Practica Recursion

Ejercicio 1 Escriba una funcion recursiva que tome un numero natural n e imprima todos los numeros desde n hasta 1.

Ejercicio 2 Escribir una función recursiva que calcule recursivamente el n-ésimo número triangular (el número 1 + 2 + 3 + ··· + n).

Ejercicio 3 Escribir una función recursiva que reciba un número positivo n y devuelva la cantidad de dígitos que tiene.

Ejercicio 4 Escribir una funcion recursiva que encuentre el mayor elemento de una lista.

Ejercicio 5 Escribir una función recursiva que reciba 2 enteros n y b y devuelva True si n es potencia de b.

Ejemplos:

- es_potencia(8, 2) -> True
- es_potencia(64, 4) -> True
- es_potencia(70, 10) -> False

Ejercicio 6 Escribir una función recursiva para replicar los elementos de una lista una cantidad n de veces. Por ejemplo:

- replicar([1, 3, 3, 7], 2) -> ([1, 1, 3, 3, 3, 3, 7, 7])

Ejercicio 7 Escriba una funcion que tome una lista y devuelva esa misma lista en orden inverso. Realice dos versiones:

- `reversaR` que resuelva utilizando recursión.
- `reversaI` que resuelva utilizando iteración.

Ejercicio 8 Escribir una funcion recursiva que reciba como parámetros dos cadenas a y b, y encuentre la posicion de la primer ocurrencia de b como subcadena de a

Ejemplo:

- posicion_de("Un tete a tete con Tete", "te") -> 3

Ayuda

- Puede ser de utilidad el método de `str.startswith` que nos dice si una cadena empieza con una subcadena dada. Puede leer mas sobre este método en la documentacion oficial

Recursión con wrappers

Al diseñar funciones recursivas muchas veces puede ser útil implementar una función **wrapper**, por ejemplo para cambiar los valores que hay que tomar, validar parámetros, inicializar datos o manejar excepciones.

Se utiliza una función recursiva, llamada funcion auxiliar, como base y luego una funcion no recursiva, llamada wrapper, que utiliza la funcion auxiliar para resolver lo pedido.

Por ejemplo, si necesitamos escribir una función recursiva para calcular el promedio, puede no ser evidente como hacerlo:

```
def promediar(lista):
  if len(lista) == 0:
    return ???
  promediar(lista[1:]) ???
```

Para calcular un promedio necesitamos computar tanto una suma como contar la cantidad de elementos. Podemos hacerlo con una función auxiliar

```
def _promediar(lista):
  if len(lista) == 0:
    return 0, 0
  cant, suma = promediar(lista[1:])
  return cant +1, suma + lista[0]

def promediar(lista):
  cant, suma = _promediar(lista)
  return suma / cant
```

Ejercicio 9 Escribir una funcion recursiva que dada una cadena determine si en la misma hay más letras A o letras E. Utilizar una funcion auxiliar.

Ejercicio 10 Escriba una función `potencia(b,n) que calcule la potencia para cualquier n entero, incluso si n es negativa. Utilizar una funcion auxiliar.

Ejercicio 11 Ya sabemos que la implementación recursiva del cálculo del número de Fibonacci (Fn = Fn−1 + Fn−2, F0 = 0, F1 = 1) es ineficiente porque muchas de las ramas calculan reiteradamente los mismos valores.

Escribir una función fibonacci(n) que calcule el enésimo número de Fibonacci de forma recursiva pero que utilice un diccionario para almacenar los valores ya computados y no computarlos más de una vez.

Nota: Será necesario implementar una función wrapper para cumplir con la firma de la función pedida.