



# UEA

UNIVERSIDAD

ESTATAL AMAZÓNICA

**UNIVERSIDAD ESTATAL AMAZÓNICA**



**INGENIERÍA EN TECNOLOGÍAS DE INFORMACIÓN**

**PROGRAMACIÓN ORIENTADA A OBJETOS**

**TEMA:**

**SISTEMA AVANZADO DE GESTIÓN DE INVENTARIO**

**ESTUDIANTE:**

**EDWIN FABIÁN NORIEGA BALDEON**

**DOCENTE:**

**ING. SANTIAGO ISRAEL NOGALES GUERRERO**

**NIVEL:**

**SEGUNDO NIVEL\_PARALELO "A"**

**2025-2025**

**PUYO-ECUADOR**



# UEA

## UNIVERSIDAD ESTATAL AMAZÓNICA

### 1. CÓDIGO DE GITHUB

[https://github.com/EDFANOBA19/PROGRAMACION\\_ORIENTADA\\_A\\_OBJETOS](https://github.com/EDFANOBA19/PROGRAMACION_ORIENTADA_A_OBJETOS)

[.git](#)

### 2. REPOSITORIO DE GITHUB

PROGRAMACION\_ORIENTADA\_A\_OBJETOS Public

main 1 Branch 0 Tags

Go to file Add file Code

EDFANOBA feat: Implementación completa de sistema de inventario en Python 4fcb462 · 4 minutes ago 27 Commits

idea feat: Añadir clase Cliente con atributos y método para mostr... 2 months ago

CAPTURAS SEMANA 8 Implementación completa del sistema de gestión de inventa... 2 weeks ago

EJERCICIO EN CLASE SEMANA 07 Implementación de ejercicios de POO en Python: last month

SEMANA\_10\_POO\_Sistema de Gestión de Inve... Implementación completa del Sistema de Gestión de Invent... last week

SEMANA\_11\_POO\_Sistema Avanzado de Gesti... feat: Implementación completa de sistema de inventario en ... 4 minutes ago

SEMANA\_2\_POO\_PILARES DE PROGRAMACIO... Add files via upload 2 months ago

SEMANA\_3\_POO\_Comparación de Programaci... Add files via upload 2 months ago

SEMANA\_4\_POO\_EJEMPLUMUNDOREAL Merge remote-tracking branch 'origin/main' last month

SEMANA\_5\_POO\_Tipos de datos, Identificadores Add files via upload 2 months ago

SEMANA\_6\_POO\_Clases, objetos, herencia, enc... Add files via upload last month

SEMANA\_7\_POO\_Constructores y Destruidores Merge remote-tracking branch 'origin/main' last month

SEMANA\_9\_POO\_Sistema de Gestión de Inven... Implementación completa del sistema de gestión de inventa... 2 weeks ago

Dashboard.vv Imolementación.comoleta del sistema.de.gestión de inventa... 2 weeks ago

About No description, website, or topics provided.

Readme Activity 0 stars 0 watching 0 forks

Releases No releases published Create a new release

Packages No packages published Publish your first package

Contributors 2 EDFANOBA EDFANOBA19

Files main + Q Go to file

SEMANA\_11\_POO\_Sistema Avanz... inventario.py inventario.txt menu.py producto.py

PROGRAMACION\_ORIENTADA\_A\_OBJETOS / SEMANA\_11\_POO\_Sistema Avanzado de Gestión de Inventario / Add file ...

EDFANOBA feat: Implementación completa de sistema de inventario en Python 4fcb462 · 5 minutes ago History

Name	Last commit message	Last commit date
..		
inventario.py	feat: Implementación completa de sistema de inventario en Python	8 minutes ago
inventario.txt	feat: Implementación completa de sistema de inventario en Python	5 minutes ago
menu.py	feat: Implementación completa de sistema de inventario en Python	8 minutes ago
producto.py	feat: Implementación completa de sistema de inventario en Python	8 minutes ago



## 2.1 DESARROLLO DE CÓDIGO CON SUS DIFERENTES PEDIDOS Y REQUISITOS

```
1 {  
2     "id": "P001",  
3     "nombre": "LAPTOPS",  
4     "cantidad": 200,  
5     "precio": 500.0  
6 },  
7 {  
8     "id": "P002",  
9     "nombre": "CELULARES",  
10    "cantidad": 1,  
11    "precio": 250.0  
12 },  
13 {  
14    "id": "P003",  
15    "nombre": "MOUSE",  
16    "cantidad": 10,  
17    "precio": 25.0  
18 },  
19 {  
20    "id": "P004",  
21    "nombre": "CABLES USB",  
22    "cantidad": 15,  
23    "precio": 1.0  
24 },  
25 {  
26    "id": "P005",  
27    "nombre": "CARGADORES DE LAPTOP",  
28    "cantidad": 200,  
29    "precio": 25.0  
30 },  
31 }  
32 }
```

