UADE

Trabajo Práctico

Universidad Argentina de la Empresa (UADE)

Asignatura: Introducción a la algoritmia

Día - Turno: Jueves - Tarde

Docente: Escandell, Gustavo Emanuel

Integrantes: Ceresetto, Valentino; Estévez, Lukas; Gandini Martin, Lucas; Gribaudo, Luca; Müller, Tiago

Fecha de presentación: jueves 14 de noviembre

UADE

Introducción

Para la realización del trabajo práctico final de la materia, elegimos hacer un sistema de gestión de stock. El objetivo de este es desarrollar un sistema de stock para una tienda donde los usuarios puedan agregar productos, registrar entradas y salidas de stock, y visualizar el stock total de cada producto de manera ordenada.

A lo largo de este informe, detallaremos el paso a paso para la realización de este código.

Contexto

Este programa de Python ha sido desarrollado con el propósito de optimizar la gestión de inventarios en una tienda minorista de productos electrónicos. La tienda ofrece una variedad de artículos, tales como dispositivos móviles, accesorios y componentes informáticos. La administración enfrenta desafíos comunes relacionados con el seguimiento de productos, como mantener un registro preciso del stock, gestionar las entradas y salidas de mercancía, y evitar tanto los desabastecimientos como los excesos de inventario.

Paso 1

La función def *buscar_producto(p)* busca el producto en *lista_producto.* El parámetro p recibe el producto, que se busca dentro de *lista_productos* mediante un bucle *while*. Este bucle recorre la lista hasta encontrar el producto y verificar que coincide con *p*. Si encuentra el producto, el bucle se detiene, y la función devuelve el índice i, que indica la posición del producto en la lista. Si no lo encuentra, el bucle termina al final de la lista y devuelve la longitud de la lista.

Variables:

- *p*: recibe el producto.
- *i*: variable que recorre la lista.

```
def buscar_producto(p):
    i = 0
    while (i < len(lista_productos)) and (lista_productos[i] != p):
        i = i + 1
    return i
```

Paso 2

Luego, en la función *agregar_o_actualizar_producto*, se solicita al usuario que ingrese el nombre del producto mediante un string(input), convirtiéndolo automáticamente a minúsculas. Posteriormente, se busca el nombre ingresado en la lista utilizando *buscar_producto*. Después, se solicita que el usuario ingrese la cantidad de unidades deseadas a través de un int(input).t), y se determina si el producto ya existe en la lista con len(lista_producto).

```
#opcion 1

def agregar_o_actualizar_producto():
    producto = str(input('Ingrese el nombre del producto: '))
    producto = producto.lower()
    indice_producto = buscar_producto(producto)

    unidades = int(input("¿Cuántas unidades?: "))
    longitud = len(lista_productos)
```

Paso 3

En el if, se verifica si el producto ya está en la lista mediante el *índice_producto*. Si el producto está en la lista, se actualizan las unidades en *lista_entradas*. En caso de que el producto sea nuevo, se solicita el precio por unidad mediante un float(input) y se agrega usando *agregar_producto()*.

```
if (indice_producto < longitud):
    lista_entradas[indice_producto] = lista_entradas[indice_producto] + unidades
else:
    precio = float(input("Ingrese el precio por unidad del producto: "))
    agregar_producto(producto,precio,unidades)
```

Paso 4

Esta función agrega el producto junto con sus detalles, como el precio, la unidad y el nombre del producto, mediante append. Además, establece los valores de entradas, salidas y stock en cero.

```
def agregar_producto (producto,precio,unidades):
    lista_productos.append(producto)
    lista_precios.append(precio)
    lista_entradas.append(unidades)
    lista_salidas.append(0)
    lista_stocktotal.append(0)
```

Paso 5

La función *stock_ordenado_de_menor_a_mayor* organiza el stock en orden ascendente. Primero, obtiene la cantidad de productos en *lista_stocktotal* y lo almacena en *n*. Luego, se ejecuta un bucle que recorre la lista almacenada en *n*, y un bucle interno recorre los elementos, incrementando el valor de *i* y reduciendo el rango del bucle, ya que cada elemento se posiciona en su lugar. Al final, se realiza una comparación de elementos para ordenarlos de menor a mayor.

```
def stock_ordenado_de_menor_a_mayor():          #ordenado SEGUN EL STOCK de menor a mayor
    n=len(lista_stocktotal)
    for i in range (n-1):
        for j in range(0, n-i-1):
            if lista_stocktotal[j] > lista_stocktotal[j+1]:
                lista_stocktotal[j], lista_stocktotal[j+1] = lista_stocktotal[j+1], lista_stocktotal[j]
                lista_productos[j], lista_productos[j+1] = lista_productos[j+1], lista_productos[j]
                lista_precios[j], lista_precios[j+1] = lista_precios [j+1], lista_precios[j]
                lista_entradas[j], lista_entradas[j+1] = lista_entradas[j+1], lista_entradas[j]
                lista_salidas[j], lista_salidas[j+1] = lista_salidas[j+1], lista_salidas[j]
```

