Luis Rodrigo Morales Florián

Lenguajes Formales Sección B

Segundo Semestre 2023

Práctica #1

Universidad de San Carlos de Guatemala

Contenido

Inicio:

Importa el modulo os para la lectura y escritura de archivos y crea una lista con todos los datos de los productos. Luego entra a un bucle infinito (que se rompe en la opción 4) y procede con la impresión del menú y con la petición de la opción que desea el usuario.

```
import os

productsData = []
```

```
# INICIO PROGRAMA
while True:
  print("USAC - Segundo Semestre 2023")
  print("Práctica 1 - Lenguajes Formales")
  print("Luis Morales - 202208521")
  print("----------------------------")
  print("MENU")
  print("1. Cargar Inventario inicial")
  print("2. Cargar Instrucciones de movimientos")
  print("3. Crear Informe de inventario")
  print("4. Salir")
  option = input("Ingrese el número de opción:")
```

Opción 1.

Primero ingresa el nombre del archivo luego chequea si nombre del archivo termina con .inv. Si no es así lanza un error, de lo contrario, lee cada línea del archivo y por cada línea formatea y lo guarda en la lista de datos de los productos.

De no funcionar (de manera externa), lanza los errores en except.

```python
if option == "1":
  print("\n")
  print("Ha elegido la opción '1. Cargar Inventario inicial'")
  print("\n")

  try:
      nombre_archivo = input("Ingresa el nombre del archivo: ")

      if not nombre_archivo.endswith('.inv'):
        print("\n")
        raise ValueError("ERROR: El archivo debe tener extensión .inv")
      else:
        lineas = leer_archivo(nombre_archivo)
        #crear_producto <nombre>;<cantidad>;<precio_unitario>;<ubicacion>
        for linea in lineas:
          dato = obtener_datos_por_linea(linea)
          nombre, cantidad, precio_unitario, ubicacion = dato[1:]
          productsData.append({
            "nombre": nombre,
            "cantidad": int(cantidad),
            "precio_unitario": float(precio_unitario),
            "ubicacion": ubicacion
          })

          print("\n")
          print(f"ÉXITO: El archivo {nombre_archivo} ha sido agregado correctamente")
```

```python
  except ValueError as ve:
      print(ve)
  except FileNotFoundError as fnfe:
      print(fnfe)
  finally:
      print("\n")
```

Opción 2

Lee el nombre del archivo. Si el nombre no termina con punto .mov, lanza un error, de lo contrario se leen los datos del archivo línea por línea y se formatean los datos ya que es un formato general. Busca el producto por nombre y ubicación y se encuentra el producto. Luego, se procede a verificar el tipo de instrucción que se ingresó en esa línea. Si es agregar producto, en el producto encontrado agrega la cantidad que se ingresó en la línea del archivo. Si el dato es igual a vender producto, chequea primero si se puede vender el producto comparando la cantidad que se encuentre contra la cantidad que ingresar en la línea del archivo. Si la cantidad el producto es mayor, se puede vender sino lanza un error. Hasta abajo están los errores de si hay algún error de la librería.

```
elif option == "2":
  print("\n")
  print("Ha elegido la opción '2. Cargar Instrucciones de movimientos'")
  print("\n")

  try:
      nombre_archivo = input("Ingresa el nombre del archivo: ")

      if not nombre_archivo.endswith('.mov'):
        print("\n")
        raise ValueError("El archivo debe tener extensión .mov")
      else:
        lineas = leer_archivo(nombre_archivo)

        for linea in lineas:
          dato = obtener_datos_por_linea(linea)
          nombre, cantidad, ubicacion = dato[1:]

          producto_encontrado = buscar_producto(nombre=nombre, ubicacion=ubicacion)

          if producto_encontrado:
            #agregar_stock <nombre>;<cantidad>;<ubicacion>
            if dato[0] == "agregar_stock":
              producto_encontrado["cantidad"] += int(cantidad)
              print("\n")
              print("ÉXITO: Cantidad actualizada exitosamente.")
              print("\n")
```

```
            #vender_producto <nombre>;<cantidad>;<ubicacion>
            elif dato[0] == "vender_producto":
              if producto_encontrado["cantidad"] >= int(cantidad):
                producto_encontrado["cantidad"] -= int(cantidad)
                print("\n")
                print("ÉXITO: Productos vendidos exitosamente.")
              else:
                print("\n")
                print("Error: La cantidad que quiere vender es mayor que la cantidad de productos existentes")
          else:
            print("\n")
            print("Error: Producto no encontrado.")

  except ValueError as ve:
      print(ve)
  except FileNotFoundError as fnfe:
      print(fnfe)
  finally:
    print("\n")
```

