FUNCTIONS

It is sometimes useful to use functions, as we saw with Reeborg. This avoids copying and pasting lines of code, when we want to use the same operations at different locations in our program. For example, it was really useful to teach Reeborg how to turn right, as in Listing 1, and call it every time we need it, 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 (if it does not return any value, we say that it produces only "side effects"). For example, a function that adds two integers has two inputs (that we may name a and b) and returns the addition of them ($a+b$). Its Python code is given in Listing 2. 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.

To define a function in `Python`, we write the keyword `def` followed by the name we want to give to the function, followed by the name of the input variables, between parentheses. Then, we indent the body of the function, which is a sequence of operations that the function will compute, and we finish by the keyword `return` followed by the value the function must return.

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

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

Remark: this function can also be called with two strings as input, and will then return the concatenation of those two strings (*i.e.*, `addition("Hel", "lo.")` will return `"Hello."`. `Python` does not ask us to type input variables, and will only generate an error upon calling this function if it is impossible to use the built-in "+" operation (this operation is ambiguous!).

In the following exercises, some examples are given, but the functions must of course give correct results on other inputs!

Exercise 1 — Equation solving

Write a function `solve_linear` that takes two numbers a and b as parameters, writes in the terminal the string `Set of solutions to the equation ... x + ... = 0` (where ... are replaced by the numbers a and b), and then the set of solutions of the equation $ax + b = 0$.

- the call `solve_linear(11, 3.2)` must print `Set of solutions to the equation 11x + 3.2 = 0` and then `S = {-0.290909091}`.
- the call `solve_linear(0, 3.2)` must print `Set of solutions to the equation 0x + 3.2 = 0` and then `S = {}`.
- the call `solve_linear(0, 0)` must print `Set of solutions to the equation 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`. Of course, you can't use it in your solution.

Exercise 3 — Strings

For this exercise, it is useful to remind the `Python` functions seen in the Work n°4. You can use them, but not other `Python` functions that manipulate strings:

http://www.barsamian.am/2022-2023/S6ICTA/TP4_Python_first_steps.pdf

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.

   - 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`