Python&Math Initiative Project (PyMath Project)

Functions

WP2 - Algorithms and Mathematics

Session 5: Built-in & User-Defined Functions

Prof. Dr. Turgay Tugay BİLGİN
Bursa Technical University
24.11.2025

Contents

CONTENTS

01 Intro & Agenda

Session Agenda

Concept

What is a Function in Programming vs. Math?

Part 1: Built-in Functions

Mathematical,
Trigonometry & Logarithms,
Random Number Generation.

Part 2: User-Defined Functions

Why create our own?
Using the Function Manager,
Parameters and Return Values.

02

What is a Function?

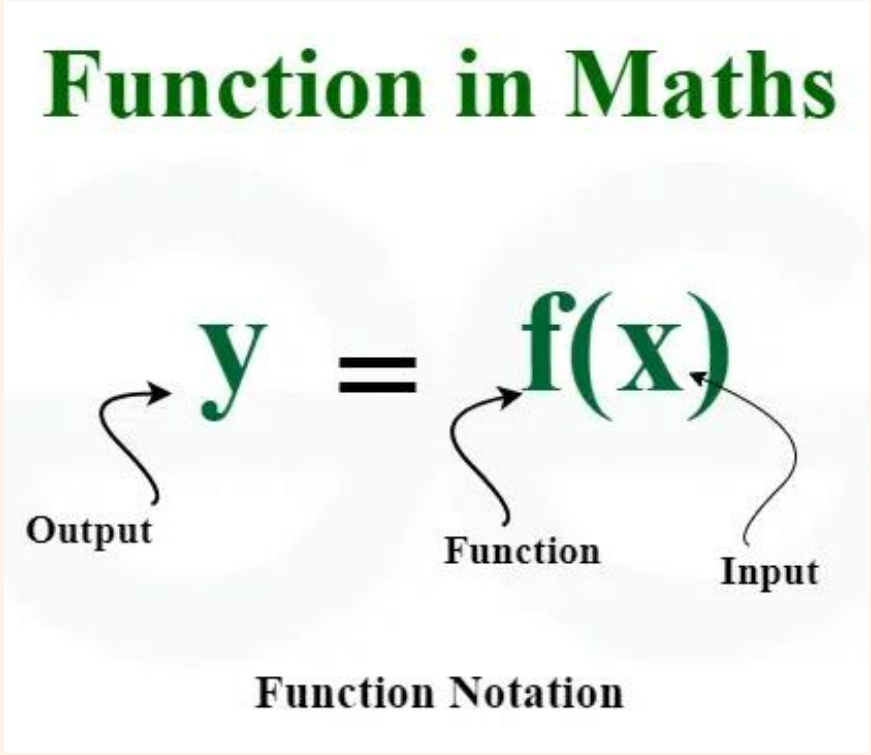
What is a Function?

In mathematics, a function relates an input to an output.
Example: f(x) = 2x + 5 .

In programming, the definition is very similar.

> **Definition:**
> A function is a set of instructions that accomplishes a well-defined task.

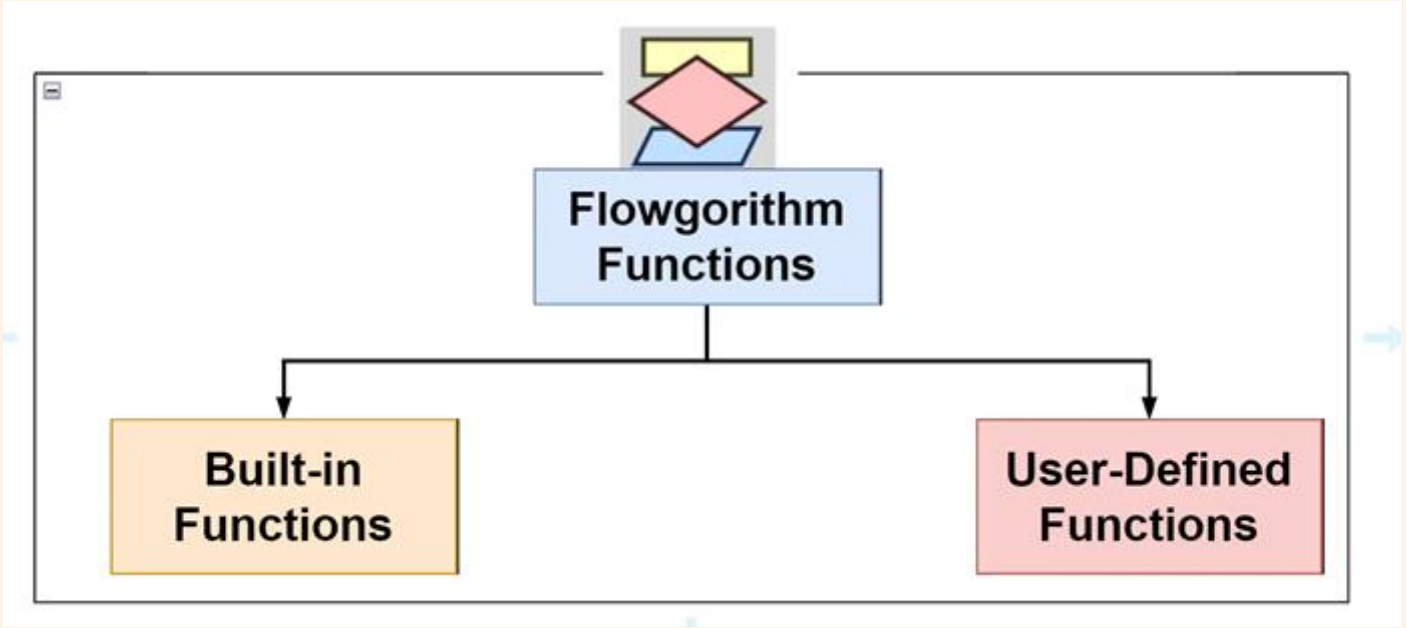
Functions are essential because they allow us to reuse code and simplify logic.

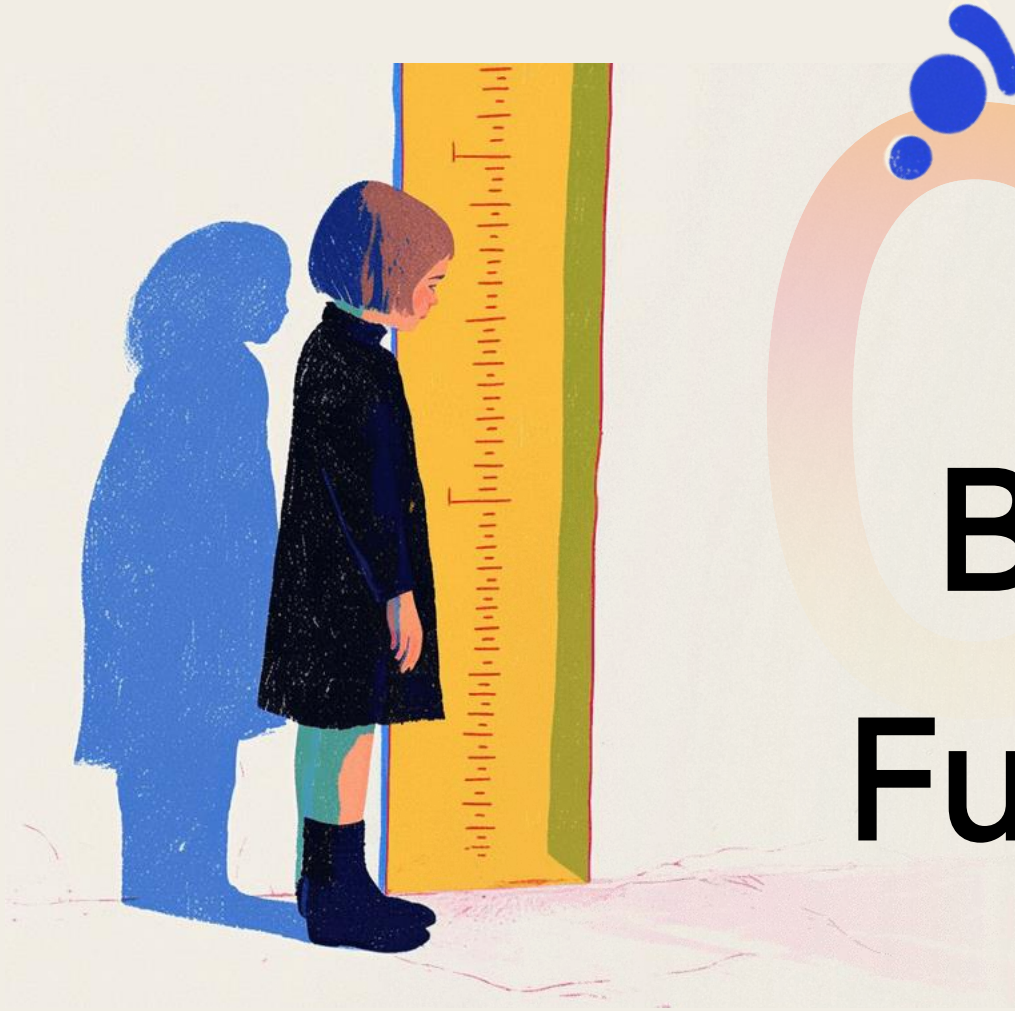


Types of Functions in Flowgorithm

Flowgorithm functions can be broadly divided into two groups:

1. **Built-in (Intrinsic) Functions:** These are ready-made tools provided by the software. We do not write the code for them; we just use them.
2. **User-Defined Functions:** These are custom functions that YOU create to solve specific problems.





03

Built-in Functions

Built-in Functions

Built-in functions are predefined, ready-to-use tools. They save time and reduce errors because the logic is already written and tested.