**2.1.1 Clase Producto:** Debe contener atributos como ID (único), nombre, cantidad y precio. Implementa métodos para obtener y establecer estos atributos.

```
1 # Clase Producto representa un producto con ID, nombre, cantidad y precio  
2 class Producto: 4 usages new *  
3     def __init__(self, id_producto, nombre, cantidad, precio): new *  
4         self.id = id_producto # ID único del producto  
5         self.nombre = nombre # Nombre del producto  
6         self.cantidad = cantidad # Cantidad disponible  
7         self.precio = precio # Precio unitario  
8  
9     def to_dict(self): 1 usage (1 dynamic) new *  
10 # Convierte el objeto Producto a un diccionario para serialización JSON  
11 return {  
12     "id": self.id,  
13     "nombre": self.nombre,  
14     "cantidad": self.cantidad,  
15     "precio": self.precio  
16 }  
17  
18 def __str__(self): new *  
19 # Representación en texto para mostrar el producto en consola  
20 return f"ID: {self.id} | Nombre: {self.nombre} | Cantidad: {self.cantidad} | Precio: ${self.precio:.2f}"  
21
```

**2.1.2 Clase Inventario:** Debe utilizar una colección adecuada (p. ej., un diccionario) para almacenar los productos. Implementa métodos para:

- Añadir nuevos productos.
- Eliminar productos por ID.



# UEA

## UNIVERSIDAD ESTATAL AMAZÓNICA

- Actualizar cantidad o precio de un producto.
- Buscar y mostrar productos por nombre.
- Mostrar todos los productos en el inventario.

```
inventario.py x
1 import json
2 import os
3 from producto import Producto # Importa la clase Producto
4
5 # Clase Inventario maneja la colección de productos y el almacenamiento en archivo
6 class Inventario: 2 usages new*
7     def __init__(self, archivo='inventario.txt'): new*
8         self.archivo = archivo # Archivo donde se guarda el inventario
9         self.productos = {} # Diccionario de productos: clave=ID, valor=Producto
10        self.cargar_desde_archivo() # Carga productos desde archivo al iniciar
11
12    def cargar_desde_archivo(self): 1 usage new*
13        # Si archivo no existe, crea uno vacío para comenzar
14        if not os.path.exists(self.archivo):
15            self.productos = {}
16            self.guardar_en_archivo()
17            return
18        try:
19            with open(self.archivo, 'r', encoding='utf-8') as f:
20                contenido = f.read().strip()
21                # Si archivo vacío, iniciar inventario vacío
22                if not contenido:
23                    self.productos = {}
24                else:
25                    # Cargar lista JSON y convertir a diccionario de Productos
26                    lista_dicts = json.loads(contenido)
27                    self.productos = {p['id']: Producto(p['id'], p['nombre'], p['cantidad'], p['precio']) for p in lista_dicts}
28            except (json.JSONDecodeError, FileNotFoundError):
29                # Si hay error lectura, crear inventario vacío para evitar fallo
30                self.productos = {}
31                self.guardar_en_archivo()
32
33    def guardar_en_archivo(self): 5 usages new*
34        # Convierte los Productos a diccionarios y guarda como JSON indentado en archivo texto
35        lista_dicts = [p.to_dict() for p in self.productos.values()]
36        with open(self.archivo, 'w', encoding='utf-8') as f:
37            json.dump(lista_dicts, f, indent=4)
38
39    def añadir_producto(self, producto): 1 usage new*
40        # Agrega un nuevo producto si su ID no existe
41        if producto.id in self.productos:
42            raise KeyError(f"Producto con ID '{producto.id}' ya existe.")
43        self.productos[producto.id] = producto
44        self.guardar_en_archivo()
45
46    def eliminar_producto(self, id_producto): 1 usage new*
47        # Elimina producto por su ID si existe
48        if id_producto in self.productos:
49            del self.productos[id_producto]
50            self.guardar_en_archivo()
51        else:
52            raise KeyError(f"No existe producto con ID '{id_producto}'.")
53
54    def actualizar_producto(self, id_producto, cantidad=None, precio=None): 1 usage new*
55        # Actualiza cantidad o precio del producto indicado
56        if id_producto not in self.productos:
57            raise KeyError(f"No existe producto con ID '{id_producto}'.")
58        if cantidad is not None:
59            self.productos[id_producto].cantidad = cantidad
60        if precio is not None:
61            self.productos[id_producto].precio = precio
62        self.guardar_en_archivo()
```



```
63
64 def buscar_por_nombre(self, nombre): 1usage new*
65     # Busca productos cuyo nombre contiene el texto indicado (case insensitive)
66     nombre_lower = nombre.lower()
67     return [p for p in self.productos.values() if nombre_lower in p.nombre.lower()]
68
69 def mostrar_todos(self): 1usage new*
70     # Muestra todos los productos presentes en el inventario
71     if not self.productos:
72         print("Inventario vacío.")
73     else:
74         print("Productos en inventario:")
75         for p in self.productos.values():
76             print(p)
77
```

