Manual Técnico Practica 1

El código de esta practica empieza con la definición de un menú en el cual se ofrecen las distintas funciones para el usuario, este funciona buscando un numero de opción ingresado por el usuario y lo comprueba que sea un numero 1, 2, o 3 y hace su función correspondiente, si el numero ingresado por el usuario no es ninguno de estos hace lo indicado en la opción de else y vuelve a cargar la clase menú().

```
def menu():
    print('Seleccione una opcion')
    print('----------------------------')
    print('1. Cargar Inventario ')
    print('2. Modificar Inventario')
    print('3. Generar Reporte')
    print('----------------------------')
    eleccion = input('Seleccione una opcion: ')
    if eleccion =='1':
        print('Opcion 1')
        cargar()
        return menu()
    elif eleccion == '2':
        print('Opcion 2')
        modificar()
        menu()

    elif eleccion == '3':
        print('Opcion 3')
        guardar_productos_en_archivo()
    else:
        print('Opcion incorrecta')
        print('Intente otra vez')
        menu()
```

Del menú anterior la elección uno redirige a la función cargar la cual esta estructura de la siguiente manera:

```
def cargar():
    with open('inventario.inv', 'r') as archivo:#usare with por que dice que esto asegura el cierre del archivo cuando termine
        for linea in archivo:
            if linea.startswith("crear_producto"):
                print('crear')#para comprobar que se cumpla la linea anterior eliminar despues
                partes = linea.strip().split(';')
                if len(partes) == 4:# len para medir la longitud de la lista partes la cual contendra nombre,stock,precio,ubicacion
                    nombre = partes[0].split(' ')[1]#indicando que nombre sera partes[0] que es crear_producto nombre
                    stock = int(partes[1])#guardando en un int el stock del producto
                    precio = float(partes[2])# y en un float el precio este es partes 2 por que es nombre0 sctock 1 precio 2
                    ubicacion = partes[3]
                    producto = Producto(nombre, stock, precio, ubicacion)#creando un objeto de la clase Producto para agregarlo al listado
                    lista_productos.append(producto)


    #imprimir_productos()
```

Esta función empieza con la línea with open la cual funciona para que se pueda abrir el archivo y esta misma función se encargara de cerrarlo cuando se deje de utilizar sin necesidad de colocar close().

Luego se ingresa a un ciclo for para buscar cada línea del archivo, la cual únicamente reconoce las líneas que empiecen por crear_producto con la función startswith luego en una variable partes se ingresan los datos de la línea indicando que los datos estarán separados por un símbolo “;” luego verifica la longitud de la línea la cual contiene nombre, stock precio y ubicación y procede a asignar a una variable cada uno de estos datos para después crear una instancia de la clase productos e ingresarla a un listado llamado

lista_productos el cual es declarado fuera de la clase para que pueda ser modificado y llamado por múltiples clases que lo necesitaran.

Funcionamiento de la clase modificar

```python
def modificar():
    try:
        with open('movimiento.mov', 'r') as archivo_movimientos:
            for linea in archivo_movimientos:
                partes = linea.strip().split(' ')
                accion = partes[0]
                datos = partes[1:]
                if accion == "agregar_stock":
                    nombre, cantidad, ubicacion = datos[0].split(';')
                    agregar_stock(nombre, int(cantidad), ubicacion)#este esta en for por eso la bandera regresa a false
                elif accion == "vender_producto":
                    nombre, cantidad, ubicacion = datos[0].split(';')
                    vender_producto(nombre, int(cantidad), ubicacion)
    except FileNotFoundError:
        print("El archivo de movimientos 'movimiento.mov' no encontrado.")
#usando clases para ordenarlo
```

Esta clase funciona de una manera muy similar a la anterior por excepción de que el archivo .mov contiene solamente tres tipo de datos los cuales son nombre, cantidad y ubicación, previamente indicando una acción la cual puede ser vender o agregar luego asigna la parte 0 a una variable acción, la parte 0 siendo el texto con el que empieza la línea del archivo verifica el texto de esta línea y manda a llamar a la clase según la acción requerida, si no encuentra el archivo imprime un mensaje de error.

