Arithmetic Operators

Lesson Progress

100% Complete

As of now, you've mastered sprites, backdrops, the Stage, drawing, animating, and creating variables. But PictoBlox is not just about drawing and animating; it's much more than that. For starters, you can perform mathematical and logical operations using it. Let's see how.

What are Operators?

> Operators are symbols that tell the computer to perform specific mathematical or logical manipulations.

Following are the different types of operators in PictoBlox:

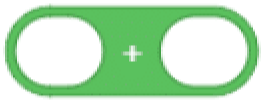
- Arithmetic Operators
- Relational Operators
- Logical Operators

In this lesson, you'll learn the arithmetic operators.

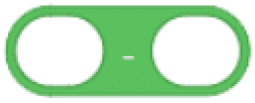
Arithmetic Operators

There are four main arithmetic operators in PictoBlox:

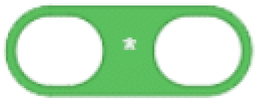
- **Addition**: Adds values on either side of the operator



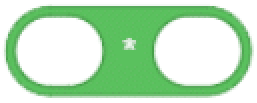
Subtraction: Subtracts the right-hand operand from the left-hand operand



Multiplication: Multiplies values on either side of the operator



Division: Divides the left-hand operand by the right-hand operand



Now, let’s perform an activity to see them in action.

Activity: Addition Bot

Objective

In this activity, you’ve to make a script that:

Asks the user to input Number 1.

Asks the user to input Number 2.

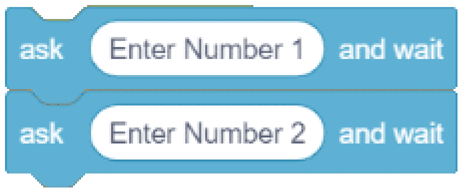
Performs addition operation on Number 1 and Number 2 and saves the result in a variable answer.

Display the answer to the user using a **say () for () seconds** block.

Coding

Let’s make it:

Add two **ask () and wait** blocks to get the input from users for Number 1 and Number 2. Using the **ask () and wait** block will add an input box (with the specified text above it) at the bottom of the Stage. The user can write the answer in it and submit it which is then stored then in the **answer** block.



```
ask (Enter Number 1) and wait
ask (Enter Number 2) and wait
```

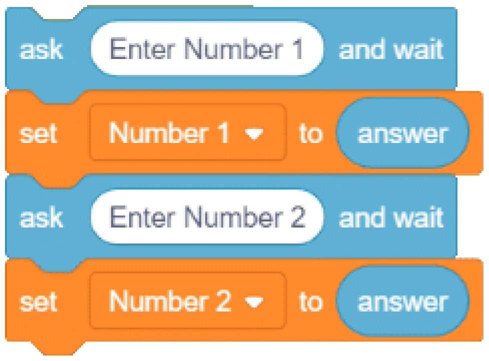
Next, make 3 variables:

Number 1: To store the value of the first input.

Number 2: To store the value of the second input.

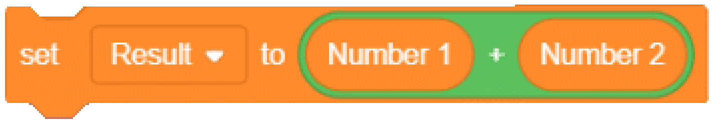
Result: To store the final output value.

Place two **set () to ()** blocks, one below each **ask () and wait** block to store the input values in Number 1 and Number 2.



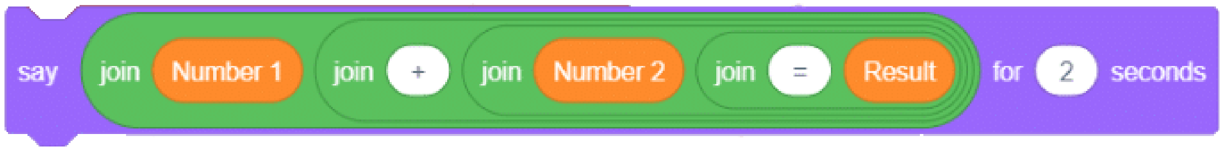
```
ask (Enter Number 1) and wait
set [Number 1 ▾] to (answer)
ask (Enter Number 2) and wait
set [Number 2 ▾] to (answer)
```

Place another **set () to ()** block at the end and choose Result from its drop-down. Place a **() + ()** block (addition operator block) in the white space and add Number 1 and Number 2.



```
set [Result ▾] to ((Number 1) + (Number 2))
```

