Ejercicios de Programación en Python

Ejercicio 1: Clase de Caja Registradora

Descripción
En este ejercicio, se implementa una clase `CajaRegistradora` que permite gestionar productos y pagos. La clase incluye métodos para agregar productos, calcular el total y dar cambio. Si el cliente no proporciona suficiente dinero, se lanza una excepción.

Datos de Prueba

```
caja = CajaRegistradora()
caja.agregar_producto('Manzana', 0.5)
caja.agregar_producto('Pan', 1)
print("Total:", caja.obtener_total())
print("Cambio:", caja.dar_cambio(2))
print("Cambio:", caja.dar_cambio(1))
```

Salida Esperada

```
Total: 1.5
Cambio: 0.5
ValueError: El pago es insuficiente.
```

Solución Propuesta

```
class CajaRegistradora:
    def __init__(self):
        self.productos = []

    def agregar_producto(self, nombre, precio):
        self.productos.append({'nombre': nombre, 'precio': precio})

    def obtener_total(self):
        return sum(producto['precio'] for producto in self.productos)

    def dar_cambio(self, pago):
        total = self.obtener_total()
        if pago < total:
            raise ValueError("El pago es insuficiente.")
        return pago - total

# Ejemplo de uso
caja = CajaRegistradora()
caja.agregar_producto('Manzana', 0.5)
caja.agregar_producto('Pan', 1)
```



```
print("Total:", caja.obtener_total())
print("Cambio:", caja.dar_cambio(2))
print("Cambio:", caja.dar_cambio(1))
```

Ejercicio 2: Clase de Cuenta Bancaria

Descripción
Aquí se implementa una clase `CuentaBancaria` que maneja depósitos, retiros y saldos, con excepciones personalizadas para manejar errores como retiros de cantidades negativas o superiores al saldo disponible.

Datos de Prueba

```
try:
    cuenta = CuentaBancaria("123456789", "Juan Perez", 1000)
    cuenta.mostrar_saldo()
    cuenta.depositar(500)
    cuenta.retirar(2000)
except ErrorRetiro as e:
    print(e)

try:
    cuenta.retirar(-100)
except ErrorRetiro as e:
    print(e)
```

Salida Esperada

```
Saldo actual: 1000
Depósito exitoso. Nuevo saldo: 1500
('Fondos insuficientes para realizar el retiro.', 401)
('La cantidad a retirar debe ser positiva.', 400)
```

Solución Propuesta

```
class ErrorRetiro(Exception):
    def __init__(self, mensaje, codigo):
        self.mensaje = mensaje
        self.codigo = codigo

class CuentaBancaria:

    def __init__(self, numero_cuenta, propietario, saldo=0):
        self.numero_cuenta = numero_cuenta
```

```
        self.propietario = propietario
        self.saldo = saldo

    def depositar(self, cantidad):
        if cantidad > 0:
            self.saldo += cantidad
            print(f"Depósito exitoso. Nuevo saldo: {self.saldo}")
        else:
            print("La cantidad a depositar debe ser positiva.")

    def retirar(self, cantidad):
        if cantidad <= 0:
            raise ErrorRetiro("La cantidad a retirar debe ser positiva.", 400)
        elif cantidad > self.saldo:
            raise ErrorRetiro("Fondos insuficientes para realizar el retiro.", 401)
        else:
            self.saldo -= cantidad
            print(f"Retiro exitoso. Nuevo saldo: {self.saldo}")

    def mostrar_saldo(self):
        print(f"Saldo actual: {self.saldo}")

try:
    cuenta = CuentaBancaria("123456789", "Juan Perez", 1000)
    cuenta.mostrar_saldo()
    cuenta.depositar(500)
    cuenta.retirar(2000)
except ErrorRetiro as e:
    print(e)

try:
    cuenta.retirar(-100)
except ErrorRetiro as e:
    print(e)
```

Ejercicio 3: Verificación de Temperatura de CPU

Descripción
Se presenta una función que verifica la temperatura de la CPU. Si la temperatura alcanza niveles críticos, se lanza una excepción personalizada. También hay mensajes de advertencia para temperaturas elevadas pero no críticas. Debes lanzar distintos mensajes de advertencia cuando la temperatura se encuentra sobre 75 y 90 grados celsius, pero debes lanzar un error si la temperatura es mayor a 100 grados celsius

Datos de Prueba

```
try:
    temperatura_actual = 95  # Cambia este valor para probar diferentes escenarios
    check_cpu_temperature(temperatura_actual)
except OverheatingError as e:
    print(e)
```

Salida Esperada

```
Advertencia: La temperatura de la CPU es muy alta.
```

Solución Propuesta

```
class OverheatingError(Exception):
    def __init__(self, mensaje):
        self.mensaje = mensaje

def check_cpu_temperature(temp):
    if temp >= 100:
        raise OverheatingError("La temperatura de la CPU ha alcanzado el límite crítico de
100 grados centígrados.")
    elif temp >= 90:
        print("Advertencia: La temperatura de la CPU es muy alta.")
    elif temp >= 75:
        print("Advertencia: La temperatura de la CPU está elevada.")
    else:
        print("La temperatura de la CPU es normal.")

try:
    temperatura_actual = 95  # Cambia este valor para probar diferentes escenarios
    check_cpu_temperature(temperatura_actual)
except OverheatingError as e:
    print(e)
```

