Institute of Technology Development of Canada

# Problem Solving & Analytical Thinking

ICR110
Fall Semester 2020

## Understanding algorithms

## What is an algorithm?

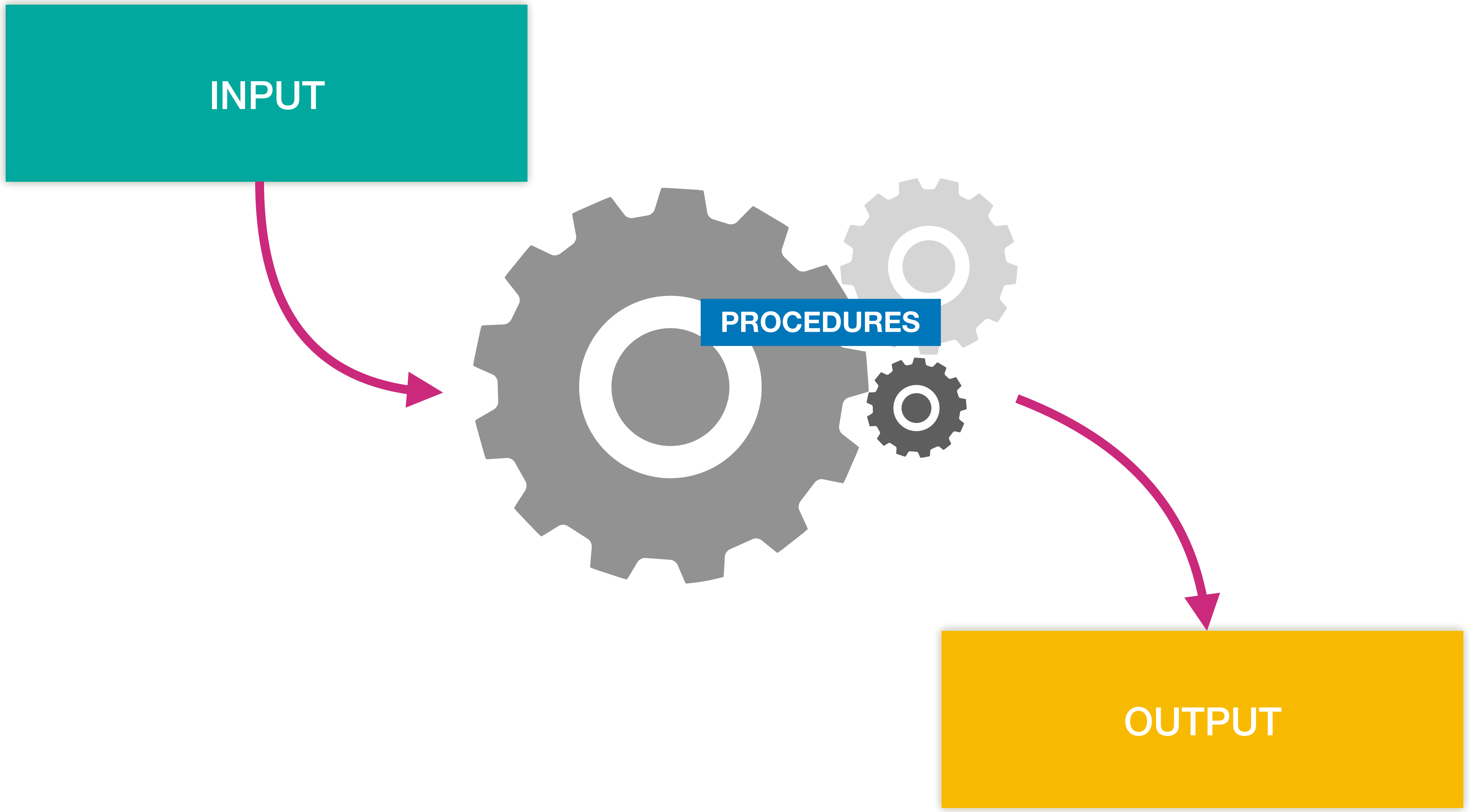
- An **algorithm** is a set of rules or process to be followed in calculations or other problem-solving operations.
- Or simply: **a set of steps to perform a task**.
- Basically, any sort of logical operation can be called algorithm.

