2A. EJERCICIO: LISTA, DICCIONARIO, OPERADORES, COLLECTIONS.

Unidad 2: Manejo avanzado de estructura de datos.

Datos
Nombre: Keith Alexis Gutiérrez Ibarra
Docente: Adolfo Aldair Duque Borja
Fecha de elaboración: 25/11/2025

Introducción

En este ejercicio se realizará un sistema de gestión de clientes en Python, encargada de procesar información relacionada con compras y registros dentro de una tienda en línea, para esto se hará uso de estructuras de construcción dinámica con operaciones de conjuntos y herramientas de collections lo que permitirá filtrar, organizar y resumir los datos de forma sencilla y directa.

El objetivo de este trabajo es aplicar los conceptos vistos en la unidad, utilizando conjuntos para identificar clientes nuevos, estructuras ordenadas para eliminar duplicados, counter para contar repeticiones y comprensiones de diccionarios para generar resúmenes personalizados, reforzando así el manejo de datos y la programación estructurada.

Desarrollo

Ejercicio 1

a) Análisis del problema: Para el ejercicio se nos piden las siguientes condiciones que debe tener el Código.

- Filtrar clientes nuevos a partir de dos listas: compras y registrados, obtener los nombres que aparecen en “compras”, pero que no están en registrados, usando conjuntos y operaciones de diferencia.
- Eliminar duplicados y mantener orden, obteniendo una lista sin repeticiones de compras, conservando el orden en el que llegaron y usando OrderedDict o set (justifica la elección).
- Contar cuántas veces se repite cada nombre, con el uso de collections.Counter para contar cuántas veces aparece cada cliente en la lista de compras.
- Crear un resumen personalizado usando dict comprehension, crear un diccionario que relacione a cada cliente con un mensaje tipo: "Luis": "Ha comprado 3 veces" e incluir solo a quienes hayan comprado más de una vez.
- El formato final debe imprime tres bloques: Clientes nuevos no registrados, lista de clientes únicos y resumen por cliente frecuente (más de 1 compra).

b) Diseño del algoritmo

Se importan las librerías necesarias (collections) y se define la estructura de datos base.

a) Se inicializan dos listas principales una lista que contiene los nombres de las personas que realizaron compras y otra lista que almacena los nombres de personas registradas.

Se crea una función llamada clientes_nuevos que identifica a las personas que compraron, pero no se encuentran registradas.

a) Se utiliza la operación de diferencia de conjuntos para obtener los clientes no registrados.
b) La función retorna el conjunto resultante.

Se crea la función llamada compradores que genera una lista que mantiene el orden en que llegaron sin duplicar registros.

a) Se emplea OrderedDict para preservar el orden y eliminar duplicados, ya que el uso de set no garantiza orden.
b) La función retorna la lista resultante.

Creación de la función contador, encargada de realizar un conteo de compras por persona.

a) Se utiliza la clase counter de collections para obtener automáticamente el número de compras realizadas por cada cliente.
b) La función retorna un diccionario con el conteo generado

Se crea una función nombrada resumen_clientes que se encarga de generar un resumen de clientes que realizaron más de una compra.

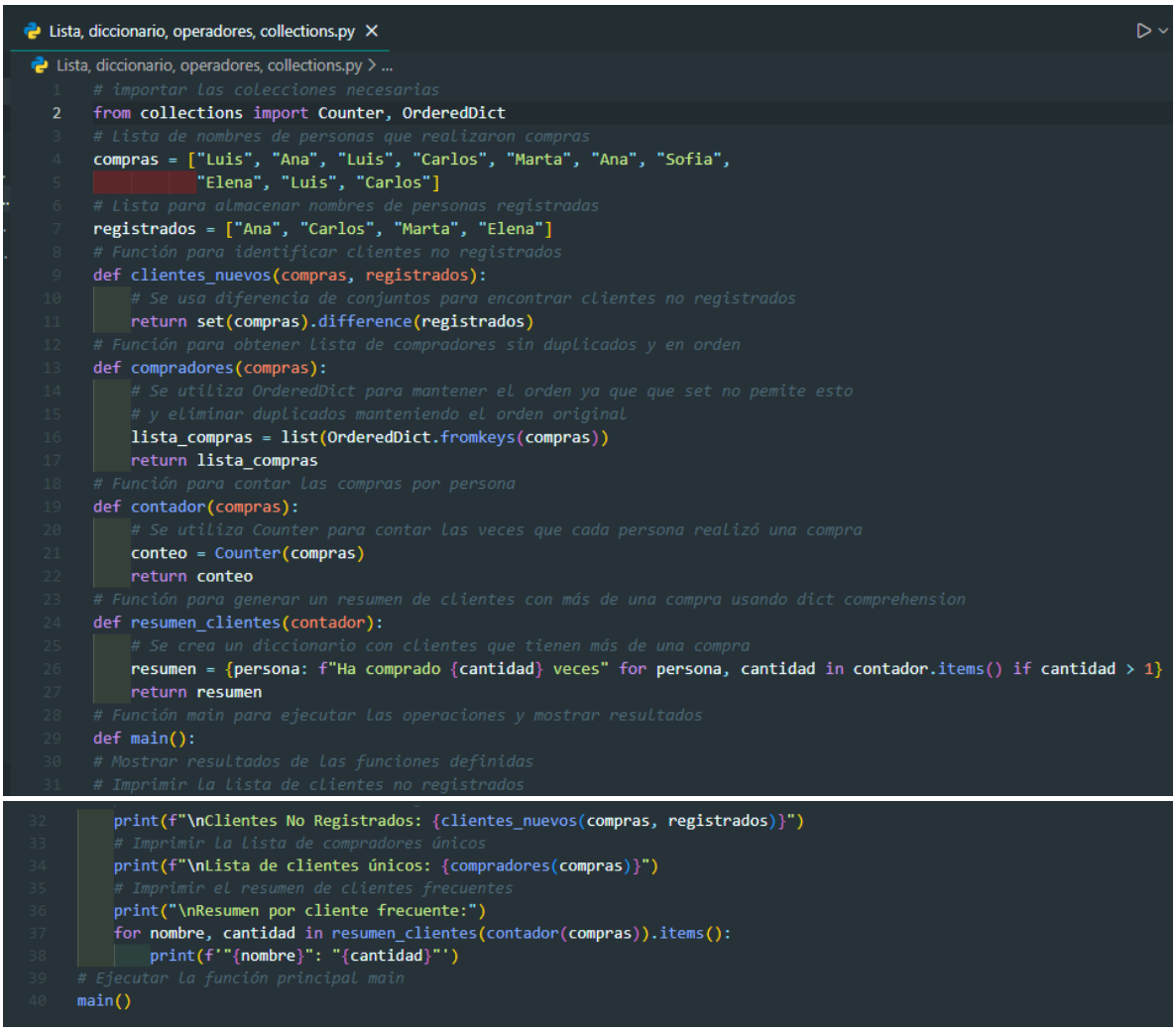
a) Se emplea dict comprehension para recorrer el conteo generado y filtrar a los clientes con compras superior a uno.
b) La función retorna un diccionario que contiene únicamente a los clientes que tienen más de una compra.

