--- Tienda Online de Pedidos ---

```
# Base de datos simulada (listas en memoria)
usuarios = []
productos = [
{"id": 1, "nombre": "Camiseta", "precio": 20}, X


{"id": 2, "nombre": "Pantalón", "precio": 35},
{"id": 3, "nombre": "Zapatillas", "precio": 50}
]
pedidos = []

# Función para registrar usuarios
def registrar_usuario():
print("=== Registrar Usuario ===")
nombre = input("Nombre: ")
email = input("Email: ")
contrasena = input("Contraseña: ")
usuario = {"nombre": nombre, "email": email, "contrasena": contrasena}
usuarios.append(usuario)
print(f"Usuario {nombre} registrado con éxito.\n")

# Función para mostrar productos
def mostrar_productos():
print("=== Productos Disponibles ===")
for producto in productos:
print(f"{producto['id']}: {producto['nombre']} - ${producto['precio']}")
print()

# Función para crear un pedido
def crear_pedido():
if len(usuarios) == 0:
print("No hay usuarios registrados. Regístrate primero.\n")
return
email = input("Introduce tu email para iniciar pedido: ")
usuario = next((u for u in usuarios if u["email"] == email), None)
if not usuario:
print("Usuario no encontrado.\n")
return

mostrar_productos()
producto_id = int(input("ID del producto a pedir: "))
producto = next((p for p in productos if p["id"] == producto_id), None)
if not producto:
print("Producto no encontrado.\n")
return
```

```
cantidad = int(input("Cantidad: "))
total = producto["precio"] * cantidad
pedido = {
"usuario": usuario["nombre"],
"email": usuario["email"],
"producto": producto["nombre"],
"cantidad": cantidad,
"total": total
}
pedidos.append(pedido)
print(f"Pedido de {cantidad} x {producto['nombre']} realizado con éxito. Total: ${total}\n")

# Función para mostrar pedidos
def mostrar_pedidos():
if len(pedidos) == 0:
print("No hay pedidos realizados.\n")
return
print("=== Pedidos Realizados ===")
for p in pedidos:
print(f"{p['usuario']} - {p['producto']} x {p['cantidad']} = ${p['total']}")
print()

# Menú principal
def menu():
while True:
print("=== Tienda Online ===")
print("1. Registrar Usuario")
print("2. Mostrar Productos")
print("3. Realizar Pedido")
print("4. Ver Pedidos")
print("5. Salir")
opcion = input("Elige una opción: ")
print()
if opcion == "1":
registrar_usuario()
elif opcion == "2":
mostrar_productos()
elif opcion == "3":
crear_pedido()
elif opcion == "4":
mostrar_pedidos()
elif opcion == "5":
print("¡Gracias por usar la tienda online!")
break
else:
print("Opción no válida.\n")

# Ejecutar la tienda
```

if __name__ == "__main__":
menu()

Explicación del código

python

 # Base de datos simulada (listas en memoria)

- Otro comentario, que explica la idea clave de diseño:
    - No se usa una base de datos real (MySQL, PostgreSQL, etc.).
    - Se van a usar listas de Python en memoria para simular tablas: una para usuarios, otra para productos y otra para pedidos.
- De nuevo, es texto para humanos; el intérprete lo ignora.

usuarios = []

- Declara una variable global usuarios y la inicializa como lista vacía.
- Esta lista actuará como una "tabla" donde cada elemento será un diccionario con los datos de un usuario (por ejemplo, {"nombre": ..., "email": ..., "contrasena": ...}).
- "En memoria" significa que mientras el programa esté corriendo esa lista existe; si cierras el programa, se pierde.

productos = [

{"id": 1, "nombre": "Camiseta", "precio": 20},

{"id": 2, "nombre": "Pantalón", "precio": 35},

{"id": 3, "nombre": "Zapatillas", "precio": 50}

]

- Aquí se define la lista productos con tres diccionarios, cada uno representando un producto disponible en la tienda.
- Cada diccionario tiene tres claves:
    - "id": identificador numérico único del producto (1, 2, 3).
    - "nombre": nombre legible del producto ("Camiseta", "Pantalón", "Zapatillas").
    - "precio": precio unitario, en este ejemplo un entero (20, 35, 50).
- Esta estructura hace de "catálogo" de la tienda:
    - La función que muestra productos recorre esta lista.
    - La función que crea pedidos busca dentro de esta lista por id para saber qué producto ha elegido el usuario y a qué precio.

pedidos = []

- Declara la lista global pedidos, inicialmente vacía.

- Cada vez que se crea un nuevo pedido, se añadirá un diccionario a esta lista con la información del usuario, el producto, la cantidad y el total.
- Igual que usuarios, está solo en memoria: si se cierra el programa, se pierden los pedidos.