## What is an algorithm for a computer?

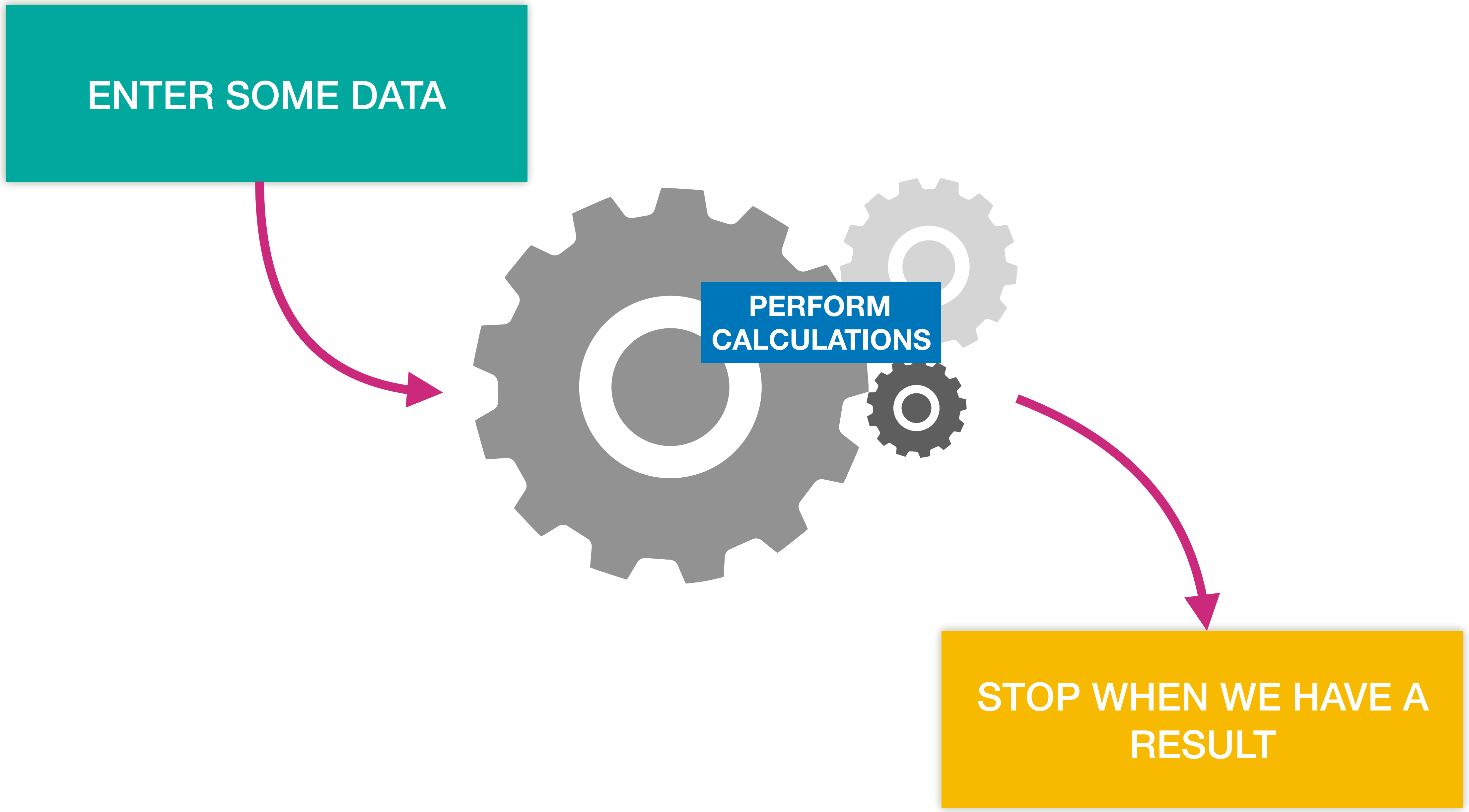
- Again, for a computer an algorithm is a set of steps to follow in order to accomplish a task.
- A computer is extremely good in performing what it’s been said, but it needs somebody who instructs it - a *programmer.*
- Basically, algorithms put *science* into computer science.
- A programmed computer can solve a problem through an algorithm if, and **only if**, this problem can be solved by the *Turing machine* as well.

