Taller Pruebas Unitarias SONAR I.S.C

El siguiente taller busca continuar con el tema de pruebas unitarias tratado en la charla del semillero.

Objetivo general
Reforzar el concepto de pruebas unitarias por medio de la práctica.

Objetivos especificos
- Realizar practicas de código y pruebas unitarias
- Definir los posibles test unitarios que aplican por función
- Analizar los resultados obtenidos y cobertura de los test unitarios creados

Descripción del taller

El taller puede ser resuelto en Python, Javascript o el lenguaje de su preferencia. **Lo importante es el concepto y entendimiento del tema, no el lenguaje.**

Cada ejercicio incluye:
- Una descripción del problema
- Una función que debe ser implementada
- Un conjunto de pruebas unitarias que se deben desarrollar.

IMPORTANTE:
En el taller encontrarás 5 ejercicios. El ejercicio 5 es **obligatorio** luego elige 2 de los 4 restantes. De esta forma podemos compartir la solución en la proxima sesión y trabajar algunos más para afianzar conocimiento y solucionar dudas.

1. **Función que retorna el factorial de un número:** Implemente una función que calcule el factorial de un número entero positivo n.
    a. Pruebas Unitarias:
        i. Prueba que la función calcule correctamente el factorial de 5.
        ii. Prueba que la función calcule correctamente el factorial de 0.
        iii. Prueba que la función levante una excepción TypeError si el argumento no es un número entero.
    b. Adicionarias alguna prueba unitaria más? Si? No?

2. **Función que suma los elementos de una lista de números:**Implemente una función que sume los elementos de una lista de números.
    a. Prueba que la función calcule correctamente la suma de [1, 2, 3, 4, 5].
    b. Prueba que la función calcule correctamente la suma de [-1, 0, 1].
    c. Prueba que la función levante una excepción TypeError si el argumento no es una lista.
    d. Prueba que la función levante una excepción TypeError si la lista contiene un elemento no numérico.

3. **La siguiente función esta desarrollada en Javascript y Python y determina si un número es primo o no**.

   Función en Javascript

```
function isPrime(n) {
  if (n < 2) {
    return false;
  }
  for (let i = 2; i <= Math.sqrt(n); i++) {
    if (n % i === 0) {
      return false;
    }
  }
  return true;
}
```

   Función en Python

```
def is_prime(n):
    if n < 2:
        return False
    for i in range(2, int(n ** 0.5) + 1):
        if n % i == 0:
            return False
    return True
```

   Desarrolle entonces:

   a. Prueba que la función determine correctamente que 7 es un número primo.
   b. Prueba que la función determine correctamente que 10 no es un número primo.
   c. Prueba que la función levante una excepción TypeError si el argumento no es un número entero.
   d. Prueba que la función levante una excepción ValueError si el argumento es un número negativo.

4. **Función que ordena una lista de números:** Implemente una función que ordene una lista de números de menor a mayor.
   a. Prueba que la función ordene correctamente [5, 1, 3, 4, 2].
   b. Prueba que la función ordene correctamente [1, 2, 3, 4, 5].
   c. Prueba que la función levante una excepción TypeError si el argumento no es una lista.
   d. Prueba que la función levante una excepción TypeError si la lista contiene un elemento no numérico.

5. **Cálculo de promedio ponderado:** Crear una función en Python que calcule el promedio ponderado de una lista de números y sus respectivos pesos. La función

debería tener dos argumentos: una lista de números y una lista de pesos, y debería devolver el promedio ponderado. También debe tener en cuenta la posibilidad de que las listas tengan diferentes tamaños y generar una excepción en ese caso.

Ejemplo de entrada
nums = [2, 4, 6]
weights = [0.3, 0.5, 0.2]

Ejemplo de salida
promedio_ponderado(nums, weights) == 3.8

Cree las pruebas unitarias para esta función, cubriendo los casos exitosos y alternos.

Good Coding!