Today, we will focus on the Mathematical functions:

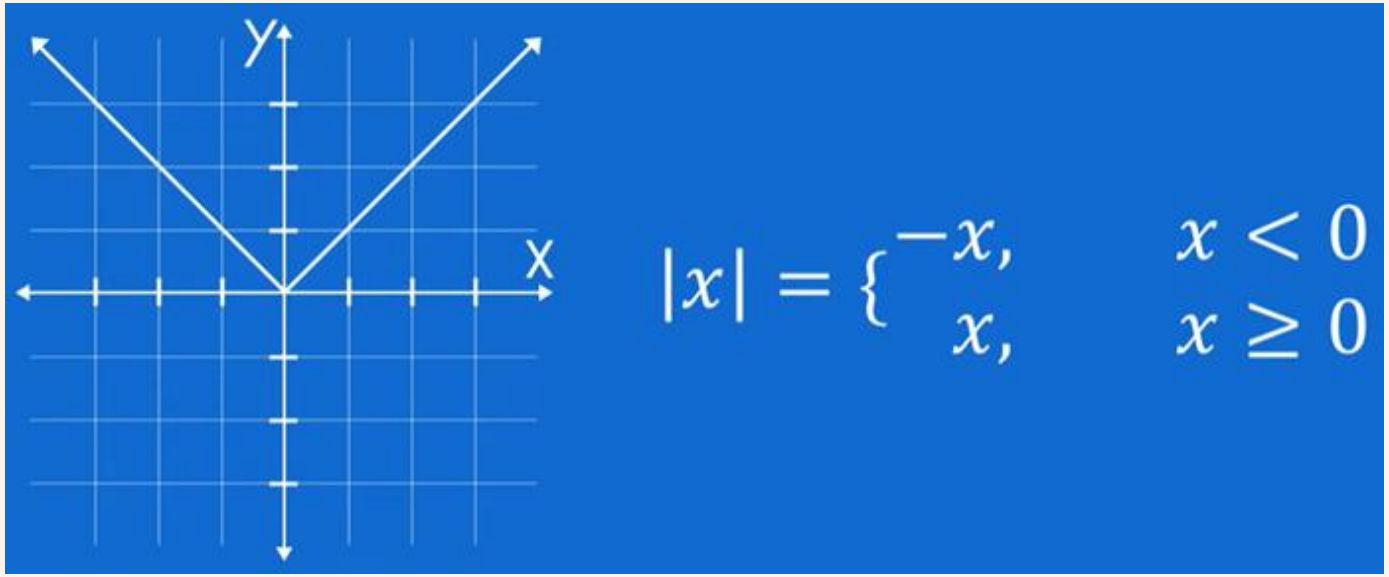
1. Basic Arithmetic
2. Roots & Logarithms
3. Trigonometry
4. Random Numbers

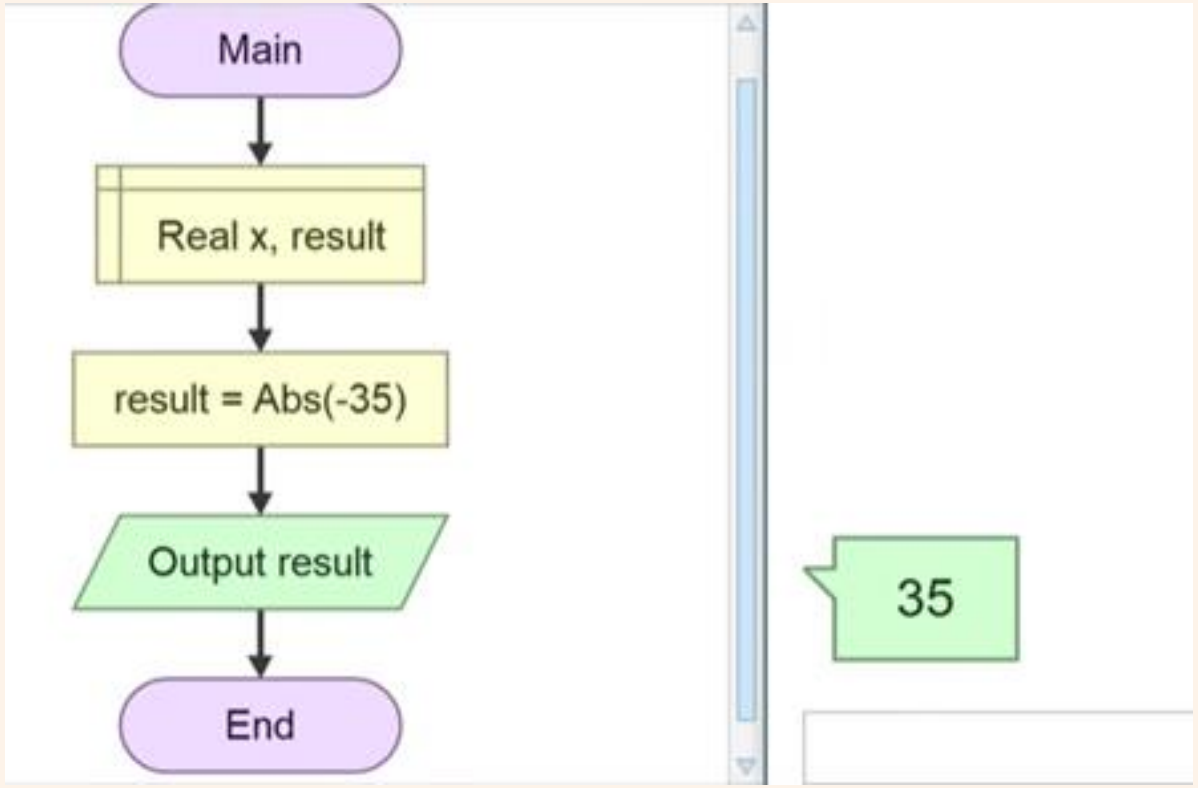
Basic Mathematical Functions

Abs(n)

Absolute Value. Returns the distance from zero.

Abs(-5) returns 5.





Flowchart Example: Abs(x)

Here is an example of the **Abs** function in action.

- We input a negative number: `-35` .
- The expression is `result = Abs(-35)` .
- The output is 35.

Roots and Logarithms

Sqrt(n)

Square Root. Input must be non-negative.

Sqrt(16) returns 4.

Log(n)

Natural Logarithm (base e).

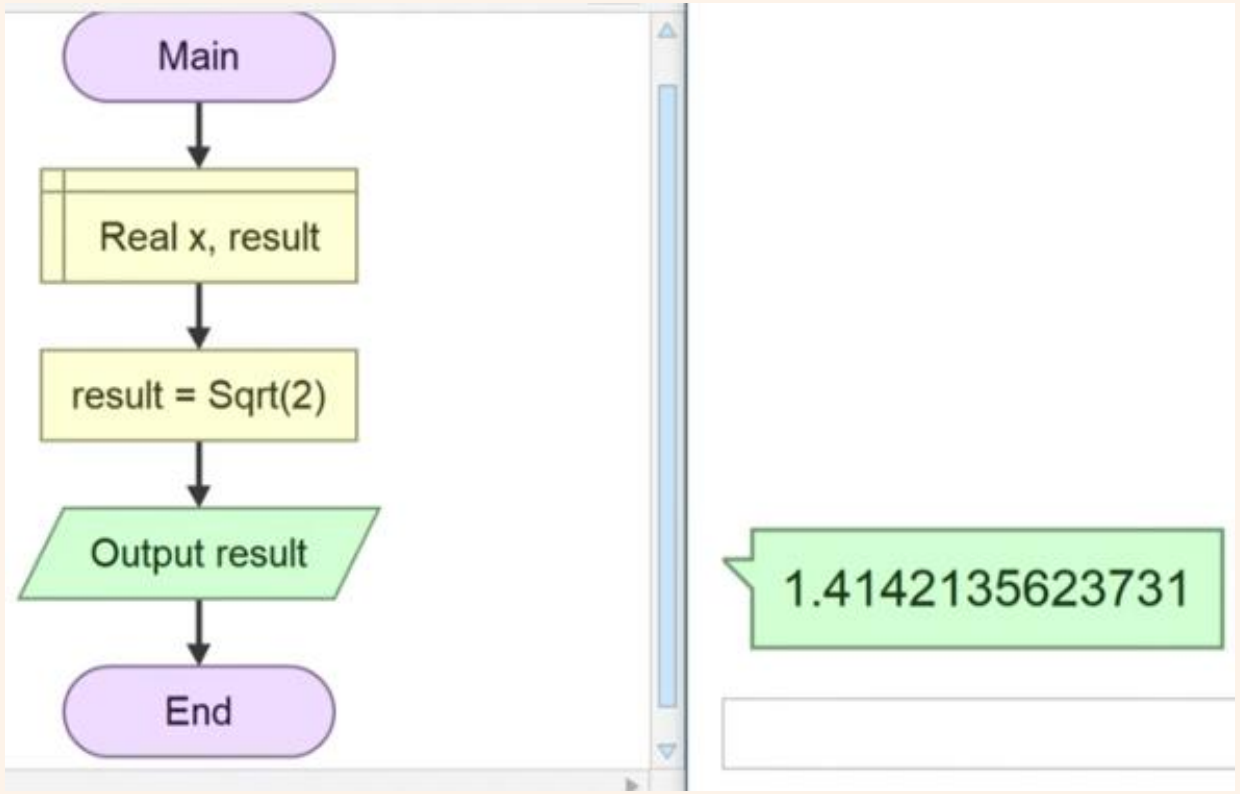
Log10(n)

Base-10 Logarithm.

Example: Using Sqrt

Let's look at a simple flowchart that calculates the square root of 2.

- **Input:** None (in this simple example).
- **Process:** `result = Sqrt(2)` .
- **Output:** `1.4142...`



Flowchart Example: Log(x)

Let's see a simple flowchart example for calculating a logarithm.

