El objetivo de este ejercicio es obtener los \_n\_ primeros números de la secuencia de Fibonacci usando un método súper rocambolesco.

El método este consiste en elevar la matriz [[1, 1], [1, 0]] a un número \_n\_, obteniendo de ahí el \_n-ésimo\_ número de la secuencia de Fibonacci.

Como queremos obtener los \_n\_ primeros números, pues iremos multiplicando la matriz [[1, 1], [1, 0]] por números cada vez más grandes e ir recolectando los números que vayan saliendo de las multiplicaciones que hemos ido haciendo.

El algoritmo para resolver este problema es demasiado largo para meterlo en una función, así que:

fibonacci\_matricial() crea la matriz [[1, 1], [1, 0]] y llama al método para hacer la exponenciación de matrices.

power() hace la exponenciación (multiplicacion iterativa), consistiendo esta en un bucle, del que se guarda un valor concreto de la matriz para cada iteración y se llama al método para multiplicar la matriz.

multiply() multiplica dos matrices.

CÓDIGO:

def fibonacci\_matricial(n):

if (n == 0):

return 0

F = [[1, 1],

[1, 0]]

return power(F, n)

def power(F, n):

M = [[1, 1],

[1, 0]]

fibonaccis = []

for i in range(n):

fibonaccis.append(F[1][1])

F = multiply(F, M)

return fibonaccis

def multiply(F, M):

result = [[0, 0], [0, 0]]

for i in range(len(F)):

for j in range(len(M[0])):

for k in range(len(M)):

result[i][j] += F[i][k] \* M[k][j]

return result

assert fibonacci\_matricial(10) == [0, 1, 1, 2, 3, 5, 8, 13, 21, 34]