

Practica 1

# Manual técnico

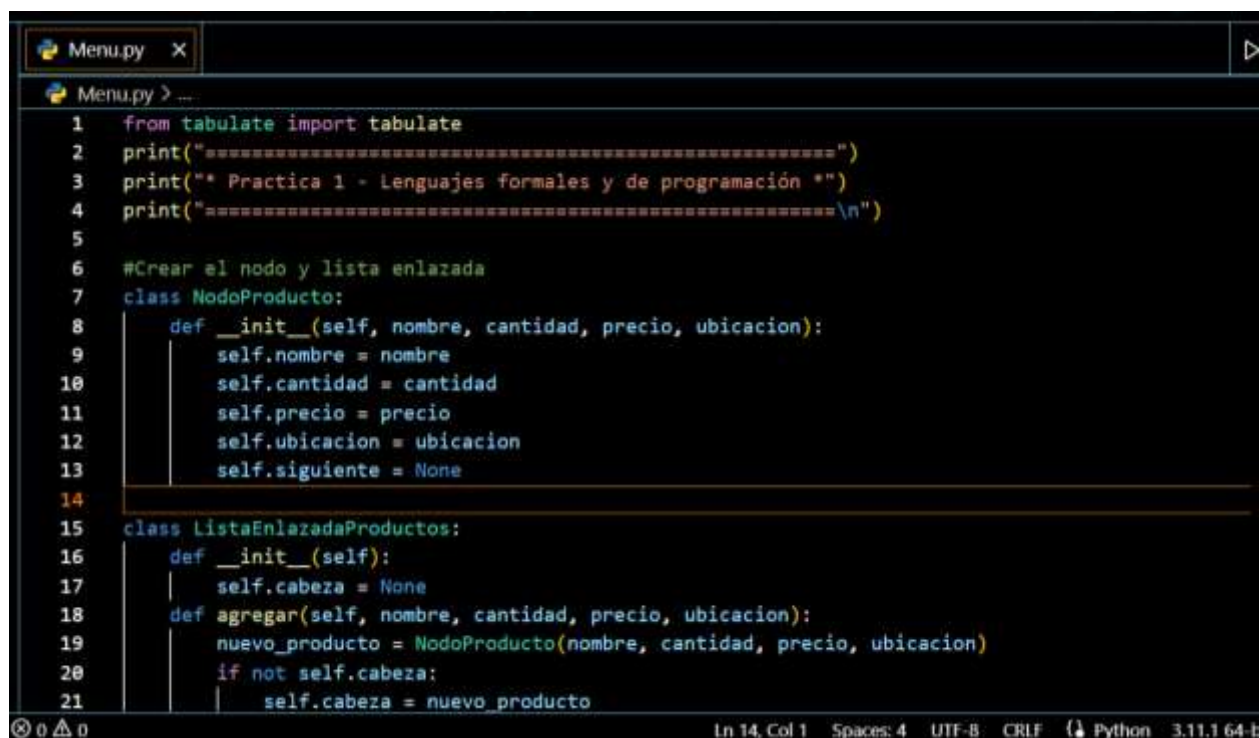
Lenguajes formales y de programación

## Manual tecnico

Este manual explica la logica del programa de python el cual tiene como fin una solución de software con base en los distintos paradigmas de programación vistos en clase y laboratorio. Y donde se utiliza el manejo de archivos, lógica de programación y manipulación de estructuras de datos en Python.

Para iniciar se establecio que se trabajaria con listas enlazadas para poder manejar los datos de los productos que estarian en un archivo diferente el cual se deberia de leer y sacar cada producto con su nombre, cantidad, precio y su ubicación.

Para esto creamos un Nodo lo cual nos permite trabajar con listas enlazadas.  
Se realizo de esta forma.

A screenshot of a Python IDE window titled 'Menu.py'. The code defines two classes: 'NodoProducto' and 'ListaEnlazadaProductos'. 'NodoProducto' has attributes 'nombre', 'cantidad', 'precio', 'ubicacion', and 'siguiente'. 'ListaEnlazadaProductos' has a 'cabeza' attribute and an 'agregar' method that creates a new 'NodoProducto' and adds it to the list. The code is as follows:

```
1 from tabulate import tabulate
2 print("=====")
3 print("* Practica 1 - Lenguajes formales y de programación *")
4 print("=====\\n")
5
6 #Crear el nodo y lista enlazada
7 class NodoProducto:
8     def __init__(self, nombre, cantidad, precio, ubicacion):
9         self.nombre = nombre
10        self.cantidad = cantidad
11        self.precio = precio
12        self.ubicacion = ubicacion
13        self.siguiente = None
14
15 class ListaEnlazadaProductos:
16     def __init__(self):
17         self.cabeza = None
18     def agregar(self, nombre, cantidad, precio, ubicacion):
19         nuevo_producto = NodoProducto(nombre, cantidad, precio, ubicacion)
20         if not self.cabeza:
21             self.cabeza = nuevo_producto
```

Se creo el NodoProducto con y se definio los datos que se utilizaran (nombre, cantidad, precio y ubicación)

Se define self.siguiente = none

Ya que es un argumento el cual se utiliza en las listas enlazadas.