Funcionamiento de la clase agregar_stock

```python
def agregar_stock(nombre, cantidad, ubicacion): #de aca corregir que no me agrega de manera correcta si se encuentra un producto sin ubicacion
    producto_encontrado = False  #probando poninendo un false y que siempre que este sea cambiado a true si se encontro el producto

    for producto in lista_productos:
        if producto.nombre == nombre and producto.ubicacion == ubicacion:
            producto.stock += cantidad#sumando la cantidad nueva mas la anterior del producto producto.nombre despues de comprobar su ubicacion
            producto_encontrado = True#haciendo la bandera true para que vuelva a repetir y no de mensaje de error
            break

    if not producto_encontrado:#o se queda en false si no
        print(f"No se encontró el producto '{nombre}' en la ubicación '{ubicacion}'")
    else:
        print("Stock agregado exitosamente")
```

Esta función empieza con una variable booleana la cual es de utilidad para verificar que productos existen en la dicha posición dada en el archivo movimiento, esta función luego verifica si el producto se encuentra en la lista y si esta suma la cantidad nueva mas la anterior, si no imprime un mensaje de error diciendo que el producto no fue encontrado

Funcionamiento de la clase vender_producto

```
def vender_producto(nombre, cantidad, ubicacion):
    for producto in lista_productos:
        if producto.nombre == nombre and producto.ubicacion == ubicacion:#comprobando que el producto exista y tenga ubicacion valida
            if producto.stock >= cantidad:
                producto.stock -= cantidad
            else:
                print(f"No hay suficiente stock de {nombre} en {ubicacion}")
            return
```

Esta clase funciona con un ciclo for buscando si el nombre y la ubicación dada existen son iguales en la lista_productos y si se busca vender mas de lo que hay imprime un mensaje de error

De la ultima función del menú guardar_productos_en_archivo

```
def guardar_productos_en_archivo():
    mdatos = []#guardar en esta lista los datos a mostrar
    for producto in lista_productos:
        if producto.stock > 0:  #Para solo mostrar los productos que tengan stock mayor que cero
            valor_total = producto.stock * producto.precio#para añadir la comlumna vtotal de prodctu.nombre
            mdatos.append([producto.nombre, producto.stock, producto.precio, valor_total, producto.ubicacion])

    if not mdatos:
        print("Se vendio todo o no se cargo correctamente el archivo")
        return
    headers = ["Nombre", "Stock", "Precio", "Valor Total", "Ubicación"]

    # usar w para sobreescribir, si fuese a necesitar agregar usar 'a'
    with open('reporte_202004071.txt', 'w') as archivo:
        archivo.write(tabulate(mdatos, headers, tablefmt="pretty"))#      https://www.youtube.com/watch?v=Yq0lbu8goeA
        print('Informe guardado en "reporte_202004071.txt".')
```

Esta funciona creando una lista llamada mdatos para poder agregarle la columna valor total la cual será mostrada, luego se verifica que la lista mdatos no este vacía y crea unos headers para la tabla, de la misma manera con los otros archivos se utiliza la función with open y se hace uso de la librería tabulate para poder mostrar de manera ordenada en el archivo de texto el reporte

Ejemplos uso de aplicación

```
Seleccione una opcion
----------------------------
1. Cargar Inventario
2. Modificar Inventario
3. Generar Reporte
----------------------------
Seleccione una opcion: 1
Opcion 1
crear
```

```
Seleccione una opcion
-------------------------
1. Cargar Inventario
2. Modificar Inventario
3. Generar Reporte
-------------------------
Seleccione una opcion: 2
Opcion 2
Stock agregado exitosamente
No se encontró el producto 'Queso' en la ubicación 'BodegaG'
No hay suficiente stock de PlÃ¡tanos en BodegaC
Stock agregado exitosamente
Stock agregado exitosamente
Stock agregado exitosamente
```

Nombre	Stock	Precio	Valor Total	Ubicación
Tomates	100	1.0	100.0	BodegaA
Peras	175	3.25	568.75	BodegaC
PlÃ¡tanos	75	1.75	131.25	BodegaD
Queso	35	20.5	717.5	BodegaE
Helado	172	6.5	1118.0	BodegaF
ArÃ¡ndanos	735	0.5	367.5	BodegaG
Tomates	1	1.0	1.0	BodegaG
Manzanas	150	3.0	450.0	BodegaF
Peras	101	3.25	328.25	BodegaE
Queso	60	20.5	1230.0	BodegaD
Helado	80	6.5	520.0	BodegaB
ArÃ¡ndanos	155	0.5	77.5	BodegaA