Opción 3:

Chequea si la ruta que yo ingrese estáticamente es posible, si lo es, procede a crear un archivo con los datos formateados (en una función aparte que se explicará más adelante).

```
elif option == "3":
    print("\n")
    print("Ha elegido la opción '3. Crear Informe de inventario'")
    print("\n")

    ruta = r'C:\Users\DELL\Desktop\archivo.txt'

    try:
        if os.path.exists(os.path.dirname(ruta)):
            with open(ruta, 'w') as archivo:
                archivo.write(obtener_tabla_datos())

            print("\n")
            print("TERMINADO: Archivo creado exitosamente en el escritorio.")
            print("\n")
        else:
            print("La ruta del escritorio no existe.")
    except Exception as e:
        print(f"Ocurrió un error: {e}")
```

Opción 4

Sale del ciclo while y termina la aplicación

```
elif option == "4":
    print("\n")
    print("HAS SALIDO DE LA APLICACIÓN")
    print("\n")
    break
```

Funciones

leer_archivo(nombre_archivo: str)-> Lista:

Lee la ruta del archivo (nombre_archivo), y si existe, procede a leerlo y devolver una lista con cada línea (iterabe).

```
def leer_archivo(nombre_archivo):
    try:
        with open(nombre_archivo, 'r') as archivo:
            lineas = archivo.readlines()
            return lineas
    except FileNotFoundError:
        raise FileNotFoundError(f"El archivo '{nombre_archivo}' no fue encontrado")
```

Obtener_tabla_datos() -> str:

Primero, crea una variable donde se almacenará todo el texto que se va a devolver.

Se crea una variable con una lista con todos los encabezados.

Se formatean poniéndolas una línea en medio y luego se procede a crear un separador de los encabezados con el tamaño de la longitud de la lista los encabezados. Luego. se agregan al texto que se va a devolver. Luego se itera cada producto existente en la lista global productsData.

Se agrega cada uno de estos datos una fila formateada qué básicamente dice que si el tamaño es menor al indicado que lo llene con espacios blancos. Donde hay una f nos dice cuántos decimales va a agregar y es así como se agrega una cadena de texto formateada y de manera iterada.

Se agrega al texto para devolver y último se devuelve.

```
def obtener_tabla_datos() -> str:
    texto_para_tabla = ""

    encabezados = ["Producto", "Cantidad", "Precio Unitario", "Valor Total", "Ubicación"]
    encabezados_formatados = " | ".join(encabezados)
    separador = "-" * len(encabezados_formatados)

    texto_para_tabla += encabezados_formatados + "\n"
    texto_para_tabla += separador + "\n"

    for producto in productsData:
        nombre = producto['nombre']
        cantidad = producto['cantidad']
        precio_unitario = producto['precio_unitario']
        valor_total = cantidad * precio_unitario
        ubicacion = producto['ubicacion']

        # Formatear y alinear los datos
        fila_formatada = "{:<8} | {:<8} | {:<15.2f} | {:<11.2f} | {:<10}".format(
            nombre, cantidad, precio_unitario, valor_total, ubicacion
        )

        texto_para_tabla += fila_formatada + '\n'
    return texto_para_tabla
```

Obtener_datos_por_linea(linea: str) -> Lista

Se crea una variable con el nombre datos y se recibe la línea literal que se recibe del archivo. Se separa la cadena de texto por “;” y el primer elemento que se obtiene es el comando junto con el primer dato. Hay que separarlos otra vez por espacios. Se agregan entonces el comando con esa técnica en el primer elemento de la variable datos. Luego, usando esta técnica otra vez, se agrega el primer elemento de los datos que se quieren ingresar. Finalmente, a partir del dato 1 de los datos obtenidos de la línea separada por “;”, se agregan los otros datos a la variable datos y se devuelve finalmente esa variable.

```python
def obtener_datos_por_linea(linea: str):
    datos = []
    datos.append(linea.strip().split(";")[0].split(" ")[0]) # el comando
    datos.append(linea.strip().split(";")[0].split(" ")[1])
    datos.extend(linea.strip().split(";")[1:])
    return datos
```

```python
def buscar_producto(nombre, ubicacion):
    for producto in productsData:
        if producto["nombre"] == nombre and producto["ubicacion"] == ubicacion:
            return producto
    return None
```