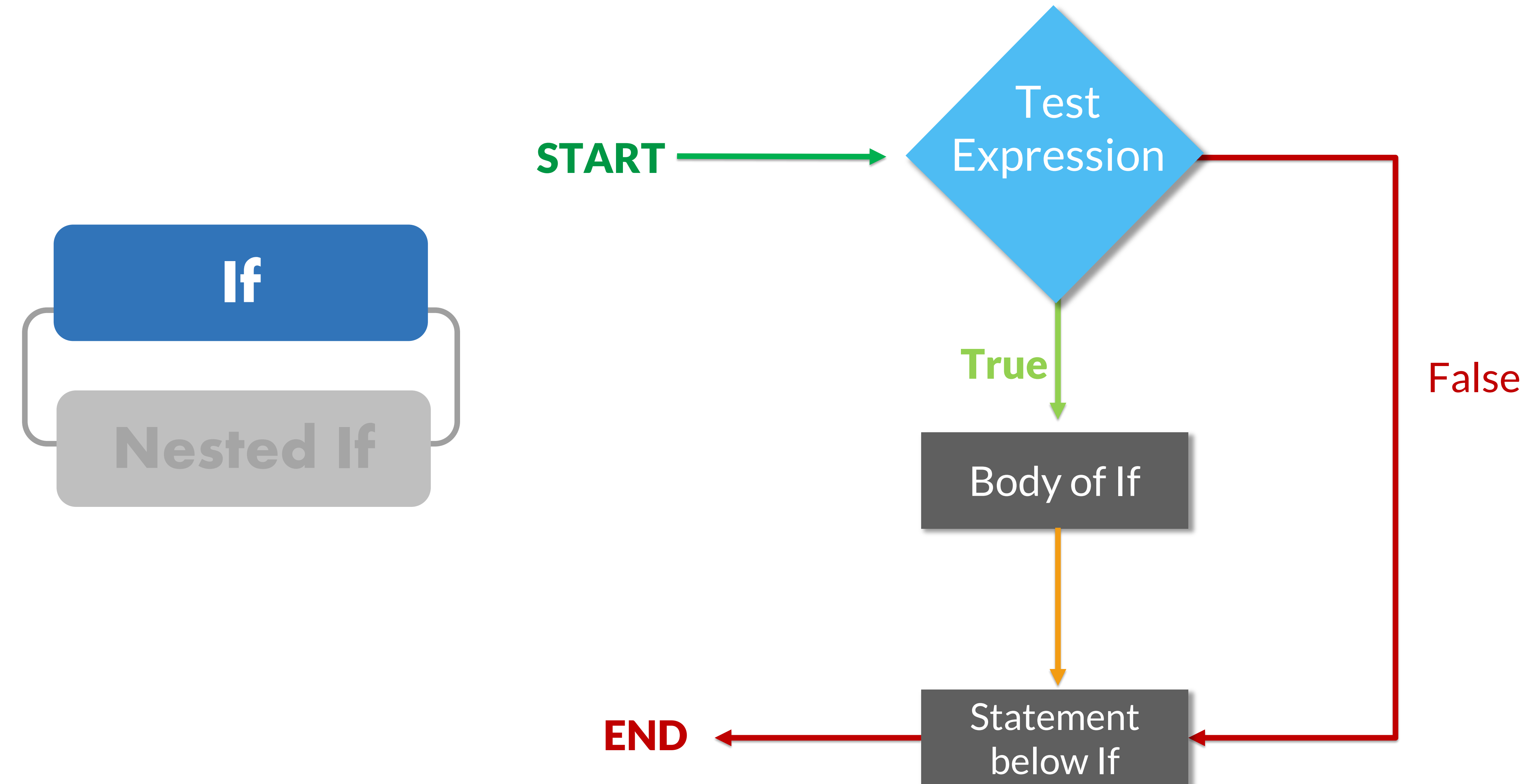
**If**

**Nested  
If**



# Conditional Statements - If

If is the most simple decision making statement that decides whether a certain statement or block of statements will be executed or not