1. Declare

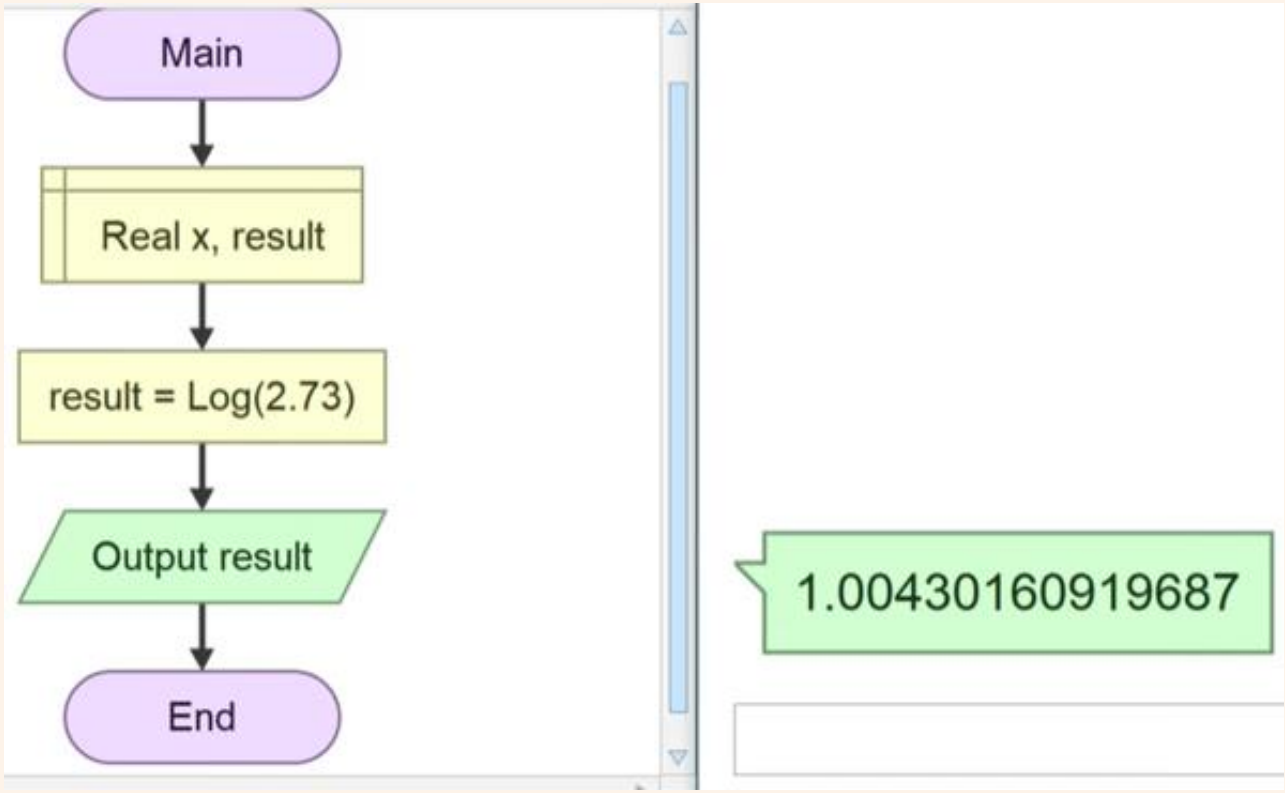
Real variable `result`

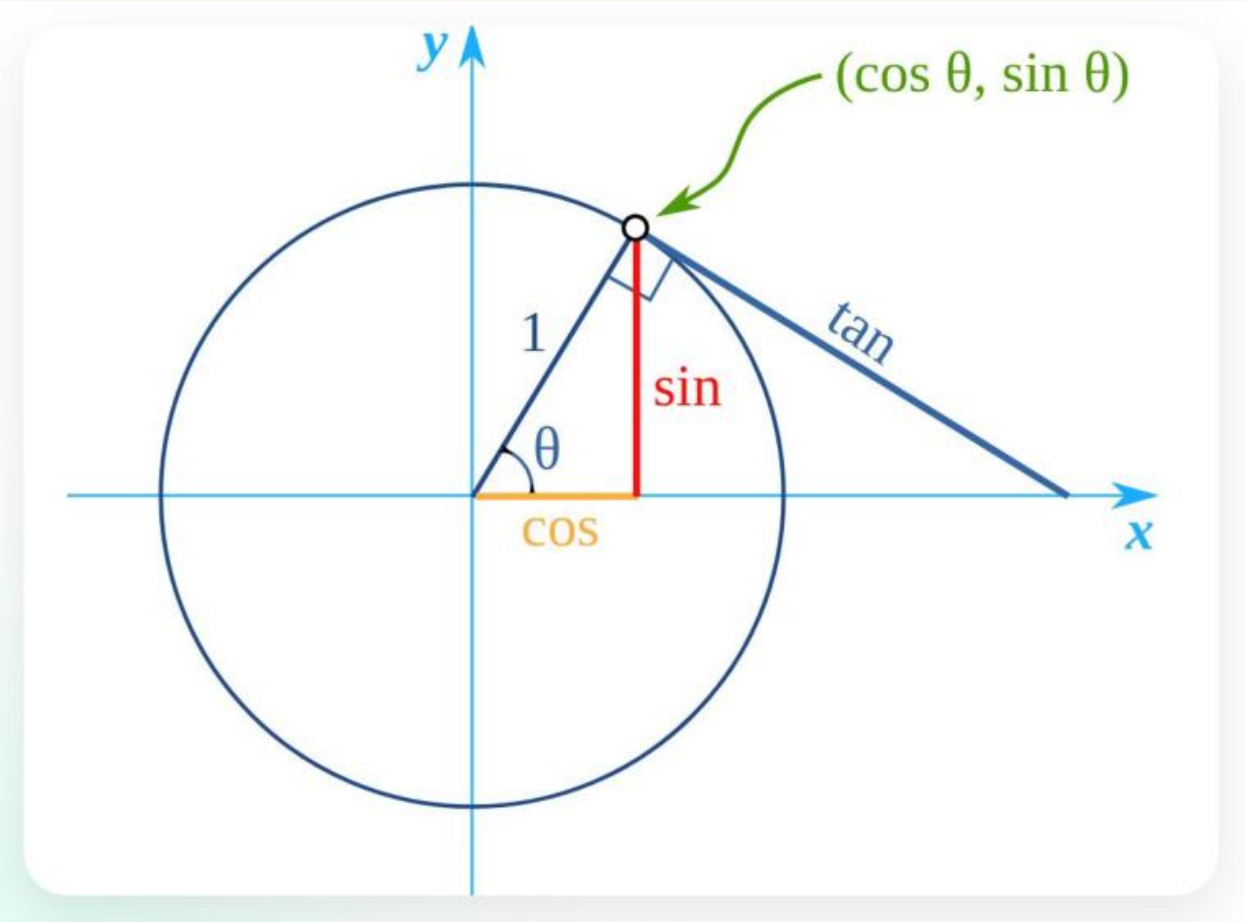
2. Calculate

`result = Log(2.73)`

3. Output

the result





Trigonometric Functions

Flowgorithm supports standard trigonometric operations:

Sin(n): Sine of angle n.

Cos(n): Cosine of angle n.

Tan(n): Tangent of angle n.

Inverse Functions: ArcSin(n), ArcCos(n), ArcTan(n).

CRITICAL NOTE

A Common Mistake

Radians vs. Degrees

In your math classes, you often use **Degrees** (0 to 360). However, Flowgorithm expects angles in RADIANS.

The Rule:

If a student enters 90 (degrees), you must convert it.

> ⚠ **Important Note**
>
> Flowgorithm expects angles in **RADIANS**, not degrees.

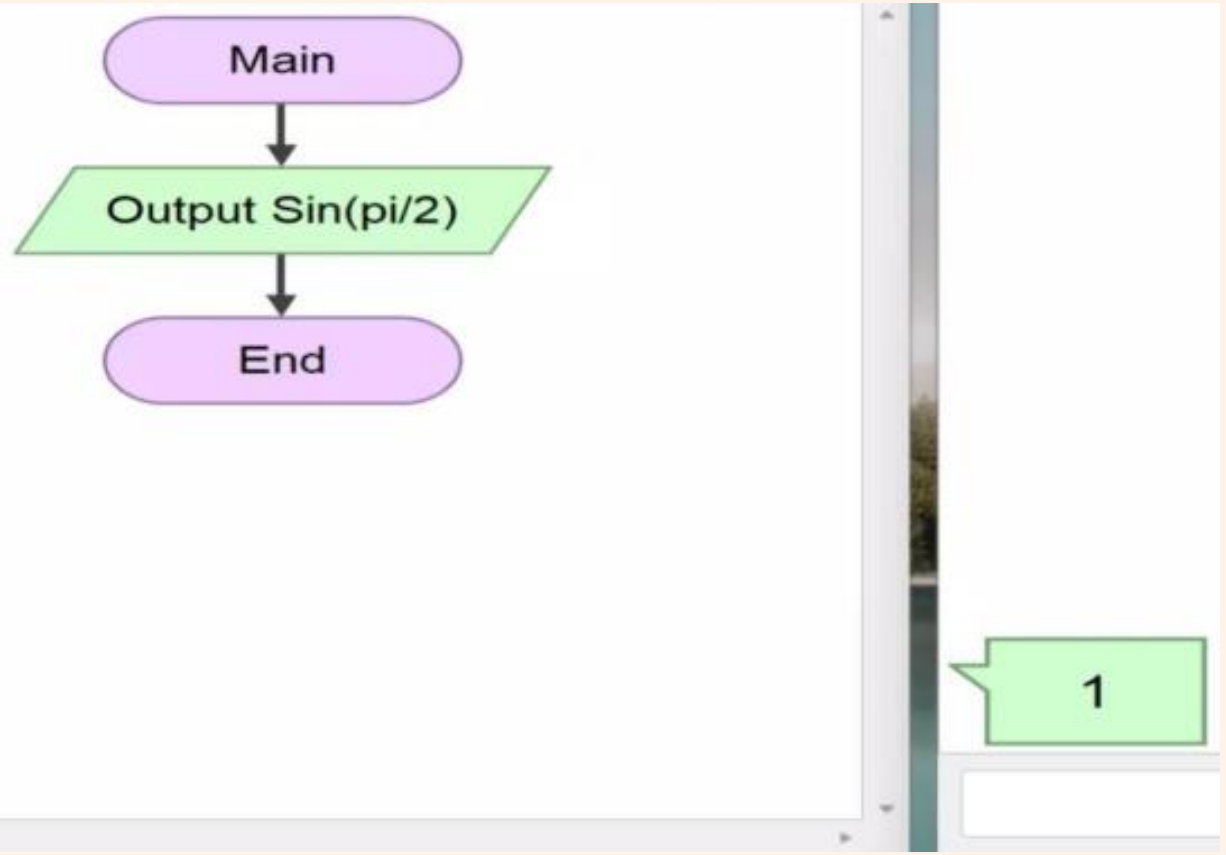
$$\text{radians} = \text{degrees} \times \frac{\pi}{180}$$

Example: Sine Function

Let's look at a simple flowchart using the Sine function.

Notice we calculate `Sin(pi/2)` .

Since `pi/2` radians equals **90 degrees**, the sine is 1.



Console Output: 1

Random Number Generation

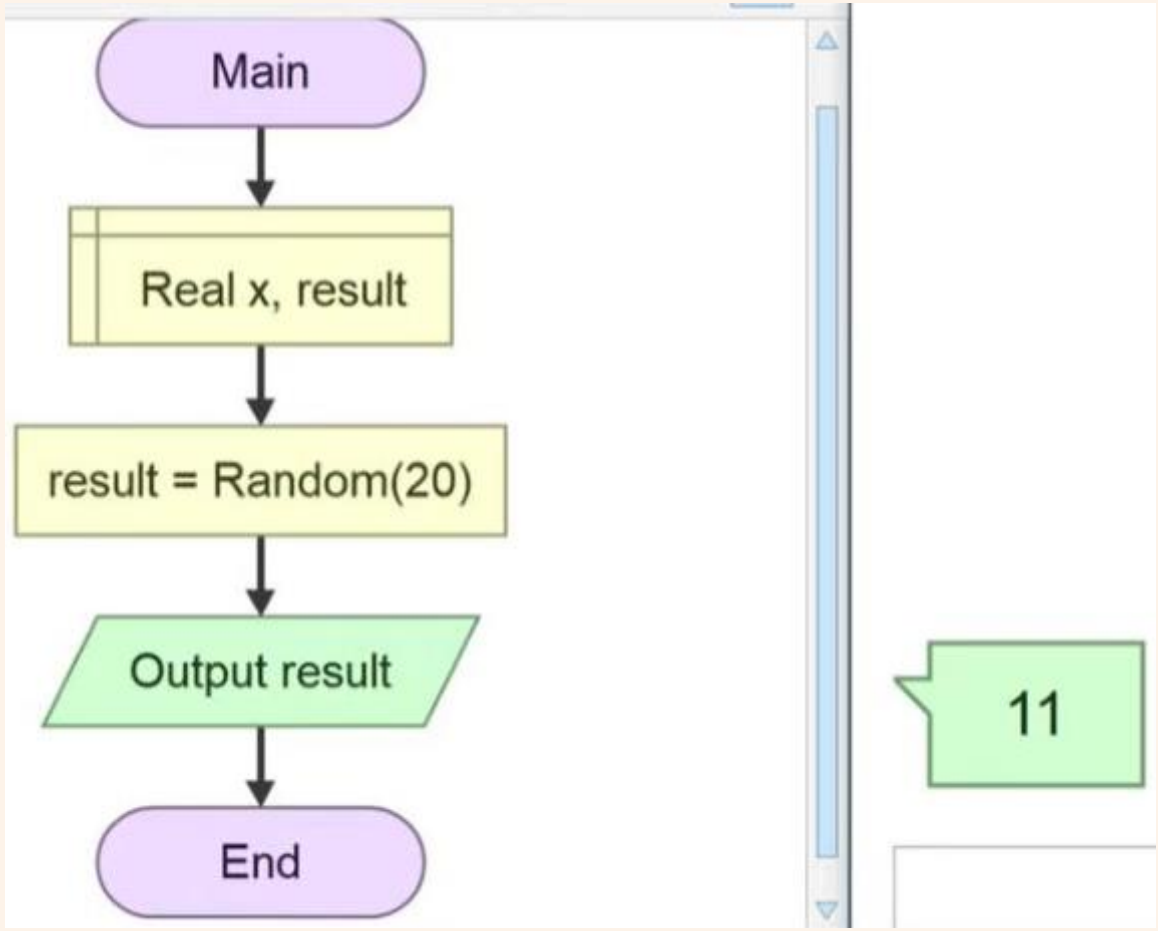
Simulating Probability with `Random(n)` .

