

```

from tkinter import *
from tkinter import messagebox
from tkinter import ttk
import tkinter as tk
import sqlite3

def main():
    root= Tk()
    app = ventanaPrincipal(root)
class ventanaPrincipal:
    def __init__(self, master):
        self.master = master
        self.master.title('Gestión de pedidos')
        self.master.geometry('00x500+0+0')
        self.master.config(bg = 'powder blue')
        self.frame =Frame(self.master, bg = 'powder blue')
        self.frame.pack()

        self.btnProducto = Button(self.frame, text='Producto', width= 17,
command=self.ventanaNueva)
        self.btnProducto.grid(row=7, column = 0)

        self.btnCliente = Button(self.frame, text='Cliente', width= 17,
command=self.ventanaNueva2)
        self.btnCliente.grid(row=7, column = 2)

        self.btnCliente = Button(self.frame, text='Pedido', width=
17,command=self.ventanaNueva)
        self.btnProducto.grid(row=7, column = 0)


    def ventanaNueva(self):
        self.ventanaNueva = Toplevel(self.master)
        self.ventanaProducto = Producto(self.ventanaNueva)
    def ventanaNueva2(self):
        self.ventanaNueva2 = Toplevel(self.master)
        self.ventanaCliente = Cliente(self.ventanaNueva2)
    def ventanaNueva3(self):
        self.ventanaNueva3 = Toplevel(self.master)
        self.ventanaPedido = Pedido(self.ventanaNueva3)

class Cliente:
    db_name = 'productos.db'

    def __init__(self, window):

```

```
self.wind = window
self.wind.title('Cliente')
```

```
frame = LabelFrame(self.wind)
frame.grid(row = 0, column = 0, columnspan = 3, pady = 20)
```

```
Label(frame, text = 'Codigo: ').grid(row = 1, column = 0)
```

```
Label(frame, text = 'Nombre: ').grid(row = 1, column = 0)
self.Nombre = Entry(frame)
self.Nombre.focus()
self.Nombre.grid(row = 1, column = 1)
```

```
Label(frame, text = 'Direccion: ').grid(row = 3, column = 0)
self.Direccion = Entry(frame)
self.Direccion.focus()
self.Direccion.grid(row = 3, column = 1)
```

```
Label(frame, text = 'Telefono: ').grid(row = 5, column = 0)
self.Telefono = Entry(frame)
self.Telefono.focus()
self.Telefono.grid(row = 5, column = 1)
```

```
Label(frame, text = 'Correo: ').grid(row = 7, column = 0)
self.Correo = Entry(frame)
self.Correo.focus()
self.Correo.grid(row = 7, column = 1)
```

```
ttk.Button(frame, text = 'Guardar', command=self.crearCliente).grid(row = 10, column = 2)
```

```
pedidoR =ttk.Button(frame, text = 'Realizar pedido',
command=self.realizarPedido).grid(row = 12, column= 3)
pedidoR.pack()
```

```
def query(self, query, parameters = ( )):
```

```
    with sqlite3.connect(self.db_name) as conn:
```

```
        cursor = conn.cursor()
```

```
        result = cursor.execute(query, parameters)
```

```
        conn.commit()
```

```
    return result
```

```
def crearCliente(self):
```

```
    if self.validarCliente():
```

```
        query = 'INSERT INTO cliente VALUES(NULL, ?, ?, ?, ?,NULL, NULL)'
```

```
        parameters = (self.Nombre.get(), self.Direccion.get(), self.Telefono.get(),
```

```
self.Correo.get())
```

```
        self.query(query, parameters)
```

```

        messagebox.showinfo(' ', 'Cliente Creado')
        self.Nombre.delete(0, END)
        self.Direccion.delete(0, END)
        self.Telefono.delete(0, END)
        self.Correo.delete(0, END)
    else:
        messagebox.showinfo(' ', 'Ingrese todos los valores ')

def validarCliente(self):
    return len(self.Nombre.get()) != 0 and len(self.Direccion.get()) != 0 and
len(self.Telefono.get()) != 0 and len(self.Correo.get()) != 0

def realizarPedido(self):
    query = ('SELECT * FROM cliente where Nombre = '+self.Nombre.get()+');
    self.query(query)
    print(query)

def obtenerCuentasCliente(self):
    query = ('SELECT * Nombre, Cuenta, saldoDisponible from cliente, cuenta where
cuenta_codigo = cuenta.Codigo');
    self.query(query)
    print(query)

if __name__ == '__main__':
    main()
    window = Tk()
    window.geometry('400x200')
    #application = Cliente(window)
    window.mainloop()

class Producto:
    db_name = 'productos.db'
    def __init__(self, window):
        self.wind = window
        self.wind.title('Productos')

        frame = LabelFrame(self.wind)
        frame.grid(row = 0, column = 0, columnspan = 3, pady = 20)

        Label(frame, text = 'Nombre: ').grid(row = 1, column = 0)
        self.Nombre = Entry(frame)
        self.Nombre.focus()
        self.Nombre.grid(row = 1, column = 1)

        Label(frame, text = 'Cantidad: ').grid(row = 3, column = 0)

