```
from tkinter import ttk
from tkinter import *

import sqlite3

class Producto:

    db_name = 'BaseDatos.db'

    def  __init__(self, window):
        self.wind=window
        self.wind.title('Control de Productos')

        # Creando un Contenedor
        frame = LabelFrame(self.wind, text='Registrar un nuevo producto')
        frame.grid(row = 0, column = 0, columnspan = 3, pady = 20)

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

        # Ingresando Precio
        Label(frame, text = 'Precio:').grid(row = 2, column = 0)
        self.precio = Entry(frame)
        self.precio.grid(row = 2, column = 1)

        # Boton para agregar producto
        ttk.Button(frame, text = 'Guardar ', command =
self.agregar_productos).grid(row = 3, columnspan = 2, sticky = W + E)

        # Boton para Borrar un dato
        ttk.Button(text = "Eliminar", command =
self.borrar_productos).grid(row = 5, column = 0, sticky = W + E)

        # boton para modificar un dato
        ttk.Button(text = "Modificar", command =
self.modificar_productos).grid(row = 5, column = 1, sticky = W + E)

        #Mensajes de  salida
        self.message = Label(text ='' , fg= 'red')
        self.message.grid(row=3, column=0, columnspan=2, sticky= W + E)

        self.tree = ttk.Treeview(height = 10, columns = 2)
```

```
        self.tree.grid(row = 4, column = 0, columnspan = 2 )
        self.tree.heading('#0', text = 'Nombre', anchor = CENTER)
        self.tree.heading('#1', text = 'Precio', anchor = CENTER)

        self.get_productos()

    def run_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  get_productos(self):
      #Limpiando la tabla
      records = self.tree.get_children()
      for element in  records:
        self.tree.delete(element)
     #Consultando la tabla
    #def get_Productos(self):
      query = 'SELECT * FROM Productos ORDER BY nombre DESC'
      db_rows = self.run_query(query)

          #rellenando los datos
      for  row in db_rows:
         self.tree.insert('', 0, text = row[1], values = row[2])

    def validacion(self):
        return len(self.nombre.get()) != 0 and len(self.precio.get()) != 0

    def agregar_productos(self):
      if self.validacion():
       query =  'INSERT INTO Productos VALUES(NULL, ?, ?)'
       parameters = (self.nombre.get(), self.precio.get())
       self.run_query(query, parameters)
       self.message['text'] = 'Producto {} agregado
satisfacatoriamente'.format(self.nombre.get())
       self.nombre.delete(0, END)
       self.precio.delete(0, END)
      else:
        self.message['text']= 'Nombre y Precio es Requerido'
      self.get_productos()

    def borrar_productos(self):
      self.message['text'] = ' '
```

```
      try:
       self.tree.item(self.tree.selection())['text'][0]
      except IndexError as e:
        self.message['text'] = 'Por favor seleccione un producto'
        return
      self.message['text'] = ' '
      nombre = self.tree.item(self.tree.selection())['text']
      query = 'DELETE FROM Productos WHERE nombre = ?'
      self.run_query(query, (nombre, ))
      self.message['text'] = 'El Producto {} ha sido borrado'.format(nombre)
      self.get_productos()

    def  modificar_productos(self):
     self.message['text'] = ' '
     try:
      self.tree.item(self.tree.selection())['text'][0]
     except IndexError as e:
      self.message['text'] = 'Por favor seleccione un producto'
      return
     nombre = self.tree.item(self.tree.selection())['text']
     precio_anterior = self.tree.item(self.tree.selection())['values'][0]
     self.edit_wind = Toplevel()
     self.edit_wind.title = 'Modificar Producto'

     #Nombre anterior
     Label(self.edit_wind, text = 'Nombre Anterior:').grid(row=0, column= 1)
     Entry(self.edit_wind, textvariable = StringVar(self.edit_wind, value =
nombre), state ='readonly').grid(row = 0, column = 2)
    #Nombre nuevo
     Label(self.edit_wind, text = 'Nuevo Nombre').grid(row= 1, column=1)
     nuevo_nombre = Entry(self.edit_wind)
     nuevo_nombre.grid(row=1, column=2)

    #Precio Anterior
     Label(self.edit_wind, text = 'Precio Anterior: ').grid(row= 2, column =
1)
     Entry(self.edit_wind, textvariable = StringVar(self.edit_wind, value =
precio_anterior), state = 'readonly').grid(row=2, column= 2)
     #Nuevo Precio
     Label(self.edit_wind, text = 'Nuevo Precio').grid(row= 3, column=1)
     nuevo_precio=Entry(self.edit_wind)
     nuevo_precio.grid(row = 3, column = 2)
```

```
     Button(self.edit_wind, text = 'Actualizar', command = lambda:
self.edit_records(nuevo_nombre.get(), nombre, nuevo_precio.get(),
precio_anterior)).grid(row = 4, column = 2, sticky = W + E)

    def  edit_records(self, nuevo_nombre, nombre, nuevo_precio,
precio_anterior):
      query = 'UPDATE Productos SET nombre = ?, precio = ? WHERE nombre = ?
AND precio = ?'
      parameters = (nuevo_nombre, nuevo_precio, nombre, precio_anterior)
      self.run_query(query, parameters)
      self.edit_wind.destroy()
      self.message['text'] = 'Registro actualizado satisfactoriamente'.
format(nombre)
      self.get_productos()




     #Datos nuevos



if __name__=='__main__':
    window =Tk()
    application=Producto(window)
    window.mainloop()
```