# Conditional Statements - If



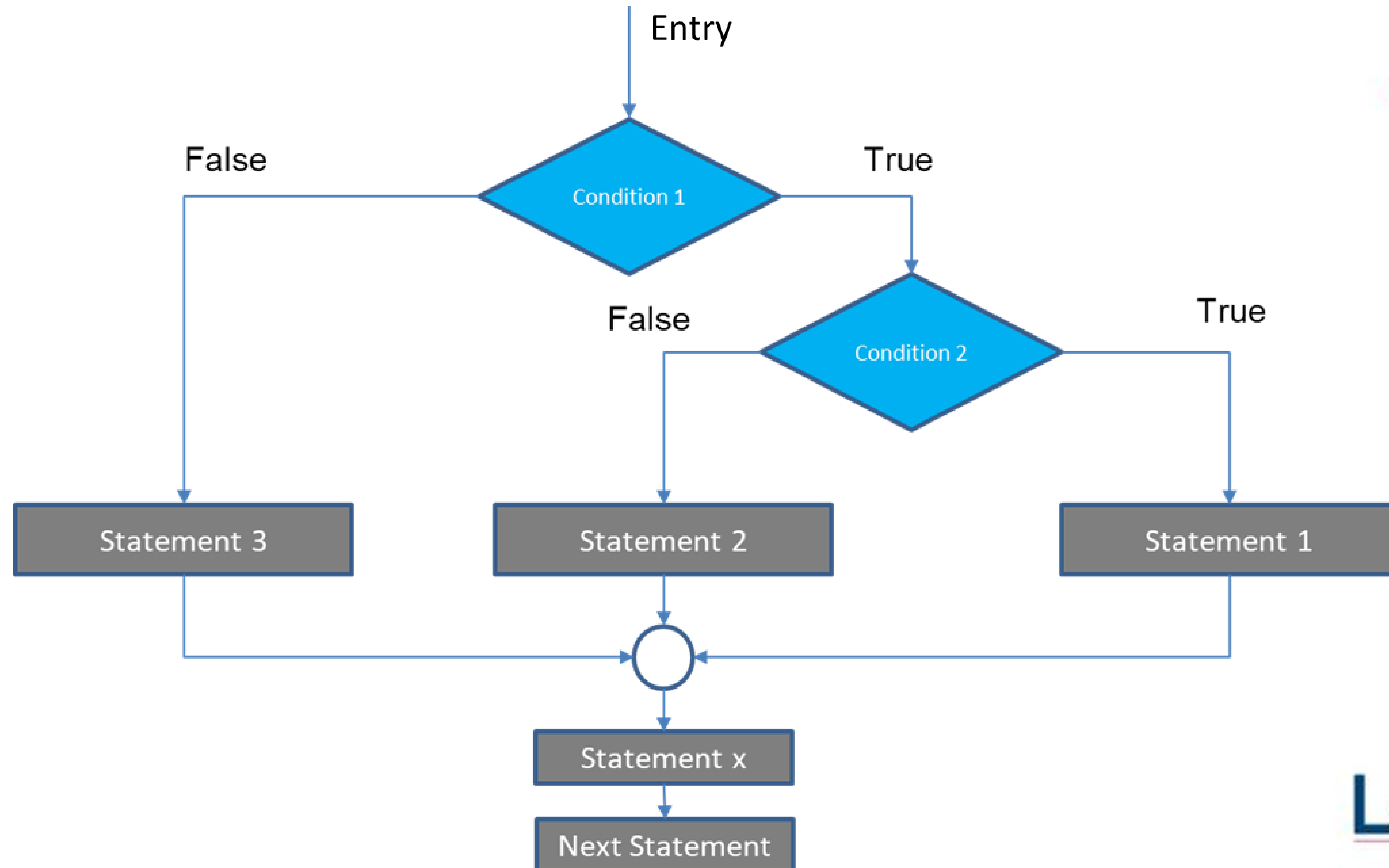
## Syntax

```
if(condition)
{
    //Statements to execute
    if condition is true
}
```



# Conditional Statements - If

It is an if statement or an if-else statement within an if statement





# Conditional Statements - If



## Syntax

```
if (condition1)
{
    // Executes when condition1 is true
    if (condition2)
    {
        // Executes when condition2 is true
    }
}
```



# Conditional Statements - If Else

It is an upgraded if statement that tests the condition and if the condition is false then 'else' statement is executed



