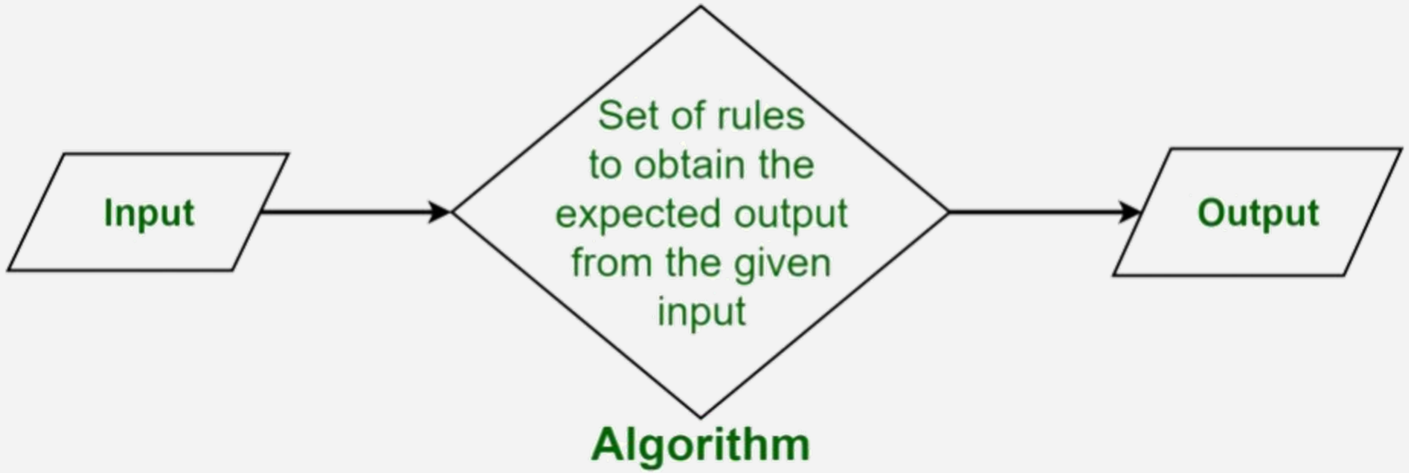
INTRODUCTION TO ALGORITHMS

Fabio Stroppa

What is an Algorithm?

DEFINITION:

An algorithm is a finite sequence of instructions to solve a particular problem.


Input
Set of rules to obtain the expected output from the given input
Output
Algorithm

Examples of Algorithms

- The daily routine of waking up in the morning.
- Cooking a new recipe.
- Making a phone call.
- Finding a book in a bookshelf.
- Dividing two numbers.
- Going from home to school by following a precise route.

Characteristics of Algorithms

In order for some instructions to be an algorithm, it must have the following characteristics:

- Clear and Unambiguous: Each of the algorithm's steps should be clear in all aspects and must lead to only one meaning. It must have only one starting and one ending point.
- Well-Defined Inputs: An algorithm may or may not take input, but when it does, it should be well-defined inputs.
- Well-Defined Outputs: The algorithm must clearly define what output will be yielded and it should be well-defined as well. It should produce at least one output.
- Finiteness: The algorithm must be finite (i.e., it should terminate after a finite time).
- Feasible: The algorithm must be simple, generic, and practical, such that it can be executed with the available resources.
- Language Independent: The Algorithm must be designed with simple and plain instructions that can be implemented in any language, and yet the output will be the same, as expected.

Pro and Cons of Algorithms

ADVANTAGES:

- They are easy to follow and understand (if well designed).
- They are a step-wise representation of a solution to a given problem.
- It breaks down a big problem into smaller sub-problems that can be solved independently.

DISADVANTAGES:

- Writing an algorithm may take long time.
- Understanding the logic can be very difficult for beginners.
- For very intricate and complex algorithms, it is very hard to understand what they do unless you follow the flow of instructions step-by-step.

Designing Algorithms

1. Problem Definition: identify the problem to be solved.
2. Constraint Definition: identify the constraints of the problem, if any.
3. Input Definition: identify the inputs of the problem and their domain, if any.
4. Output Definition: identify the expected outputs of the problem.

How to express an Algorithm.

Algorithms can be expressed with different techniques:

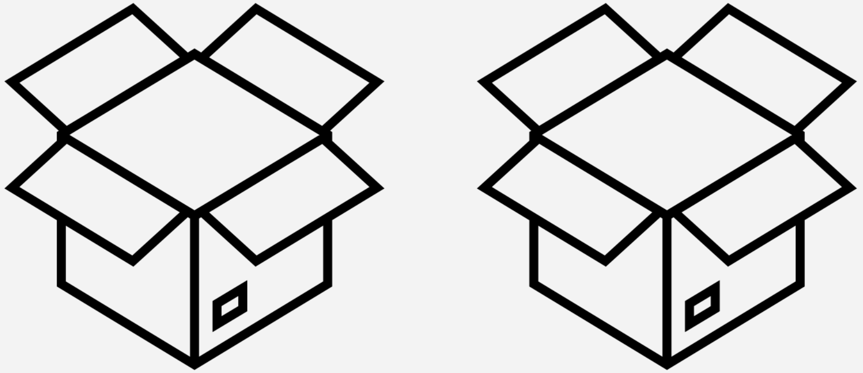
- Natural Language: Here we express the Algorithm in natural language (e.g., English). It is too hard to understand the algorithm from it.
- Flow Chat: Here we express the Algorithm by making graphical/pictorial representation of it. It is easier to understand than Natural Language.
- Pseudo Code: Here we express the Algorithm in the form of annotations and informative text written in plain English which is very much similar to the real code but as it has no syntax like any of the programming language, it can't be compiled or interpreted by the computer. It is the best way to express an algorithm because it can be understood by even a layman with some school level programming knowledge.