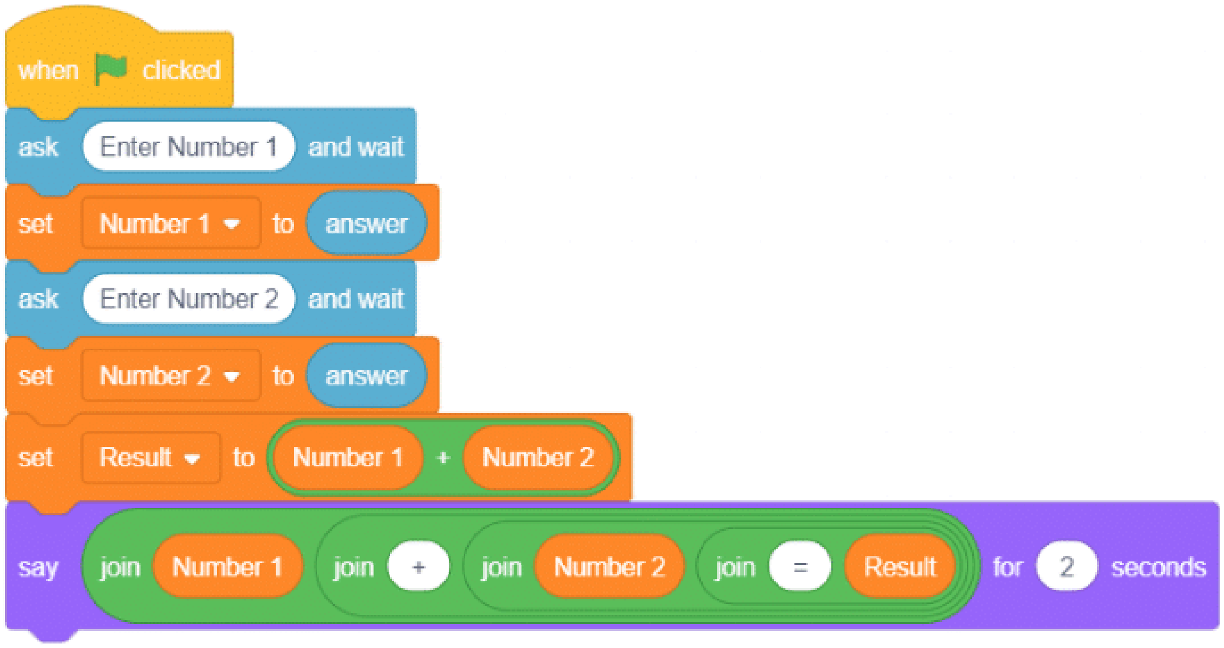
Make Tobi say the result out loud using the **say () for () seconds** block.



```
say (join (Number 1) (join (+) (join (Number 2) (join (=) (Result))))) for (2) seconds
```

Complete the script by adding a **when flag clicked** block.

Below is the complete script:



```
when green flag clicked
ask (Enter Number 1) and wait
set [Number 1 v] to (answer)
ask (Enter Number 2) and wait
set [Number 2 v] to (answer)
set [Result v] to ((Number 1) + (Number 2))
say (join (Number 1) (join (+) (join (Number 2) (join (=) (Result))))) for (2) seconds
```

Relational Operators

A relational operator is an operator that tests a relation between two entities. The result of a relational operator is either true or false.

There are 3 operators in PictoBlox:

Greater Than >

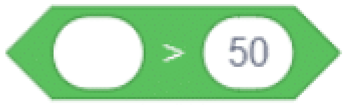
Less Than <

Equals to =

Let’s have a look at each one by one.

Greater Than

This block reports true *only* if the first number is greater than the second number. If the first number is equal to or less than the second, it results in false.

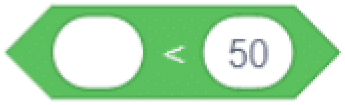


Example:

- 15 > 10 will return true.
- 10 > 15 will return false.

Less Than

This block reports true *only* if the first number is less than the second number. If the first number is equal to or greater than the second, it results in false.

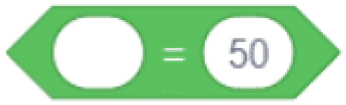


Example:

- 15 > 10 will return false.
- 10 > 15 will return true.

Equals to

It results in true *only* if the first number is equal to the second number; otherwise, it results in false.



Precedence

Like arithmetic operators, logical operators also have precedence that determines how the operations are grouped without parentheses. Parentheses can be used to group the operands with their correct operator,
just like how we do it in arithmetic.

Have a look at the below table to check the precedence of logical operators:

NOT (!) – High
AND (&&) – Medium
OR (||) – Low

In the below example, A1, A2, A3, and A4 stand for the relational expressions. They have a mix of “&&”, “||” and “!” operators.

A1 && A2 && A3 || A4 can be interpreted as **((A1 && A2) && A3) || A4**
A1 || A2 && A3 can be interpreted as **A1 || (A2 && A3)**
! A1 && A2 || A3 can be interpreted as **((!A1) && A2) || A3**
A1 && A2 || A3 && A4 can be interpreted as **(A1 && A2) || (A3 && A4)**

It is a common practice to use parenthesis to group operands together rather than relying on logical operator precedence rules.