



# UEA

UNIVERSIDAD

ESTATAL AMAZÓNICA

**UNIVERSIDAD ESTATAL AMAZÓNICA**



**INGENIERÍA EN TECNOLOGÍAS DE INFORMACIÓN**

**PROGRAMACIÓN ORIENTADA A OBJETOS**

**TEMA:**

COMPARACIÓN DE PROGRAMACIÓN TRADICIONAL Y POO  
EN PYTHON

**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.git](https://github.com/EDFANOBA19/PROGRAMACION_ORIENTADA_A_OBJETOS.git)

### 2. REPOSITORIO DE GITHUB

The screenshot displays the GitHub repository interface for 'PROGRAMACION\_ORIENTADA\_A\_OBJETOS'. The repository is owned by 'EDFANOBA19'. The main branch is 'main'. The repository contains several files and folders, including '.idea', 'POO\_EJEMPLUMUNDREAL', 'SEMANA\_2\_POO\_PILARES DE PROGRAMACIO...', 'SEMANA\_3\_POO\_Comparación\_de\_Programaci...', and 'README.md'. The 'README.md' file is selected, showing the repository's title and description. The right sidebar provides repository statistics: 0 stars, 0 forks, and 0 releases. The bottom section shows the code for the file 'online\_tienda.py' in the 'POO\_EJEMPLUMUNDREAL' directory. The code defines a 'Producto' class with attributes 'nombre', 'precio', and 'stock', and methods 'mostrar\_info()' and 'actualizar\_stock()'.

### 2.1 EJEMPLOS DEL MUNDO REAL Y CARACTERÍSTICAS DE LA PROGRAMACIÓN ORIENTADA A OBJETOS



En el siguiente ejemplo que se realizó se muestra un sistema básico de una tienda en línea donde se encuentran, productos, pedidos y clientes, en el cual se interactúa para simular una compra.

### 2.1.1. CÓDIGO

```
1 class Producto: 2 usages
2 """
3 Clase que representa un producto en la tienda.
4 Atributos:
5     nombre (str): nombre del producto
6     precio (float): precio unitario del producto
7     stock (int): cantidad disponible en inventario
8 Métodos:
9     mostrar_info(): imprime información del producto
10    actualizar_stock(cantidad): actualiza el stock tras una venta
11 """
12 def __init__(self, nombre, precio, stock):
13     self.nombre = nombre
14     self.precio = precio
15     self.stock = stock
16
17 def mostrar_info(self):
18     print(f"Producto: {self.nombre} | Precio: ${self.precio:2f} | Stock: {self.stock}")
19
20 def actualizar_stock(self, cantidad): 1 usage (1 dynamic)
21     if cantidad <= self.stock:
22         self.stock -= cantidad
23         return True
24     else:
25         print(f"No hay suficiente stock de {self.nombre}.")
26         return False
27
28 class Cliente: 1 usage
29 """
30 Clase que representa un cliente de la tienda.
31 Atributos:
32     nombre (str): nombre del cliente
33     correo (str): correo electrónico del cliente
34 Métodos:
35     mostrar_info(): imprime información del cliente
36 """
37 def __init__(self, nombre, correo):
38     self.nombre = nombre
39     self.correo = correo
40
41 def mostrar_info(self): ...
42
43 class Pedido: 1 usage
44 """
45 Clase que representa un pedido realizado por un cliente.
46 Atributos:
47     cliente (Cliente): cliente que realiza el pedido
48     productos (dict): diccionario con Producto como clave y cantidad como valor
49 Métodos:
50     agregar_producto(producto, cantidad): añade productos al pedido si hay stock
51     mostrar_pedido(): muestra los detalles del pedido y total a pagar
52 """
53 def __init__(self, cliente):
54     self.cliente = cliente
55     self.productos = {}
56
57 def agregar_producto(self, producto, cantidad): 3 usages
58     if producto.actualizar_stock(cantidad):
59         if producto in self.productos:
60             self.productos[producto] += cantidad
61         else:
62             self.productos[producto] = cantidad
63
```



```
class Pedido:
    def __init__(self, cliente, productos):
        self.cliente = cliente
        self.productos = productos

    def agregar_producto(self, producto, cantidad):
        if producto in self.productos:
            self.productos[producto] += cantidad
            print(f"Agregado {cantidad} unidades de {producto.nombre} al pedido.")
        else:
            print(f"No se pudo agregar {producto.nombre} al pedido.")

    def mostrar_pedido(self):
        print(f"Pedido de {self.cliente.nombre}:")
        total = 0
        for producto, cantidad in self.productos.items():
            subtotal = producto.precio * cantidad
            print(f"- {producto.nombre}: {cantidad} x ${producto.precio} = ${subtotal}")
            total += subtotal
        print(f"Total a pagar: ${total}")

# Simulación de uso
if __name__ == "__main__":
    # Crear productos
    p1 = Producto(nombre="Camisa", precio=20.0, stock=10)
    p2 = Producto(nombre="Pantalón", precio=35.0, stock=5)

    # Crear cliente
    cliente1 = Cliente(nombre="Ana Pérez", correo="ana.perez@email.com")

    # Crear pedido
    pedido1 = Pedido(cliente1)

    # Agregar productos al pedido
    pedido1.agregar_producto(p1, cantidad=2)
    pedido1.agregar_producto(p2, cantidad=1)
    pedido1.agregar_producto(p2, cantidad=10) # Intento de agregar más de stock disponible

    # Mostrar pedido
    pedido1.mostrar_pedido()
```

Run online\_tienda

```
No se pudo agregar Pantalón al pedido.
Pedido de Ana Pérez:
- Camisa: 2 x $20.00 = $40.00
- Pantalón: 1 x $35.00 = $35.00
Total a pagar: $75.00
Process finished with exit code 0
```

Este es un sistema que simula la gestión de una tienda online, abarcando productos, clientes y pedidos. Su objetivo es, agilizar las ventas al registrar cada compra. Así, todo su inventario se actualizará automáticamente, evitando vender artículos agotados y manteniendo un control de stock en tiempo real. Además, almacena información de clientes para personalizar el servicio. Al realizar un pedido el sistema verifica de forma inmediata la disponibilidad y calcula su total. Es una herramienta clave para tiendas físicas en plataformas online, asegurando organización y control. Simula situaciones reales, como intentar comprar mas de lo disponible, siendo una base practica para optimizar la administración y el servicio al cliente.