## 2.1.2.1 Menú

```
1 from producto import Producto
2 from inventario import Inventario
3
4 def menu(): 1usage new*
5     inventario = Inventario() # Crear instancia de inventario
6     while True:
7         print("\nSistema de Gestión de Inventario")
8         print("1. Añadir nuevo producto")
9         print("2. Eliminar producto por ID")
10        print("3. Actualizar cantidad o precio de un producto")
11        print("4. Buscar productos por nombre")
12        print("5. Mostrar todos los productos")
13        print("0. Salir")
14        opcion = input("Seleccione opción: ").strip()
15
16        try:
17            if opcion == '1':
18                id_prod = input("ID (único): ").strip()
19                nombre = input("Nombre: ").strip()
20                cantidad = int(input("Cantidad: "))
21                precio = float(input("Precio: "))
22                p = Producto(id_prod, nombre, cantidad, precio)
23                inventario.añadir_producto(p)
24                print("Producto añadido exitosamente.")
25            elif opcion == '2':
26                id_prod = input("ID del producto a eliminar: ").strip()
27                inventario.eliminar_producto(id_prod)
28                print("Producto eliminado con éxito.")
29            elif opcion == '3':
30                id_prod = input("ID del producto a actualizar: ").strip()
31                cantidad_input = input("Nueva cantidad (deje vacío para no modificar): ").strip()
32                precio_input = input("Nuevo precio (deje vacío para no modificar): ").strip()
```



```
33     cantidad = int(cantidad_input) if cantidad_input else None
34     precio = float(precio_input) if precio_input else None
35     inventario.actualizar_producto(id_prod, cantidad, precio)
36     print("Producto actualizado.")
37     elif opcion == '4':
38         nombre = input("Nombre o parte del nombre para buscar: ").strip()
39         resultados = inventario.buscar_por_nombre(nombre)
40         if resultados:
41             print(f"Se encontraron {len(resultados)} productos:")
42             for p in resultados:
43                 print(p)
44         else:
45             print("No se encontraron productos.")
46     elif opcion == '5':
47         inventario.mostrar_todos()
48     elif opcion == '0':
49         print("Saliendo del programa.")
50         break
51     else:
52         print("Opción inválida, intente de nuevo.")
53 except ValueError:
54     print("Error: cantidad y precio deben ser números válidos.")
55 except KeyError as e:
56     print(e)
57
58 if __name__ == "__main__":
59     menu()
60
```

**2.1.3. Integración de Colecciones:** Utiliza colecciones para optimizar las operaciones del inventario, como búsqueda rápida de productos y manejo eficiente de los datos.

```
4 def menu(): 1usage new*
5     inventario = Inventario() # Crear instancia de inventario
6     while True:
7         print("\nSistema de Gestión de Inventario")
8         print("1. Añadir nuevo producto")
9         print("2. Eliminar producto por ID")
10        print("3. Actualizar cantidad o precio de un producto")
11        print("4. Buscar productos por nombre")
12        print("5. Mostrar todos los productos")
13        print("0. Guardar y Salir")
14        opcion = input("Seleccione opción: ").strip()
15
16        elif opcion == '4':
17            nombre = input("Nombre o parte del nombre para buscar: ").strip()
18            resultados = inventario.buscar_por_nombre(nombre)
19            if resultados:
20                print(f"Se encontraron {len(resultados)} productos:")
21                for p in resultados:
22                    print(p)
23            else:
24                print("No se encontraron productos.")
25
```





# UEA

## UNIVERSIDAD ESTATAL AMAZÓNICA

**2.1.4. Almacenamiento en Archivos:** Implementa funciones para guardar y cargar el inventario desde archivos. Esto incluye la serialización de la colección del inventario para su almacenamiento y la deserialización al cargar el programa.

```
4 def menu(): usage new *
5     inventario = Inventario() # Crear instancia de inventario
6     while True:
7         print("\nSistema de Gestión de Inventario")
8         print("1. Añadir nuevo producto")
9         print("2. Eliminar producto por ID")
10        print("3. Actualizar cantidad o precio de un producto")
11        print("4. Buscar productos por nombre")
12        print("5. Mostrar todos los productos")
13        print("0. Guardar y Salir")
14        opcion = input("Seleccione opción: ").strip()
15
48        elif opcion == '0':
49            print("Guardando y Saliendo del programa.")
50            break
51        else:
52            print("Opción inválida, intente de nuevo.")
53    except ValueError:
54        print("Error: cantidad y precio deben ser números válidos.")
55    except KeyError as e:
56        print(e)
```

**2.1.5. Interfaz de Usuario:** Crea un menú interactivo en la consola que permita al usuario realizar operaciones sobre el inventario (añadir, eliminar, actualizar, buscar, mostrar).