Luego definimos las funciones a utilizar para manipular nuestra lista los cuales son:

- ListaEnLazadaProductos: Con esta definimos la cabeza de el nodo
- Agregar: Esta funcion nos sirve para agregar datos en la lista.
- Mostrar: Esta funcion se utilizo para visualizar como se estaba colocando los productos en la lista.

```
Menu.py x
Menu.py > ListaEnlazadaProductos
15 class ListaEnlazadaProductos:
16     def __init__(self):
17         self.cabeza = None
18     def agregar(self, nombre, cantidad, precio, ubicacion):
19         nuevo_producto = NodoProducto(nombre, cantidad, precio, ubicacion)
20         if not self.cabeza:
21             self.cabeza = nuevo_producto
22             return
23         actual = self.cabeza
24         while actual.siguiente:
25             actual = actual.siguiente
26         actual.siguiente = nuevo_producto
27 #Funcion para ver como se estaba ingresando la lista
28     def mostrar(self):
29         actual = self.cabeza
30         while actual:
31             print(f"Nombre: {actual.nombre}, Cantidad: {actual.cantidad}, Precio: {actual.precio}")
32             actual = actual.siguiente
33     def __str__(self) -> str:
34         pass
35
```

Ln 15, Col 30 Spaces: 4 UTF-8 CRLF Python 3.11.1 64-bit

Con esto obtenemos la lista enlazada a la cual ingresaremos los productos, a estos productos se le agregaran productos y se venderan productos de la misma. Entonces definimos las funciones para agregar productos y vender productos de la siguiente forma.

- Buscar\_producto: Con esta definicion nos ubicamos en el lugar que ocupa el producto que se quiere modificar.
- Agregar\_stock: Con esta agregamos productos validando que la cantidad que se agregara se sume a la que ya se encuentra en el inventario (lista enlazada).

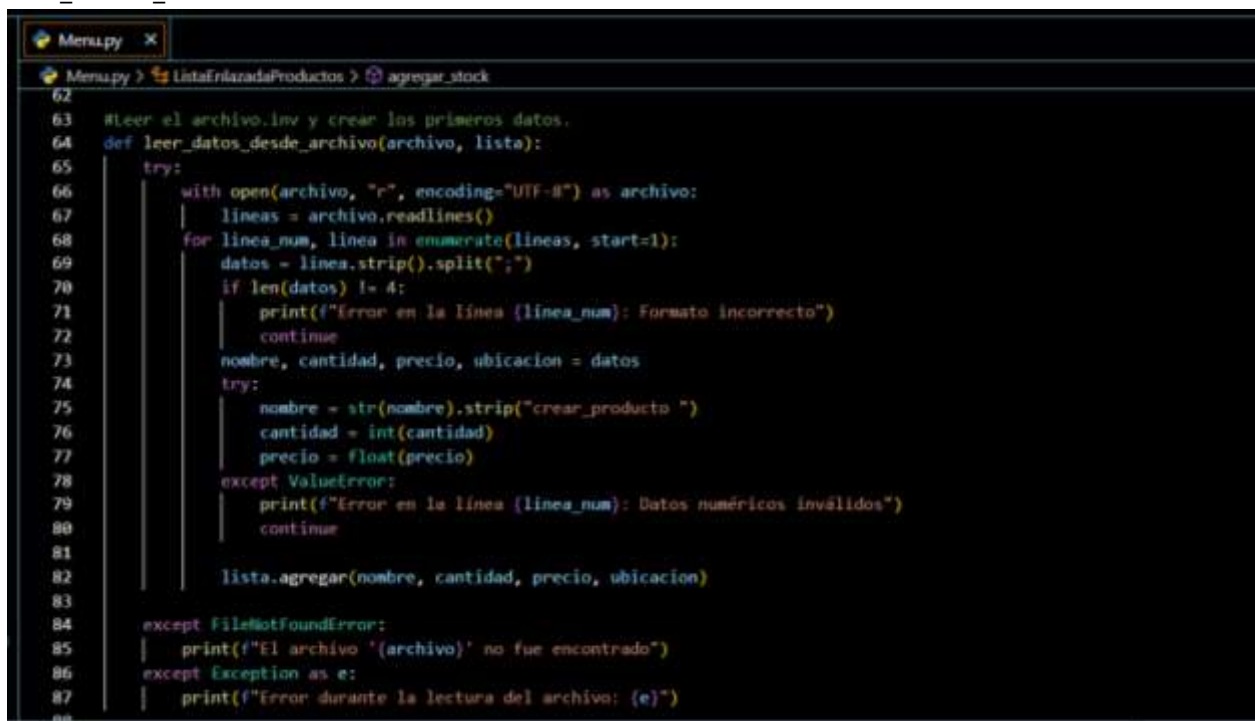
```
Menu.py x
Menu.py > ListaEnlazadaProductos
35
36 # Métodos para agregar y vender productos
37 def buscar_producto(self, nombre):
38     actual = self.cabeza
39     while actual:
40         if actual.nombre == nombre:
41             return actual
42         actual = actual.siguiente
43     return None
44
45 def agregar_stock(self, nombre, cantidad_a_agregar, ubicacion_nueva):
46     producto = self.buscar_producto(nombre)
47     if producto:
48         producto.cantidad += cantidad_a_agregar
49         producto.ubicacion = ubicacion_nueva
50     else:
51         print("Producto no encontrado.")
52
```