- Abdullah Muhammad bin Musa al-**Khwarizmi** during the 9th century.
- A Persian scientist, astronomer and mathematician considered the *father of Algebra.*
- He invented the word *algorithm* - while the concept itself can already be found in the Ancient Greek culture.

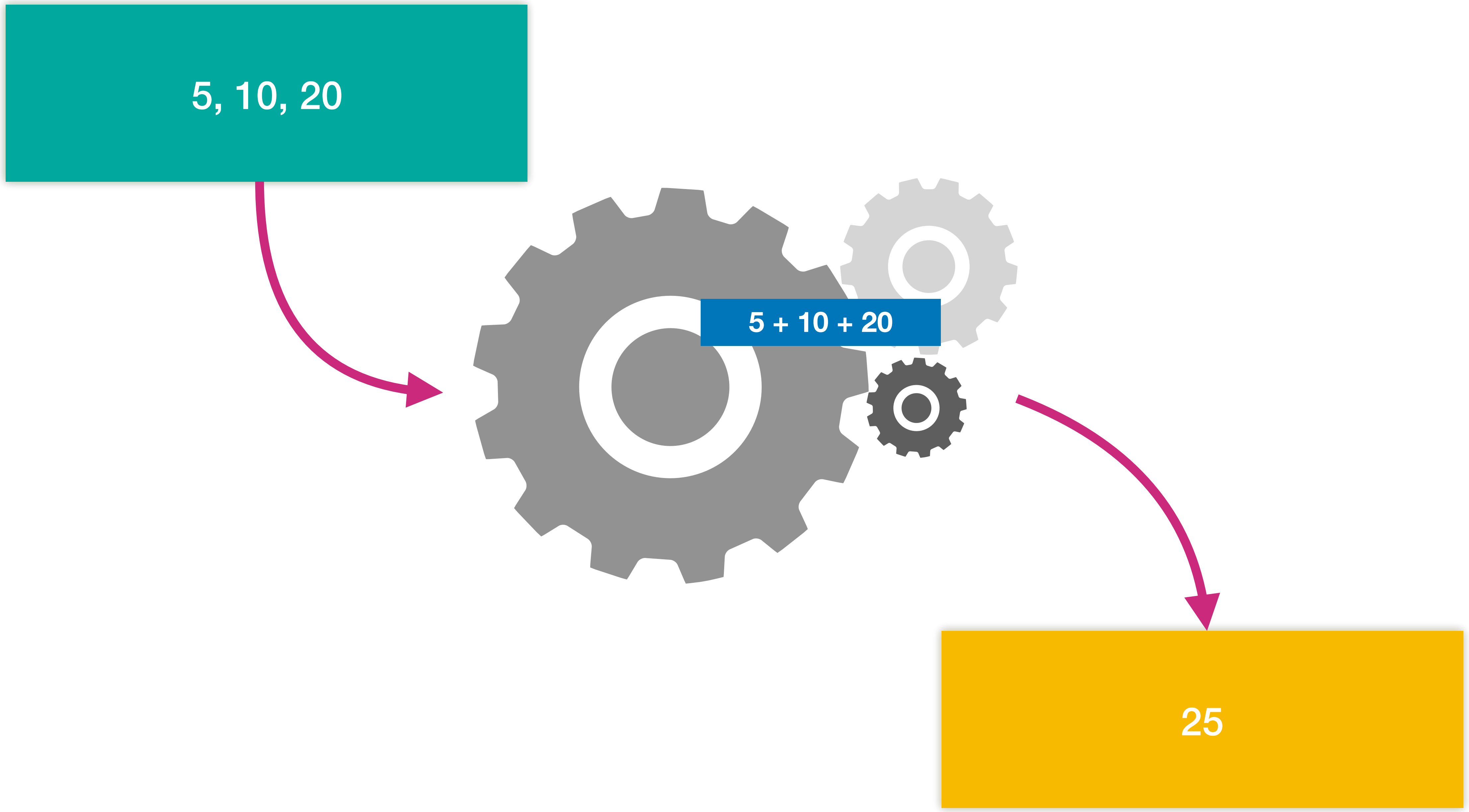
## What is an algorithm for a computer?