The screenshot shows a terminal window titled "Run" with a Python icon and "menu (1)". The output of the program is displayed in a dark-themed console. A red rectangle highlights the menu options:

```
Sistema de Gestión de Inventario
1. Añadir nuevo producto
2. Eliminar producto por ID
3. Actualizar cantidad o precio de un producto
4. Buscar productos por nombre
5. Mostrar todos los productos
0. Guardar y Salir
Seleccione opción:
```



### 3. INSTRUCCIONES DE LA TAREA:

- Utiliza PyCharm para desarrollar tu solución. Asegúrate de que tu código esté bien organizado y comentado adecuadamente para explicar la lógica detrás de tus decisiones de diseño.

#### 3.1 CÓDIGO COMPLETO

##### 3.1.1 Class Producto

```
producto.py x inventario.py menu.py inventario.txt
1 # Clase Producto representa un producto con ID, nombre, cantidad y precio
2 class Producto: 4 usages new *
3     def __init__(self, id_producto, nombre, cantidad, precio): new *
4         self.id = id_producto # ID único del producto
5         self.nombre = nombre # Nombre del producto
6         self.cantidad = cantidad # Cantidad disponible
7         self.precio = precio # Precio unitario
8
9     def to_dict(self): 1 usage (1 dynamic) new *
10        # Convierte el objeto Producto a un diccionario para serialización JSON
11        return {
12            "id": self.id,
13            "nombre": self.nombre,
14            "cantidad": self.cantidad,
15            "precio": self.precio
16        }
17
18    def __str__(self): new *
19        # Representación en texto para mostrar el producto en consola
20        return f"ID: {self.id} | Nombre: {self.nombre} | Cantidad: {self.cantidad} | Precio: ${self.precio:.2f}"
21
```

##### 3.1.2 Class Inventario

```
producto.py x inventario.py x menu.py inventario.txt
1 import json
2 import os
3 from producto import Producto # Importa la clase Producto
4
5 # Clase Inventario maneja la colección de productos y el almacenamiento en archivo
6 class Inventario: 2 usages new *
7     def __init__(self, archivo='inventario.txt'): new *
8         self.archivo = archivo # Archivo donde se guarda el inventario
9         self.productos = {} # Diccionario de productos: clave=ID, valor=Producto
10        self.cargar_desde_archivo() # Carga productos desde archivo al iniciar
11
12    def cargar_desde_archivo(self): 1 usage new *
13        # Si archivo no existe, crea uno vacío para comenzar
14        if not os.path.exists(self.archivo):
15            self.productos = {}
16            self.guardar_en_archivo()
17            return
18        try:
19            with open(self.archivo, 'r', encoding='utf-8') as f:
20                contenido = f.read().strip()
21                # Si archivo vacío, iniciar inventario vacío
22                if not contenido:
23                    self.productos = {}
24                else:
25                    # Cargar lista JSON y convertir a diccionario de Productos
26                    lista_dicts = json.loads(contenido)
27                    self.productos = {p['id']: Producto(p['id'], p['nombre'], p['cantidad'], p['precio']) for p in lista_dicts}
28            except (json.JSONDecodeError, FileNotFoundError):
29                # Si hay error lectura, crear inventario vacío para evitar fallo
30                self.productos = {}
31                self.guardar_en_archivo()
32
```





# UEA

## UNIVERSIDAD ESTATAL AMAZÓNICA

```
34 def guardar_en_archivo(self): 1usage new *
35 # Convierte los Productos a diccionarios y guarda como JSON indentado en archivo texto
36 lista_dicts = [p.to_dict() for p in self.productos.values()]
37 with open(self.archivo, 'w', encoding='utf-8') as f:
38     json.dump(lista_dicts, f, indent=4)
39
40 def añadir_producto(self, producto): 1usage new *
41 # Agrega un nuevo producto si su ID no existe
42 if producto.id in self.productos:
43     raise KeyError(f"Producto con ID '{producto.id}' ya existe.")
44 self.productos[producto.id] = producto
45 self.guardar_en_archivo()
46
47 def eliminar_producto(self, id_producto): 1usage new *
48 # Elimina producto por su ID si existe
49 if id_producto in self.productos:
50     del self.productos[id_producto]
51     self.guardar_en_archivo()
52 else:
53     raise KeyError(f"No existe producto con ID '{id_producto}'.")
54
55 def actualizar_producto(self, id_producto, cantidad=None, precio=None): 1usage new *
56 # Actualiza cantidad o precio del producto indicado
57 if id_producto not in self.productos:
58     raise KeyError(f"No existe producto con ID '{id_producto}'.")
59 if cantidad is not None:
60     self.productos[id_producto].cantidad = cantidad
61 if precio is not None:
62     self.productos[id_producto].precio = precio
63 self.guardar_en_archivo()
64
65 def buscar_por_nombre(self, nombre): 1usage new *
66 # Busca productos cuyo nombre contiene el texto indicado (case insensitive)
67 nombre_lower = nombre.lower()
68 return [p for p in self.productos.values() if nombre_lower in p.nombre.lower()]
69
70 def mostrar_todos(self): 1usage new *
71 # Muestra todos los productos presentes en el inventario
72 if not self.productos:
73     print("Inventario vacío.")
74 else:
75     print("Productos en inventario:")
76     for p in self.productos.values():
77         print(p)
```