What is a Variable?

To write algorithms that perform useful tasks that really save us work, we need to introduce the concept of variables.

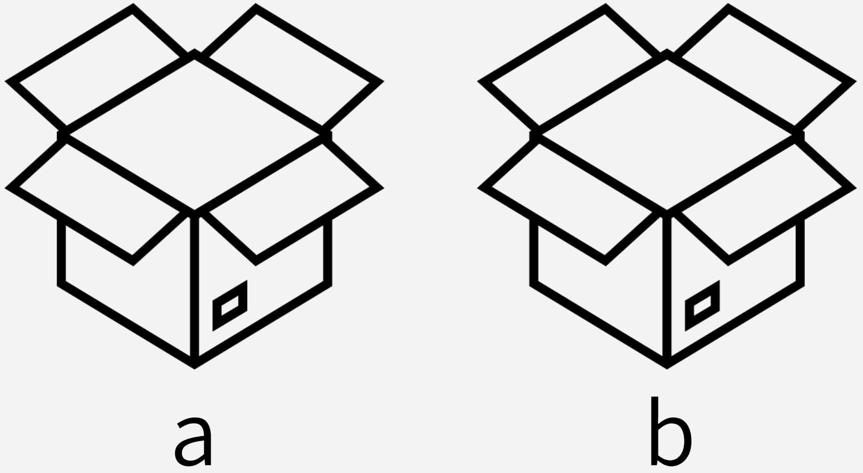
What is a Variable?

To write algorithms that perform useful tasks that really save us work, we need to introduce the concept of **variables**.



What is a Variable?

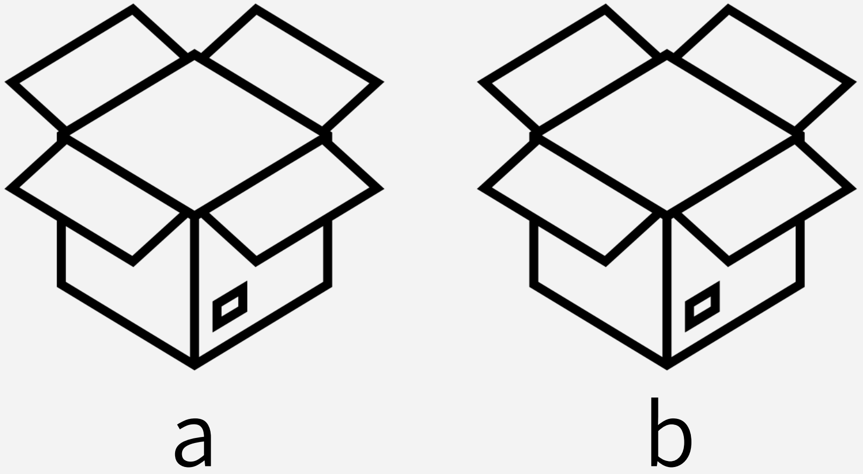
To write algorithms that perform useful tasks that really save us work, we need to introduce the concept of variables.



What is a Variable?

To write algorithms that perform useful tasks that really save us work, we need to introduce the concept of **variables**.

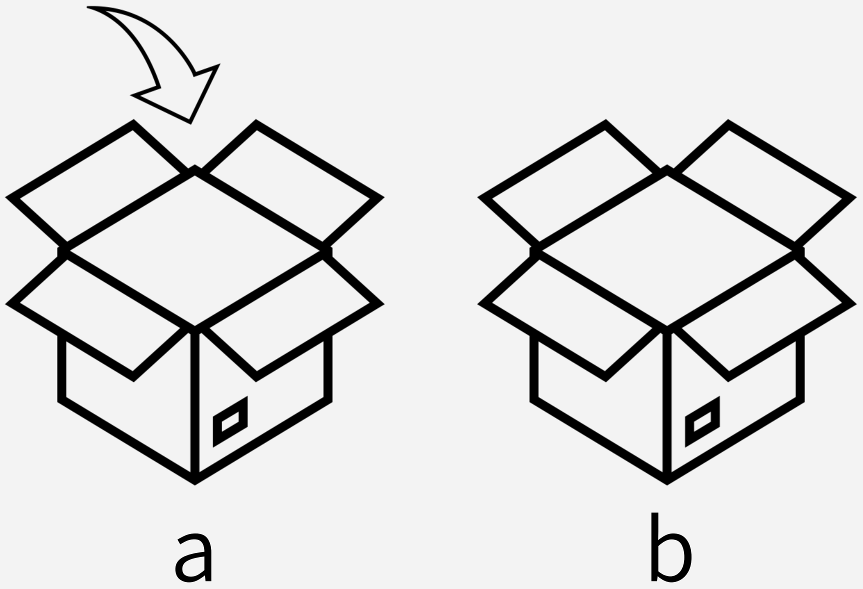
- 5



What is a Variable?

To write algorithms that perform useful tasks that really save us work, we need to introduce the concept of variables.

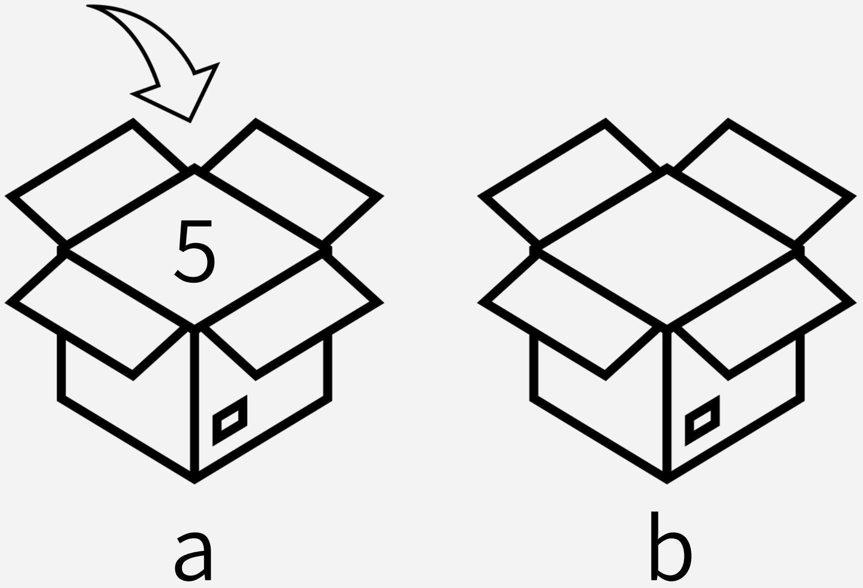
- 5



What is a Variable?