- **Random(n):** Returns a random integer between 0 and n-1.
- **Range:** 0 (inclusive) to n (exclusive).
- **Example:** Random(100) gives a number from 0 to 99.
- **Application:** To simulate a dice roll (1 to 6):
    - Use: **Random(6) + 1**

Flowchart Example: Random(n)

Here is an example of the `Random` function.

- We use the expression `result = Random(20)` .
- This will generate a number between **0** and **19**.
- In this example run, the output is **11**.



Practical Application: Hypotenuse

Putting it all together to calculate the hypotenuse of a right triangle using `Sqrt`.

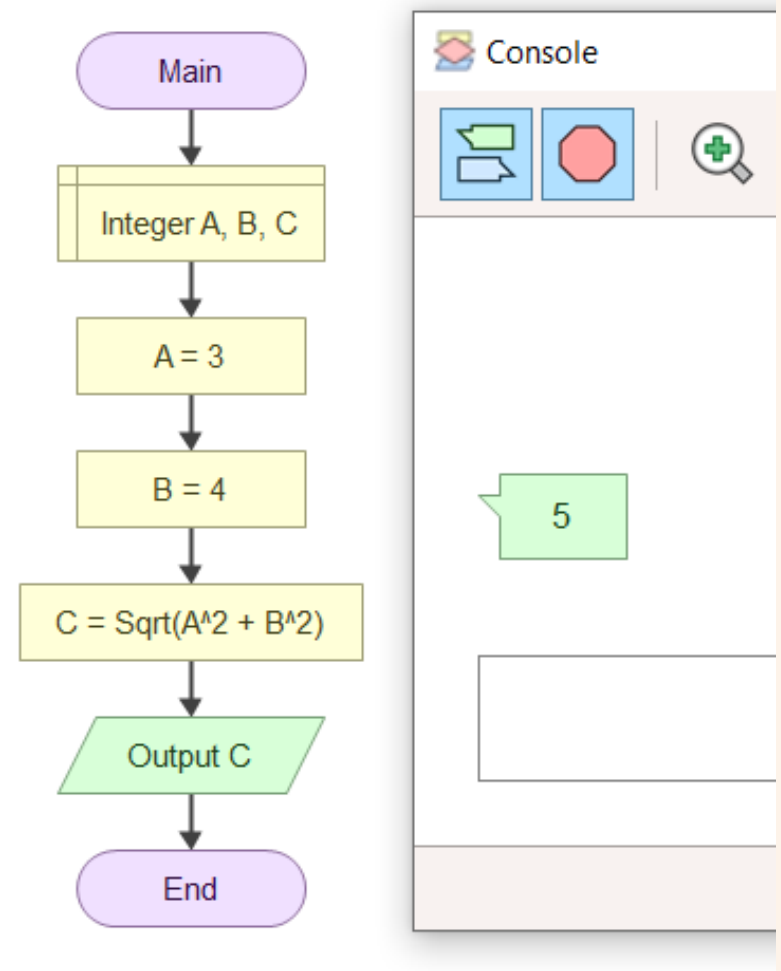
Formula:

$$C = \sqrt{A^2 + B^2}$$

Flowgorithm Code:

```
C = Sqrt(A^2 + B^2)
```

Note: ^ is the power operator.



04

User-Defined Functions

User-Defined Functions

What if the function you need doesn't exist?
Sometimes we need a function specific to our lesson.

User-Defined Functions allow you to create custom logic blocks.

Examples:

- CalculateDiscriminant(a, b, c)
- AreaOfTrapezoid(a, b, h)
- FindHypotenuse(a, b)

User-defined functions allow us to write this logic once and use it many times.

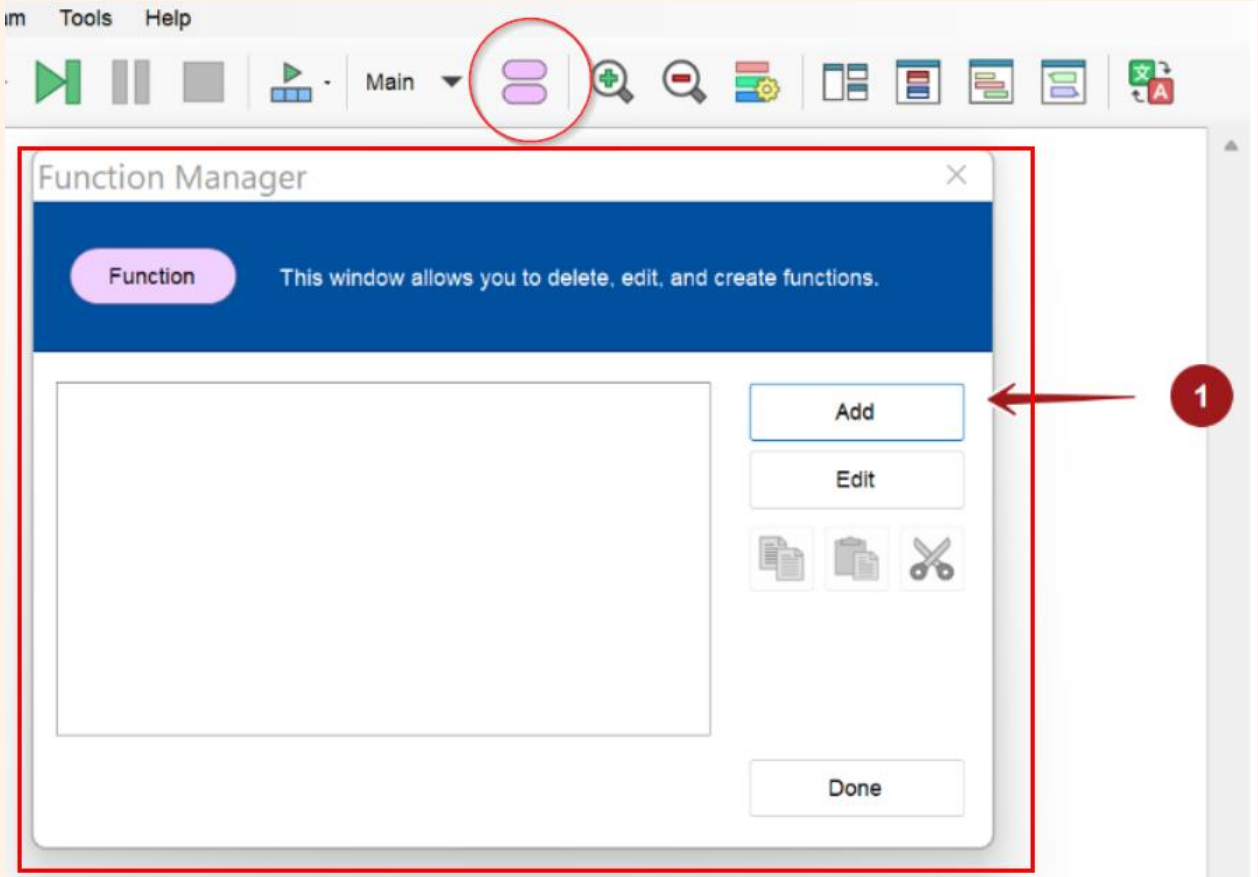
Benefits of User-Defined Functions

Modularity

Break a large flowchart into smaller, readable pieces.

Reusability

Write the logic once, call it many times.



The Function Manager

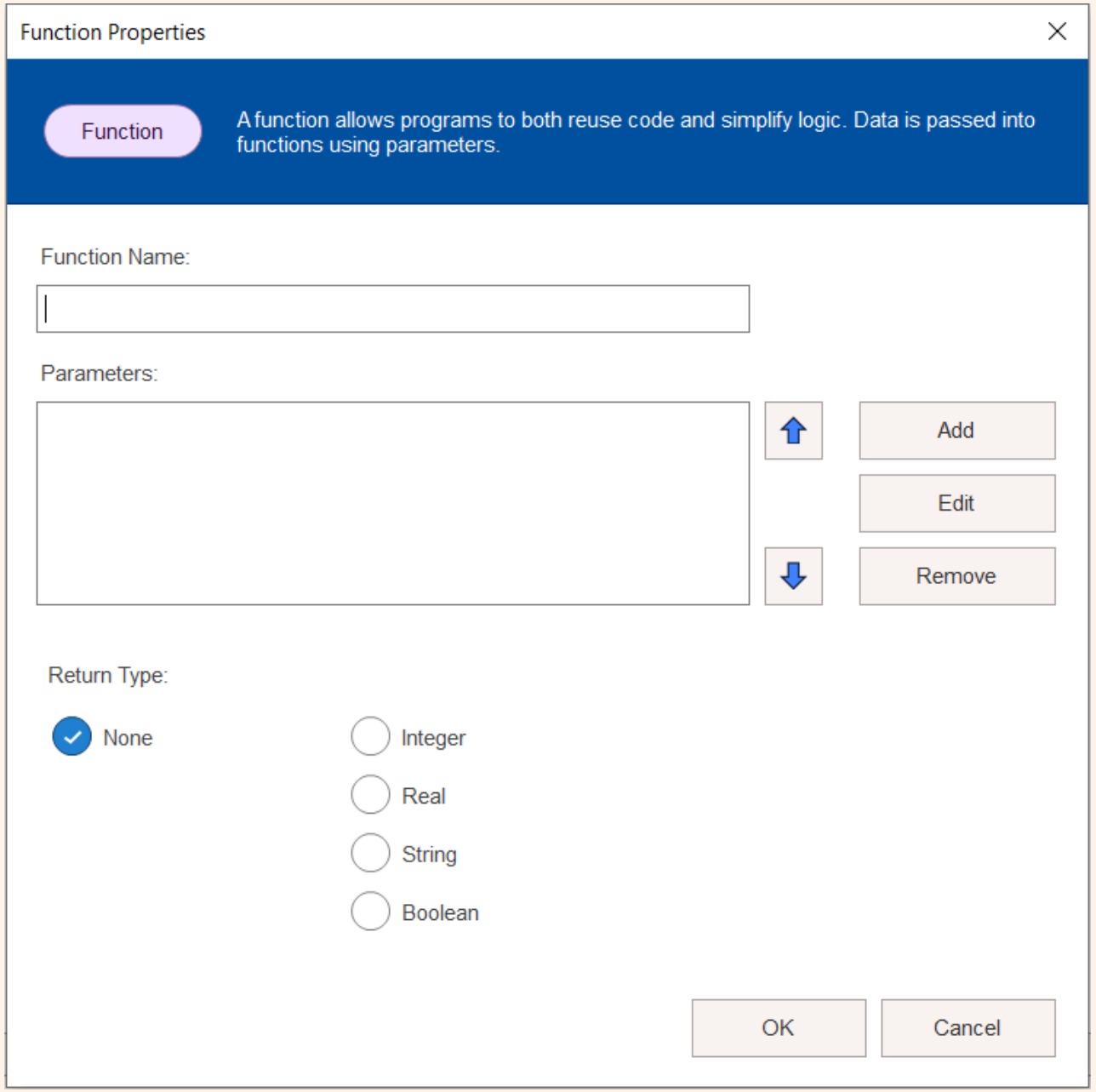
The Function Manager is the tool used to create custom functions.

