Practice 14: Iterating Through Lists

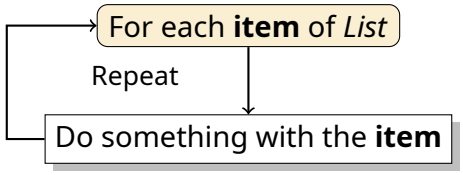
The ”For Each” loop and data processing
Module 4: Composite Data

How to read the entire cabinet?
Imagine you have a list of 100 prices and you want to know their sum. You are not going to add `item 1 + item 2...` by hand! For that, we use loops. Iterating through a list means visiting every drawer one by one automatically.

Key Concepts

1. The ”For Each” block

This is the most powerful block for working with lists in Snap!. The program manages how many items there are and knows when it reaches the end:



- **The ”Item” Variable:** It is a ”temporary” variable. In the first turn, it holds the content of drawer 1; in the second, drawer 2, and so on.

2. Visual Iteration Scheme

Imagine a list of fruits: `[apple, pear, banana]`.

Turn (Iteration)	Value of ”item”	Program Action
1st	apple	Say ”apple” for 2 sec.
2nd	pear	Say ”pear” for 2 sec.
3rd	banana	Say ”banana” for 2 sec.

THE CHALLENGE: Roll Call

Let's program a character to act like a teacher taking attendance. It must greet every student stored in our list.

Instructions in Snap!:

1. Create a list named `Class` and add 4 or 5 names (e.g., Ana, Luis, Marta, Brais).
2. Use the `for each (item) of (Class)` block.
3. Inside the loop, the sprite should `say` a greeting.
4. **Pro Tip:** Use the `join [Hello ] (item)` block so the greeting includes the student's name.

Pseudocode: The Summation Algorithm

```
Algorithm SumPrices
   List Prices <- [10, 5, 20]
   Variable Total <- 0

   For Each p in Prices:
      Total <- Total + p // Accumulating the value

   Print "The total bill is: ", Total
EndAlgorithm
```