To write algorithms that perform useful tasks that really save us work, we need to introduce the concept of variables.

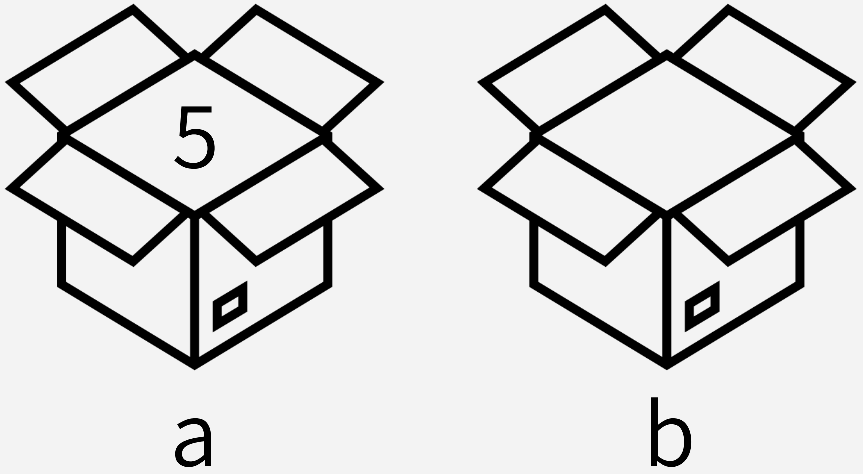
- 5



What is a Variable?

To write algorithms that perform useful tasks that really save us work, we need to introduce the concept of variables.

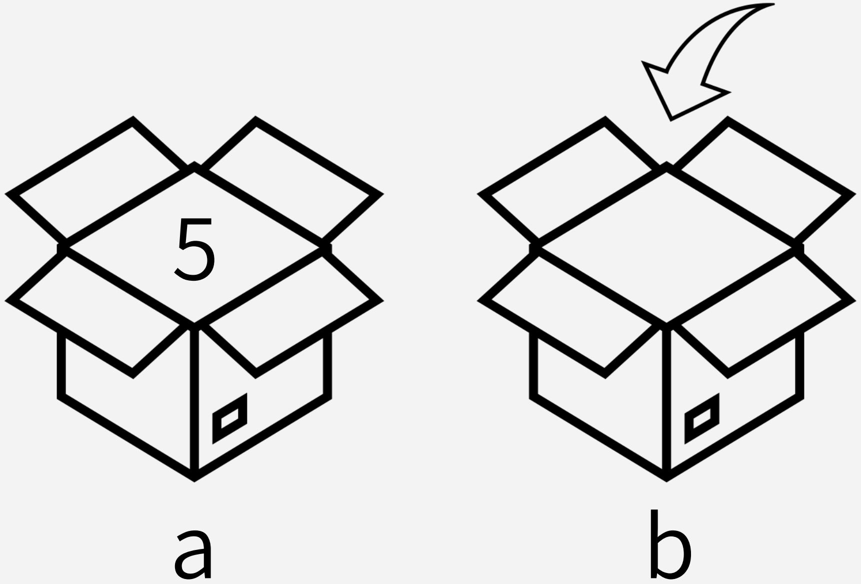
- 5
- 2



What is a Variable?

To write algorithms that perform useful tasks that really save us work, we need to introduce the concept of variables.

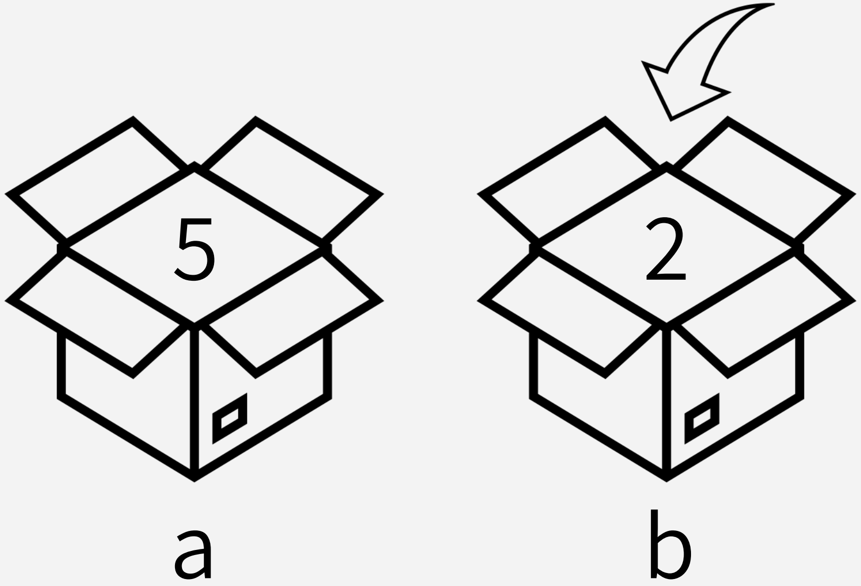
- 5
- 2



What is a Variable?