- Vender\_producto: Con esta funcion validamos que la cantidad a vender sea menor o igual a la cantidad en inventario y que la ubicación sea la misma.

```
48         producto.cantidad += cantidad_a_agregar
49         producto.ubicacion = ubicacion_nueva
50     else:
51         print("Producto no encontrado.")
52
53     def vender_producto(self, nombre, cantidad_a_vender, verificar_ubi):
54         producto = self.buscar_producto(nombre)
55         if producto:
56             if producto.cantidad >= cantidad_a_vender and producto.ubicacion == verificar_ubi:
57                 producto.cantidad = producto.cantidad - cantidad_a_vender
58             else:
59                 print("No hay suficiente stock para la venta.")
60         else:
61             print("Producto no encontrado.")
62
```

Ahora teniendo definidas las funciones que utilizaremos para agregar y vender se paso a leer el archivo **.inv** para definir los productos que encontraremos en la lista enlazada.

#### Leer\_datos\_desde\_archivo:



```
62
63 #Leer el archivo.inv y crear los primeros datos.
64 def leer_datos_desde_archivo(archivo, lista):
65     try:
66         with open(archivo, "r", encoding="UTF-8") as archivo:
67             lineas = archivo.readlines()
68             for linea_num, linea in enumerate(lineas, start=1):
69                 datos = linea.strip().split(";")
70                 if len(datos) != 4:
71                     print(f"Error en la línea {linea_num}: Formato incorrecto")
72                     continue
73                 nombre, cantidad, precio, ubicacion = datos
74                 try:
75                     nombre = str(nombre).strip("crear_producto ")
76                     cantidad = int(cantidad)
77                     precio = float(precio)
78                 except ValueError:
79                     print(f"Error en la línea {linea_num}: Datos numéricos inválidos")
80                     continue
81                 lista.agregar(nombre, cantidad, precio, ubicacion)
82
83     except FileNotFoundError:
84         print(f"El archivo '{archivo}' no fue encontrado")
85     except Exception as e:
86         print(f"Error durante la lectura del archivo: {e}")
87
```