Se define la función principal main (decorada) que inicializa la salida al usuario.

a) Se imprimen los clientes no registrados usando la función correspondiente.
b) Se muestra la lista de compradores únicos cuidando el orden original.
c) Se imprime un resumen de clientes frecuentes iterando sobre el diccionario generado por la función resumen clientes.

Se realiza la llamada a la función main, ejecutando las funciones mencionadas anteriormente.

c) Código en Python



```python
# importar las colecciones necesarias
from collections import Counter, OrderedDict
# Lista de nombres de personas que realizaron compras
compras = ["Luis", "Ana", "Luis", "Carlos", "Marta", "Ana", "Sofia",
           "Elena", "Luis", "Carlos"]
# Lista para almacenar nombres de personas registradas
registrados = ["Ana", "Carlos", "Marta", "Elena"]
# Función para identificar clientes no registrados
def clientes_nuevos(compras, registrados):
    # Se usa diferencia de conjuntos para encontrar clientes no registrados
    return set(compras).difference(registrados)
# Función para obtener lista de compradores sin duplicados y en orden
def compradores(compras):
    # Se utiliza OrderedDict para mantener el orden ya que que set no pemite esto
    # y eliminar duplicados manteniendo el orden original
    lista_compras = list(OrderedDict.fromkeys(compras))
    return lista_compras
# Función para contar las compras por persona
def contador(compras):
    # Se utiliza Counter para contar las veces que cada persona realizó una compra
    conteo = Counter(compras)
    return conteo
# Función para generar un resumen de clientes con más de una compra usando dict comprehension
def resumen_clientes(contador):
    # Se crea un diccionario con clientes que tienen más de una compra
    resumen = {persona: f"Ha comprado {cantidad} veces" for persona, cantidad in contador.items() if cantidad > 1}
    return resumen
# Función main para ejecutar las operaciones y mostrar resultados
def main():
# Mostrar resultados de las funciones definidas
# Imprimir la lista de clientes no registrados
    print(f"\nClientes No Registrados: {clientes_nuevos(compras, registrados)}")
    # Imprimir la lista de compradores únicos
    print(f"\nLista de clientes únicos: {compradores(compras)}")
    # Imprimir el resumen de clientes frecuentes
    print("\nResumen por cliente frecuente:")
    for nombre, cantidad in resumen_clientes(contador(compras)).items():
        print(f'"{nombre}": "{cantidad}"')
# Ejecutar la función principal main
main()
```

d) Ejecución y pruebas

Ejemplos del funcionamiento del código:

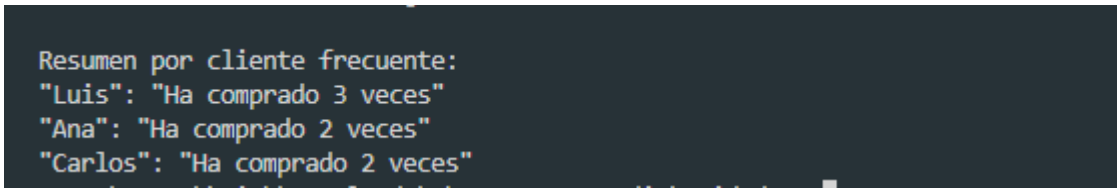
- Clientes nuevos no registrados:

```
Clientes No Registrados: {'Luis', 'Sofia'}
```

- Lista de clientes únicos:

```
Lista de clientes únicos: ['Luis', 'Ana', 'Carlos', 'Marta', 'Sofia', 'Elena']
```

- Resumen por cliente frecuente (más de una compra):



Conclusiones

El desarrollo de este ejercicio me permitió aplicar de manera práctica los conceptos vistos en la unidad como el uso de comprensiones, operaciones con conjuntos y uso del módulo collections, estas herramientas facilitaron el procesamiento de las listas de clientes registrados y compras, permitiendo identificar compradores nuevos, eliminar duplicados, contar registros y generar resúmenes de forma más ordenada y eficiente.

El trabajo realizado demuestra cómo el uso de estructuras dinámicas y técnicas propias de Python puede simplificar tareas que podrían ser más complejas o repetitivas, además permitió reforzar la lógica de programación y comprender mejor cómo organizar los datos para obtener información útil a partir de ellos, este ejercicio permite ver la importancia de emplear soluciones claras, reutilizables y estructuradas para el manejo de información.