# UEA

## UNIVERSIDAD ESTATAL AMAZÓNICA

### 3.1.3 Menú

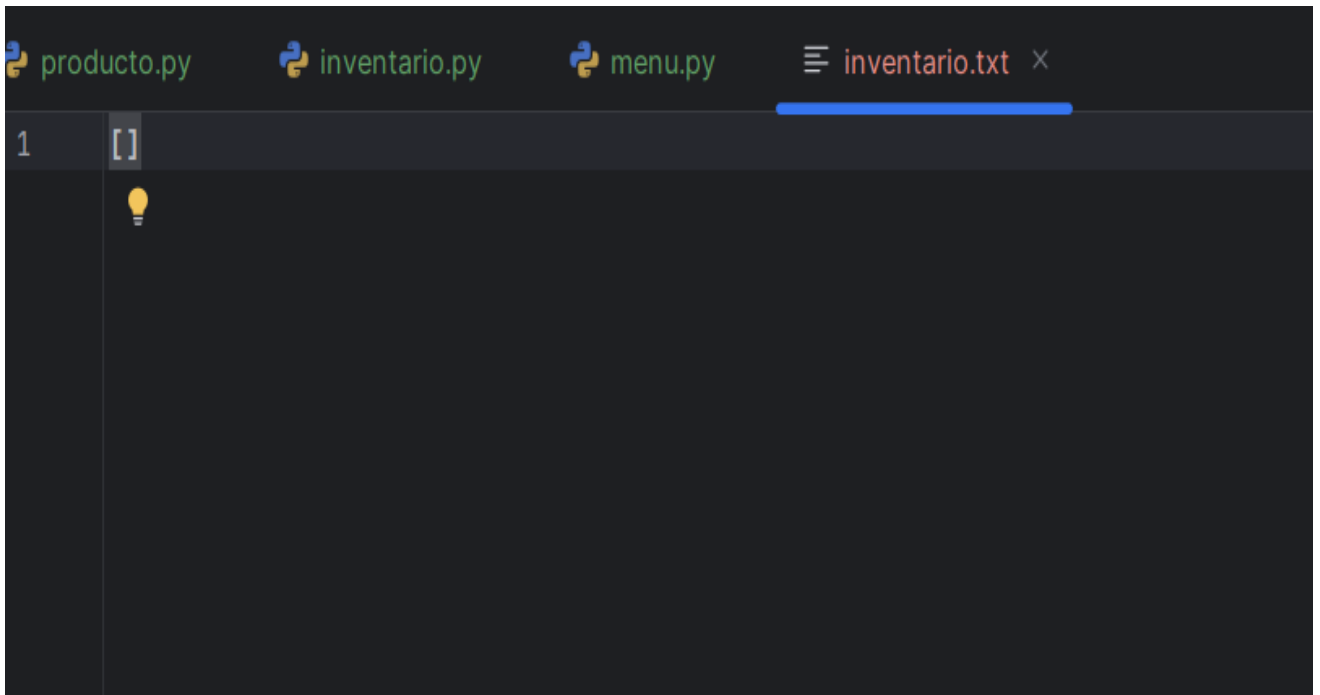
```
producto.py inventario.py menu.py x inventario.txt
1 from producto import Producto
2 from inventario import Inventario
3
4 def menu(): 1 usage new *
5     inventario = Inventario() # Crear instancia de inventario
6     while True:
7         print("\nSistema de Gestión de Inventario")
8         print("1. Añadir nuevo producto")
9         print("2. Eliminar producto por ID")
10        print("3. Actualizar cantidad o precio de un producto")
11        print("4. Buscar productos por nombre")
12        print("5. Mostrar todos los productos")
13        print("0. Guardar y Salir")
14        opcion = input("Seleccione opción: ").strip()
15
16    try:
17        if opcion == '1':
18            id_prod = input("ID (único): ").strip()
19            nombre = input("Nombre: ").strip()
20            cantidad = int(input("Cantidad: "))
21            precio = float(input("Precio: "))
22            p = Producto(id_prod, nombre, cantidad, precio)
23            inventario.agregar_producto(p)
24            print("Producto añadido exitosamente.")
25        elif opcion == '2':
26            id_prod = input("ID del producto a eliminar: ").strip()
27            inventario.eliminar_producto(id_prod)
28            print("Producto eliminado con éxito.")
29        elif opcion == '3':
30            id_prod = input("ID del producto a actualizar: ").strip()
31            cantidad_input = input("Nueva cantidad (deje vacío para no modificar): ").strip()
32            precio_input = input("Nuevo precio (deje vacío para no modificar): ").strip()
33
34            cantidad = int(cantidad_input) if cantidad_input else None
35            precio = float(precio_input) if precio_input else None
36            inventario.actualizar_producto(id_prod, cantidad, precio)
37            print("Producto actualizado.")
38        elif opcion == '4':
39            nombre = input("Nombre o parte del nombre para buscar: ").strip()
40            resultados = inventario.buscar_por_nombre(nombre)
41            if resultados:
42                print(f"Se encontraron {len(resultados)} productos:")
43                for p in resultados:
44                    print(p)
45            else:
46                print("No se encontraron productos.")
47        elif opcion == '5':
48            inventario.mostrar_todos()
49        elif opcion == '0':
50            print("Guardando y Saliendo del programa.")
51            break
52        else:
53            print("Opción inválida, intente de nuevo.")
54    except ValueError:
55        print("Error: cantidad y precio deben ser números válidos.")
56    except KeyError as e:
57        print(e)
58
59 if __name__ == "__main__":
60     menu()
```