To write algorithms that perform useful tasks that really save us work, we need to introduce the concept of variables.

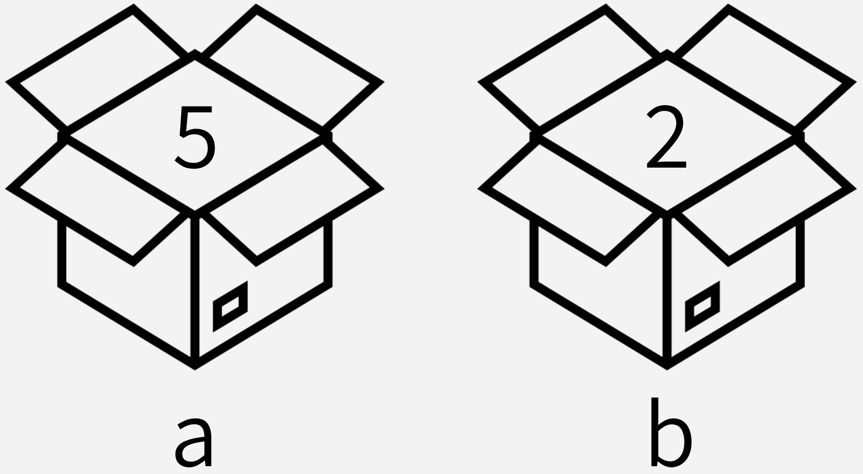
- 5
- 2



What is a Variable?

To write algorithms that perform useful tasks that really save us work, we need to introduce the concept of variables.

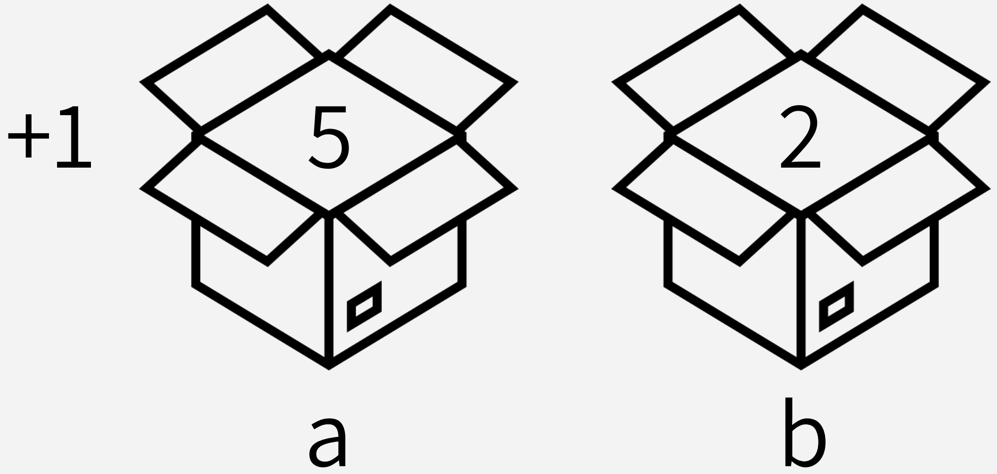
- 5+1
- 2



What is a Variable?

To write algorithms that perform useful tasks that really save us work, we need to introduce the concept of **variables**.

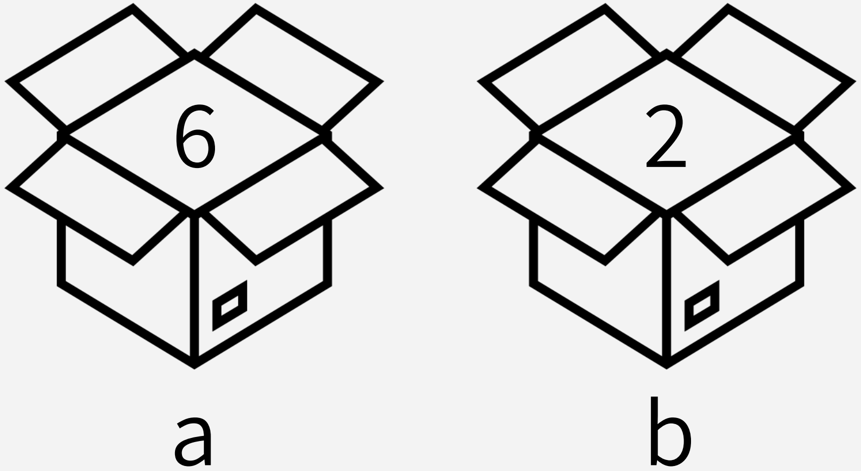
- 5+1
- 2



What is a Variable?

To write algorithms that perform useful tasks that really save us work, we need to introduce the concept of variables.

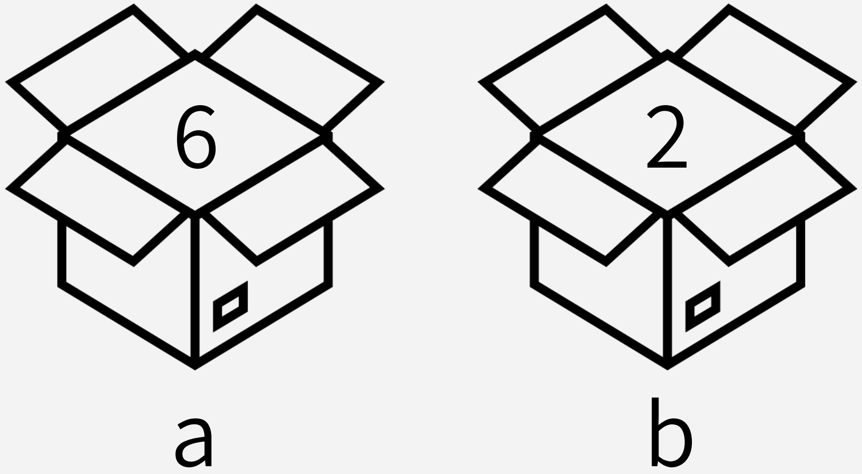
- 5+1
- 2



What is a Variable?

