
Practica 1

202100692 – Josue Ricardo Carias Ordoñez

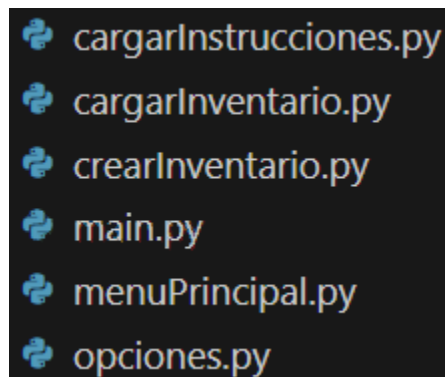
Resumen:

Se brindo solución a un problema de inventario usando el lenguaje de programación Python en su versión 3, usando habilidades en el manejo de archivos, lógica de programación y manipulación de estructura de datos

Objetivos:

- Que el estudiante implemente una solución de software con base en los distintos paradigmas de programación vistos en clase y laboratorio.
- Que el estudiante adquiera habilidades en el manejo de archivos, lógica de programación y manipulación de estructuras de datos en Python.

Clases dentro del programa



cargarInstrucciones.py
cargarInventario.py
crearInventario.py
main.py
menuPrincipal.py
opciones.py

- 1). main.py: clase principal del proyecto, su función es dar inicio al proyecto, dentro de él se hace llamado a la clase encargada de generar el menú de selección

```
from menuPrincipal import getMenuPrincipal
#constructor principal del programa
getMenuPrincipal()
```

2). menuPrincipal.py: esta clase se encarga de hacer el menú, y llama al módulo encargado de hacer la opción, que recibe como parámetro un numero entre el 1 y el 4

```
#generador del formulario principal
def getMenuPrincipal():
    from opciones import getOpciones
    print("-----")
    print("Pratica 1-Lenguajes formales y de programacion")
    print("-----")
    print("# Sistema de inventario:")
    print("1. Cargar inventario inicial")
    print("2. Cargar instrucciones de movimientos")
    print("3. Crear informe de invetario")
    print("4. Salir")
    getOpciones()
```

```
-----
Pratica 1-Lenguajes formales y de programacion
-----
# Sistema de inventario:
1. Cargar inventario inicial
2. Cargar instrucciones de movimientos
3. Crear informe de invetario
4. Salir
Ingrese una opcion:
```

Visualmente se mira así, se queda esperando a recibir un valor, si el valor esta fuera del rango de selección vuelve a preguntar por la opción hasta recibir una dentro del rango

3). cargarInventario.py: esta clase es la encargada de leer el archivo .inv, dentro de ella hay varias funciones la primera es leerArchivoInv(ruta), esta función recibe como parámetro la dirección relativa del archivo .inv, lee el archivo y lo guarda en un diccionario

```
#clase para leer el archivo .inv y almacenarlo en un diccionario
def leerArchivosInv(ruta):
    ubicaciones = {}
    try:
        with open(ruta, encoding="utf-8") as file:
            for linea in file:
                producto, cantidad, precio, ubicacion= linea.strip().split(" ")[1].split(";")
                if ubicacion in ubicaciones:
                    ubicaciones[ubicacion][producto]= {
                        "cantidad": float(cantidad),
                        "precio": float(precio)
                    }
                else:
                    ubicaciones[ubicacion]= {
                        producto: {
                            "cantidad": float(cantidad),
                            "precio": float(precio)
                        }
                    }
            return ubicaciones
    except FileNotFoundError:
        print("El archivo no fue encontrado.")
        getCargarInicial()
    except IOError:
        print("Error al leer el archivo.")
        getCargarInicial()
```

Otra función dentro de esta clase es `imprimirInventario(inventario)`, esta función recibe como parámetro un diccionario, usando la librería json para dar una salida un poco más estética y entendible en la terminal

```
#clase para escribir el inventario en un formato un poco mas leible
def imprimirInventario(inventario):
    print("-----")
    print(json.dumps(inventario, indent=4))
```

La función principal de esta clase es `getCargarInivial()` este llama de nuevo `getMenuPrincipal()` al termina, su función principal es preguntar por la dirección relativa del archivo .inv y llamar a la función `leerArchivosInv(ruta)` y mandarle el parámetro de la ruta, después llama a la función encargada de imprimir `imprimirInventario(inventario)` , mandando el parámetro de lo leído en `leerArivicosInv()`