How to access it:

1. Go to the **Menu bar**.
2. Select **Function Manager**.
3. Click **Add**.

Creating a New Function

Add button opens the **Function Properties** window, where you define the "signature" of your function.



Function Properties: Name

1

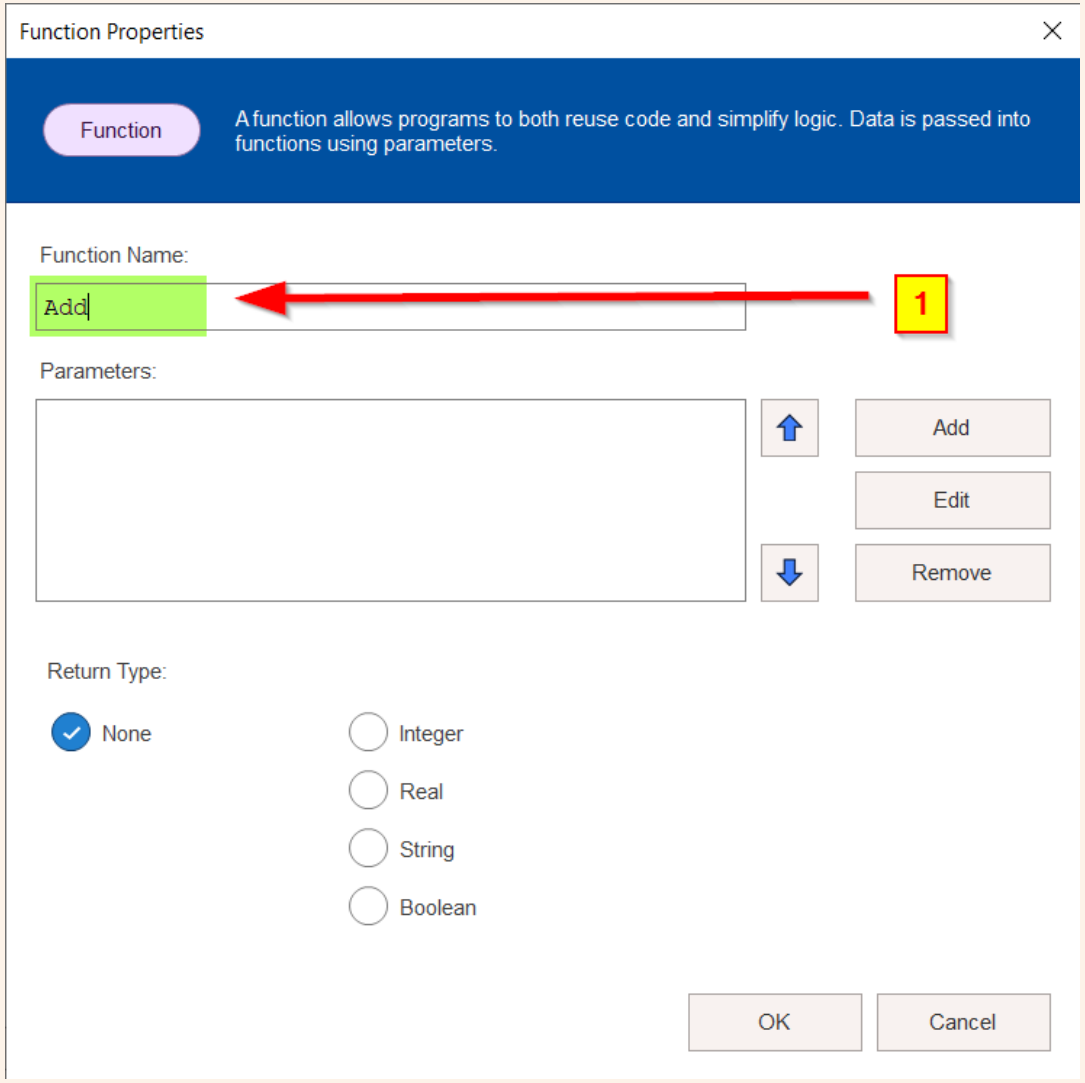
Function Name

Give it a descriptive name. It should describe what the function does.

Examples:

- `Add`
- `CalculateArea`
- `IsPrime`

Note: Names cannot contain spaces.



Function Properties: Parameters

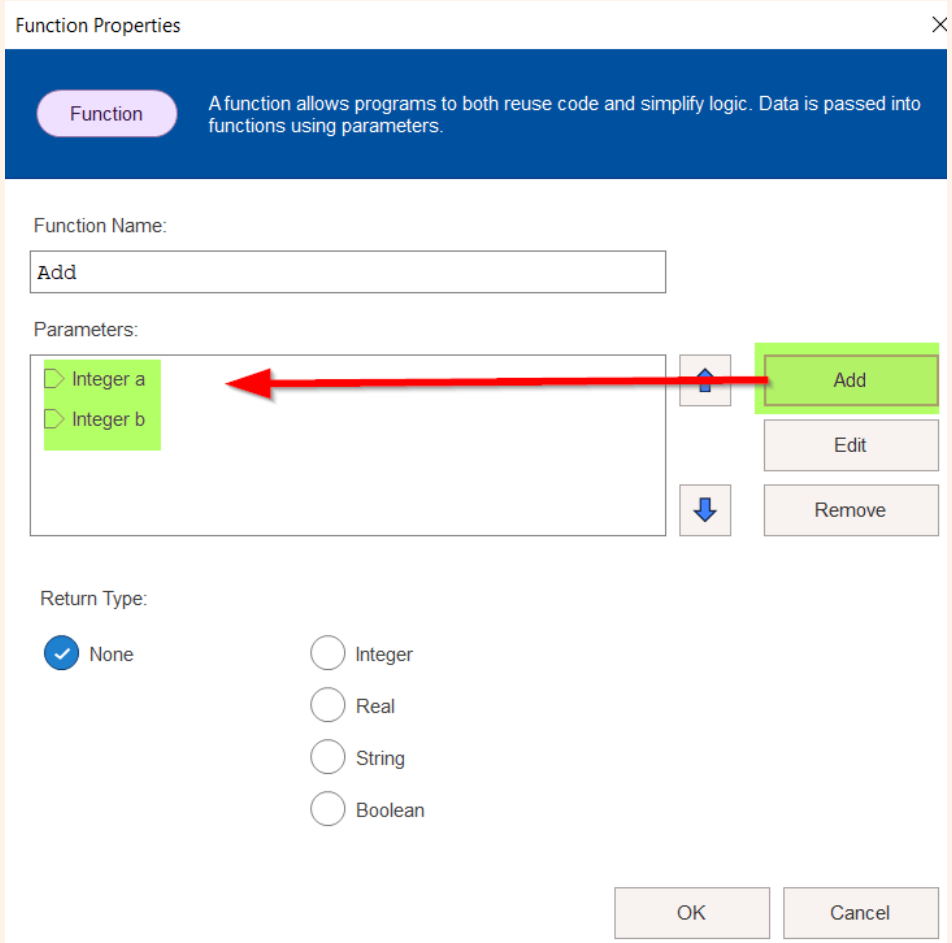
2

Parameters (The Inputs)

These are the inputs the function needs to do its job. They are like the variables **x** and **y** in **f(x, y).**

You must specify the **Variable Type** (Integer, Real, etc.) and the **Name** for each parameter.

Example:
For an Addition function, we need `Integer a` and `Integer b`.



Function Properties: Return Type

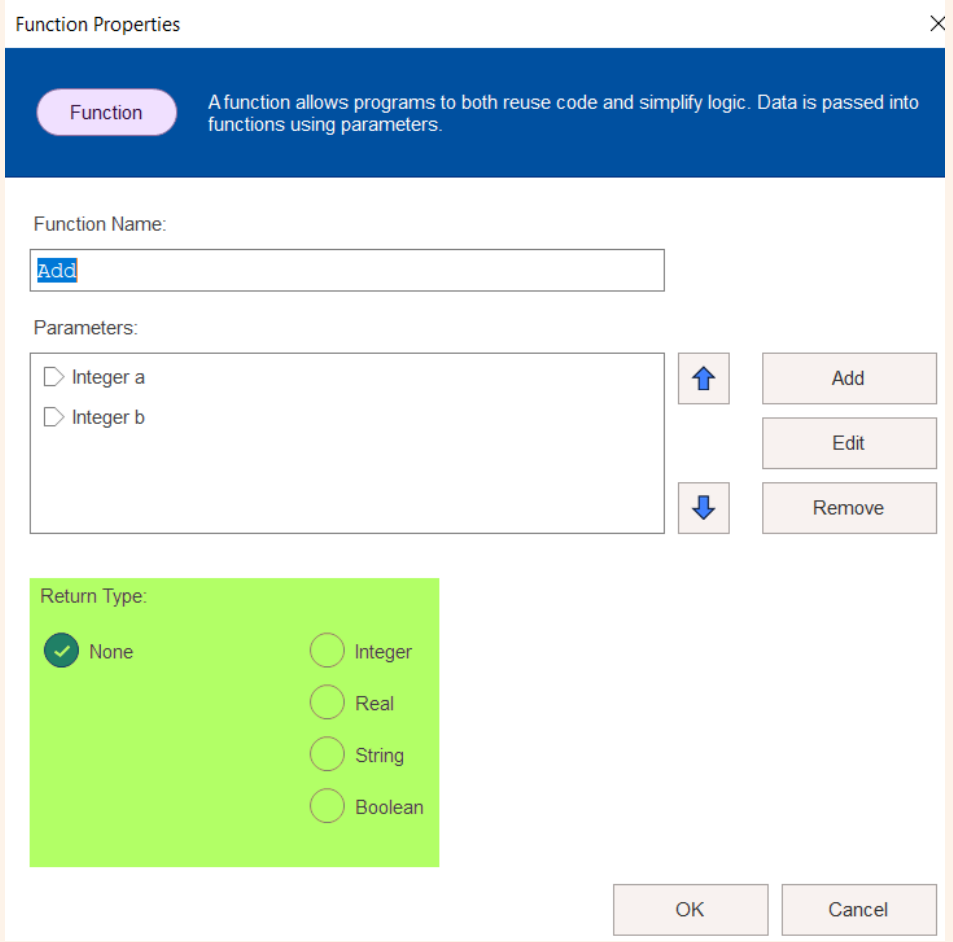
3

Return Type (The Output)

What kind of data does this function give back?