To write algorithms that perform useful tasks that really save us work, we need to introduce the concept of **variables**.

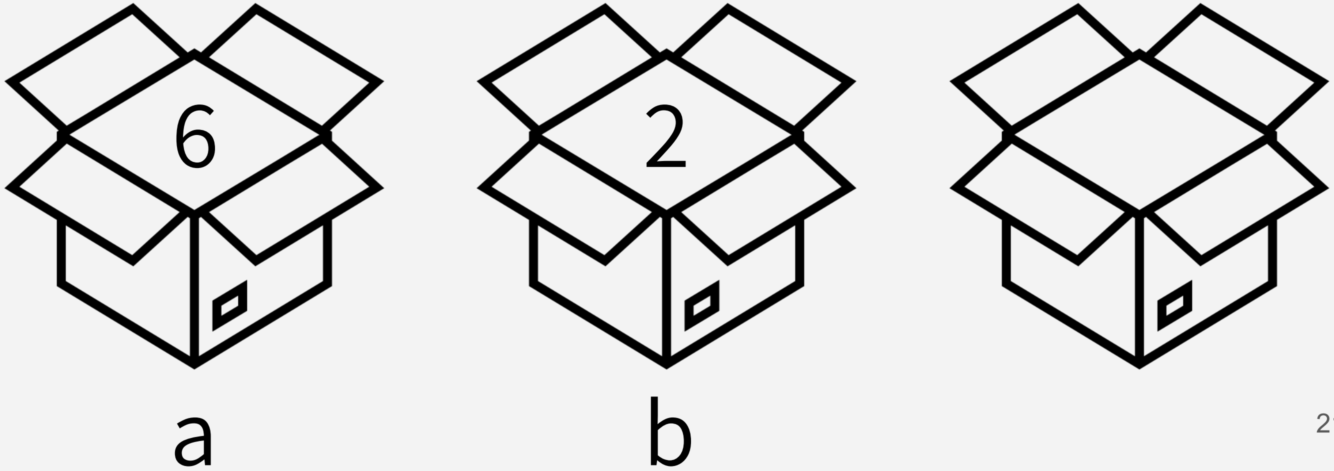
$$\left.\begin{array}{l} \text{➢ } 5+1 \\ \text{➢ } 2 \end{array}\right\} \text{subtract}$$



What is a Variable?

To write algorithms that perform useful tasks that really save us work, we need to introduce the concept of variables.

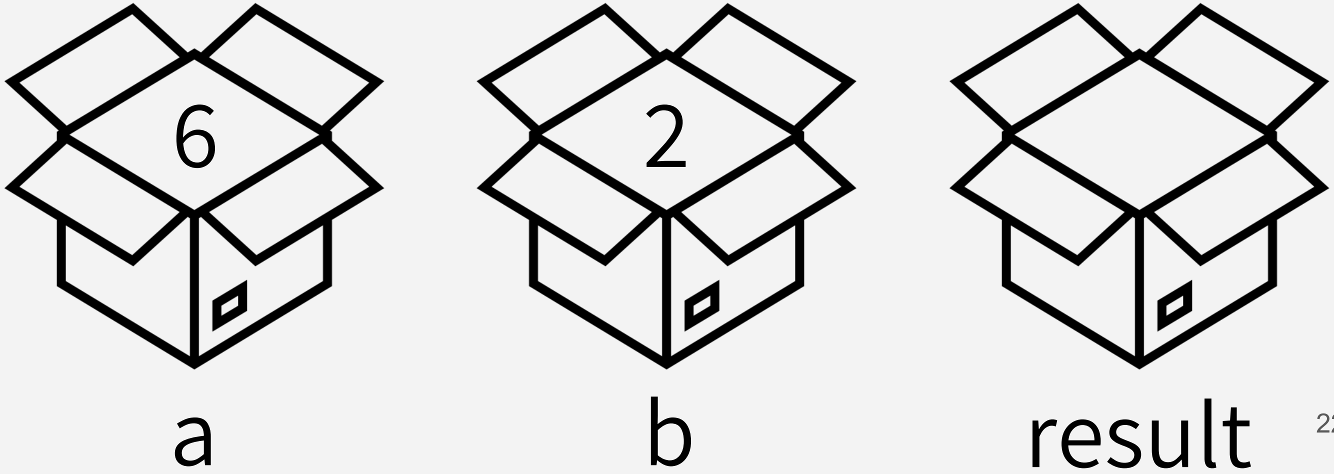
$$\left.\begin{array}{l} \text{➢ } 5+1 \\ \text{➢ } 2 \end{array}\right\} \text{subtract}$$



What is a Variable?

To write algorithms that perform useful tasks that really save us work, we need to introduce the concept of **variables**.

$$\left.\begin{array}{l} \text{➢ } 5+1 \\ \text{➢ } 2 \end{array}\right\} \text{subtract}$$