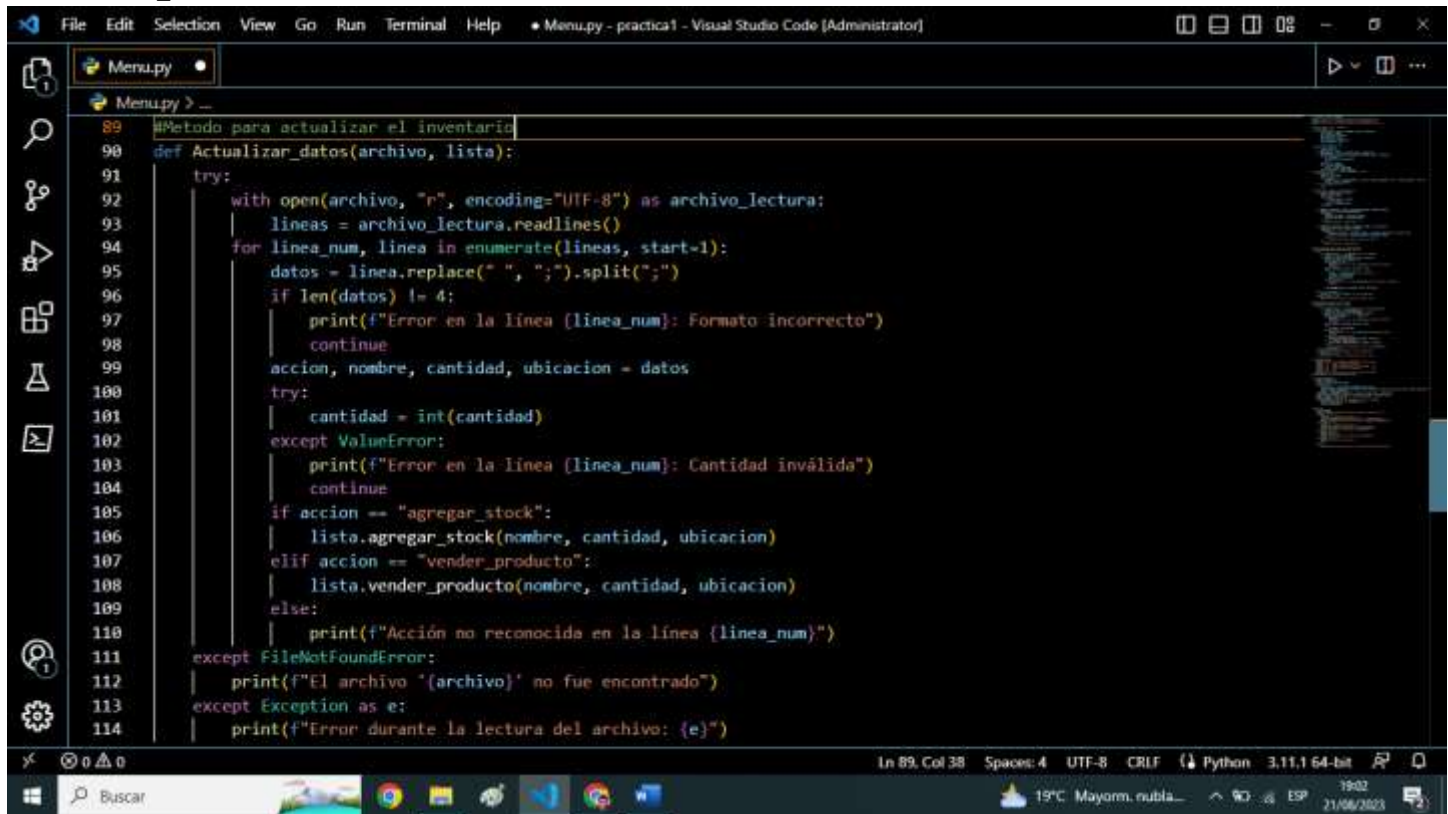
Primero leemos el archivo, este lo leemos línea por línea y separamos cada palabra de esa línea con **.split(";")**. Con esto partimos la línea en 4 partes de las cuales se obtiene nombre, cantidad, precio, ubicación.

La primera parte de cada producto nos queda como "crear\_producto nombre".

Para obtener el nombre utilizamos **.strip("crear\_producto")**.

Con el **try** nos aseguramos que si hay errores en el archivo se pueda seguir corriendo el programa

## Actualizar\_datos:



```
89 #Metodo para actualizar el inventario
90 def Actualizar_datos(archivo, lista):
91     try:
92         with open(archivo, "r", encoding="UTF-8") as archivo_lectura:
93             lineas = archivo_lectura.readlines()
94             for linea_num, linea in enumerate(lineas, start=1):
95                 datos = linea.replace(" ", ";").split(";")
96                 if len(datos) != 4:
97                     print(f"Error en la línea {linea_num}: Formato incorrecto")
98                     continue
99                 accion, nombre, cantidad, ubicacion = datos
100                 try:
101                     cantidad = int(cantidad)
102                 except ValueError:
103                     print(f"Error en la línea {linea_num}: Cantidad inválida")
104                     continue
105                 if accion == "agregar_stock":
106                     lista.agregar_stock(nombre, cantidad, ubicacion)
107                 elif accion == "vender_producto":
108                     lista.vender_producto(nombre, cantidad, ubicacion)
109                 else:
110                     print(f"Acción no reconocida en la línea {linea_num}")
111             except FileNotFoundError:
112                 print(f"El archivo '{archivo}' no fue encontrado")
113             except Exception as e:
114                 print(f"Error durante la lectura del archivo: {e}")
```

Con esta funcion leemos el archivo **.mov** primero utilizamos el try nuevamente para asegurarnos de que el programa siga corriendo aunque el archivo tenga errores.

