**Análisis de Casos**

Según el código dado en la clase ***AnalizadorFinaciero*** se analiza lo siguiente

**1.- Análisis de la Estructura de Datos del sistema DataSolver**

**Listas**

La estructura lista es utilizada en los métodos *calcular\_total\_ingreso* y *filtrar\_ingresos\_altos* en ellos almacena y procesa transacciones financieras.

En el método *calcular\_total\_ingresos*, la lista de transacciones se recorre mediante una sentencia for, sumando cada elemento para calcular el total general de ingresos. Esta estructura permite iterar secuencialmente cada monto.

En el método *filtrar\_ingresos\_altos*la lista se utiliza nuevamente para recorrer todas las transacciones, esta vez comparando cada ingreso contra un umbral definido. Si el ingreso supera dicho umbral, se agrega a una lista llamada *ingresos\_altos* que luego se retorna. Esto permite realizar un filtro selectivo de elementos, manteniendo solo los que cumplen con la condición establecida.

**Ventajas del uso de listas:** Son estructuras ordenadas y mutables, ideales para almacenar datos que se van a recorrer o modificar. Permiten operaciones como suma total, filtrado y recolección dinámica de elementos. Son compatibles con compresiones de listas lo que permite optimizar procesos como el filtrado.

**Limitaciones** **del uso de listas:** No valida el tipo de dato, no valida registros dulicados, las listas por si solas no indican a que tipo de ingreso pertenecen, si el volumen de transacciones es uy lto puede ser costos en rendimiento.

**Posibles mejoras a utilizar:** validar cada elemento con isinstance(ingreso, (int, float)) esto asegura solo el proceso de ingresos de tipo numéricos, usar de manera combinada con diccionarios esto es necesario para aportar estructura semántica, para grandes volúmenes de datos considerar Numpy o Pandas.

**Diccionarios**

En el método *agrupar\_por\_categoria* se emplea la estructura diccionario para organizar los ingresos según su categoría. Este método recibe dos listas, *transacciones* (montos de ingresos)y *categorías* (etiquetas que indican a que grupo pertenece cada ingreso).

La función *zip(categorias, transacciones)* permite iterar sobre paresy construir un diccionario de listas donde cada clave es una categoría y cada valor es una lista de ingresos asociados a esa categoría. De esta manera permite clasicar y agrupar los datos por categorías.

**Ventajas del uso de diccionarios:** Los diccionarios son colecciones desordenadas de pares clave-valor, son mutables, permiten un acceso rápido a los valores a través de sus claves únicas, son deales para representar datos jerárquicos o clasificados.

**Limitaciones** **del uso de diccionarios:** Las claves del diccionario son sensibles a errores tipográficos o mayúsculas y minúsculas; se pueden mezclar string, floats o datos mal ingresados.

**Posibles mejoras:** normalizar las claves con .lower().strip() antes de usarlas, implementar bloque try-except.

**2.- Optimización de Sentencias Iterativas de la clase AnalazidorFinanciero**

**Explicación de la optimización**

Una posible optimización de la clase es usar la función integrada sum(), simplifica y optimiza el cálculo eliminando la iteración manual y además se agrega la validación isintance() para evitar errores si algún dato no es numérico, como muestra la Figura 1 de acontinuación.

Texto

El contenido generado por IA puede ser incorrecto.Figura 1

**Descripción del uso de la expresión generadora y comprensión de lista**

**Expresión generadora:** En la optimización de la Figura 1, en la línea 4 se usa una expresión generadora para iterar cada ingreso dentro de la lista *transacciones* filtrando solo los elementos que son numéricos por medio de isinatnce(), la expresión generadora produce los valores uno a uno y los pasa directamente a la función sum() para calcular el total sin construir una lista intermedia retornando directamente dicho valor.

**Comprensión de lista:** En la línea 7 se usa una compresión de lista que recorre la lista *transacciones* filtrando los elementos que sean numéricos por medio de isinstance() y que sean mayores que un valor umbral, el resultado es una nueva lista implícita que retorna directamente.

**3.- Implementación de pruebas para las funciones**

Para la prueba de las funciones de la clase *AnalazidorFinanciero,* que muestra la Figura 1, se explica lo siguiente de la Figura 2:

Línea 2: Instancia *af* de la clase *AnalazidorFinanciero.*

Línea 4: Lista *cat*, que contiene las etiquetas de los ingresos.

Línea 6: Lista *tran*, que contiene las Transacciones.

Línea 8 y 9: Obtiene y muestra el cálculo total de los ingresos mediante la instancia *af* y el método *calcular\_total\_ingresos.*

Línea 14 y 15: Obtiene y muestra los ingresos altos mediante la instancia *af* y el método *filtrar\_ingresos\_altos.*

Línea 20 y 21: Obtiene y muestra las Transacciones agrupadas por categorías mediante la instancia *af* y el método *agrupar\_por\_categoria.*

Texto

El contenido generado por IA puede ser incorrecto. Figura 2

**4.- Aplicación de estructuras de daos avanzados**

Los sets son colecciones desordenadas de elementos únicos. Se utilizan para

operaciones de conjuntos matemáticos.

**Características de un Set**

* Desordenados: No mantienen un orden específico.
* Elementos únicos: No permiten duplicados.
* Mutables: Pueden agregar o eliminar elementos.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.Figura 3

**Beneficios de utilizar Set**

* Eliminación automática de duplicados
* Verificación más rápido que en una lista

**5.- Refactorización del código**

Se refactoriza el código optimizando también el método *agrupar\_por\_categoria* usando defaultdict(list) lo que permite inicializar listas vacías automáticamente al acceder a una nueva clave, también evita validar manualmente con un bloque if else como muestra la línea 22 de la Figura 4.

Texto

El contenido generado por IA puede ser incorrecto.Figura 4