- If it calculates a sum of integers, the return type is **Integer**.
- If it calculates a square root, the return type might be **Real**.

Return Variable: The name of the variable that holds the final answer (e.g., result).

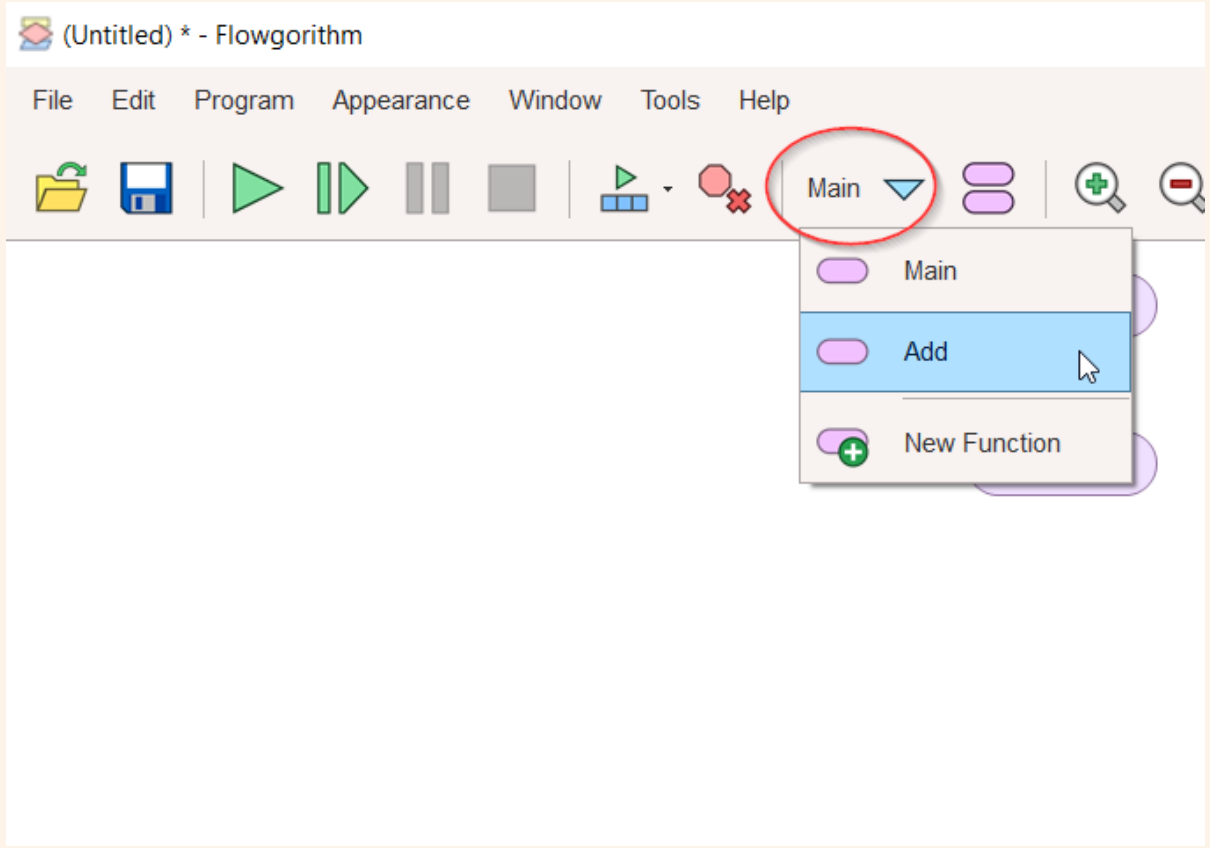


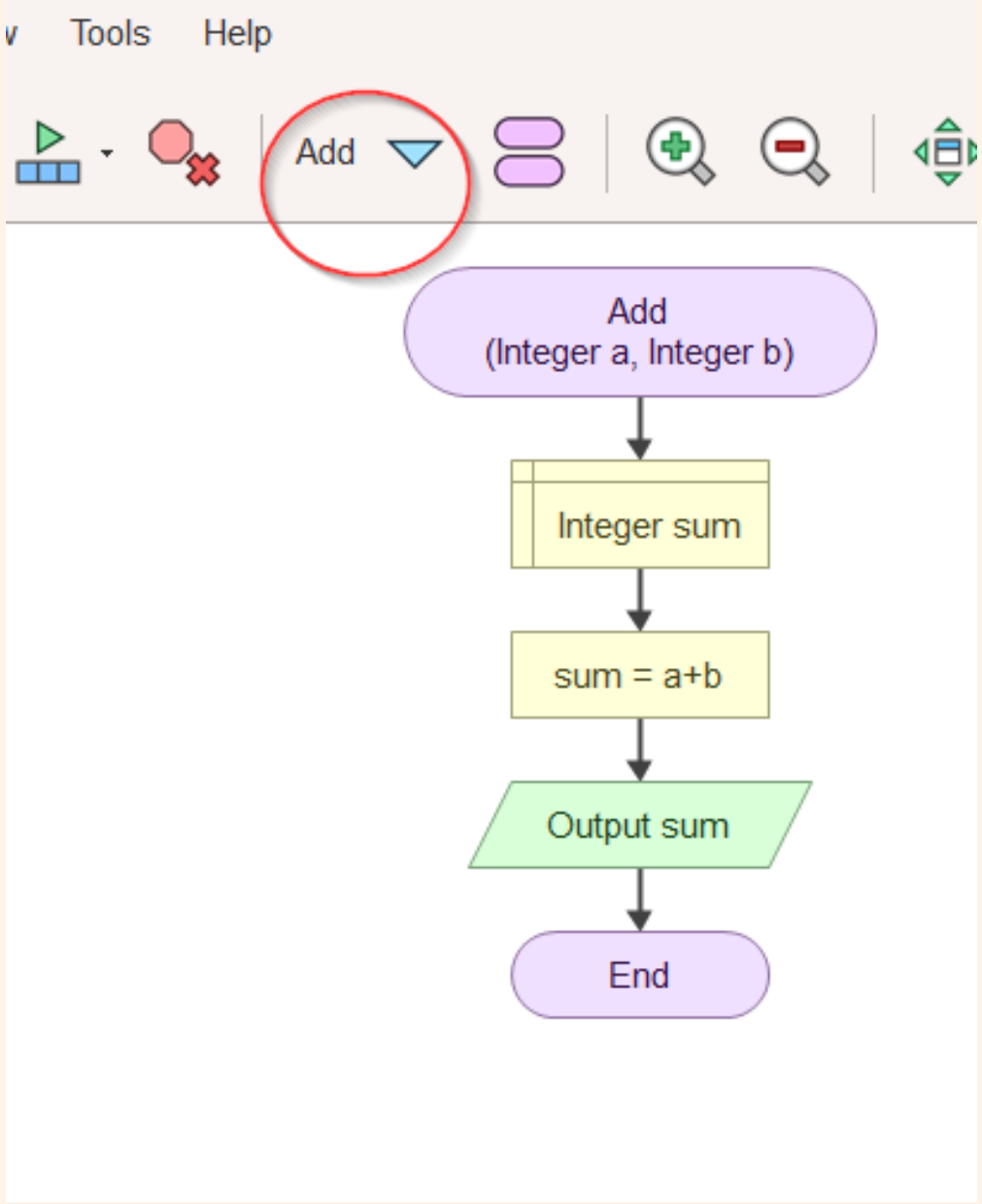
The Function Interface

Once you click OK, Flowgorithm opens a **new tab** for your function.

Look at the top of the screen. You will see `Main` and your new function (e.g., `Add`).

You write the flowchart for the function **inside this new tab**.





Visualizing the 'Add' Function

Let's create the logic for our `Add` function.

Goal:

1. Take two numbers, add them, and return the result.

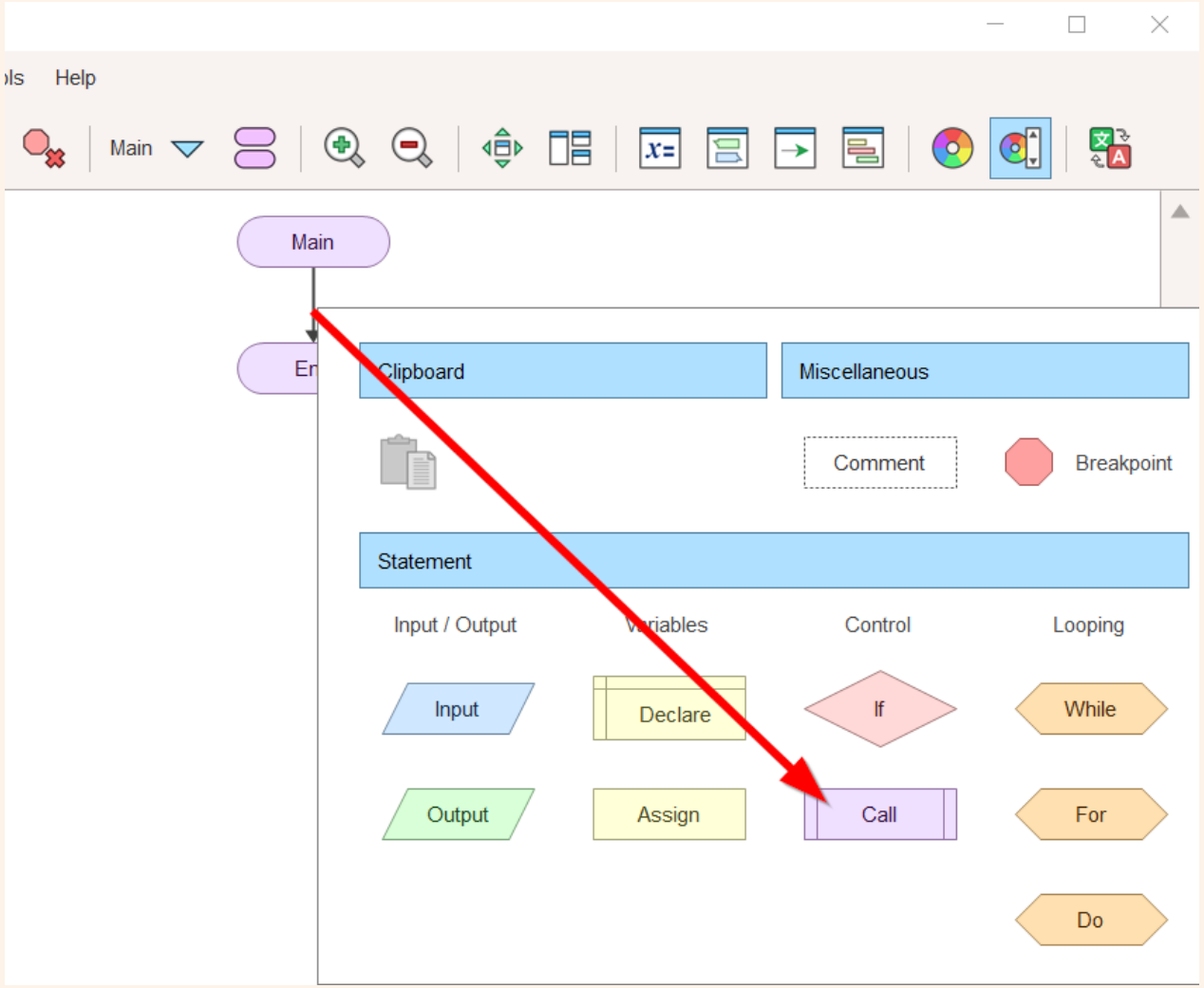
Steps:

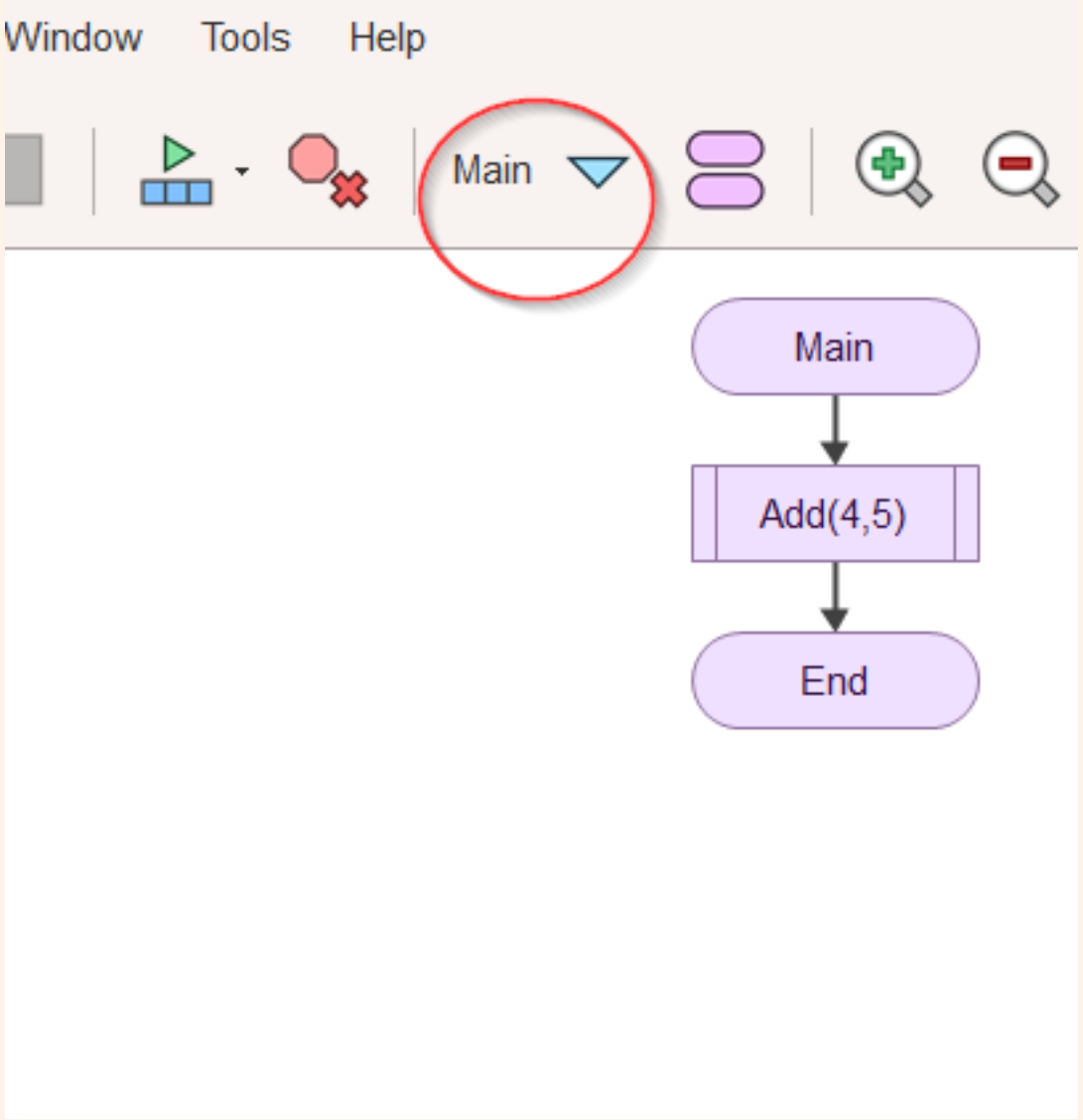
1. The function starts with parameters `a` and `b`.
2. Assign `sum = a + b`.
3. The function automatically returns the value in `sum`.

Using the Function: The 'Main' Tab

Now that the function is defined, we need to use it.

1. Switch back to the **Main** tab.
2. To run a function, we use a specific symbol called the **Call** statement.





The Call Statement

To use your new function, use the **Call** symbol in the Main flowchart.

Syntax:

`FunctionName(arguments)`

Example:

`Add(4, 5)`

Here, 4 is passed to `a`, and 5 is passed to `b`.

How It Works (Execution Flow)

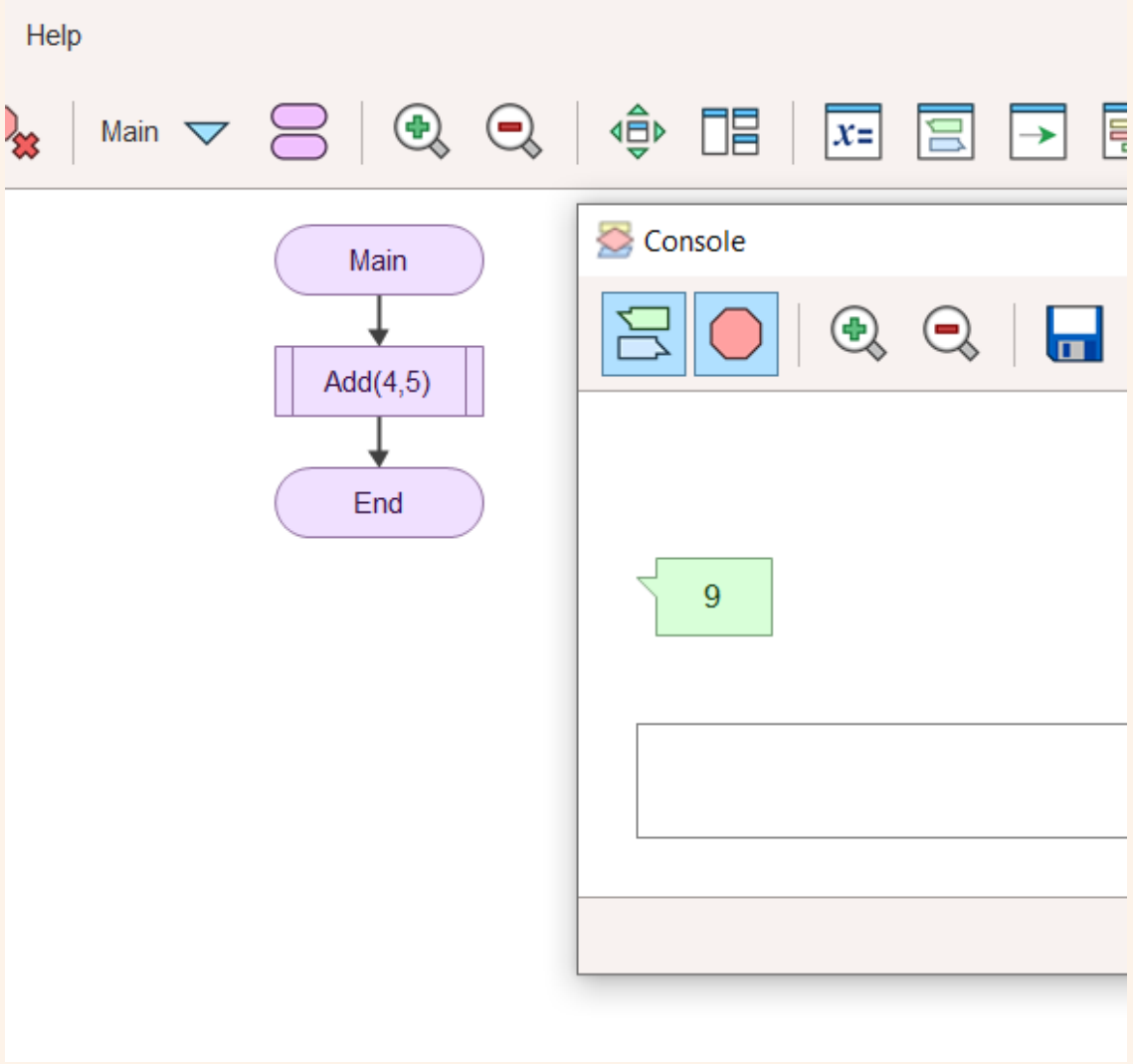
1. Program starts at **Main**.

