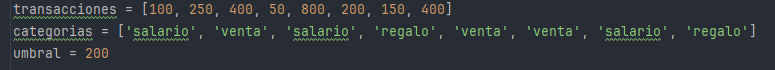
Benjamín Díaz

1. Estructura de datos:
   1. Listas: Permiten almacenar secuencias de datos de cualquier tipo, se recorren con un ciclo for fácilmente, además de mantener siempre el orden mediante los índices o posiciones
      1. Ventajas: Facilidad, sencillez de manipulación, versátiles, permiten hacer operaciones fácilmente con los datos de estas
      2. Desventajas: Listas muy grandes quizás no están optimizadas para buscar algún valor, por falta de las keys, son solo datos.
   2. Diccionarios: Los diccionarios permiten guardar datos (valores) con sus keys, similar a como se hace en DB con las tablas, fácil de identificar a que tipo de dato se está accediendo gracias a las keys, fácil de actualizar
      1. Ventajas: Acceso inmediato a datos agrupados como tablas según la key, mejor organización debido a esta estructura ya mencionada
      2. Desventajas: Como las DB no permiten keys duplicadas
2. Optimización:
   1. Dentro del código está el detalle de la optimización
   2. Las comprensiones de listas y expresiones generadoras son herramientas que en Python son muy útiles para optimizar la manipulación de los datos, en solo una linea de código se puede crear una lista que se debe rellenar por ciclos iterativos (ciclo for) o bien crear expresiones sin necesidad de guardarlas en memoria.
3. Pruebas:
   1. Pruebas hechas con diferentes conjuntos de datos/configuraciones, todas sin problemas
   2. Se pueden verificar los datos haciendo las operaciones manualmente, ya que tampoco son muchos datos, en caso contrario, lo mejor sería (según mi conocimiento), hacer un código simple que recorra la estr. De datos y verificar si son los mismos outputs.
4. Aplicación de estructuras de datos avanzados:
   1. Para verificar se podría usar el código set(categorías), ya que este tipo de dato representa una colección que filtra automáticamente todos los datos duplicados, por ejemplo si tengo una lista con [1,2,2,3,4,4,5] y uso set(lista), me devuelve automáticamente [1,2,3,4,5] quitando así los duplicados
   2. Beneficios: evita los duplicados automáticamente, es más eficiente que usar un ciclo for comparando, quizás con estructura de datos más grandes ocupa menos memoria.
5. Código refactorizado y reflexión: Todo eso está en el archivo entregado, con las líneas que venían por “default” comentadas para facilitar comparación. Por otro lado, con respecto a la optimización, se ve un código más limpio, más fácil de analizar, además del ahorro de memoria que tiene como beneficio, que si bien con pocos datos no se nota, es una muy buena práctica para cuando se trabaja con conjuntos de datos enormes (como imagino será la tónica del curso más adelante)

Anexos:

P1: 

Texto

El contenido generado por IA puede ser incorrecto.

P2:



Texto

El contenido generado por IA puede ser incorrecto.