1 Variables and functions

The **variables** are the memory of the program. We will store values inside them to use them later. The **function** are pieces of the program that we want to use several times without copy-pasting them. They can also be parameterized to be used in different cases with the same behavior (but not the same result !).

Exercise 1 : Let's start

Create a variable of each following types :

— integer

— decimal number

— string

Display them in Python console by using the **print** function. Verify their types by displaying them with the **type** and the **print** functions.

Exercise 2 : Type conversion

Sometimes, you want to change the type of a variable to fit the requirements of your program. The **int** function will convert the value passed as a parameter to an integer. The value can be either a number (e.g. 4.2 will be converted to 4) or a string (e.g. "8" will be converted to 8). The **float** function will do the same but will return a decimal number. By using these two functions, compute the following calculation

a) "6" - 2

b) "18.2" + 4.6

You will notice something strange with the second calculation ; that is due to the lack of precision with the floating-point representation of numbers. Because of it, you should never try an equality between decimal numbers !

Exercise 3 : Conditional structures (if)

a) Create a function that display "Hello World !" in the Python console and call it.

b) Add a parameter to this function, and modify the **print** to display the value passed as a parameter after the "Hello World !"

c) Add a conditional structure (an **if**) to change the display of the function depending on the value passed as parameter. If this value is equals to 0, you will **print** "Bonjour le monde !", in all other cases, you will **print** "Hello World !"

d) Modify this structure to display your greetings according to a parameter that is either "fr" or "en". If the value passed as a parameter is none of them, **print** "Error".

Exercise 4 : Mathematical functions

a) Create the **circle_perimeter** function which takes as an argument the radius of the circle and returns its perimeter.

b) Do the same for the **circle_area** function.

c) Define the two functions **radians_to_degres** and **degres_to_radians** that convert angle notation from one to the other.

d) Define the two functions **celsius_to_fahrenheit** and **fahrenheit_to_celsius** that convert temperature measure from one to the other.

Exercise 5 : Calculator

a) Create the **calculator** function that takes as parameters : a a first number, *operator* a string corresponding to the operation to proceed, and b the second number.

b) Inside the body of this function, use a conditional structure to return $a + b$ when the operator is equals to "+".

c) Test your function by calling it with the value 2, "+", and 3, and by printing the return value.

d) Inside the body of your function, add the other operator ("-", "*", "/") and test them

---

Exercise 6 : Triangles

a) Create the **is_equilateral** function that takes as parameters the length of the three sides of a triangle and return **True** if they are all equals, and **False** in other cases.

b) Create the **is_isoceles** function that takes as parameters the length of the three sides of a triangle and return **True** if at least two of them are of same length, and **False** if they are all different.

c) Create the **maximum** functions that take three number as parameters and return the greatest among them.

d) Use this function to create the **right_angled_triangle** function that return if the three lengths passed as parameters correspond to a right-angled triangle (do you remember Pythagore ?)

---

2 Loops

Loops are blocks of code created to be used several consecutive times. We distinguish two kinds of loops :

— **for** : we know the number of iterations before starting the loop

— **while** : we iterate until some condition is not respected anymore

Exercise 1 : My first loops

a) Create a **for** loop that **print** the number from 1 to 100

b) Add a condition to **print** only the even number (check on Internet how to do a modulo)

c) Modify this condition to **print** both the even number and the multiple of 7

d) Modify it again to **print** both the even number which are not multiple of 7 and the multiples of 7 that are not even

---

Exercise 2 : Binomial coefficient

a) Create the **factorial** function that takes a positive integer n as a parameter and returns $n! = 1 \times ... \times n$
To do it, you need to create a variable equals to 1 and multiply it inside a loop by the index of the loop.

b) Test that **factorial**(12) is equals to 479 001 600

c) Create the **binomial** that takes two positive integers p and n and returns the number of combinations of "p among n".
The formula to compute it is :

$$\binom{n}{p} = \frac{n!}{p!(n-p)!}$$

You may use the **factorial** function that you have already defined to achieve this computation

d) Verify that **binomial**(5,8) is equals to 56

Exercise 3 : Fibonacci sequence

a) Create the **fibonacci** function that takes a positive integer n as a paramter and returns the n^{th} number of the Fibonacci sequence. The Fibonacci sequence is defined as :

$$\begin{cases} u_0 &= 0 \\ u_1 &= 1 \\ u_{n\geq 2} &= u_{(n-1)} + u_{(n-2)} \end{cases}$$