2. Reaches **Call**: `Add(4, 5)`.

3. Pauses Main, **jumps** to `Add`.

4. Executes logic (calculates 4+5).

5. Returns result, **returns** to Main.

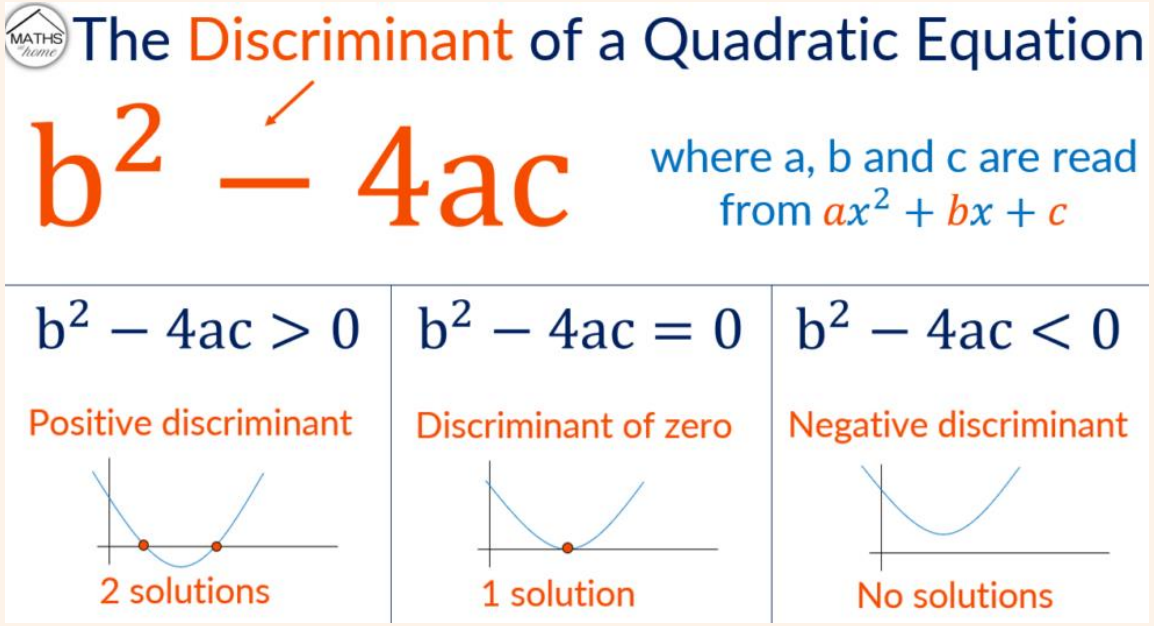


Why is this useful for Math Teachers?

Imagine teaching the **Quadratic Formula**:

$$b^2 - 4ac$$

Instead of one huge, messy flowchart, you can create helper functions to make the logic clear and teachable.



Workshop Activity: Quadratic Formula

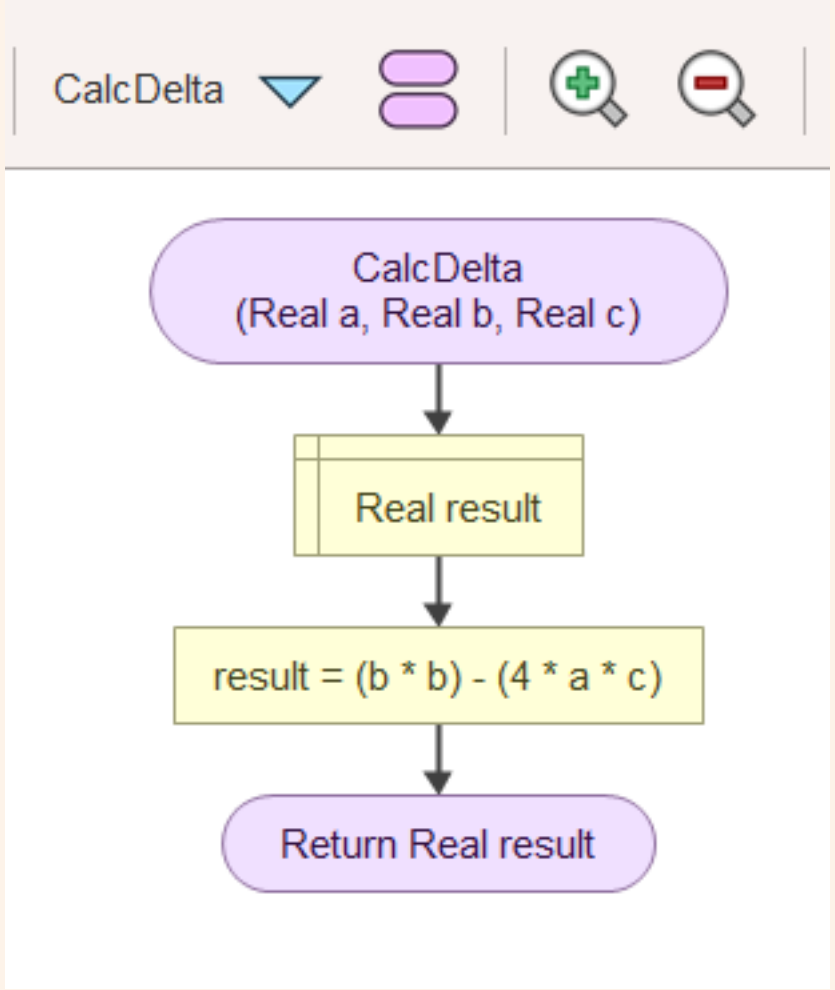
Let's Build a Function to Calculate the Discriminant (Δ) Together.

Function Name:
`CalcDelta`

Parameters:
`Real a, Real b, Real c`

Return Type:
`Real`

Logic:
`result = (b * b) - (4 * a * c)`



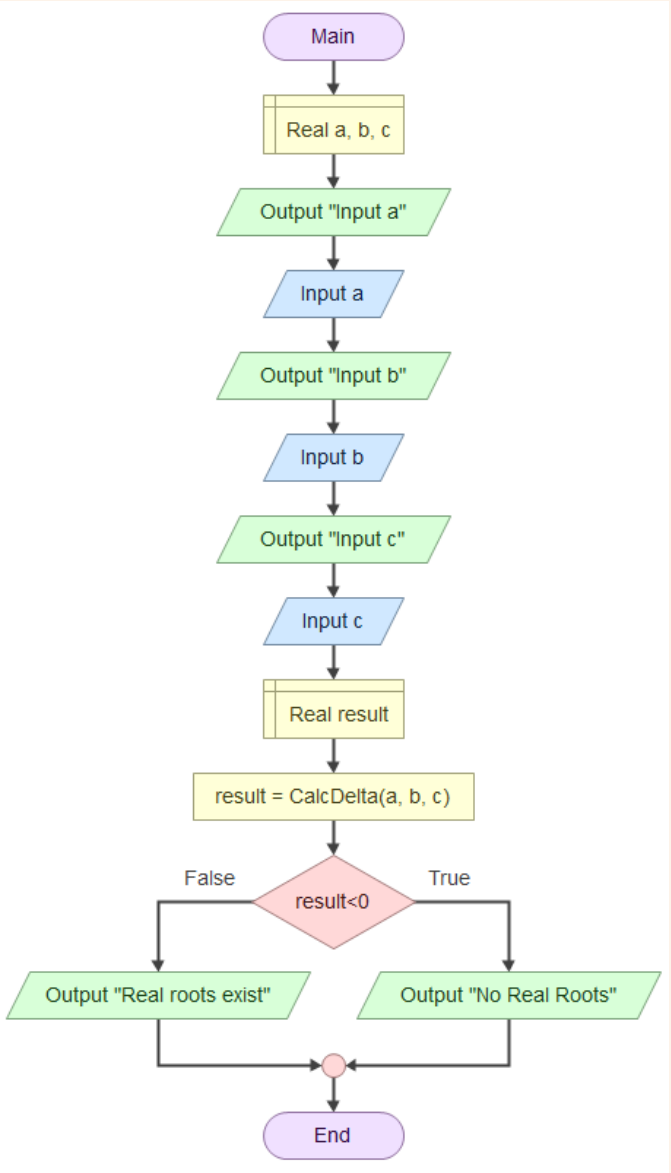
Workshop Activity: Step 2

Using the Function in the Main Tab

1. Ask the user for a, b, and c .
2. Call `CalcDelta(a, b, c)` and store the result.
3. Use an **If Statement**:

If Delta < 0:
Output "No Real Roots".

Else:
Real Roots exist.



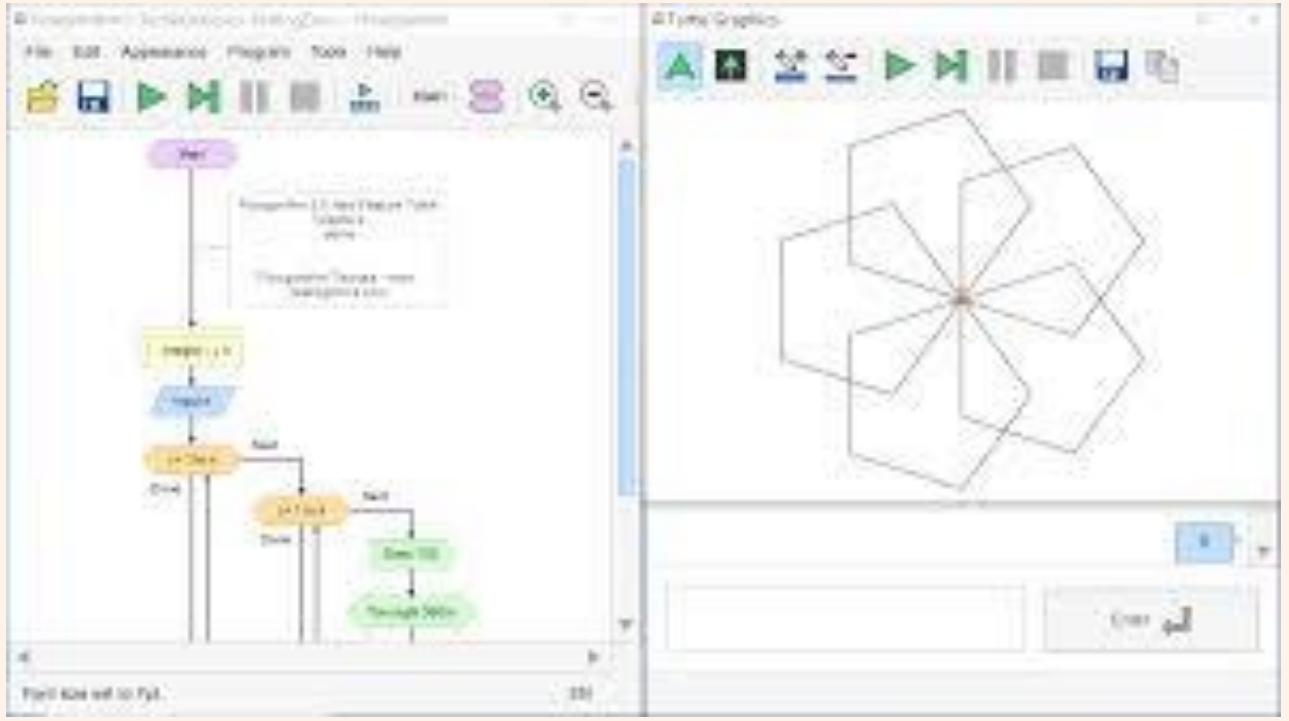
Next Steps

Coming up in **Session 6**:

Now that we understand Loops and Functions, we are ready for the most visual part of the training.

Turtle Graphics!

We will use code to draw geometric shapes, patterns, and fractals, and see how functions help us create complex art.



Q&A

Do you have any questions?

(End of Session 5)

End of Session 5