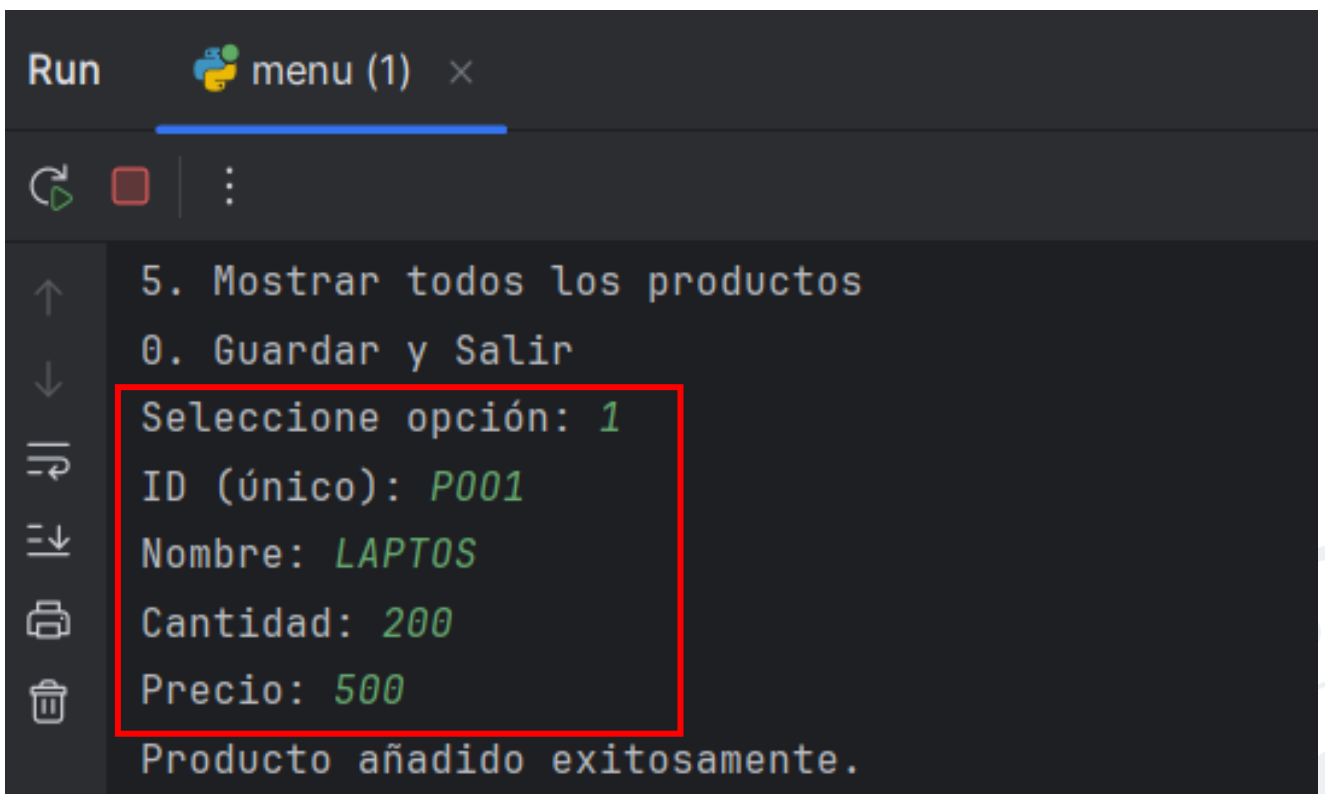
# UEA

## UNIVERSIDAD ESTATAL AMAZÓNICA

- Prueba todas las funcionalidades de tu programa para asegurarte de que funcionan correctamente.
- **INVENTARIO TXT DICCIONARIO**



### 1. AÑADIR PRODUCTOS





## 1.1 Visualización de diccionario

```
producto.py inventario.py menu.py inventario.txt x
1  [
2      {
3          "id": "P001",
4          "nombre": "LAPTOPS",
5          "cantidad": 200,
6          "precio": 500.0
7      },
8      {
9          "id": "P002",
10         "nombre": "CELULARES",
11         "cantidad": 1,
12         "precio": 250.0
13     },
14     {
15         "id": "P003",
16         "nombre": "MOUSE",
17         "cantidad": 10,
18         "precio": 25.0
19     },
20     {
21         "id": "P004",
22         "nombre": "CABLES USB",
23         "cantidad": 15,
24         "precio": 1.0
25     },
26     {
27         "id": "P005",
28         "nombre": "CARGADORES DE LAPTOP",
29         "cantidad": 200,
30         "precio": 25.0
31     }
32 ]
```



## 2. ELIMINAR PRODUCTO POR ID

```
Run menu (1) x
5. Mostrar todos los productos
0. Guardar y Salir
Seleccione opción: 2
ID del producto a eliminar: P001
Producto eliminado con éxito.

Sistema de Gestión de Inventario
1. Añadir nuevo producto
2. Eliminar producto por ID
```

```
producto.py x inventario.py menu.py inventario.txt x
1 [
2     {
3         "id": "P002",
4         "nombre": "CELULARES",
5         "cantidad": 1,
6         "precio": 250.0
7     },
8     {
9         "id": "P003",
10        "nombre": "MOUSE",
11        "cantidad": 10,
12        "precio": 25.0
13    },
14    {
15        "id": "P004",
16        "nombre": "CABLES USB",
17        "cantidad": 15,
18        "precio": 1.0