To achieve this generation, you will need to use three variables : two containing the last two Fibonacci numbers you have generated and the third one containing their sum. Then you need to shift the content of the variable before the next iteration of your loop.

b) Call this function inside a loop, and **print** the result. You should see the Fibonacci sequence : $0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55...$

---

3 Array

Arrays are the most common data structure. It is composed of a bunch of variables that are stored together in memory in a consecutive way. Each element of the array contains a value and is referenced by an index.

Exercise 1 : My first array

a) Create an array that contains the value [1, 2, 3, 4, 5] and display it
b) Display the value of the element number 2
c) Put a 0 in the element number 3 and display the array
d) According to the two previous question, what property of array do you deduce ?
e) Try to display the element number 5, and see what happen. Can you explain it ?

Exercise 2 : Array iteration

a) Copy the following line of code : **tab = [0]*10**
b) Display *tab*. What does it contain ?
c) Iterate through this array with a for-loop to store the values from 1 to 10 in this array, and then display the array
d) If you have not do it that way yet, use the **len** function in the header of your for-loop to have an adaptative code.
e) Iterate through this array again to square every element.

Exercise 3 : Sum of two arrays

a) Create three arrays containing only zero whose length is 10.
b) Iterate through the first one to fill it with multiple of 3 from 0 to 27
c) Iterate through the second one to fill it with values from -5 to 4
d) Iterate through the third and last one and fill it with the sum of the two others (element by element).

Exercise 4 : Fibonacci sequence

a) Create an array containing only zero whose length is 20.
b) Store the values 0 and 1 in the two first elements.
c) Iterate through the rest of the array and store in each element the sum of the two directly to its left.
d) Display your array, and you should see the Fibonacci sequence : 0, 1, 1, 2, 3, 5, 8, 13, 21, 34...

---

4 Nested loops

Every block of code can contain another (or multiple) block of code which can also contains blocks of code and so on... When a loop is inside another, we call that a nested loop.

Exercise 1 : Multiplication tables

a) Create a simple **for** loop that **print** the multiplication table of 5 ($5 \times 1 = 5 \ldots 5 \times 10 = 50$)

b) Create another loop, that iterate from 1 to 10 and surrounds the previous one, whose index will replace the hard-coded 5 in the nested loop.

c) Run your program : you should see all the multiplication table from 1 to 10

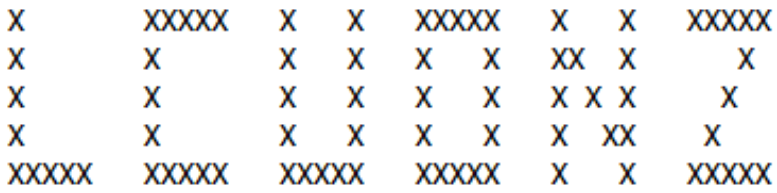
d) Create a function **multiplication_table** that takes two integers m and n as parameters and display the multiplication table from 1 to m associated with the multiples from 1 to n.

Exercise 2 : Pixel art for dummies

a) Create a nested loop

b) Inside this loop add a conditional structure that sometimes **print** a "#" and sometimes a space

c) Find the boolean condition to display every following letters (one condition by letter)



Exercise 3 : Pascal and Sierpiński

Pascal triangle is a mathematical construction where every number is the sum of the one directly above and the one in the top-left diagonal from it. The first rows of Pascal triangle are :

1
1 1
1 2 1
1 3 3 1
1 4 6 4 1

There is another way to compute every number of the Pascal triangle : the number in the p^{th} position of the n^{th} row is equal to "p among n" that you have coded in a previous exercise.

a) Create a nested loop

b) Inside it, call your **binomial** function and **print** its result

c) Replace your display of the result by a conditional structure that **print** a "#" when the result is odd and a space when it is even

d) Display the first 31 rows and admire the Sierpiński triangle

---