4). `cargarInstrucciones.py`: a este modulo se accede ingresando la opcion 2 del menú este archivo se encarga de leer el archivo .mov, dentro del el hay varias funciones una de ellas es `cargarCamvio(rutaMovimiento, ubicaciones)` esta función recibe 2 paramatros, la ruta del archivo .mov y el diccionario leído con la función `leer Archivolnv(ruta)` y regresa el inventario actualizado con los cambios de productos si son posibles o avisar por que no son posibles

```
#Clase para identificar los movimientos del inventario y sus respectivas acciones
#recibe 2 atributos la ruta del archivo .mov y el inventario cargado del archivo .inv
def cargarCambios(rutaMovimiento, ubicaciones):
    #intento de procesar el archivo .mov
    try:
        with open(rutaMovimiento, encoding="utf-8") as file:
            #leer el archivo en líneas
            for linea in file:
                instruccion, informacion = linea.strip().split(" ")
                producto, cantidad, ubicacion = informacion.split(";")
                #proceso de agregar
                if instruccion == "agregar_stock":
                    if ubicacion not in ubicaciones:
                        print("Esa ubicacion no existe")
                        continue
                    if producto not in ubicaciones[ubicacion]:
                        print("Ese producto no existe en esa ubicacion")
                        continue
                    ubicaciones[ubicacion][producto]['cantidad'] += float(cantidad)
                #procesos de vender
                elif instruccion == "vender_producto":
                    if ubicacion not in ubicaciones:
                        print("Esa ubicacion no existe")
                    if ubicacion not in ubicaciones or producto not in ubicaciones[ubicacion]:
                        print("Ese producto no esta en esa ubicacion")
                    cantidadVender = float(cantidad)
                    cantidadDisponible = ubicaciones[ubicacion][producto]['cantidad']
                    if cantidadVender > cantidadDisponible:
                        print("Error: La cantidad de producto no está disponible", producto, cantidad, ubicacion, informacion)
                    else:
                        ubicaciones[ubicacion][producto]['cantidad'] -= float(cantidadVender)
    return ubicaciones
except FileNotFoundError:
    print("El archivo no fue encontrado.")
    getCargarInstrucciones()
except IOError:
    print("Error al leer el archivo.")
    getCargarInstrucciones()
```

otra función es actualizarInventario(rutaInventario, inventario), esta función recibe 2 parametros la ruta del archivo .inv y el inventario actualizado con los cambios de productos, y reescribe el archivo .inv

```
#clase para reescribir el inventario con los cambios de agregar y ventas
def actualizarInventario(rutaInventario, inventario):
    #intento de reescritura
    try:
        with open(rutaInventario, "w") as archivo_inv:
            for ubicacion, ubicaciones in inventario.items():
                for producto, detalles in ubicaciones.items():
                    cantidad = detalles["cantidad"]
                    precio = detalles["precio"]
                    archivo_inv.write(f"crear_producto {producto};{cantidad};{precio};{ubicacion}\n")
            print("Archivo de inventario actualizado correctamente.")
    except IOError:
        print("Error al escribir en el archivo de inventario.")
        getCargarInstrucciones()
```

La función principal de esta clase es getCargarInstrucciones(): esta se encarga de preguntar la dirección de los archivos .inv y .mov y enviarlos a las funciones que los piden como leerArchivoInv(rutaInventario) y cargarCambios(rutaMovimiento, inventario), imprimirInventario(inventario) y actualizar inventario.

4). crearInventario.py esta clase se encarga de crear un informe con los productos de inventario, dentro de ella hay varias funciones una de ellas es salida(inventario) recibe el inventario actualizado del archivo .inv con los cambios de productos

```
#impresion del informe de inventario
def salida(inventario):
    print("Producto"+" "+"Cantidad"+" "+"Precio unitario"+" "+"Valor total"+" "+"Ubicacion")
    print("-----")
    for producto, ubicaciones in inventario.items():
        for ubicacion, detalles in ubicaciones.items():
            cantidad = detalles["cantidad"]
            precio = detalles["precio"]
            print(f"<producto> <cantidad> ${precio:.2f} ${cantidad * precio:.2f} <ubicacion>")
```

Otra función de es crearInforme(inventario) recibe de parámetro el inventario y escribe un archivo .txt con el resumen del inventario

```
#clase para crear el archivo .txt con el informe
def crearInforme(inventario):
    fecha=str(date.today())+".txt"
    try:
        with open(fecha, "w") as archivo:
            archivo.write("Producto Cantidad Precio Unitario Valor Total Ubicacion\n")
            archivo.write("-----\n")
            for producto, ubicaciones in inventario.items():
                for ubicacion, detalles in ubicaciones.items():
                    cantidad = detalles["cantidad"]
                    precio = detalles["precio"]
                    linea = f"<producto> <cantidad>      ${precio:.2f}      ${cantidad * precio:.2f}"
                    archivo.write(linea)
            print("Inventario guardado en el archivo correctamente.")
    except IOError:
        print("Error al escribir informe.")
```

La función principal es setInventario() se encargar de pedir la ruta del archivo .inv y envía los parámetros a sus respectivas funciones al finalizar da la opción de regresar al menú principal.

```
from datetime import date
#clase llamada en la opcion 3 del inventario
def setInventario():
    #importacion de clases llamadas mas adelante
    from menuPrincipal import getMenuPrincipal
    from cargarInventario import leerArchivosInv
    #bienvenida del formulario
    print("-----")
    print("Informe de Inventario")
    print("-----")
    #captura de direcciones
    print("Ingrese la ruta relativa del inventario")
    ruta=input()
    inventario=leerArchivosInv(ruta)
    salida(inventario)
    crearInforme(inventario)
    #salida del formulario
    print("-----")
    print("Presione cualquier tecla para continuar")
    input()
    getMenuPrincipal()
```