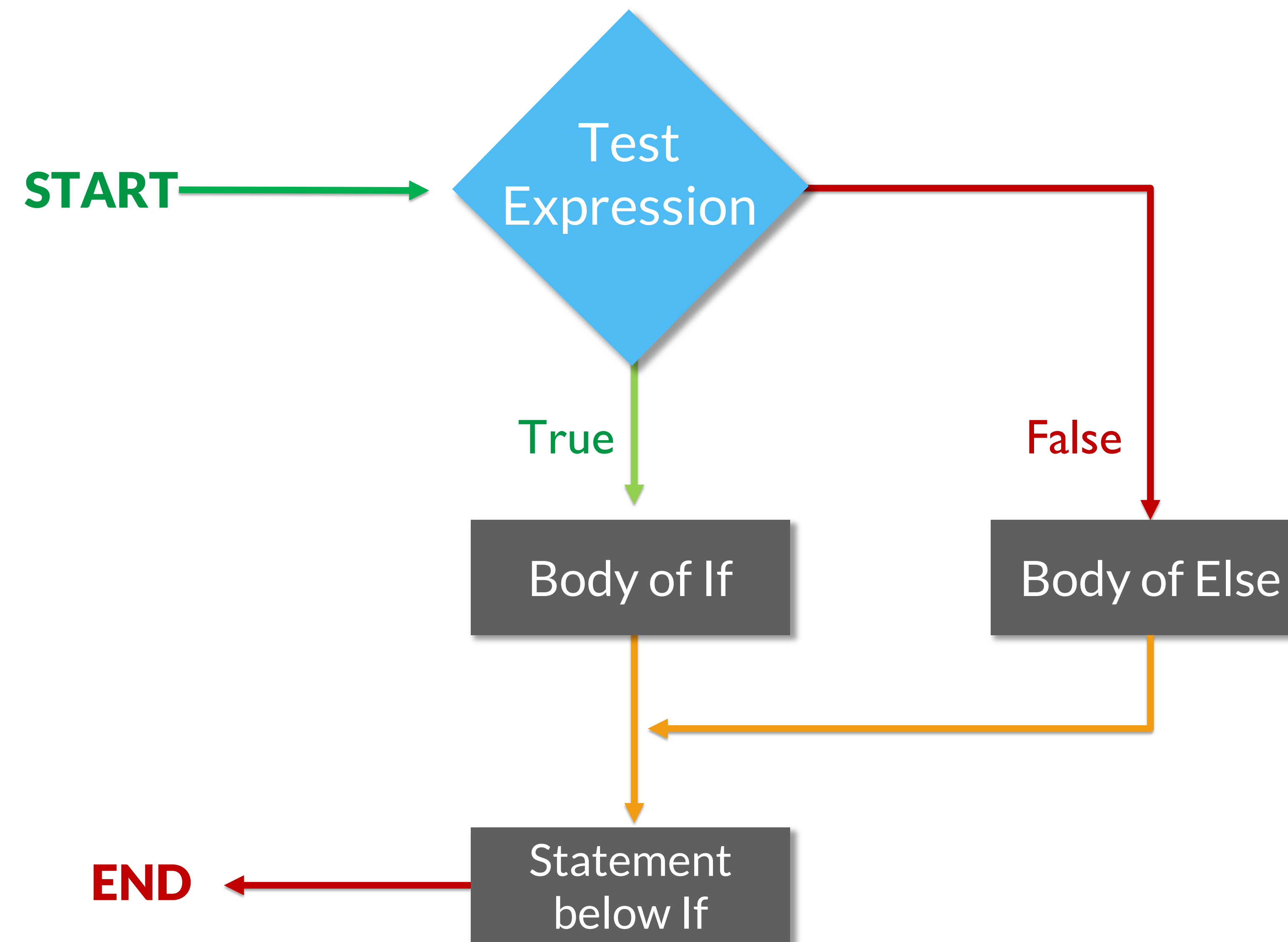
**If-Else**

**If-Else-If Ladder**



# Conditional Statements - If Else

It tests the condition and if the condition is false then 'else' statement is executed





# Conditional Statements - If Else

## Syntax

```
if (condition)
{
    //Executes this block if condition is
true
}
else
{
    //Executes this block if condition
is false
}
```

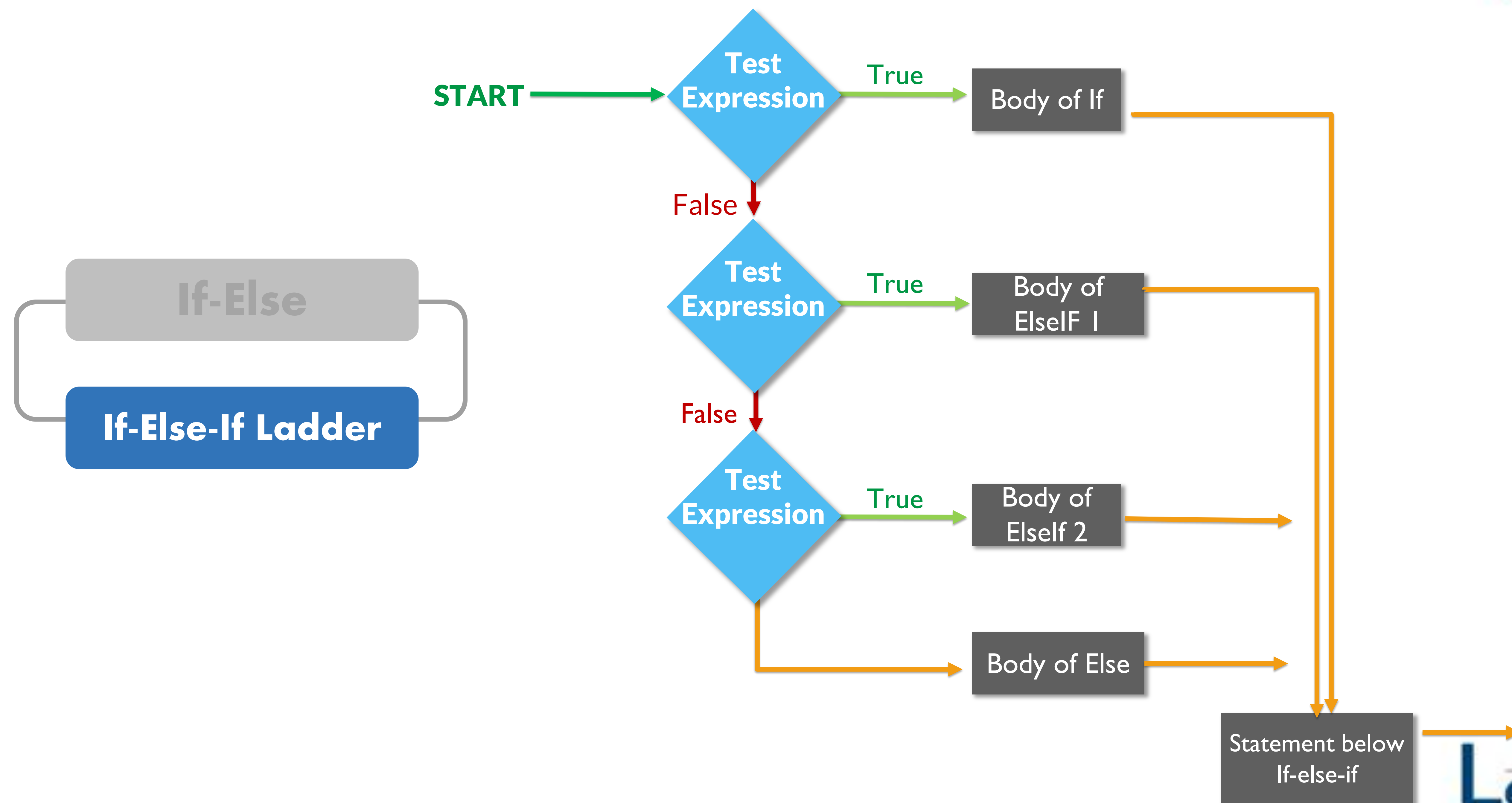
If-Else

If-Else-If Ladder



# Conditional Statements - If Else

If-else-if ladder allows the user to use many if else statement within a loop and in case one of the condition holds true the rest of the loops is bypassed





