5. Tutorial/practical exercises

Exercise 1

1 – Write the algorithm and the Python program to compute the greatest common divisor of two integers provided by the user.
2 – Give the flowchart of your program.

Exercise 2

1 – Write the algorithm and the Python program to decide if an integer is a prime number.
2 – Give the flowchart of your program.

Exercise 3

Consider a program that reads real numbers corresponding to the marks of students in an exam (between 0 and 20). The input ends when the user enters the word **end**. The program computes the average of the numbers.

Give the algorithm of the game, then its Python program.

Exercise 4

The goal of this exercise is to design an algorithm to compute the square root of number, with a given error ε. In reality, the square root of a number `a` is simply computed by `a**2` (in Python). However, we suppose that we don't have the right to use powers. We consider two algorithms:

A – Consider an interval $[u, v]$ and the function $f(x) = x^2 - a$. We suppose that u and v are chosen such that $f(u).f(v) \leq 0$. The algorithm computes the center of the interval $w = \frac{u+v}{2}$, then test if $f(u).f(c) \leq 0$. In this case, the same processing is repeated on the interval $[u, w]$, otherwise it is repeated on $[w, v]$. The algorithm continues until $|u - v| < \varepsilon$. The algorithm starts with the interval $[0, 2a]$.
B – We build the following series: $x_{n+1} = \frac{x_n^2 + a}{2x_n}$ with $x_0 = 1$. The computation stops when $|x_{n+1} - x_n| < \varepsilon$ (a given error).

1 – Write the Python program of each algorithm.
2 – For each algorithm, the program should print the number of iterations. Compare between the two algorithms. What do you remark?

Exercise 5

Write a Python program that prompts the user to enter an integer n (let's say it is 6). It then make an output as the following:

```
          1
        2 1 2
      3 2 1 2 3
    4 3 2 1 2 3 4
  5 4 3 2 1 2 3 4 5
6 5 4 3 2 1 2 3 4 5 6
```

Exercise 6

The constant $e = 2.71...$ can be numerically computed thanks to the series: $\sum_{i=0}^{\infty} \frac{1}{i!}$.
1 – Write a simple Python program that gives an approximation of e by calculating the sum with a specified error.
2 – If the sum is computed up to an integer n, how many iterations are performed by your code?
3 – Build a more optimal version of the code.