Cambio Monedas

1. Se define una clase llamada `CalculadoraDeCambio` que se utilizará para manejar el cálculo del cambio.

2. En el constructor `__init__`, se inicializa la calculadora con una lista de `monedas_disponibles` que se ordena de mayor a menor valor.

3. El método público `calcular_cambio` se llama con una cantidad, y este método llama al método privado `_calcular_cambio_recursive` para realizar el cálculo de cambio.

4. El método privado `_calcular_cambio_recursive` es el corazón de la calculadora y realiza un cálculo recursivo para determinar todas las posibles combinaciones de monedas que suman la cantidad deseada.

5. Los casos base de la recursión son:

- Si la `cantidad` es cero, hemos encontrado un conjunto de monedas que suma al cambio, y se devuelve un diccionario que representa las monedas utilizadas y su cantidad.

- Si la `cantidad` es negativa o hemos agotado todas las monedas disponibles, no hay solución, y se devuelve `None`.

6. Para cada moneda disponible, el método calcula dos opciones:

- Usar la moneda actual y reducir la cantidad.

- No usar la moneda actual y pasar a la siguiente.

7. Si es posible usar la moneda actual (el resultado de la llamada recursiva no es `None`), se agrega al diccionario de solución con su cantidad correspondiente.

8. El método continúa con las siguientes monedas y devuelve la solución final si es posible dar el cambio.

9. Luego, se crea una instancia de `CalculadoraDeCambio` con una lista de monedas disponibles y se llama al método `calcular_cambio` para obtener la solución del cambio requerido. Finalmente, se imprime la solución o un mensaje si no es posible dar el cambio.

Hanoi

1. Se define una clase llamada `TorresDeHanoi` que se utilizará para resolver el problema de las Torres de Hanoi.

2. El constructor `__init__` no realiza ninguna acción en este caso.

3. El método `resolver_hanoi` es un método recursivo para resolver el problema de las Torres de Hanoi. Toma cuatro argumentos:

- `n`: El número de discos que se deben mover.
- `origen`: El nombre de la torre de origen.
- `destino`: El nombre de la torre de destino.
- `auxiliar`: El nombre de la torre auxiliar.

4. El caso base de la recursión es cuando `n` es igual a 1. En este caso, se imprime un mensaje indicando el movimiento del disco desde la torre de origen a la torre de destino y se retorna.

5. En los pasos recursivos, se utiliza la estrategia de dividir y conquistar:

- Se mueven `n-1` discos de la torre de origen a la torre auxiliar, usando la torre de destino como torre auxiliar.
- Se mueve el disco restante (el disco más grande) de la torre de origen a la torre de destino.
- Se mueven nuevamente los `n-1` discos desde la torre auxiliar a la torre de destino, utilizando la torre de origen como torre auxiliar.

6. Esto se repite hasta que todos los discos se muevan de la torre de origen a la torre de destino, respetando las reglas del problema de las Torres de Hanoi.

7. Se crea una instancia de la clase `TorresDeHanoi`, se solicita al usuario ingresar el número de discos y se llama al método `resolver_hanoi` para resolver el problema de las Torres de Hanoi. Cada movimiento se imprime en la consola.