# Conditional Statements - If Else

## Syntax

```
if (condition)
{
    statement;
}
else if (condition)
{
    statement;
}
.
.
else
    statement;
```

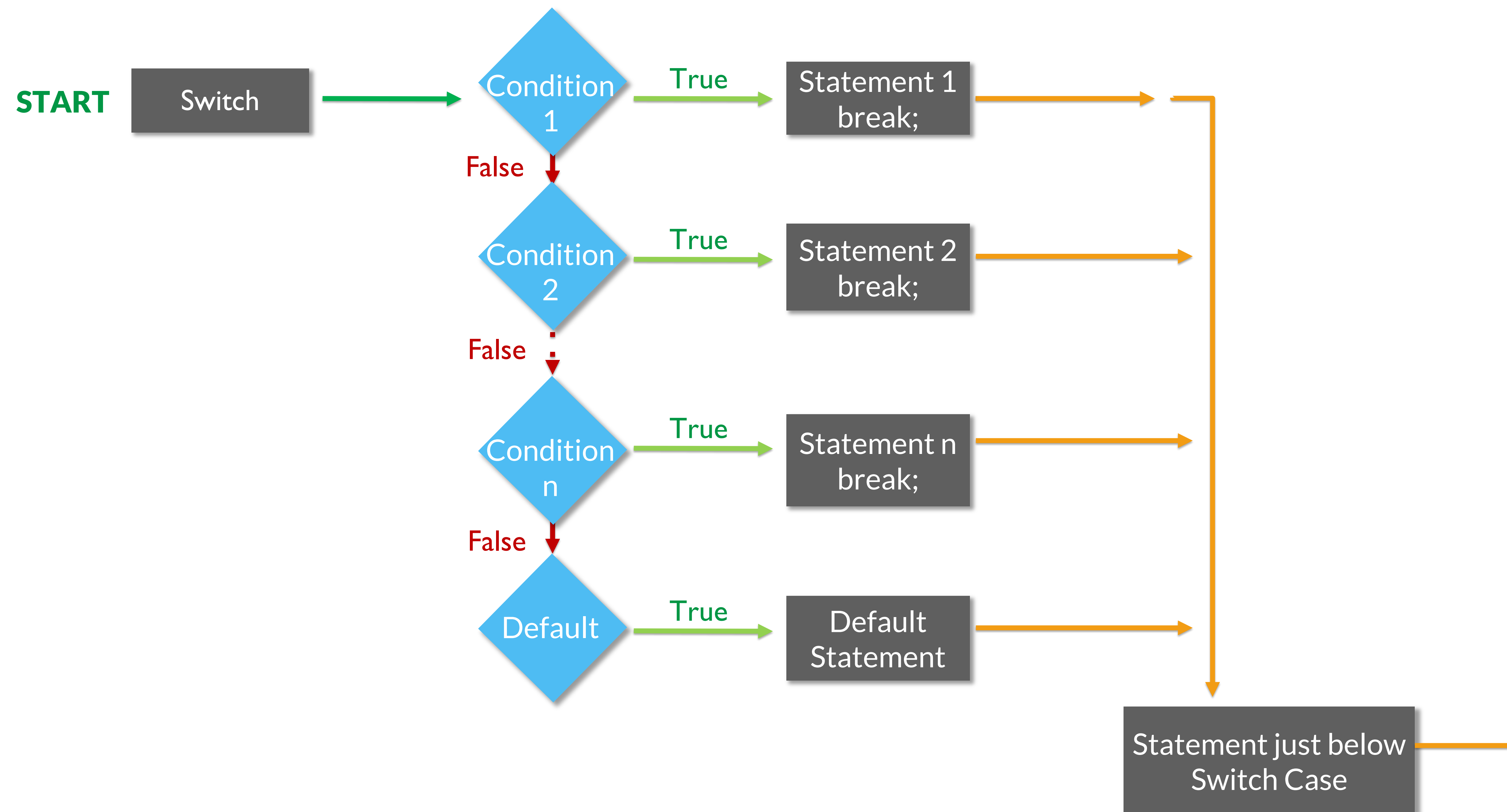
If-Else

If-Else-If Ladder



# Conditional Statements - Switch

The switch statement provides an easy way to execute conditions to different parts of the code





# Conditional Statements - Switch

## Syntax

```
switch (expression)
{
    case value1:
        statement1;
        break;

    case value2:
        statement2;
        break;
    .
    .
    case valueN:
        statementN;
        break;

    default:
        statementDefault;
}
```