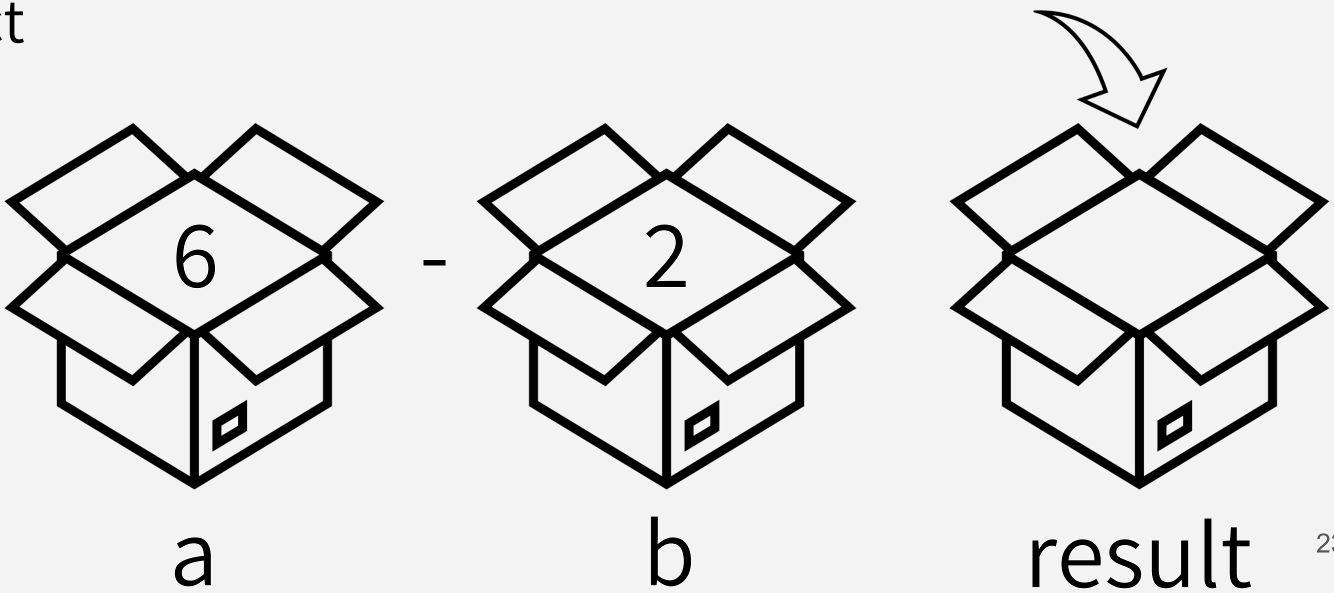


What is a Variable?

To write algorithms that perform useful tasks that really save us work, we need to introduce the concept of **variables**.

- 5+1
- 2

subtract

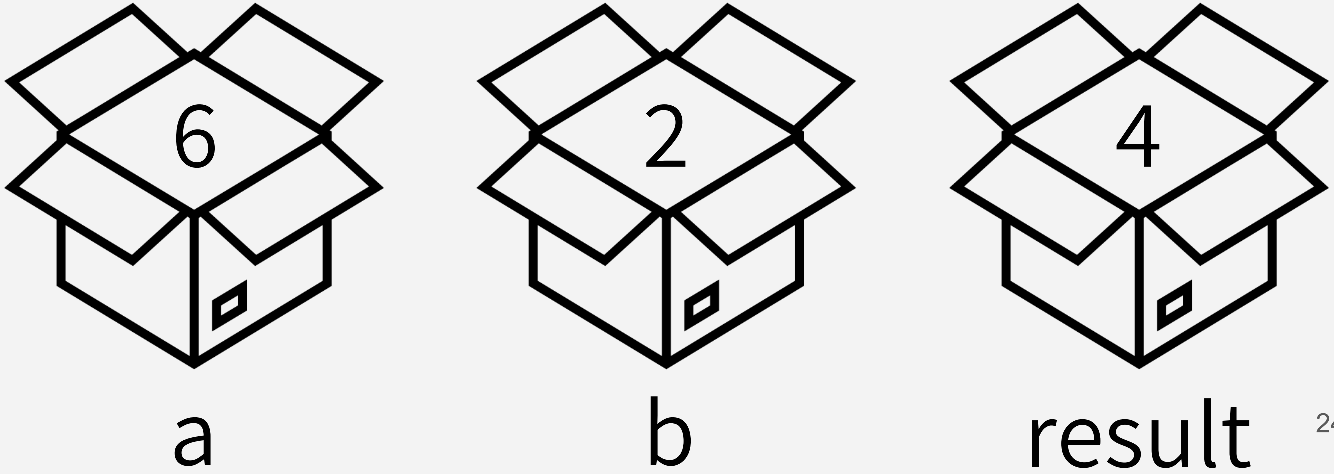


What is a Variable?

To write algorithms that perform useful tasks that really save us work, we need to introduce the concept of **variables**.

- 5+1
- 2

$\left.\begin{array}{l} \\ \\ \end{array}\right\}$ subtract



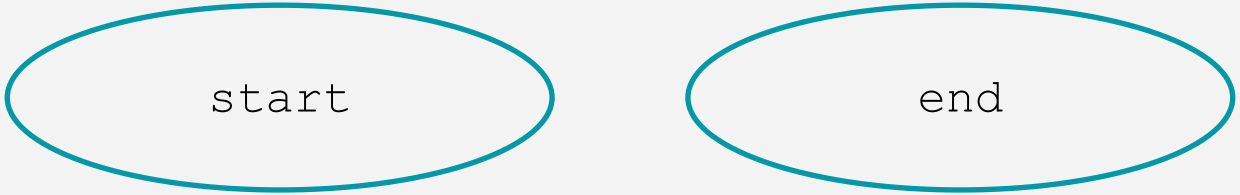
What is a Flow Chart?

A Flow Chart is a graphical representation of an algorithm. Programmers often use it as a program-planning tool to solve a problem. It makes use of symbols which are connected among them to indicate the flow of information and processing.

Flow Charts – Terminal

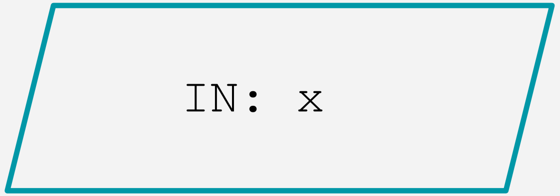
The oval symbol indicates **`start`**, **`stop`**, and **`pause/halt`** in a program's logic flow. A **`pause/halt`** is generally used in a program logic under some error conditions.

