Trabajo final para la diplomatura en PYTHON3

Profesor: Juan Barreto

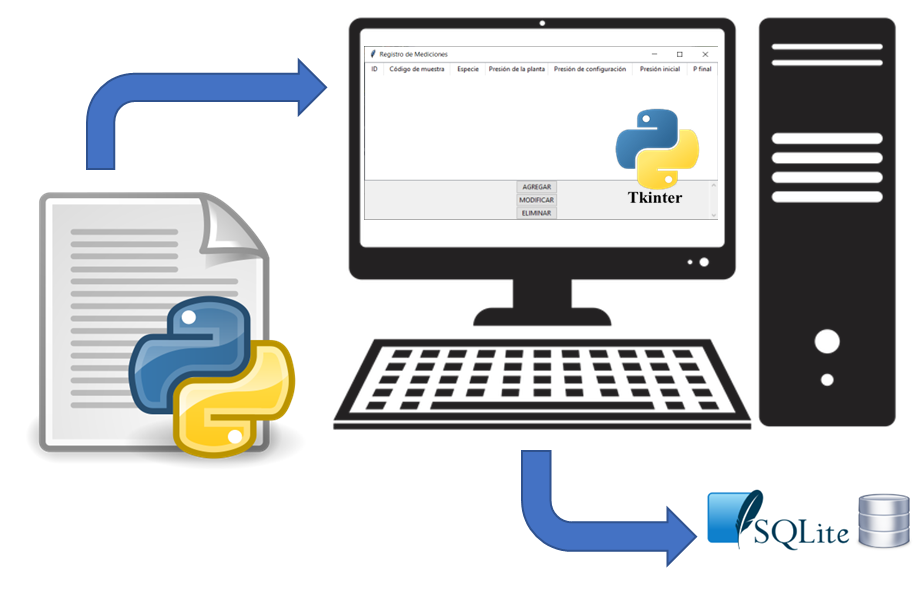
Alumno: Pablo Masera

# Descripción de la app.

Esta app se diseñó con el fin de tomar un registro de mediciones del sistema hidráulico de una planta. La idea fue utilizar el programa como complemento para registrar las presiones hidráulicas medidas con otras máquinas (Neumatrón y Cámara de Scholander).



El programa fue construido con el lenguaje de programación Python3, utilizando para su interfaz gráfica el paquete Tkinter de la biblioteca gráfica Tcl/Tk, para dicho lenguaje. También se utilizó el sistema de gestión de bases de datos relacionales SQLite3.



# Uso y funcionamiento de la app.

## Uso

Al iniciarse la app, muestra una pantalla muy simple con un registro de datos que tiene siete columas o campos:

ID: campo autoincremental que sirve de identificación.

Muestra: un campo alfanumérico para identificar con un código la muestra

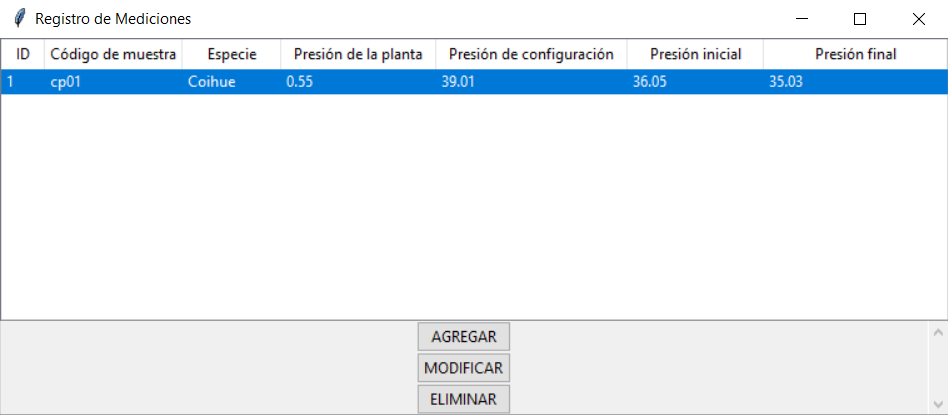
Especie: un campo alfabético para ingresar la especie de árbol

Pplant (Presión de la planta): un campo numérico decimal que se completa con el dato de presión medido con la cámara de scholander.

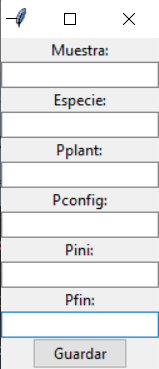
Pconfig (Presión de configuración): un campo numérico decimal que se completa con el dato medido desde el Neumatrón y consiste en la presión con la que se cierra el sistema.

Pini (Presión inicial): un campo numérico decimal que se completa con el dato medido desde el Neumatrón y consiste en la presión inicial que registra dicho dispositivo.

Pfin (Presión final): un campo numérico decimal que se completa con el dato medido desde el Neumatrón y consiste en la presión final que registra dicho dispositivo.



