**_4bucles/_7logger.py**

```python
# Ejercicio 7: Logger con Tiempo de Ejecución

# Imagina que estás desarrollando un sistema complejo que incluye múltiples funciones críticas. Para asegurarte de que todo funcione correctamente y para realizar un seguimiento del tiempo de ejecución de estas funciones, decides implementar un decorador de registro (logger) con tiempo de ejecución.

# El decorador debería realizar las siguientes acciones:

# Antes de llamar a la función original (puedes incluir cualquier función), debe imprimir un mensaje indicando que la función está a punto de ejecutarse.
# Después de que la función se haya ejecutado, debe imprimir un mensaje que incluya el tiempo que tardó la función en ejecutarse.
# Si la función original arroja una excepción, el decorador debe manejarla e imprimir un mensaje adecuado, indicando que se ha producido una excepción.

import time

def logger_con_tiempo_de_ejecucion(funcion):
  def wrapper():
  #Imprimir el tiempo de ejecucion
    inicio=time.time()
    print(f"Invocando a la funcion '{funcion.__name__}' ...")

    try:
      resultado=funcion()
    except Exception as e:
      print(f" Se produjo un error en la funcion '{funcion.__name__}': {e}")
      raise
    #Llamar al tiempo final de ejecucion
    fin=time.time()
    print(f"La funcion '{funcion.__name__}' ha tardado {fin-inicio} segundo en ejecutarse ")

    return resultado

  return wrapper

@logger_con_tiempo_de_ejecucion
#Funcion principal que calcula la serie de fibonacci
def mi_funcion():
  fib_series=[0,1]
  for i in range(2,20):
    fib_series.append(fib_series[i-1]+fib_series[i-2])
  return fib_series
print(mi_funcion())
```