Terminal is the first and last symbols in the flowchart.

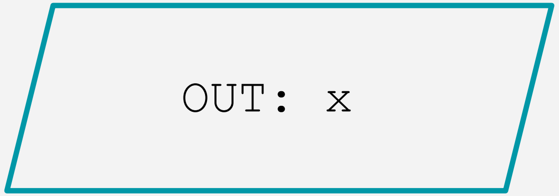


Flow Charts – Input/Output

A parallelogram denotes any function of input/output type. Program instructions that take input from input devices and display output on output devices are indicated with parallelogram in a flow chart.



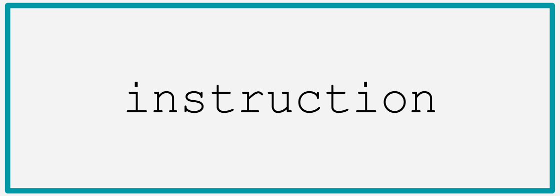
In this example, the value of the variable **x** is taken as input while the user is typing on the keyboard.



In this example, the value of the variable **x** is given as output and printed on screen.

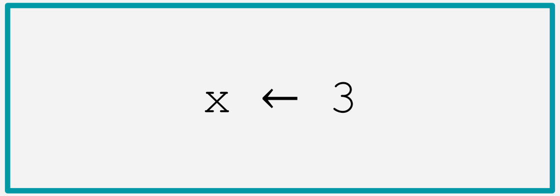
Flow Charts – Processing

A box represents arithmetic instructions. All arithmetic processes such as adding, subtracting, multiplication and division are indicated by action or process symbol.



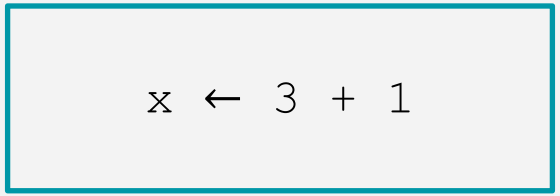
Flow Charts – Processing

A box represents arithmetic instructions. All arithmetic processes such as adding, subtracting, multiplication and division are indicated by action or process symbol.



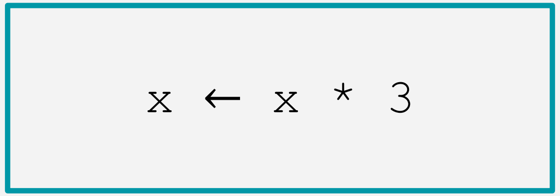
Flow Charts – Processing

A box represents arithmetic instructions. All arithmetic processes such as adding, subtracting, multiplication and division are indicated by action or process symbol.



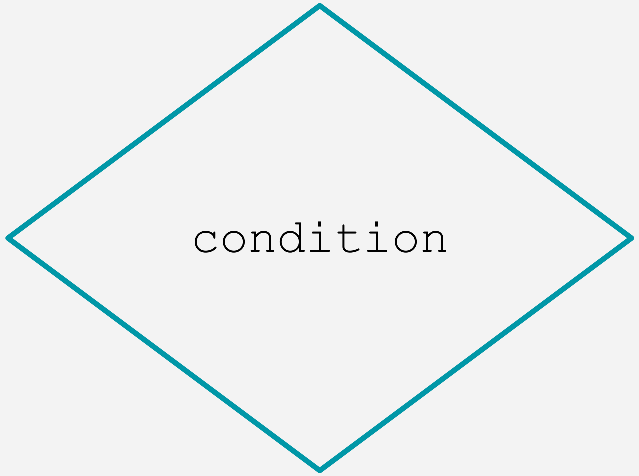
Flow Charts – Processing

A box represents arithmetic instructions. All arithmetic processes such as adding, subtracting, multiplication and division are indicated by action or process symbol.



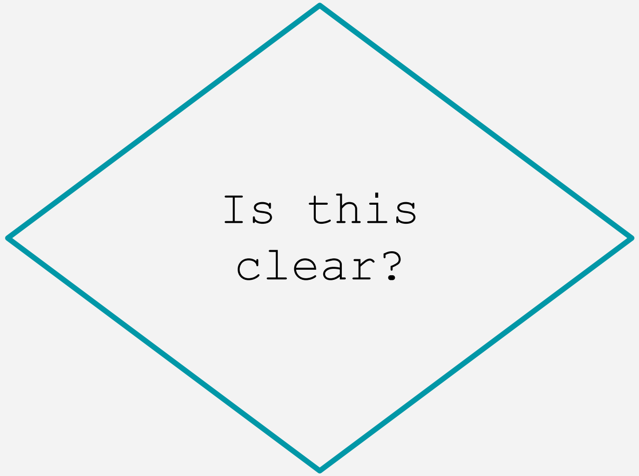
Flow Charts – Decision/Condition

A diamond symbol represents a decision or condition point. Decisions are Boolean operations such as yes/no questions.