```

```

self.Cantidad = Entry(frame)
self.Cantidad.focus()
self.Cantidad.grid(row = 3, column = 1)

Label(frame, text = 'Valor: ').grid(row = 5, column = 0)
self.Valor = Entry(frame)
self.Valor.focus()
self.Valor.grid(row = 5, column = 1)

ttk.Button(frame, text = 'Guardar',command=self.crearProducto ).grid(row = 6,
columnspan = 2, sticky = W + E)
def query(self, query, parameters = ()):
    with sqlite3.connect(self.db_name) as conn:
        cursor = conn.cursor()
        result = cursor.execute(query, parameters)
        conn.commit()
    return result
def crearProducto(self):
    if self.validarProducto():
        query = 'INSERT INTO producto VALUES(NULL, ?, ?, ?)'
        parameters = (self.Nombre.get(), self.Cantidad.get(), self.Valor.get())
        self.query(query, parameters)
        messagebox.showinfo(' ', 'Producto Creado')
        self.Nombre.delete(0, END)
        self.Cantidad.delete(0, END)
        self.Valor.delete(0, END)
    else:
        messagebox.showinfo(' ', 'Ingrese los campos')
def validarProducto(self):
    return len(self.Nombre.get()) != 0 and len(self.Cantidad.get()) != 0 and
len(self.Valor.get())

if __name__ == '__main__':
    main()
    window = Tk()
    application = Producto(window)
    window.geometry('280x170')
    window.mainloop()

class Cuenta:
    db_name = 'productos.db'
    def __init__(self, window):
        self.wind = window

        self.wind.title('Cuenta')

        frame = LabelFrame(self.wind)

```

```

frame.grid(row = 0, column = 0, columnspan = 3, pady = 20)

Label(frame, text = 'Saldo: ').grid(row = 1, column = 0)
self.Saldo = Entry(frame)
self.Saldo.focus()
self.Saldo.grid(row = 1, column = 1)

Label(frame, text = 'Tarjeta: ').grid(row = 3, column = 0)
self.Tarjeta = Entry(frame)
self.Tarjeta.focus()
self.Tarjeta.grid(row = 3, column = 1)

ttk.Button(frame, text = 'Guardar', command=self.crearCuenta ).grid(row = 5, column =
0 )
ttk.Button(frame, text = 'Editar', command = self.editarSaldo).grid(row = 6, column = 1)
ttk.Button(frame, text = 'Buscar', command = self.obtenerCuenta).grid(row = 7, column
= 2)

def query(self, query, parameters = ()):
    with sqlite3.connect(self.db_name) as conn:
        cursor = conn.cursor()
        resultado = cursor.execute(query, parameters)
        conn.commit()
    return resultado
def crearCuenta(self):
    if self.validarCuenta():
        query = 'INSERT INTO cuenta VALUES(NULL, ?, ?)'
        parameters = (self.Saldo.get(), self.Tarjeta.get())
        self.query(query, parameters)
        messagebox.showinfo(' ', 'Cuenta Creada')
        self.Saldo.delete(0, END)
        self.Tarjeta.delete(0, END)

    else:
        messagebox.showinfo(' ', 'Ingrese los campos')
def validarCuenta(self):
    return len(self.Saldo.get()) != 0 and len(self.Tarjeta.get())

def editarSaldo(self):

    Label(frame, text = 'Saldo: ').grid(row = 1, column = 0)
    self.Saldo =
    self.Saldo.focus()
    self.Saldo.grid(row = 1, column = 1)

    Label(frame, text = 'Tarjeta: ').grid(row = 3, column = 0)

```

```

self.Tarjeta = Entry(frame)
self.Tarjeta.focus()
self.Tarjeta.grid(row = 3, column = 1)

def editarDato(self, SaldoNuevo, Saldo, Tarjeta):
    query = 'UPDATE cuenta SET Saldo = ?, WHERE Saldo=?'
    parameters = (SaldoNuevo, Saldo, Tarjeta)
    self.run_query(query, parameters)
    self.edit_wind.destroy()
    messagebox.showinfo(' ', 'Editado ')
    self.obtenerCuenta()

def obtenerCuenta(self):
    query = ('SELECT * FROM cuenta where Tarjeta = '+self.Tarjeta.get()+");
    self.query(query)
    print(query)

if __name__ == '__main__':
    main()
    window = Tk()
    application = Cuenta(window)
    window.geometry('380x370')
    window.mainloop()

class Pedidos:
    db_name = 'productos.db'
    def __init__(self, window):
        self.wind = window
        self.wind.title('Pedidos')

        frame = LabelFrame(self.wind)
        frame.grid(row = 0, column = 0, columnspan = 3, pady = 20)

        Label(frame, text = 'Total: ').grid(row = 1, column = 0)
        self.Total = Entry(frame)
        self.Total.focus()
        self.Total.grid(row = 1, column = 1)

        Label(frame, text = 'Estado Pedido: ').grid(row = 3, column = 0)
        self.Estado = Entry(frame)
        self.Estado.focus()
        self.Estado.grid(row = 3, column = 1)

        ttk.Button(frame, text = 'Guardar',command=self.crearPedido ).grid(row = 6,
columnspan = 2, sticky = W + E)
    def query(self, query, parameters = ()):