Paso 6

La función *recalcular_stock* calcula el stock actual restando las salidas de las entradas. Una vez realizado este cálculo, se ordena el stock mediante *stock_ordenado_de_menor_a_mayor*. Al finalizar, se recorre *lista_producto* y se imprime cada producto con su precio por unidad y su stock.

```
def recalcular_stock ():
    for i in range(len(lista_productos)):
        lista_stocktotal[i] = (lista_entradas[i] - lista_salidas[i])
    stock_ordenado_de_menor_a_mayor()
    for i in range (len(lista_productos)):
        print(f'\n El producto {lista_productos[i]} tiene un precio por unidad de ${lista_precios[i]} y tiene un stock de {lista_stocktotal[i]}')
    print()
```

Paso 7

La siguiente función permite registrar la salida de un producto. Mediante un if, se verifica si hay productos disponibles; de ser así, se solicita el nombre mediante un str(input) y se busca en la lista con *buscar_producto*.

```
#opcion 2

def registrar_salida_producto():
    if lista_productos != []:
        producto = str(input('ingrese el producto a retirar: '))
        producto = producto.lower()
        indice_producto = buscar_producto(producto)
        while (indice_producto == len(lista_productos)):
            producto = str(input('ingrese el producto a retirar: '))
            producto = producto.lower()
            indice_producto = buscar_producto(producto)
        cant_salida = int(input('ingrese la cantidad de producto a retirar: '))
        while ((lista_entradas[indice_producto] - lista_salidas[indice_producto]) < cant_salida ):
            cant_salida = int(input('no hay suficiente stock, ingrese un número menor: '))

        lista_salidas[indice_producto] = lista_salidas[indice_producto] + cant_salida
    else:
        print('Aún no hay productos, debe añadir uno para poder retirar')
        pass
```

Paso 8

Ahora, se crean tres listas vacías para almacenar los datos ingresados, las cuales son:

- lista_productos = [ ].
- lista_precios = [ ].
- lista_entradas = [ ].
- lista_salidas = [ ].
- lista_stocktotal = [ ].

Paso 9

Se crean listas vacías para almacenar datos. Luego, se solicita al usuario que ingrese la acción a realizar mediante un int(input), y se ejecuta un bucle *while* hasta que el usuario seleccione la opción 4. Si el número ingresado no es válido, se le pide que lo ingrese nuevamente. A continuación, mediante un if, se ejecuta una acción específica según el número ingresado, como *agregar_o_actualizar_producto()*, *registrar_salida_producto()*, *recalcular_stock()*, o terminar = True. Si el programa aún no ha terminado, se vuelve a solicitar la acción deseada. Finalmente, se recalcula el stock y se imprime el mensaje de finalización.

```
terminar = False

accion = int(input('que acción derea ejecutar?. 1 actualizar o agregar, 2 retirar, 3 mostrar stock ordenado, 4 salir: '))

while (accion != 4) and (terminar == False):
    while (accion < 1) or (accion > 5):
        accion = int(input('que acción derea ejecutar?. 1 actualizar o agregar, 2 retirar, 3 mostrar stock ordenado, 4 salir: '))
    if accion == 1:
        agregar_o_actualizar_producto()
    elif accion == 2:
        registrar_salida_producto()
    elif accion == 3:
        recalcular_stock()
    elif accion == 4:
        terminar = True
    if (terminar == False):
        accion = int(input('que acción derea ejecutar?. 1 actualizar o agregar, 2 retirar, 3 mostrar stock ordenado, 4 salir: '))

recalcular_stock()

print('El sistema a finalizado')
```

Conclusión

Este programa en Python ofrece una solución práctica para manejar el inventario en una tienda de productos electrónicos. Al automatizar el seguimiento de las existencias, las entradas y salidas de productos, ayuda a evitar problemas comunes como el desabastecimiento o el exceso de stock.

Además, al organizar los productos de manera clara y permitir actualizaciones rápidas, hace que el proceso de gestión sea más eficiente, lo que facilita la toma de decisiones y mejora la organización general del negocio. En definitiva, es una herramienta clave para mantener un control efectivo del inventario y mejorar la operación de la tienda.