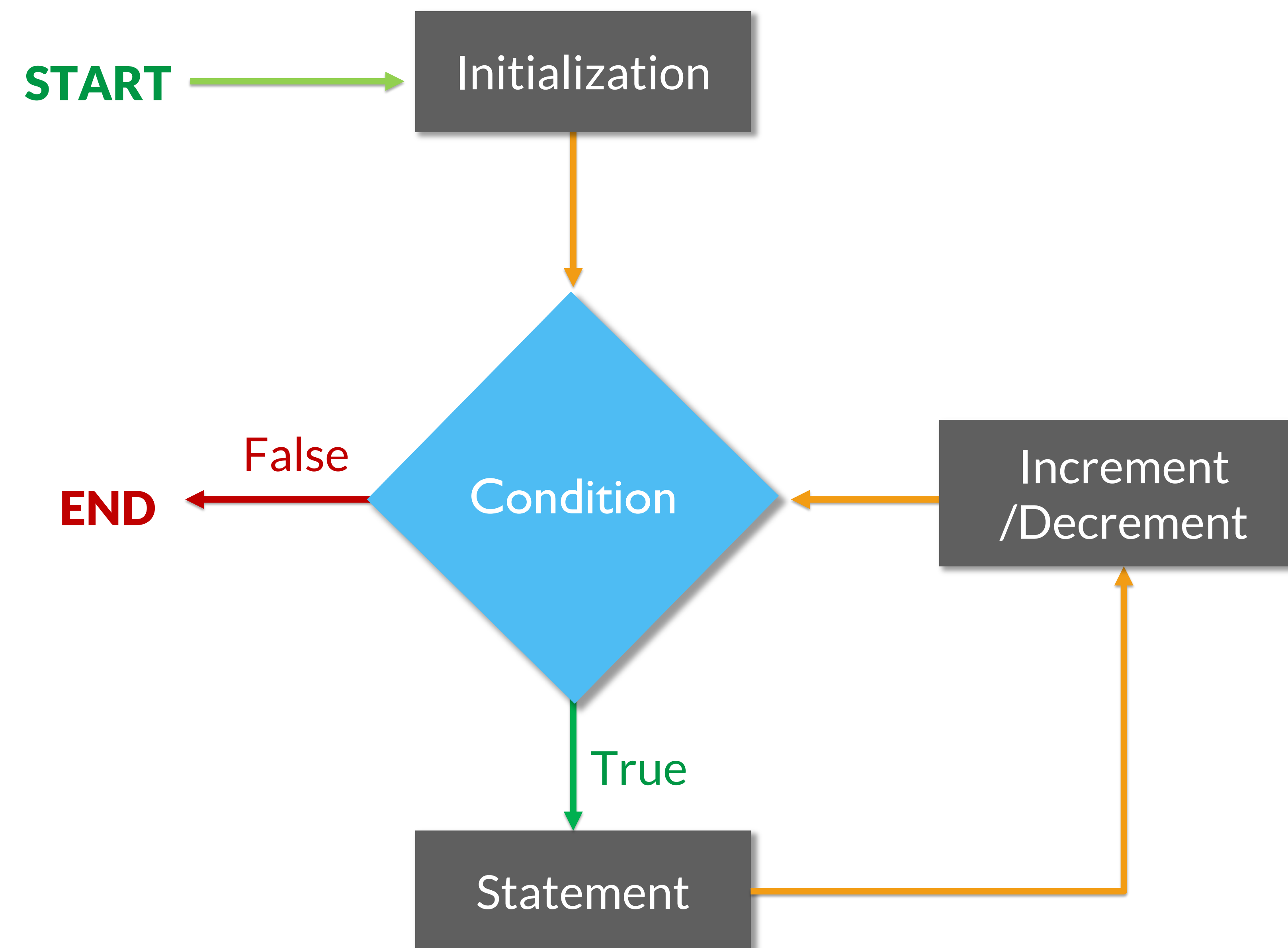


# Iterative Statements



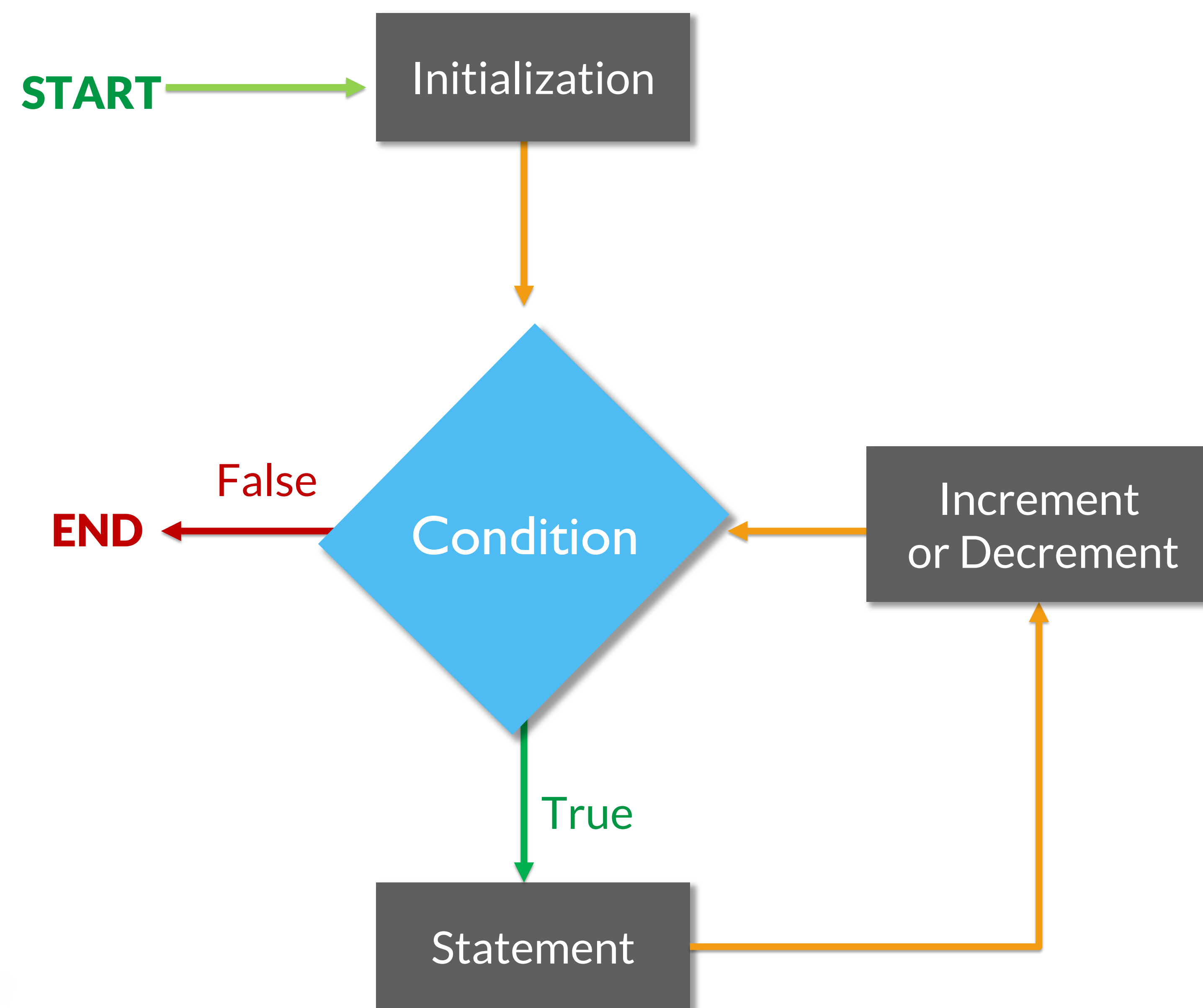
# Iterative Statements – For Loop

It is a control structure that allows us to repeat certain operations by incrementing and evaluating a loop counter