```

```

with sqlite3.connect(self.db_name) as conn:
    cursor = conn.cursor()
    resultado = cursor.execute(query, parameters)
    conn.commit()
    return resultado
def crearPedido(self):
    if self.validarPedido():
        query = 'INSERT INTO pedido VALUES(NULL, ?, ?,NULL)'
        parameters = (self.Total.get(), self.Estado.get())
        self.query(query, parameters)
        messagebox.showinfo(' ', 'Pedido Realizado')
        self.Total.delete(0, END)
        self.Estado.delete(0, END)

    else:
        messagebox.showinfo(' ', 'Ingrese los campos')
def aniadirProducto ():
    query = ('SELECT Nombre, cuenta_codigo, Estado from cliente, pedido where
pedido.Estado = '+pendiente+'');
    self.query(query)
def validarPedido(self):
    return len(self.Total.get()) != 0 and len(self.Estado.get())

if __name__ == '__main__':
    main()
    window = Tk()
    application = Pedidos(window)
    window.geometry('280x170')
    window.mainloop()

class controladorPedido():
    def cobroPedido():
        query = ('SELECT Nombre, cuenta_codigo, Estado from cliente, pedido where
pedido.Estado = '+pendiente+'');
        self.query(query)
        if query != null:
            consulta2 = ('SELECT Valor, Cantidad from cliente , pedido, producto where
producto_codigo = producto.Codigo AND pedido.Codigo = pedido_codigo');
            self.query(consulta2)
            if consulta2 != null:
                for valor in consulta2:
                    saldo = valor [0]
                    cantidad= valor [1]
                    valorCobrar = saldo * cantidad;

```

```

        consulta3 = ('SELECT Nombre, saldoDisponible from cliente, cuenta where
saldoDisponible >=' + valorCobrar + ');
        self.query(consulta3)
        if (consulta3 != null):
            for valorPago in consulta3:
                valorRestante = ValorPago[1]
                valorFinalRestante = ValorRestante - valorCobrar.
        else:
            print ('Se rechaza el pedido por falta de dinero')
    else:
        print('No existe producto o pedido con ese código')
else:
    print ('No hay pedidos pendientes')

```

```

def ordenDistribucion():
    if cobroPedido != null:
        print ('Ordenar Distribución')
    else:
        print ("Cancelar distribucion")

```

```

def confirmarPedido():
    if ordenDistribucion != null:
        print ('Pedido confirmado')
    else:
        print ("No se confirma el pedido todavía")

```

```

class pedidoSimple():

```

```

    def obtenerTotal(int totalS):
        consulta = ('SELECT Valor, Cantidad from cliente , pedido, producto where
producto_codigo = producto.Codigo AND pedido.Codigo = pedido_codigo');
        self.query(consulta)
        if consulta != null:
            for simple in consulta:
                pedidoS = simple [1]
                if pedidoS > 9:
                    print ("PAQUETE COMPUESTO")
                else:
                    totalPedido = pedidoS * len(pedidoS)
            else:
                print('No hay pedidos simples')
def cobrarPedido (void):

```

```

    def obtenerCuenta (List cuenta):

```

```

class pedidoCompuesto(void):

```

```

    def obtenerTotal(int totalC):

```



```
        consulta = ('SELECT Valor, Cantidad from cliente , pedido, producto where
producto_codigo = producto.Codigo AND pedido.Codigo = pedido_codigo');
        self.query(consulta)
        if consulta != null:
            for simple in consulta:
                pedidoS = simple [1]
                if pedidoS < 9:
                    totalPedido = pedidoS * len(pedidoS)
                else:
                    print ("PAQUETE Simple")
            else:
                print('No hay pedidos compuestos')
def cobrarPedido (boolean bandera):

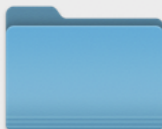
def obtenerDetalle(void):

def obtenerCuenta (List cuenta):
```

ANEXOS

Gestión de pedidos	
tk #103	Gestión de pedidos
Producto	Cliente

Nombre:	<input type="text" value="ADRIAN"/>
Direccion:	<input type="text" value="ESTADIO"/>
Telefono:	<input type="text" value="97992779"/>
Correo:	<input type="text" value="adrina@gmail.com"/>
<input type="button" value="Guardar"/>	
<input type="button" value="Realizar pedido"/>	



Cliente Creado

Nombre:

Cantidad:

Valor:

Productos Gestión de pedidos

Nombre:

Cantidad:

Valor:

Pedidos Gestión de pedidos

Total:

Estado Pedido:

Ingrese los campos

Total:

Estado Pedido:

Producto Creado

Nombre:

Cantidad:

Valor:

```
un_query(que
dit_wind.des
ebox.showinf
btenerCuenta
rCuenta(self
= ('SELECT *
uery(query)
```