Overview of what we have seen so far

1 Level 0: Basic applications

1.1 Variable handling (condition)

What are the values of the variables of Listing 1 at the end of the program? You can check the correct answer with `Python` but this is not required.

```
a=22
b=5
if (a < 50):
    b = b * 2 + a
    a += 20
if (a < 10):
    b = b * 3 + a
elif (a < 30):
    b = b * 4 + a
else:
    b = a
```

Listing 1: A sample program.

1.2 Bug handling (syntax)

There are some bugs in Listing 2. Please correct this `Python` program so that it prints 13 times the text `9 * 8 = 72`.

```
for loop in range(13)
print("9 * 8 = 72)
```

Listing 2: Syntax error.

1.3 Bug handling (loop iterations)

The `Python` program in Listing 3 was supposed to print the square of each number from 10 to 15 but it does not. Please correct it.

```
for i in range(10, 15):
    square = i**2 #Computes the square of i
    print("The square of " + str(i) + " is " + str(square))
```

Listing 3: Incorrect loop bounds.

1.4 Absolute value (condition)

The absolute value $|x|$ of a number x is the number x when $x \geq 0$ and $-x$ when $x < 0$.

Write a function `abso` that takes as input a real x and outputs the number $|x|$.

The goal of this exercise is to use only conditions, assignments and the basic mathematics operators $(+, -, \times, \div)$, without using anything else provided by `Python`. In particular, I don't want you to use the function `abs` from the `math` library (available if you start your program with `from math import *`) that would answer the question : in `Python`, `math.abs(x)` is precisely $|x|$.

1.5 Insurance deductible (condition)

An insurance company reimburses their customers when a damage occurs, but there is always an amount of money, the deductible, that must be paid by the customer. This deductible is 10% of the total amount of the damage, but is never less than 15€ and never more than 500€.

Write a sequence of instructions that asks the user to give the total amount of the damage, and then prints the amount the insurance will reimburse, and the amount of the deductible.

1.6 Exponentiation (loop)

The exponentiation x^n of a number x to an integer power $n \geq 1$ is the number $x^n = \underbrace{x \times x \cdots \times x \times x}_{n \text{ times}}$.

Write a function `expo` that takes as input a real x and an integer n and outputs the number x^n. The function must first ensure that $n \geq 1$ and print an error message if it is not the case. To conduct the multiplications, it is possible to use a temporary variable (a, for example), and a loop. At the beginning of the function, a may be set to 1, and at each iteration of the loop, a may be multiplied by x.

The goal of this exercise is to use only loops, assignments and the basic mathematics operators $(+, -, \times, \div)$, without using anything else provided by `Python`. In particular, I don't want you to use the operator `**` that would answer the question : in `Python`, `x**n` is precisely x^n.

Remark: We can of course raise a number to the exponent 0, and $x^0 = 1$. But we won't bother with that in this exercise.

2 Level 1

2.1 Administration opening hours (conditions)

We want to go to the administration. The opening hours are :

- Monday-Friday: 8h–13h and 14h-17h
- Saturday: 8h–13h

Write a sequence of instructions that asks the user the day of the week and the hour (between 0h and 24h), and prints a message indicating whether the administration is open or not.

Tip: It is very likely that the user will type indifferently Monday or monday, etc. So, a good idea is to compare strings using the `.lower()` function we have seen in work n°5.

2.2 Factorial (loop)

The factorial $n!$ of a integer $n \geq 1$ is the number $n! = 1 \times 2 \times 3 \times 4 \times \cdots \times (n-1) \times n$.

Write a function `fact` that takes as input an integer n and outputs the number $n!$. The function must first ensure that $n \geq 1$ and print an error message if it is not the case. To conduct the multiplications, it is possible to use a temporary variable (a, for example), and a loop over a second temporary variable i. At the beginning of the function, a may be set to 1, and at each iteration of the loop, a may be multiplied by i.

The goal of this exercise is to use only loops, assignments and the basic mathematics operators $(+, -, \times, \div)$, without using anything else provided by `Python`. In particular, I don't want you to use the function `factorial` from the `math` library (available if you start your program with `from math import *`) that would answer the question : in `Python`, `math.factorial(n)` is precisely $n!$.

Remark: The factorial of 0 is also defined, and $0! = 1$. But we won't bother with that in this exercise.

2.3 Give the change

We want to know what is the minimal amount of (bills and coins) needed for any amount of money. In this exercise, we will only focus on the bills: the coins needed, if any, will not be taken into account. We will use only the standard bills of value 5€ ; 10€ ; 20€ ; 50€ ; 100€.

Write a function `nb_bills` that takes as input a number *amount*, and returns the number of bills (of 5€ ; 10€ ; 20€ ; 50€ ; 100€) needed if we use a minimal amount of (bills and coins) to come up to this amount. Before returning the value, the function must print the number of bills of each kind that are needed.

Example : the call `nb_bills(1193)` will print

```
Number of 100€ bills : 11.
Number of 50€ bills : 1.
Number of 20€ bills : 2.
```

And will return the value 14 (to give the remaining 3€ we would use 2 coins, but we just discard everything that remains after we gave the 5€ bill(s)).

3 Level 2

3.1 Throwing a die (condition)

The `random` function from the `random` library (this function is available if you start your program with `from random import *`) outputs a number uniformly generated in $[0, 1)$. Example:

```
>>> from random import *
>>> random()
0.9811896401527802
>>> random()
0.9889103424621456
```

Write a function that uses this function and simulates the throwing of a well-equilibrated six-sided die (it has to output a random uniform integer in $\{1, 2, 3, 4, 5, 6\}$).

The goal of this exercise is to use only the function `random`, assignments and the basic mathematics operators $(+, -, \times, \div)$, without using anything else provided by `Python`. In particular, I don't want you to use the functions `randint`, `randrange`, `choice` or `uniform` from the `random` library that would facilitate the answer (`randint(1, 6)` or `randrange(1, 7)` or `choice(range(1, 7))` or `int(uniform(1, 7))` would answer the question).

3.2 File names (condition)

You have the responsability to organize a set of files. Each file has a given date (for example, this document has the date "October 13th, 2023"), and a name (for example, this document has the name `TP7_Overview.pdf`). You must write a function `rename_file` that takes as input a string (the original name) and three numbers (day, month, year) and returns a string which is the new name of the file, of the form `date_original_name`. The constraint on the new names is the following: given two documents a and b, the alphabetical order of their new names must be the same as the order of their dates.

For instance, renaming this document to `October_13th_2023_TP7_Overview.pdf` will not work, because, following the same logic, you would then rename the document from Work 4 `September_22nd_2023_TP4_Guess_the_number.pdf`, and in alphabetical order, `O` is before `S`, which implies that the order would be

- First `October_13th_2023_TP7_Overview.pdf`,
- then `September_22nd_2023_TP4_Guess_the_number.pdf`

whereas the date "October 13th, 2023" is after the date "September 22nd, 2023".