## What is an algorithm for a computer?



## What is an algorithm for a computer?



## Algorithms examples

**CALCULATION EXAMPLE**

- Take 2 numbers. One is **A** the other one is **B**.
- Take A and multiply by 5. **The value of A has now changed.**
- Take B and multiply by 10. **The value of B has now changed.**
- Add A and B together. The result is **sum.**
- Divide sum by 5. **The value of sum has now changed.**
- Subtract A from sum. **The value of sum has now changed.**
- Multiply A by 2. **The value of A has now changed.**
- Add A to sum.
- Divide sum by 2. **The value of sum has now changed.**
- Print sum on the board.

- Algorithms are **everywhere** and we use them **everyday** without even knowing.
- Everything we do, that has to do with logic, is an algorithm:
- Doing the laundry. Grocery shopping. Tooth brushing. Dressing up. The list can go on forever.

## How can I buy milk from a store?

CHOOSE FAT PERCENTAGE

CHECK THE EXPIRY DATE

PAY FOR IT

TAKE YOUR RECEIPT

FIND THE FRIDGE

ASK IN WHICH FRIDGE IS MILK

CHOOSE THE SIZE

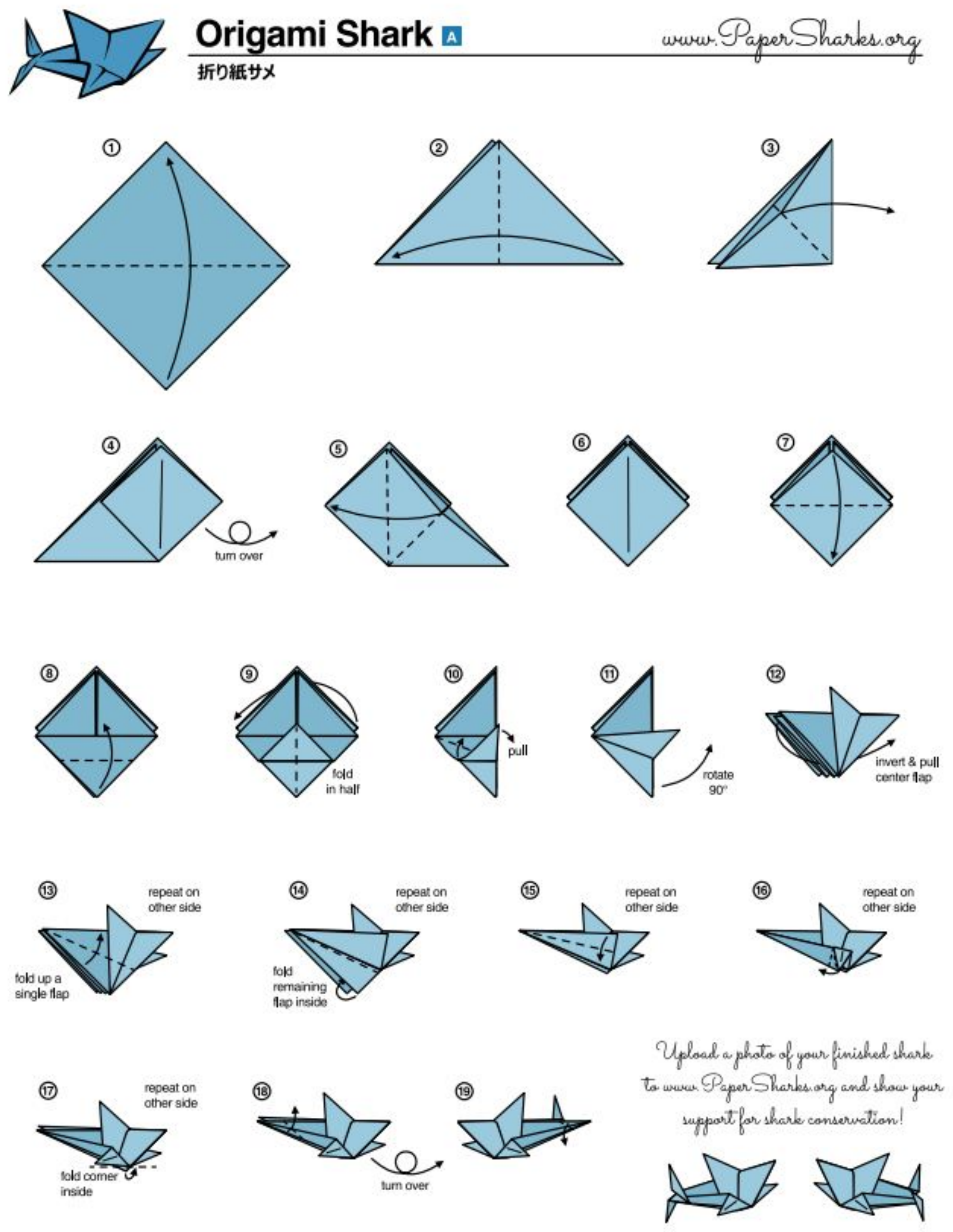
GO TO CHECKOUT

GO TO A STORE

EXIT THE STORE

CHECK IF STORE IS OPEN

BRING THE WALLET WITH YOU



- As we have seen, not just the **steps** are essentials, but also **their order**.
- When you brush your teeth you first put the toothpaste and then you brush them, not the opposite.
- When you make origami, the correct order is essential.

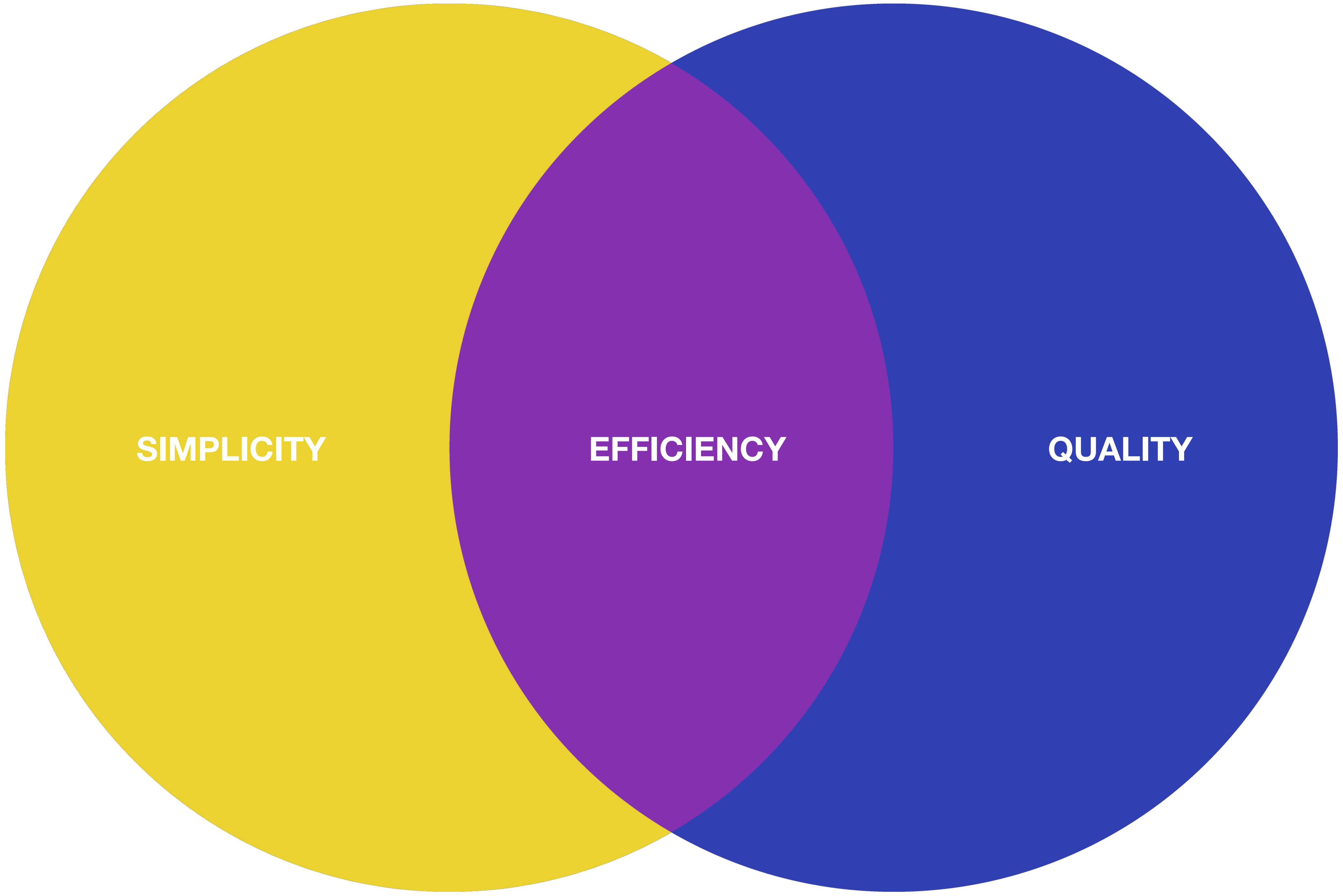
## Keep it good!

