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”

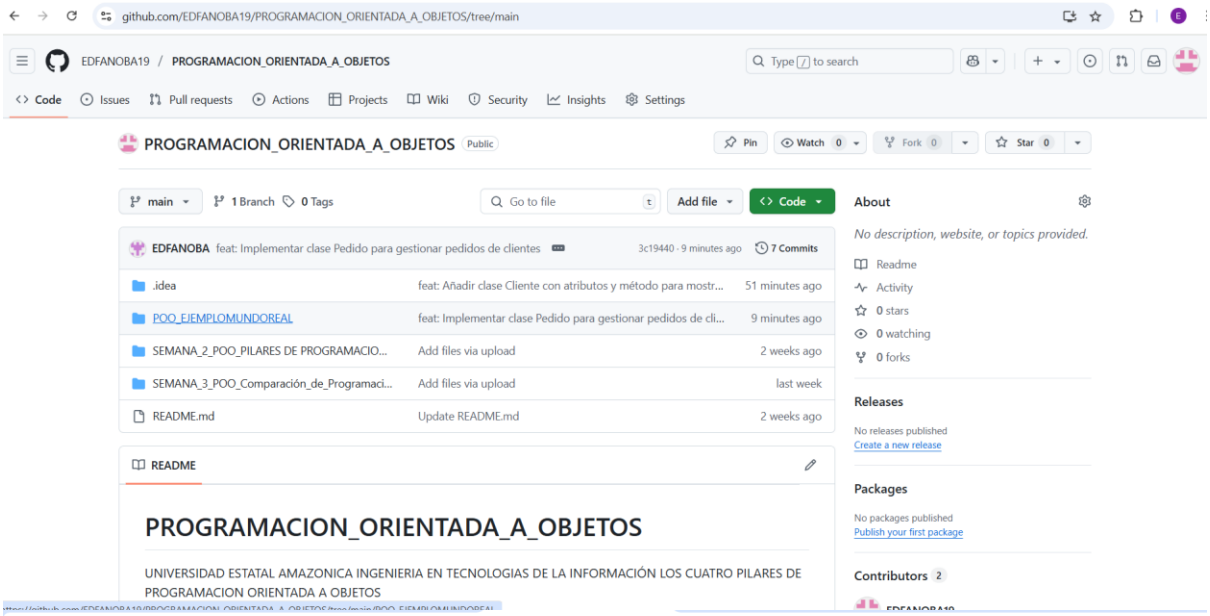
2025-2025

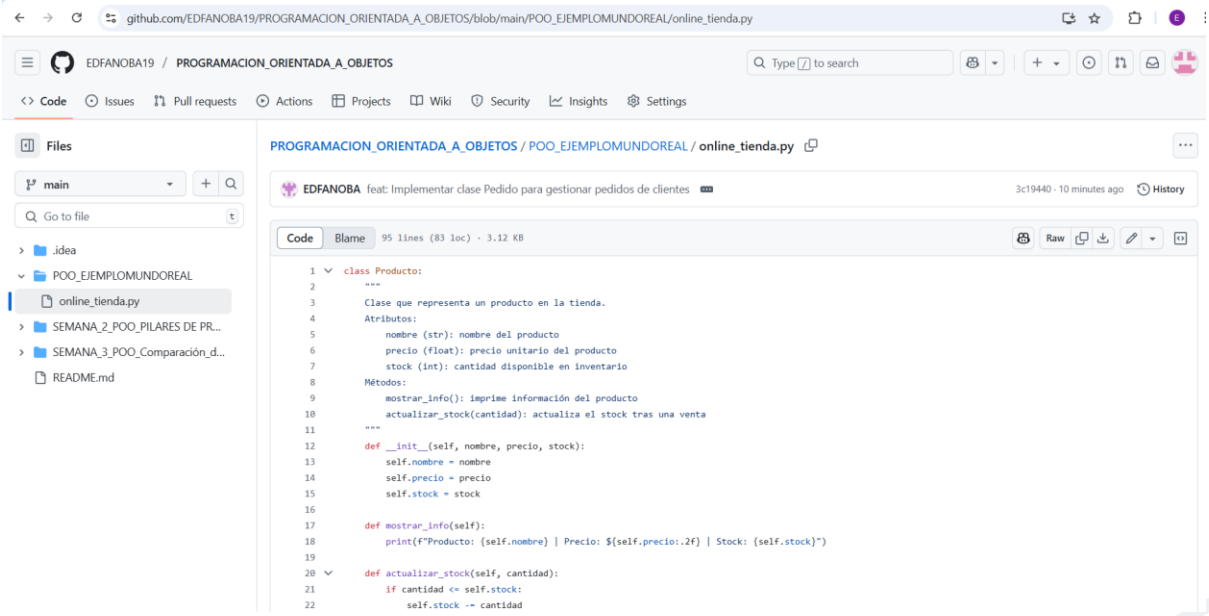
PUYO-ECUADOR

1. CÓDIGO DE GITHUB

https://github.com/EDFANOBA19/PROGRAMACION_ORIENTADA_A_OBJETOS.git

2. REPOSITORIO DE GITHUB

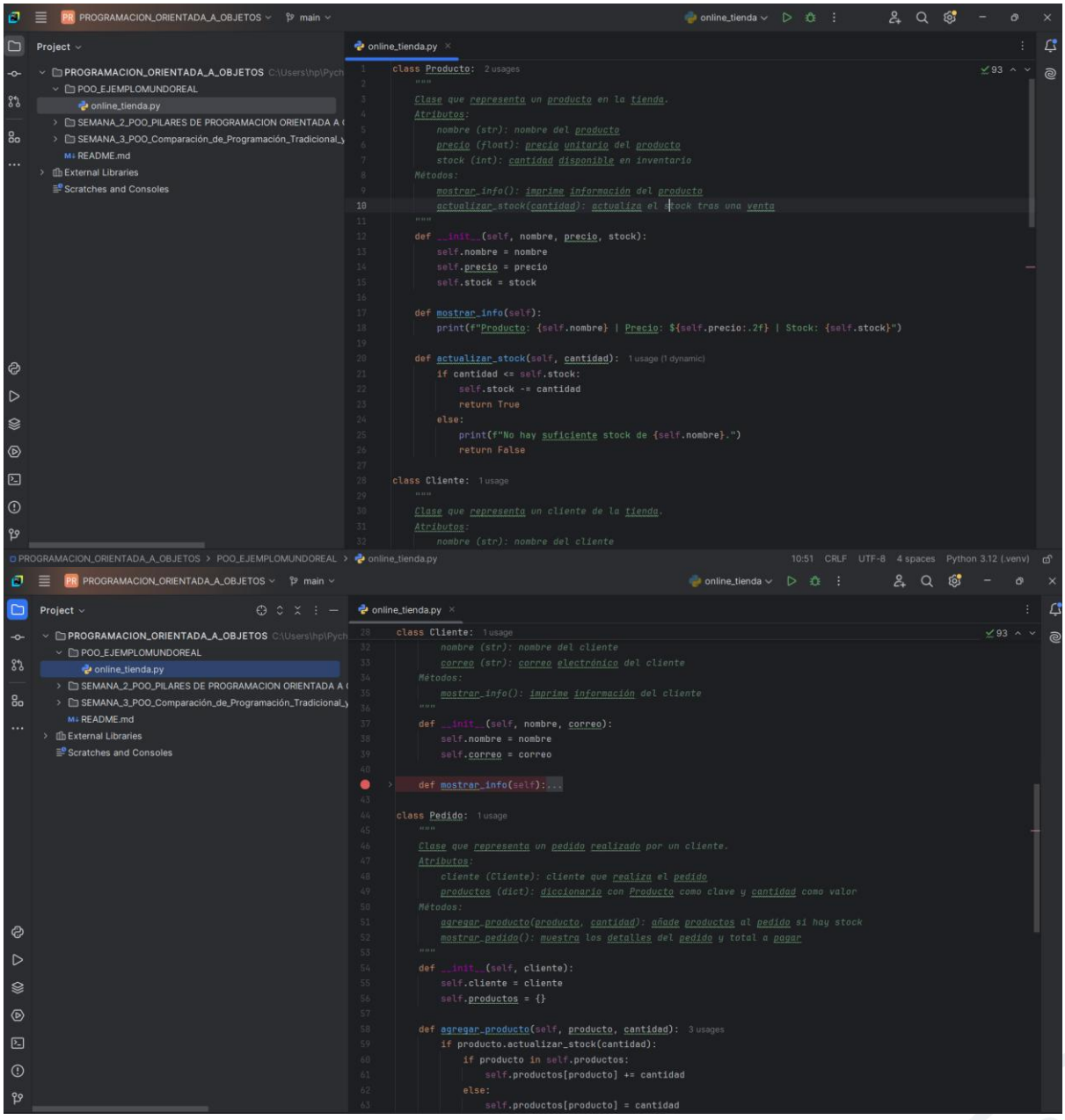




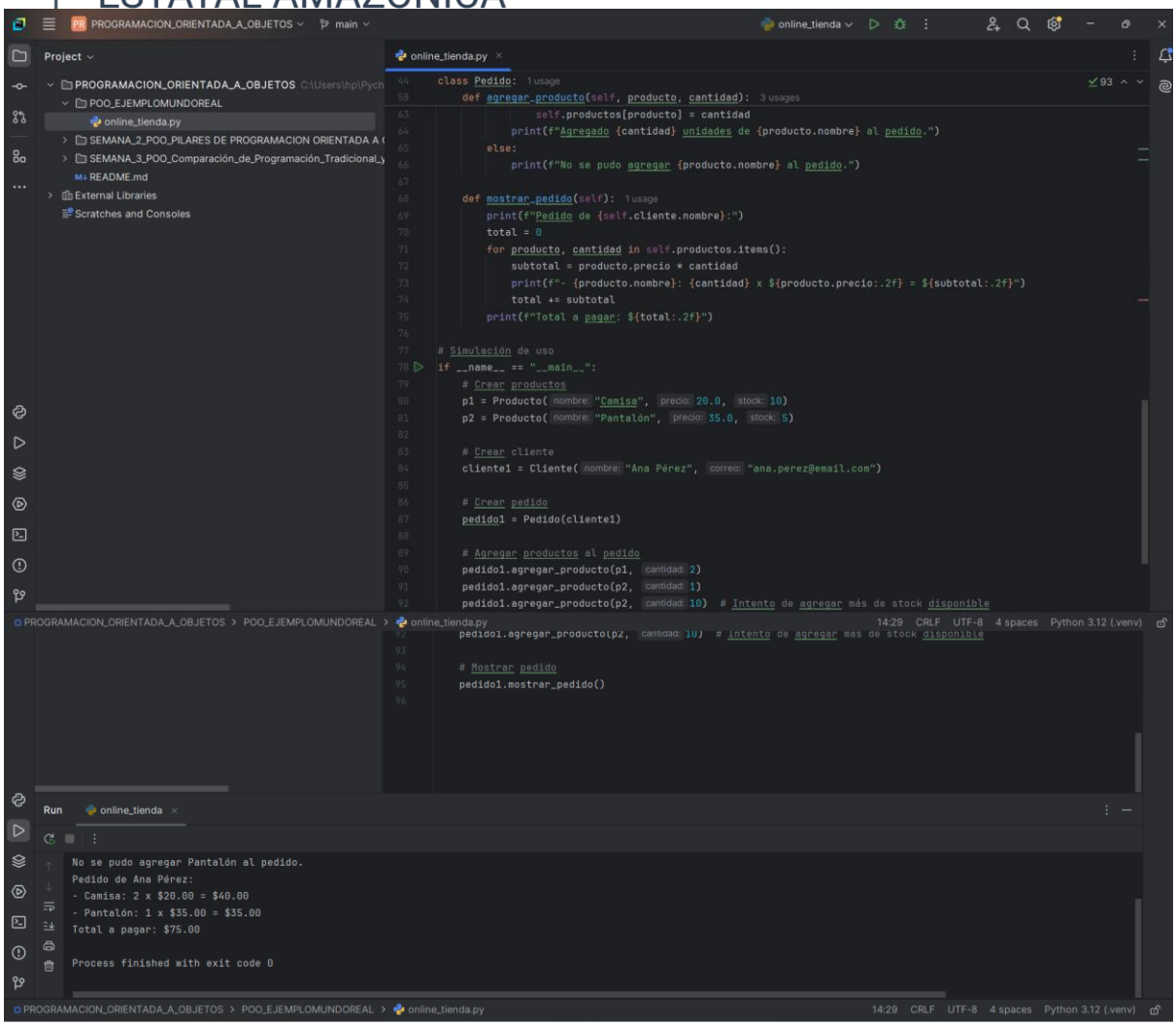
2.1 EJEMPLOS DEL MUNDO REAL Y CARACTERÍSTICAS DE LA PROGRAMACIÓN ORIENTADA A OBJETOS

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



```python
class Producto:
    """
    Clase que representa un producto en la tienda.
    Atributos:
        nombre (str): nombre del producto
        precio (float): precio unitario del producto
        stock (int): cantidad disponible en inventario
    Métodos:
        mostrar_info(): imprime información del producto
        actualizar_stock(cantidad): actualiza el stock tras una venta
