Python programming Lab. Work nº 2 : Recursion

Preliminaries : Please remind that teachers can be called to help you on any problem you get. Don't get stuck on an issue for too long.

We remind that there is mainly two kind of definitions of the factorial function.

• as a product, or sequence of products

$$n! = \prod_{i=1}^{n} i$$

• as a recursion (self-definition)

$$n! = \begin{cases} 1, & \text{if } n = 1\\ n \times (n-1)!, & \text{else} \end{cases}$$

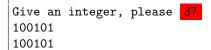
Exercise $n^{\circ}1$: Recursion

Create a module recurs.py.

1. Create two functions fact_iterative(n) which computes n! for any given n > 0, and fact_rec(n) which computes the same using recursion. Test.

```
Give an integer, please 8
8!=40320 40320
```

2. Create two functions to_binary(n) that prints the binary representation of any integer $n \ge 0$ using recursion, to_binary_string(n) that computes the string (the binary representation of n).



- **3.** Create a function fibonacci(n) that computes the Fibonacci number \mathcal{F}_n defined as $\mathcal{F}_0 = \mathcal{F}_1 = 1$ and $\mathcal{F}_n = \mathcal{F}_{n-1} + \mathcal{F}_{n-2}$ (for any n > 1).
- 4. Create a function that computes the greatest common divisor of two positive integers gcd(a,b) using Euclidean algorithm. gcd(a,0) = a, gcd(a,b) = gcd(a-b,b) (if $a \ge b$) and gcd(a,b) = gcd(b,a) (if a < b).
- 5. Create a function pascal(n) that draws the Pascal's triangle up to n using recursion. From Wikipedia :

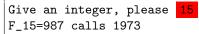
In Pascal's triangle, each number is the sum of the two numbers directly above it.

We just have to start from the bi-infinite sequence of number $^\infty 010^\infty$ and ignore the zeroes on drawing.

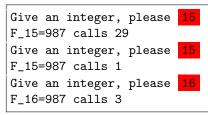


Give	an	in	tege	er,	please			ł	
					1				
				1		1			
			1		2		1		
		1		3		3		1	
	1		4		6		4		1

6. Modify fibonacci(n) so that it will return both the result and the number of calls.



7. Create a function fibonacci_opt(n) that will optimize the fibonnacci by the use of a list of already computed values that can be reused when needed.



8. Create a function fibonacci_matrix(n) that will optimize the computation of Fibonacci's function by the use of fast exponentiation matrix. We remind that we have the following relation :

											(:	\mathcal{F}_{n+2} \mathcal{F}_{n+1}	$\binom{2}{2} =$	$=\begin{pmatrix}1\\1\end{pmatrix}$	$\begin{pmatrix} 1\\ 0 \end{pmatrix}$	$egin{pmatrix} \mathcal{F}_{n+1} \ \mathcal{F}_n \end{pmatrix}$
1	1	2	3	5	8	13	21	34	55	89	144	233	377	610	987	1597

9. In every preceding question, add a counter of operations so that at the end of the computation the total number of operations is printed and the number of calls to every recursive function.... Like :

Give	an	integ	ger,	ple	ease	19	
Give	an	integ	ger,	ple	ease	12	
gcd(1	19,1	2)=1	ops=	-38	call	s=13	3