Functions

Functions are useful: they avoid copying and pasting lines of code, when we want to use the same operations at different locations in our program. For example, it was useful to teach Reeborg how to turn right, as in Listing 1, instead of writing three `turn_left()` each time.

```
1 def turn_right():
2     repeat 3:
3         turn_left()
```

Listing 1: Sample code for a right turn with Reeborg.

In a cook book, it is convenient to describe only once how to make a "béchamel" sauce, and to refer to this description every time it is needed, instead of rewriting the recipe each time!

A function can take inputs or not, and can return values, or not. The addition function given in Listing 2 has two inputs (named a and b) and returns the addition of them ($a + b$). The `turn_right()` function has no input and no return value.

It is important to note the type of the input variables, to understand what can be done with those input variables. It is also important to specify the type of the function, which is the type of the values returned by the function. Some programming languages automatically type the input variables and the functions, some ask the programmer to specify the types.

```
1 def addition(a, b):
2     return a + b
```

Listing 2: Sample code for the addition of two integers.

Exercise 0

What happens if you call `addition("Hel", "lo.")`? Why?

Exercise 1 — Equation solving

Download the skeleton code given in Listing 3 that solves a linear equation, then complete it. This function takes two numbers a and b as parameters, writes in the terminal the string `Set of solutions to ... x + ... = 0` (where ... are replaced by the numbers a and b), and then the set of solutions of the equation $ax + b = 0$.

```
1 def solve_linear(a, b):
2     print("Set of solutions to " + str(a) + "x + " + str(b) + " = 0")
3     if (...):
4         print("S = {}")
5     elif (...):
6         print("S = all real numbers")
7     else:
8         solution = ...
9         print("S = {" + str(solution) + "}")
```

Listing 3: Skeleton code for first-order equation solving.
Download it at: http://www.barsamian.am/2023-2024/S6ICTA/TP6_Linear_equation.py

- the call `solve_linear(11, 3.2)` must print `Set of solutions to 11x + 3.2 = 0` and then `S = {-0.290909091}`.
- the call `solve_linear(0, 3.2)` must print `Set of solutions to 0x + 3.2 = 0` and then `S = {}`.
- the call `solve_linear(0, 0)` must print `Set of solutions to 0x + 0 = 0` and then `S = all real numbers`.

Exercise 2 — The minimum

Write a function "min2numbers" that takes two numbers as input and returns the minimum of them.

- the call `min2numbers(2, 3)` must return `2`
- the call `min2numbers(3.14, 3.14)` must return `3.14`

Remark: in `Python` this function already exists: it is the function `min`. You're not allowed to use it in your solution.

Exercise 3 — Strings

For this exercise, it is useful to know some `Python` syntax that manipulate strings:

- `"I love ICT."[-1]` gives `'.'`
- `"I love ICT 2 periods."[2:6]` gives `"love"`
- `"Hi " + "there."` gives `"Hi there."`
- `"well " * 4` gives `"well well well well "`
- `"I love ICT." > "I love Nature."` gives `False` (alphabetical order)

1. Write a function "accumulate" that takes as inputs a string s and an integer n, and returns a string containing n times the string s, separated by spaces.

   - the call `accumulate("ATGC", 4)` must return `"ATGC ATGC ATGC ATGC"`.

2. Write a function "correct_sentence" that takes a string s as input and return a boolean: it will return `True` if and only if s starts with an upper case letter, then contains only spaces or lower case letters, then ends with a point.

   For this question, it is a good idea to:

   (a) if the 1st character is not an upper case letter (between `'A'` and `'Z'`), return `False` ;
   (b) write a loop that will look at each character in the string (`for c in s:`), and return `False` if we found an incorrect character ;
   (c) at the end, if we are still there, it means there was no problem, so we return `True`.

   - the call `correct_sentence("Hel lo.")` must return `True`
   - the call `correct_sentence("School3.")` must return `False`

3. Write a function "points" that takes as input a string s (for this question, we can assume that any s given as input only contains letters and spaces), and return the number of points this string would give if played in a Scrabble® game, without bonus. The points of each letter are those of a French version, given by the following picture:

   - the call `points("ABCDEF")` must return `14`
   - the call `points("ZX")` must return `20`