# Iterative Statements – For Loop



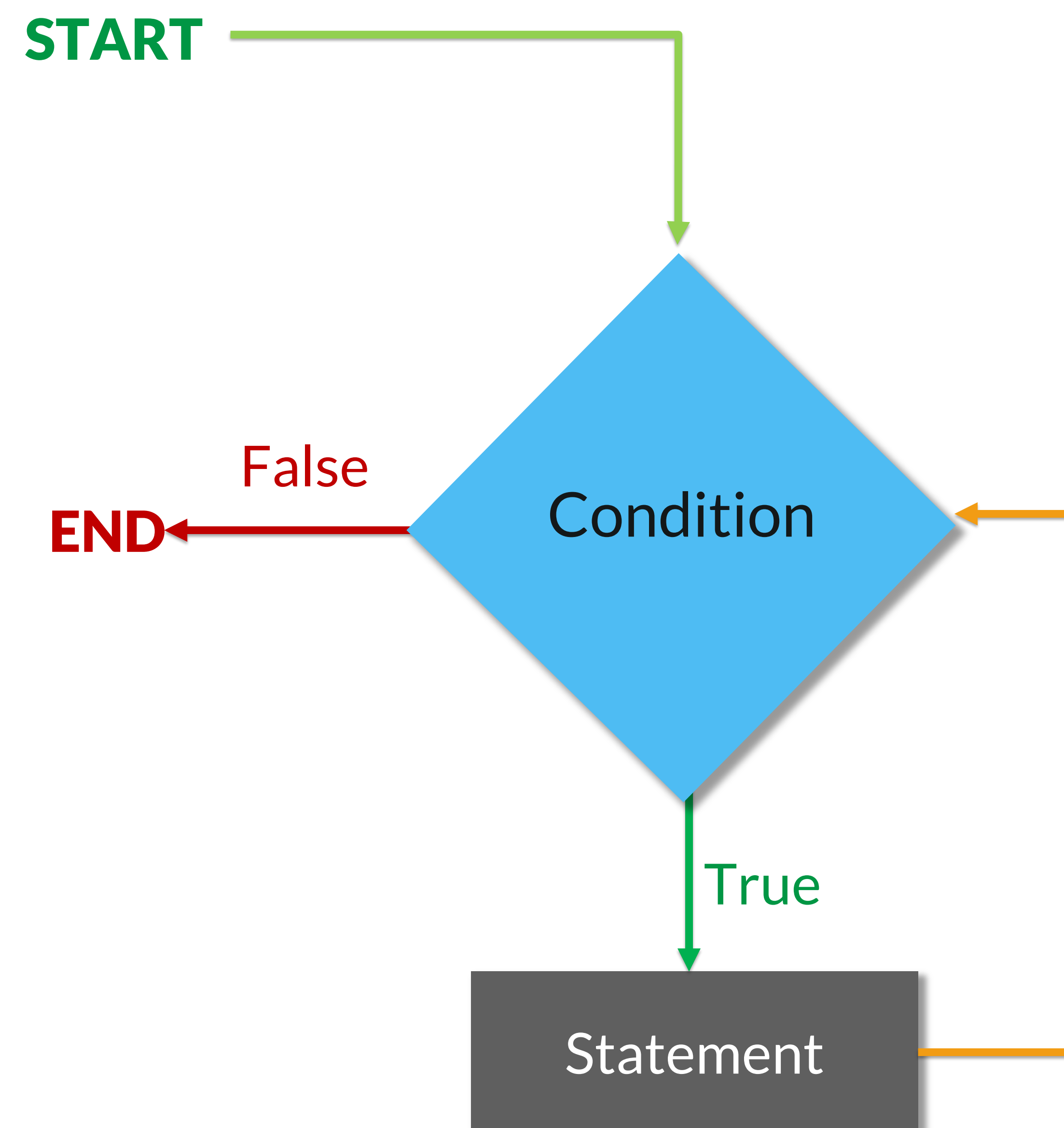
## Syntax

```
for(init; condition; incr/dcr)
{
    code block;
}
```



# Iterative Statements – While Loop

It is a control structure that allows us to specify that a certain statement is to be executed repetitively until the loop condition is false