19    },
20    {
21        "id": "P005",
22        "nombre": "CARGADORES DE LAPTOP",
23        "cantidad": 200,
24        "precio": 25.0
25    }
26 ]
```



### 3. ACTUALIZAR CANTIDAD O PRECIO DE UN PRODUCTO

```
producto.py inventario.py menu.py inventario.txt x
1  [
2      {
3          "id": "P002",
4          "nombre": "CELULARES",
5          "cantidad": 1,
6          "precio": 250.0
7      },
8      {
9          "id": "P003",
10         "nombre": "MOUSE",
11         "cantidad": 10,
12         "precio": 25.0
13     },
14     {
15         "id": "P004",
16         "nombre": "CABLES USB",
17         "cantidad": 15,
18         "precio": 1.0
19     },
20     {
```

Run menu (1) x

4. Buscar productos por nombre  
5. Mostrar todos los productos  
0. Guardar y Salir

Seleccione opción: 3  
ID del producto a actualizar: P004  
Nueva cantidad (deje vacío para no modificar): 25  
Nuevo precio (deje vacío para no modificar): 2.00  
Producto actualizado.





```
producto.py x inventario.py menu.py inventario.txt x
1  [
2      {
3          "id": "P002",
4          "nombre": "CELULARES",
5          "cantidad": 1,
6          "precio": 250.0
7      },
8      {
9          "id": "P003",
10         "nombre": "MOUSE",
11         "cantidad": 10,
12         "precio": 25.0
13     },
14     {
15         "id": "P004",
16         "nombre": "CABLES USB",
17         "cantidad": 25,
18         "precio": 2.0
19     },
20 ]
```

#### 4. BUSCAR PRODUCTOS POR NOMBRE

```
Run menu (1) x
0. Guardar y Salir
Seleccione opción: 4
Nombre o parte del nombre para buscar: CELULARES
Se encontraron 1 productos:
ID: P002 | Nombre: CELULARES | Cantidad: 1 | Precio: $250.00
Sistema de Gestión de Inventario
1. Añadir nuevo producto
```