 # Función para registrar usuarios

def registrar_usuario():

- Comentario que indica el propósito de la función: registrar usuarios en la "base de datos" simulada.
- def registrar_usuario(): define una nueva función llamada registrar_usuario que no recibe parámetros; todo lo pedirá por teclado.

print("=== Registrar Usuario ===")

- Muestra un título en la consola para que el usuario sepa qué está haciendo (proceso de registro).

nombre = input("Nombre: ")

- input("Nombre: ") escribe el texto "Nombre: " y espera a que el usuario teclee algo y pulse Enter.
- El texto introducido se guarda en la variable nombre como cadena (str).

email = input("Email: ")

- Igual que la línea anterior, pero ahora se pide el email.
- Se guarda en la variable email.

contrasena = input("Contraseña: ")

- Pide la contraseña para esa cuenta.
- Se almacena la cadena en la variable contrasena.
- En este ejemplo se guarda tal cual, lo que es útil didácticamente para hablar de riesgos (contraseña en claro).

usuario = {"nombre": nombre, "email": email, "contrasena": contrasena}

- Crea un diccionario usuario con tres pares clave-valor:
  - "nombre" → valor de la variable nombre.
  - "email" → valor de la variable email.
  - "contrasena" → valor de la variable contrasena.
- Este diccionario representa un registro de usuario.

usuarios.append(usuario)

- Añade el diccionario usuario a la lista global usuarios.
- append mete el elemento al final de la lista, modificándola "in place".

print(f"Usuario {nombre} registrado con éxito.\n")

- Muestra un mensaje de confirmación, usando una f-string para incluir el nombre del usuario.
- \n añade una línea en blanco al final para separar visualmente el siguiente bloque de salida.

 # Función para mostrar productos

def mostrar_productos():

- Comentario: indica que la función está pensada para enseñar el catálogo.
- def mostrar_productos(): define una función sin parámetros que simplemente recorre la lista productos.

print("=== Productos Disponibles ===")

- Imprime un título antes del listado para que quede claro qué se está mostrando.

for producto **in** productos:

- Inicia un bucle for que recorre todos los elementos de la lista productos.
- Cada producto es un diccionario del tipo {"id": 1, "nombre": "Camiseta", "precio": 20}.

print(f"{producto['id']}: {producto['nombre']} - ${producto['precio']}")

- Para cada producto, imprime una línea con:
  - el id
  - el nombre
  - el precio, precedido de $.
- La f-string permite acceder a los campos del diccionario usando producto['id'], etc.

print()

- Imprime una línea en blanco al acabar el bucle, para dejar espacio en la consola.

 # Función para crear un pedido

def crear_pedido():

- Comentario: esta función gestiona el flujo de hacer un pedido nuevo.
- def crear_pedido(): define esa función, que pedirá datos al usuario y acabará creando un diccionario pedido y añadiéndolo a la lista pedidos.

if len(usuarios) == 0:

print("No hay usuarios registrados. Regístrate primero.\n")

return

- Comprueba si la lista usuarios está vacía (no hay clientes registrados).

- len(usuarios) == 0 significa que no hay ningún usuario.
- Si no hay usuarios:
    - Muestra el mensaje de aviso.
    - return termina la función crear_pedido sin hacer nada más (no se puede pedir sin usuario).

email = input("Introduce tu email para iniciar pedido: ")

- Pide al usuario que introduzca su email para asociar el pedido a esa persona.
- Se guarda en la variable email.

usuario = next((u **for** u **in** usuarios **if** u["email"] == email), None)

- Usa una expresión generadora para buscar en la lista de usuarios:
    - (u for u in usuarios if u["email"] == email) recorre usuarios y devuelve solo aquellos diccionarios u cuyo "email" coincide con el email introducido.
    - next(..., None) toma el primer resultado de ese generador; si no hay ninguno, devuelve None.
- Resultado:
    - usuario será el diccionario del usuario encontrado, o None si ese email no existe en la lista.

if not usuario:

print("Usuario no encontrado.\n")

return

- if not usuario se cumple cuando usuario es None (o una estructura vacía, pero aquí el caso es None).
- Si el email no estaba registrado:
    - Muestra mensaje “Usuario no encontrado”.
    - return sale de la función sin crear pedido.

mostrar_productos()

- Llama a la función mostrar_productos() para enseñar el catálogo disponible antes de pedir el ID del producto.
- Así, el usuario puede ver qué opciones hay (IDs, nombres, precios).

producto_id = int(input("ID del producto a pedir: "))

- Pide por teclado el ID numérico del producto que se quiere pedir.
- input devuelve una cadena, por eso se envuelve con int(...) para convertirla a entero, y se guarda en producto_id.

