Practice 3: Unit Converter

Handling mathematical data types and rounding
Programming Course in Snap! and Pseudocode

Data Types and Precision
Computers are excellent calculators, but when performing division or multiplying by fractions, they often generate numbers with many decimal places (e.g., 15.6666667). To make our programs professional and easy to read, we must learn to **round** those values.

Objectives of this practice

- Handle numerical data types (integers and decimals/reals).
- Understand the concept of the *conversion constant* (multiplier factors).
- Apply the **rounding** function to improve the user interface.
- Nest complex mathematical operations.

Step by Step: Snap! vs Pseudocode

1. The Conversion Constant

To change from one unit to another, we multiply by a factor. For example, for Euros to Dollars (1.08):

- **In Snap!:** Use the multiplication block: `(euros * 1.08)`.
- **In Pseudocode:** `dollars <- euros * 1.08`

2. Manipulation: Rounding

To avoid numbers like 10.799999:

- **In Snap!:** In the **Operators** category, use the `round ( )` block. You must place the entire mathematical operation inside this block.
- **In Pseudocode:** Use the built-in function `ROUND()`.
- *Example:* `clean_dollars <- ROUND(dollars)`

3. Output Formatting

The user needs to understand the final result by combining text and variables:

- **In Snap!:** Use `join [The value is ] [clean_dollars]`.
- **In Pseudocode:** `Write "The value is ", clean_dollars`

THE CHALLENGE: Travel Calculator (Km to Miles)

Create a tool for a tourist. Your program must convert Kilometers to Miles and round the result so it has no decimal places. *(Formula:* 1 *km* $= 0.62$ *miles)*.

PART A: In Snap!

1. Ask for the distance in **Kilometers** and store it in a variable called `km`.
2. Create a variable called `miles`.
3. Set `miles` to the result of **rounding** the multiplication of `km` by 0.62.
4. Display a final message joining text and the rounded result.

PART B: In Pseudocode Write the corresponding algorithm in your notebook:

```
Algorithm TravelConverter
   Define km, miles As Real
   Write "How many kilometers are you going to travel?"
   Read km
   // We apply the formula and round to the nearest integer
   miles <- ROUND(km * 0.62)
   Write "Your destination is about ", miles, " miles away."
EndAlgorithm
```

Did you know...?

In most programming languages like Python, C, or Java, rounding is a native function. In Python, the `round()` function is used:

```
km = float(input("Kilometers to travel: "))
miles_rounded = round(km * 0.62)
print(f"Your destination is {miles_rounded} miles away.")
```