    """
    def __init__(self, nombre, precio, stock):
        self.nombre = nombre
        self.precio = precio
        self.stock = stock

    def mostrar_info(self):
        print(f"Producto: {self.nombre} | Precio: ${self.precio:.2f} | Stock: {self.stock}")

    def actualizar_stock(self, cantidad):
        if cantidad <= self.stock:
            self.stock -= cantidad
            return True
        else:
            print(f"No hay suficiente stock de {self.nombre}.")
            return False

class Cliente:
    """
    Clase que representa un cliente de la tienda.
    Atributos:
        nombre (str): nombre del cliente
        correo (str): correo electrónico del cliente
    Métodos:
        mostrar_info(): imprime información del cliente
    """
    def __init__(self, nombre, correo):
        self.nombre = nombre
        self.correo = correo

    def mostrar_info(self):...

class Pedido:
    """
    Clase que representa un pedido realizado por un cliente.
    Atributos:
        cliente (Cliente): cliente que realiza el pedido
        productos (dict): diccionario con Producto como clave y cantidad como valor
    Métodos:
        agregar_producto(producto, cantidad): añade productos al pedido si hay stock
        mostrar_pedido(): muestra los detalles del pedido y total a pagar
    """
    def __init__(self, cliente):
        self.cliente = cliente
        self.productos = {}

    def agregar_producto(self, producto, cantidad):
        if producto.actualizar_stock(cantidad):
            if producto in self.productos:
                self.productos[producto] += cantidad
            else:
                self.productos[producto] = cantidad
```



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.