## 5. MOSTRAR TODOS LOS PRODUCTOS

```
Run menu (1) x
0. Guardar y Salir
Seleccione opción: 5
Productos en inventario:
ID: P001 | Nombre: LAPTOPS | Cantidad: 200 | Precio: $500.00
ID: P002 | Nombre: CELULARES | Cantidad: 1 | Precio: $250.00
ID: P003 | Nombre: MOUSE | Cantidad: 10 | Precio: $25.00
ID: P004 | Nombre: CABLES USB | Cantidad: 15 | Precio: $1.00
ID: P005 | Nombre: CARGADORES DE LAPTOP | Cantidad: 200 | Precio: $25.00
```

```
producto.py inventario.py menu.py inventario.txt x
1 [
2     {
3         "id": "P001",
4         "nombre": "LAPTOPS",
5         "cantidad": 200,
6         "precio": 500.0
7     },
8     {
9         "id": "P002",
10        "nombre": "CELULARES",
11        "cantidad": 1,
12        "precio": 250.0
13    },
14    {
15        "id": "P003",
16        "nombre": "MOUSE",
17        "cantidad": 10,
18        "precio": 25.0
19    },
20    {
21        "id": "P004",
22        "nombre": "CABLES USB",
23        "cantidad": 15,
24        "precio": 1.0
25    },
26    {
27        "id": "P005",
28        "nombre": "CARGADORES DE LAPTOP",
29        "cantidad": 200,
30        "precio": 25.0
31    }
32 ]
```



## 6. GUARDAR Y SALIR

```
Run  menu (1) x
[Icons]
4. Buscar productos por nombre
5. Mostrar todos los productos
0. Guardar y Salir
Seleccione opción: 0
Guardando y Saliendo del programa.
Process finished with exit code 0
```

- Documenta cómo tu programa utiliza las colecciones para gestionar el inventario y cómo implementaste el almacenamiento en archivos.

### Documentación del Programa de Gestión de Inventario

#### Uso de colecciones para gestionar el inventario

- El inventario se maneja internamente mediante un diccionario de Python.
- Este diccionario usa como clave el ID único de cada producto y como valor un objeto Producto que contiene todos los atributos (nombre, cantidad, precio).
- Gracias a que un diccionario permite acceso rápido en tiempo constante  $O(1)$  por clave, las operaciones de búsqueda, actualización y eliminación se hacen eficientes y directas.
- Para mostrar y listar productos, se itera sobre los valores del diccionario, que son objetos Producto.
- Para buscar por nombre se recorre el diccionario filtrando los productos cuyo nombre contenga el texto proporcionado, con búsqueda insensible a mayúsculas.



Esta estructura facilita la gestión dinámica y eficiente:

- No se permite duplicidad de IDs.
- Actualizar o borrar un producto es inmediato al acceder a su clave.
- Añadir un producto es simplemente asignar una nueva entrada al diccionario.

## Implementación de almacenamiento en archivos

- El inventario se guarda en un archivo texto llamado inventario.txt con formato JSON legible, que es una lista de diccionarios (cada uno representa un producto).
- La serialización a JSON se hace usando:

```
lista_dicts = [p.to_dict() for p in self.productos.values()]  
json.dump(lista_dicts, archivo, indent=4)
```

- La deserialización (carga desde el archivo) se hace leyendo el archivo JSON y recreando los objetos Producto desde los diccionarios con:

```
lista_dicts = json.load(archivo)  
self.productos = {d["id"]: Producto(d["id"], d["nombre"], d["cantidad"], d["precio"]) for d in lista_dicts}
```

- Se incluye manejo de errores para el archivo (archivo no existente, corrupto o vacío), creándolo o inicializando el inventario en caso de problemas para asegurar que el programa funcione sin fallos.
- Cada vez que se añade, elimina o actualiza un producto, se actualiza inmediatamente el archivo JSON para mantener persistencia y evitar pérdida de datos.
- El archivo es fácilmente legible por humanos y puede editarse o consultarse manualmente si se desea.



# UEA

## UNIVERSIDAD ESTATAL AMAZÓNICA

### Ventajas de este enfoque

- Usar un archivo de texto plano con JSON hace que el archivo sea fácilmente legible y portable sin necesidad de herramientas binarias específicas.
- Mantener el inventario en memoria con un diccionario permite operaciones rápidas con acceso a productos por ID.
- La estructura JSON en archivo conserva la semántica y las relaciones de los datos (lista de objetos con atributos).
- Es un formato ideal para interoperabilidad con otras aplicaciones o para migrar datos a bases de datos si el proyecto crece.
- Permite escala sencilla, ya que modificar el archivo en disco es simple y el código es mantenible.



[www.uea.edu.ec](http://www.uea.edu.ec)



Km. 2. 1/2 vía Puyo a Tena (Paso Lateral)



032892-118 / 032892-188 032892-098 / 032896-188 032896-476

**#UEAesExcelencia**