- Si se introduce algo que no es un número válido, se produciría un ValueError (en este ejemplo no se captura, lo que es otra oportunidad para ejercicios).

producto = next((p **for** p **in** productos **if** p["id"] == producto_id), None)

- Igual que con el usuario, se busca ahora el producto en la lista productos:
    - Generador (p for p in productos if p["id"] == producto_id) filtra por ID.
    - next(..., None) devuelve el primer producto que cumpla la condición o None si no existe.
- producto será el diccionario del producto elegido o None.

if not producto:

print("Producto no encontrado.\n")

return

- Si no se ha encontrado un producto con ese ID, muestra un mensaje de error y sale de la función sin crear pedido.

cantidad = int(input("Cantidad: "))

- Pide cuántas unidades del producto quiere el usuario.
- Convierte la entrada a entero y la guarda en cantidad.

total = producto["precio"] * cantidad

- Calcula el total del pedido multiplicando el precio unitario (producto["precio"]) por la cantidad.
- Guarda el resultado numérico en total.

pedido = {

"usuario": usuario["nombre"],

"email": usuario["email"],

"producto": producto["nombre"],

"cantidad": cantidad,

"total": total

}

- Crea un diccionario pedido con la información relevante:
    - "usuario": nombre del usuario que hace el pedido.
    - "email": su email, para identificarlo o enviar confirmaciones.
    - "producto": nombre del producto pedido.
    - "cantidad": número de unidades.
    - "total": importe total calculado.

- Este diccionario representa una “fila” en la tabla de pedidos.

pedidos.append(pedido)

- Añade el diccionario pedido a la lista global pedidos.
- Esa lista mantiene el historial de todos los pedidos creados hasta ahora.

print(f"Pedido de {cantidad} x {producto['nombre']} realizado con éxito. Total: ${total}\n")

- Muestra en pantalla un resumen del pedido recién creado: cuántas unidades, de qué producto y el total en dinero.
- también añade una línea en blanco al final.

 # Función para mostrar pedidos

def mostrar_pedidos():

- Comentario indicando el propósito: listar los pedidos existentes.
- def mostrar_pedidos(): define la función encargada de eso.

if len(pedidos) == 0:

print("No hay pedidos realizados.\n")

return

- Comprueba si la lista pedidos está vacía.
- Si no hay ningún pedido registrado, muestra el mensaje y termina la función.

print("=== Pedidos Realizados ===")

- Si sí hay pedidos, muestra el título del listado.

for p **in** pedidos:

print(f"{p['usuario']} - {p['producto']} x {p['cantidad']} = ${p['total']}")

- Recorre la lista pedidos, donde cada p es un diccionario de pedido.
- Para cada pedido, imprime en una línea:
    - nombre del usuario,
    - nombre del producto,
    - cantidad,
    - total.

print()

- Línea en blanco al final del listado.

 # Menú principal

def menu():

- Comentario: aquí se define el menú principal de la aplicación, la “interfaz” con el usuario.
- def menu(): crea la función que mostrará el menú y responderá según la opción elegida.

while True:

- Inicia un bucle infinito (while True).
- El menú se repetirá una y otra vez hasta que el usuario elija “Salir” y se haga break.

print("=== Tienda Online ===")

print("1. Registrar Usuario")

print("2. Mostrar Productos")

print("3. Realizar Pedido")

print("4. Ver Pedidos")

print("5. Salir")

- Muestra el título y las cinco opciones de la tienda online.
- Cada print es una línea del menú.

opcion = input("Elige una opción: ")

print()

- input pide al usuario que introduzca una opción (por ejemplo, "1", "2", etc.) y guarda la cadena en opcion.
- La línea en blanco posterior (print()) separa visualmente el menú de la salida de la opción.

if opcion == "1":

registrar_usuario()

- Si el usuario ha tecleado "1", se llama a registrar_usuario().
- Al acabar el registro, se vuelve al inicio del while y se vuelve a mostrar el menú.

elif opcion == "2":

mostrar_productos()

- Si la opción es "2", se ejecuta mostrar_productos().

elif opcion == "3":

crear_pedido()

- Si la opción es "3", se lanza crear_pedido().

elif opcion == "4":

mostrar_pedidos()

- Si la opción es "4", se llama a mostrar_pedidos().

elif opcion == "5":

print("¡Gracias por usar la tienda online!")

break

- Si la opción es "5":
    - Muestra un mensaje de despedida.
    - break rompe el bucle while True y hace que la función menu termine.

else:

print("Opción no válida.\n")

- Si el usuario escribe cualquier cosa que no sea "1", "2", "3", "4" o "5", entra en el else.
- Se avisa de que la opción no es válida y, al terminar este bloque, el bucle vuelve a empezar mostrando de nuevo el menú.

 # Ejecutar la tienda

if __name__ == "__main__":

menu()

- Comentario: indica que lo que viene sirve para arrancar la tienda cuando se ejecuta el script.
- if __name__== "__main__": comprueba si este archivo se está ejecutando directamente (por ejemplo python tienda.py) y no importado desde otro script.
- Si es el archivo principal, se llama a menu(), dando inicio a la interacción con el usuario.