- **QUALITY**: Some tasks might have one or a few ways to be performed. Some others, a lot more.
- Finding the best one, which means the **good** and **fast** one, affect the quality of an algorithm.
- For example: if you have to come to school, how many options you have? Public transportations, car, bicycle, walking, taxi, skateboard, etc.
- A taxi might be faster but it’s for sure more expensive than the public transportations. So, maybe the bus is the perfect compromise between cost and travel time.

## Keep it simple!

- **SIMPLICITY:** The clarity of the instructions (steps) we write affect many factors as well: for example, the readability of the code itself.
- If the code is clean and simple, it’s easier for the programmer to maintain it. Plus, 90% of the time is easier for the computer to execute it as well.

## Simplicity + quality = efficiency



## What is code efficiency?

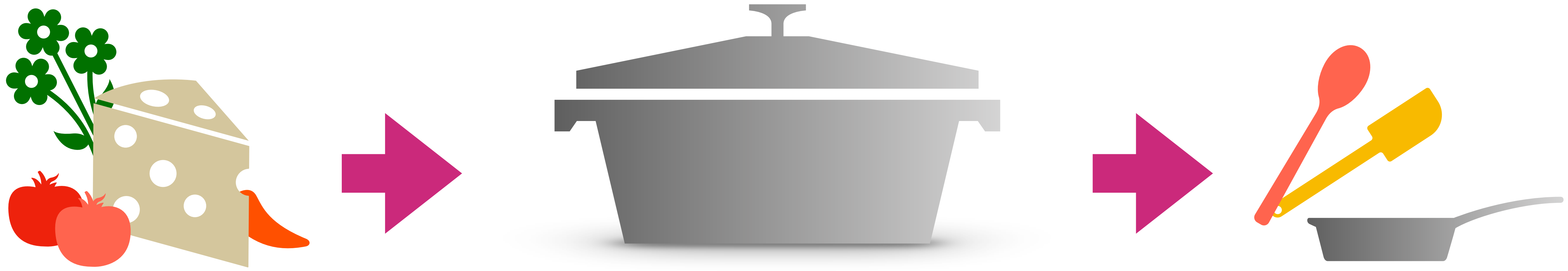
- There can be lots of different answers on what is code efficiency - depending on what we mean with *efficient*.
- Generically speaking, an efficient code is:
- Fast.
- Clean.
- Light.