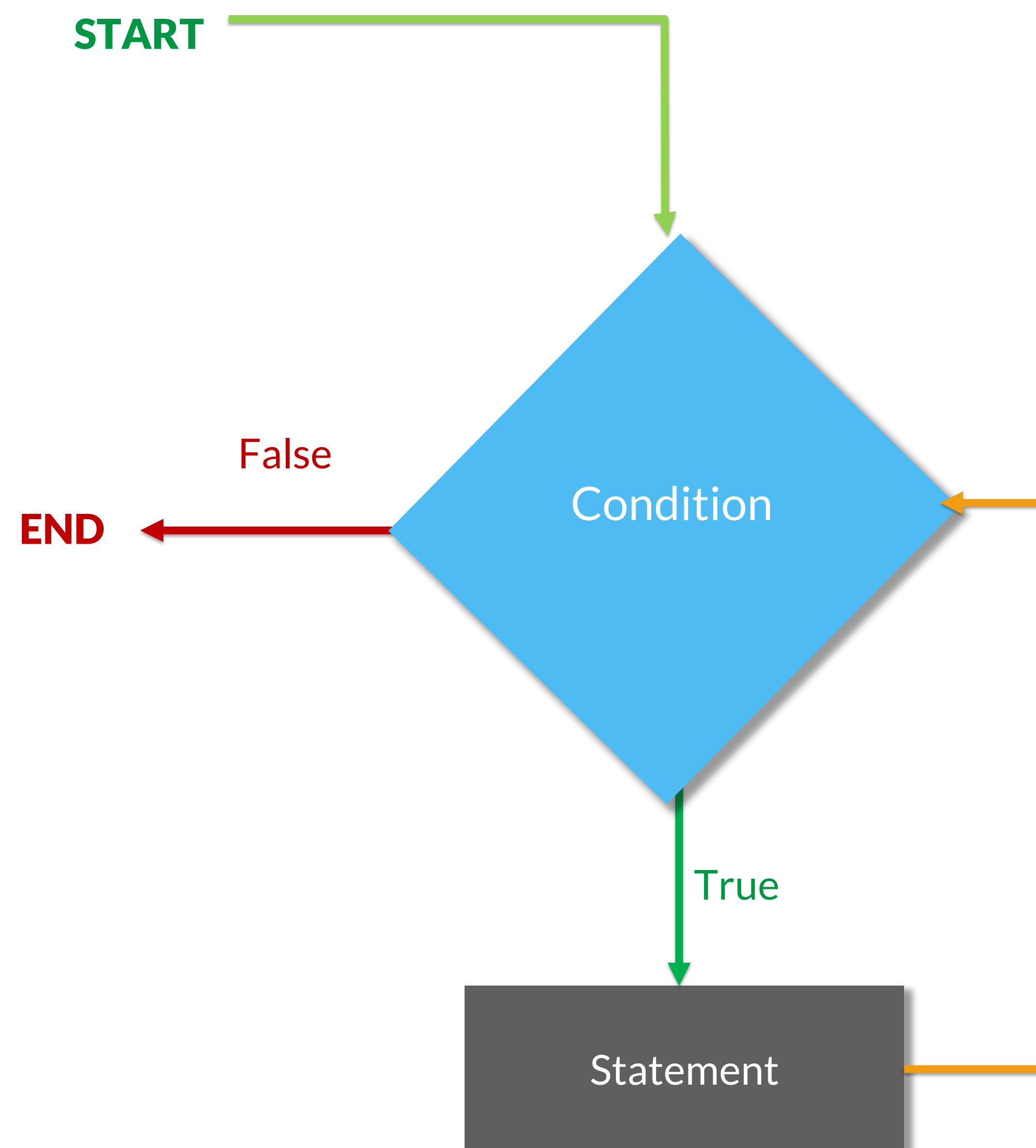




# Iterative Statements – While Loop

## Syntax

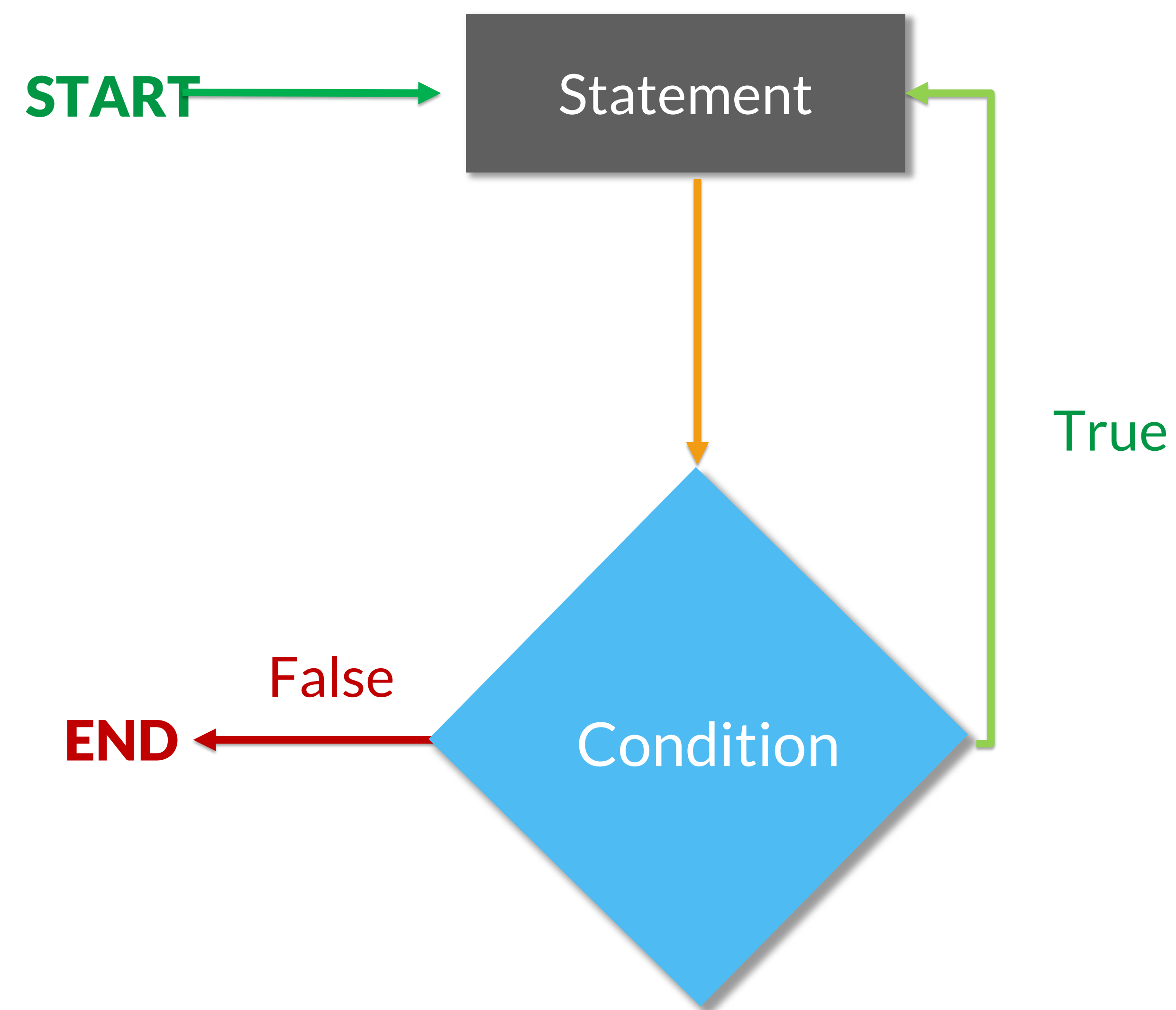
```
while (boolean condition)
{
    loop statements...
}
```





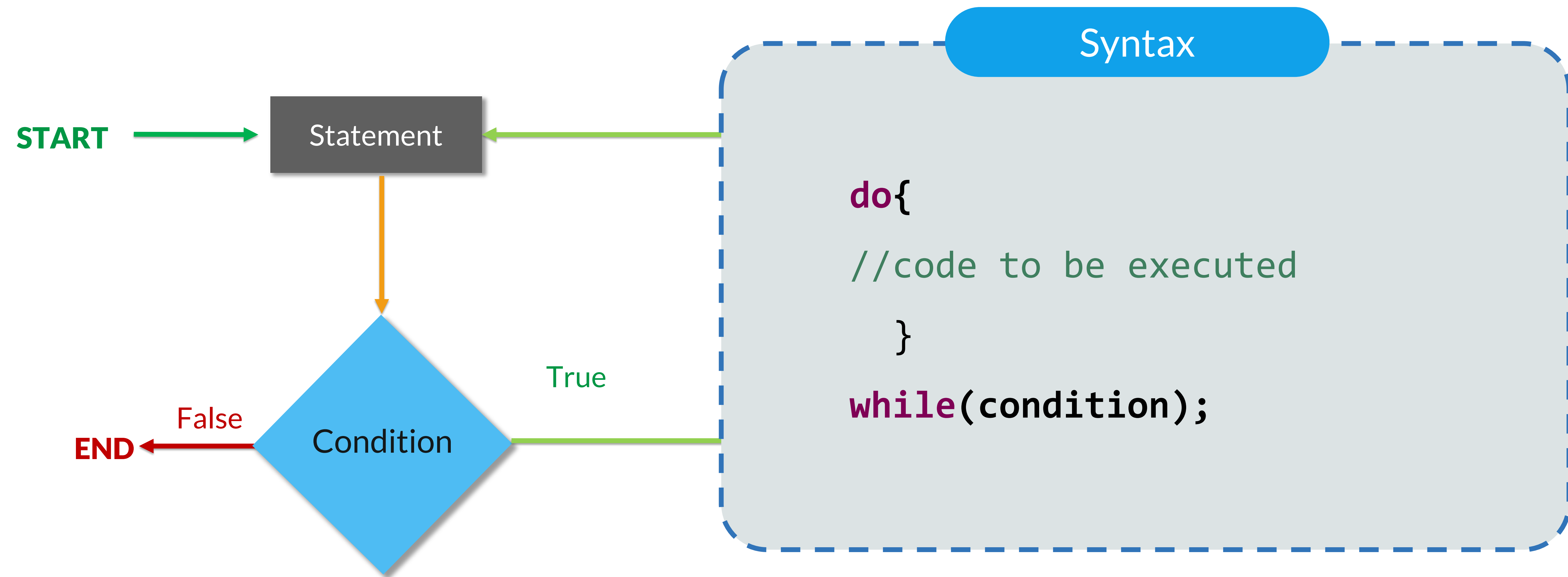
# Iterative Statements – Do While Loop

It is a control structure that allows code to be executed repeatedly based on a given Boolean condition and tests the condition before executing the loop body





# Iterative Statements – Do While Loop





# Java 8 Features



# Java 8 Features

Lambda  
Expressions

Pipelines and  
Streams

Date and Time  
API

Default Methods

Type  
Annotations

Nashorn  
JavaScript  
Engine

Concurrent  
Accumulators

Parallel  
operations

PermGen Space  
Removed

TLS SNI



**Thank You!**