Leemos el archivo linea por linea y al igual que el anterior utilizamos el **replace(" ", ";").strip(";")** Para quitar el espacio y poner un ";" y poder dividir cada linea en 4 partes : accion, nombre, cantidad, ubicación para guardarlos en la variable datos

Ya que aquí tenemos 2 acciones diferentes las cuales son agregar\_stock y vender\_producto. Tenemos la parte accion la cual comparamos y utilizamos para actualizar nuestro inventario. Despues de validar cual de las acciones se realizara utilizamos las funciones previamente definidas para trabajar con estos datos del archivo de movimientos.

Ahora definimos el menu que mostraremos al usuario:

Con este menu damos 4 opciones al usuario para que el pueda realizar lo que desee.

Luego creamos una lista con la cual trabajaremos nuestra lista enlazada  
Esto es bastante importante para el funcionamiento de nuestro programa

**Lista\_productos = ListaEnlazadaProductos()**



```

Menu.py > Generar_reporte
113     except Exception as e:
114         print(f"Error durante la lectura del archivo: {e}")
115
116     def mostrar_menu():
117         print("===== SISTEMA DE INVENTARIO =====")
118         print("| 1|  Cargar inventario inicial      |")
119         print("| 2|  Cargar instrucciones de movimientos |")
120         print("| 3|  Crear informe de inventario      |")
121         print("| 4|  Salir                          |")
122         print("=====")
123     lista_productos = ListaEnlazadaProductos()
124

```

Definimos la funcion para Generar\_reporte

Creamos la lista la cual utilizaremos para sacar los datos

Recorremos la lista enlazada para llenar nuestra lista datos\_tabulados

Vemos que realizamos una consrante precio\_total, esta nos sirve para obtener el valor total multiplicando la cantidad y el precio de cada producto

Para crear la tabla utilizamos la librería **Tabulate** la cual importamos al inicio del codigo

Esta nos ayuda para crear una tabla brindando el nombre de las columnas en la constante **encabezados**

Luego definimos el nombre del archivo que generaremos

Ahora escribimos la tabla en el archivo creado.

```

122     print("=====")
123     lista_productos = ListaEnlazadaProductos()
124
125     def Generar_reporte():
126         datos_tabulados = []
127         actual = lista_productos.cabeza
128         while actual:
129             precio_total = float(actual.cantidad*actual.precio)
130             datos_tabulados.append([actual.nombre, actual.cantidad, actual.precio, precio_total, actual.ubicacion])
131             actual = actual.siguiente
132         encabezados = ["NOMBRE", "CANTIDAD", "PRECIO", "VALOR TOTAL", "UBICACIÓN"]
133         tabla = tabulate(datos_tabulados, headers=encabezados, tablefmt="grid")
134         nombre_archivo = "Resultados_201901403.txt"
135         with open(nombre_archivo, "w", encoding="UTF-8") as archivo:
136             archivo.write("Informe del Inventario: \n")
137             archivo.write(tabla)
138
139     while True:
140         mostrar_menu()
141         opcion = input("Seleccione una opción (1/2/3) o 4 para salir: ")
142

```

Por ultimo creamos un while el cual nos sirve para brindarle al usuario la opcion que escoja y si esta es una diferente a la que le ofrecemos volver a preguntarle

Si elige la opcion 1 : Esta lee los datos del archivo . inv y los ingresa en nuestra lista\_productos para poder trabajar con estos datos

Si elige la opcion 2 : Esta opcion lee los movimientos del archivo .mov y actualiza la lista\_productos.

Si elige la opcion 3 : Esta opcion genera un archivo Reporte\_201901403 con una tabla la cual tiene registrado los ultimos movimientos realizados en la lista\_productos.

Si elige la opcion 4 : Esta opcion cierra el programa.

```
Menu.py > _
138
139 while True:
140     mostrar_menu()
141     opcion = input("Seleccione una opción (1/2/3) o 4 para salir: ")
142     if opcion == "1":
143         #Ingresar el nombre del archivo .inv
144         leer_datos_desde_archivo("inventario.inv", lista_productos)
145         print("\nInventario con datos del archivo cargado Correctamente\n")
146     elif opcion == "2":
147         #Ingresar el nombre del archivo .mov
148         Actualizar_datos("movimientos.mov", lista_productos)
149         print("\nMovimientos realizados exitosamente\n")
150     elif opcion == "3":
151         Generar_reporte()
152         print("\nSe ha Generado el reporte exitosamente\n")
153     elif opcion == "4":
154         print("\nSaliendo del programa...")
155         break
156     else:
157         print("Opción inválida. Por favor, seleccione una opción válida.")
```