## Let’s make some examples…

- If I want to prepare a plate of pasta, there’s some procedures which always stay the same - regardless of the kind of dish I am going to do. For example, the steps for **cooking the pasta itself**:

1. Fill a pot with cold water.
2. Put the pot on the flame and turn it on.
3. When the water is boiling, add the coarse salt.
4. Then, add the pasta.
5. When the pasta is *al dente*, drip the pasta with a colander.

## Let’s make some examples…



DO SOME STEPS FOR PREPARE THE SEASONING

ALWAYS CHANGING, DEPENDING BY THE RECIPE

COOK THE PASTA

THESE STEPS ARE CONSTANT REGARDLESS OF THE RECIPE

PERFORM FINAL STEPS

ALWAYS CHANGING, DEPENDING BY THE RECIPE

## Let’s make some examples…

- If we have to write the code for these recipes, **it does not make sense to repeat the steps** for cooking the pasta because they never change.
- The operations that have to be performed are always the same, so we can just write them **once** and tell to the computer to perform them when needed.
- This make the code more **efficient**.

## Algorithms examples

**COOK SOMETHING FOR ME!**

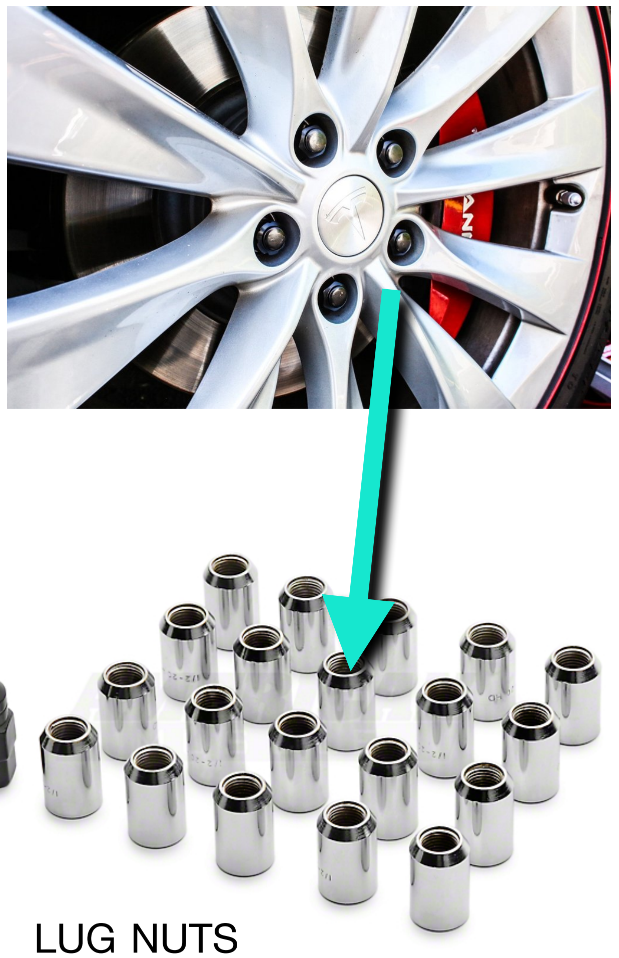
- Choose a simple recipe you know and write the algorithm for it.

# Algorithms examples

**HOW CAN I FIX A FLAT TIRE?**

TRUNK

JACK



LUG NUTS

SPARE TIRE

WRENCH

## Algorithms examples

**HOW DO YOU COME TO SCHOOL?**

- Describe step by step everything you do in order to come to school.

## Algorithms examples

**HOW DO YOU COME TO SCHOOL?**

- Exit the house and go to the bus stop.
- Wait for bus 23.
- Once the bus has arrived, take the bus.
- Once it reaches Yaletown Roundhouse, get off the bus and go to the skytrain station and wait for the skytrain for Waterfront.
- Once the skytrain has arrived, take the skytrain.
- When the skytrain makes 2 stops (Waterfront), get off the skytrain.
- Exit the skytrain station and, if the school is open, enter the school.

Thank You,