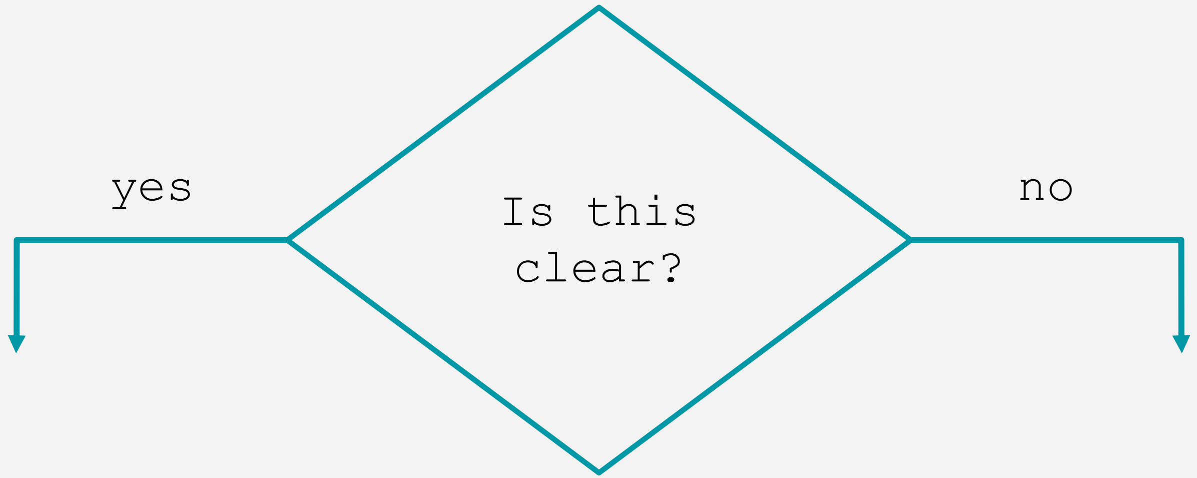
Flow Charts – Decision/Condition

A diamond symbol represents a decision or condition point. Decisions are Boolean operations such as yes/no questions.



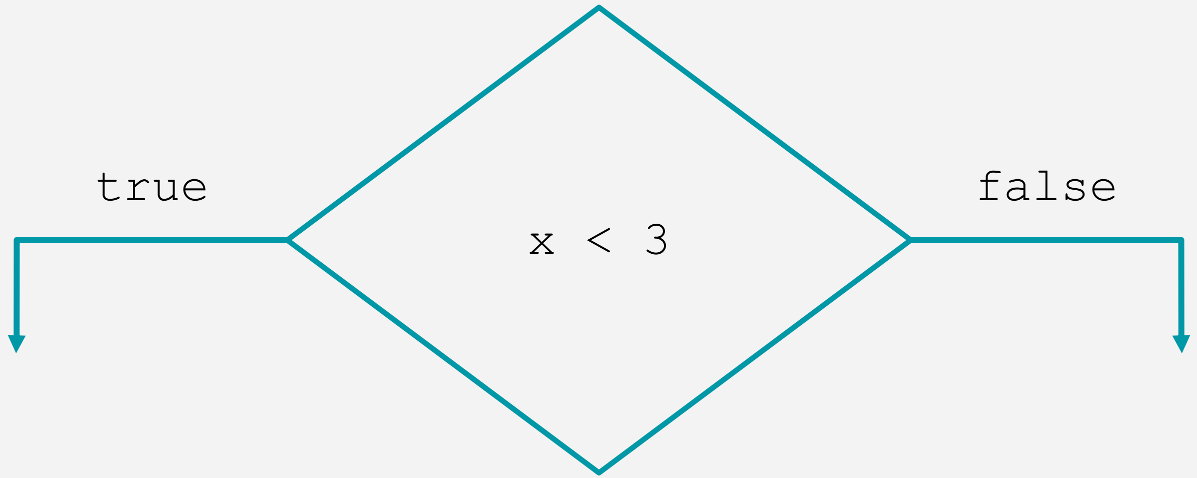
Flow Charts – Decision/Condition

A diamond symbol represents a decision or condition point. Decisions are Boolean operations such as yes/no questions.



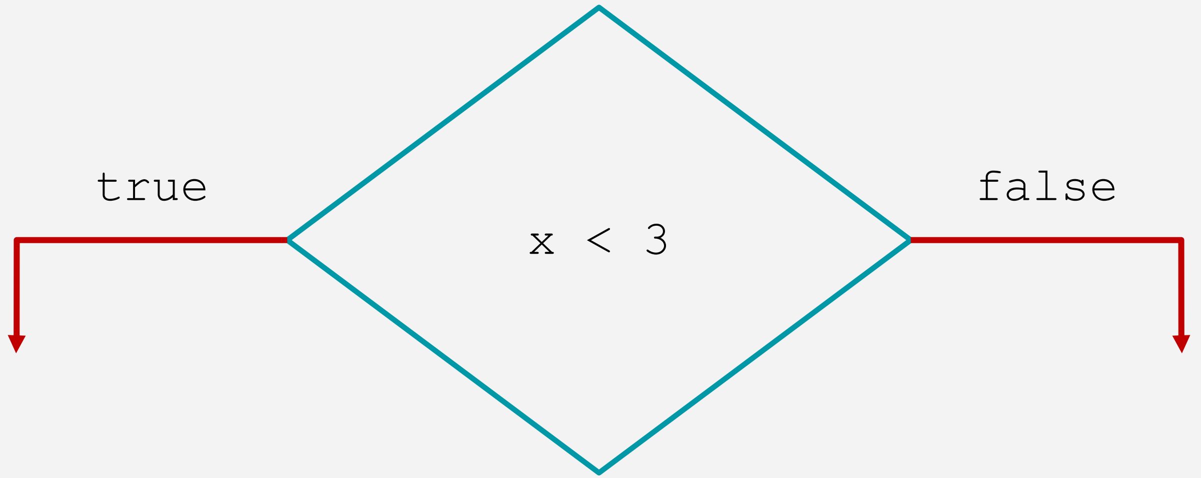
Flow Charts – Decision/Condition

A diamond symbol represents a decision or condition point. Decisions are Boolean operations such as yes/no questions.



Flow Charts – Lines and Arrows

Lines indicate the exact sequence in which instructions are executed. Arrows represent the direction of flow of control and relationship among different symbols of flowchart.



Flow Charts – Connectors

Whenever flow chart becomes complex or it spreads over more than one page, it is useful to use connectors to avoid any confusions. It is represented by a circle.