Para poder agregar nuevos registros se debe utilizar el botón “AGREGAR”, que abre un formulario en donde se completan los nuevos datos. Para modificarse un registro previamente cargado se utiliza el botón “MODIFICAR”. Este abre la misma ventana de formulario que en el caso de “AGREGAR”, pero trae los datos que fueron cargados previamente para ser modificados. Todas las modificaciones se graban con el botón “GUARDAR”



## Funcionamiento

El programa funciona con cinco funciones principales, en primer lugar, la que crea o abre (si ya está creada) la base de datos, llamada “BD\_neumatron” y la tabla “mediciones”.

def crear\_tabla():

    conn = sqlite3.connect("BD\_neumatron.db")

    c = conn.cursor()

    # Crear la tabla mediciones si no existe

    c.execute(

        """CREATE TABLE IF NOT EXISTS mediciones (

                    ID INTEGER PRIMARY KEY AUTOINCREMENT,

                    Muestra TEXT,

                    Especie TEXT,

                    Pplant REAL,

                    Pconfig REAL,

                    Pini REAL,

                    Pfin REAL

                )"""

    )

    conn.commit()

    conn.close()

Las cuatro funciones restantes se encargan de mostrar, agregar, modificar o eliminar registros de mediciones en la base de datos. Dichas funciones son:

### Función mostrar\_registros

def mostrar\_registros():

    # Limpiar la tabla

    tabla.delete(\*tabla.get\_children())

    conn = sqlite3.connect("BD\_neumatron.db")

    c = conn.cursor()

    # Obtener los registros de la tabla mediciones

    c.execute("SELECT \* FROM mediciones")

    registros = c.fetchall()

    # Mostrar los registros en la tabla

    for registro in registros:

        tabla.insert("", "end", values=registro)

    conn.close()

### Función agregar\_registro y la subfunción guardar\_registro

def agregar\_registro():

    def guardar\_registro():

        muestra = entrada\_muestra.get()

        especie = entrada\_especie.get()

        pplant = entrada\_pplant.get()

        pconfig = entrada\_pconfig.get()

        pini = entrada\_pini.get()

        pfin = entrada\_pfin.get()

        # Validar que los campos no estén vacíos

        if muestra and especie and pplant and pconfig and pini and pfin:

            conn = sqlite3.connect("BD\_neumatron.db")

            c = conn.cursor()

            # Insertar el nuevo registro en la tabla mediciones

            c.execute(

                "INSERT INTO mediciones (Muestra, Especie, Pplant, Pconfig, Pini, Pfin) VALUES (?, ?, ?, ?, ?, ?)",

                (muestra, especie, pplant, pconfig, pini, pfin),

            )

            conn.commit()

            conn.close()

            # Actualizar la tabla y cerrar la ventana de formulario

            mostrar\_registros()

            ventana\_agregar.destroy()

        else:

            messagebox.showerror("Error", "Por favor, complete todos los campos.")

    ventana\_agregar = tk.Toplevel(root)

    ventana\_agregar.title("Agregar Registro")

    etiqueta\_muestra = ttk.Label(ventana\_agregar, text="Muestra:")

    etiqueta\_muestra.pack()

    entrada\_muestra = ttk.Entry(ventana\_agregar)

    entrada\_muestra.pack()

    etiqueta\_especie = ttk.Label(ventana\_agregar, text="Especie:")

    etiqueta\_especie.pack()

    entrada\_especie = ttk.Entry(ventana\_agregar)

    entrada\_especie.pack()

    etiqueta\_pplant = ttk.Label(ventana\_agregar, text="Pplant:")

    etiqueta\_pplant.pack()

    entrada\_pplant = ttk.Entry(ventana\_agregar)

    entrada\_pplant.pack()

    etiqueta\_pconfig = ttk.Label(ventana\_agregar, text="Pconfig:")

    etiqueta\_pconfig.pack()

    entrada\_pconfig = ttk.Entry(ventana\_agregar)

    entrada\_pconfig.pack()

    etiqueta\_pini = ttk.Label(ventana\_agregar, text="Pini:")

    etiqueta\_pini.pack()

    entrada\_pini = ttk.Entry(ventana\_agregar)

    entrada\_pini.pack()

    etiqueta\_pfin = ttk.Label(ventana\_agregar, text="Pfin:")

    etiqueta\_pfin.pack()

    entrada\_pfin = ttk.Entry(ventana\_agregar)

    entrada\_pfin.pack()

    boton\_guardar = ttk.Button(

        ventana\_agregar, text="Guardar", command=guardar\_registro

    )

    boton\_guardar.pack()

### Función modificar\_registro

def modificar\_registro():

    seleccionado = tabla.focus()

    if seleccionado:

        datos = tabla.item(seleccionado)["values"]

        def guardar\_modificacion():

            muestra = entrada\_muestra.get()

            especie = entrada\_especie.get()

            pplant = entrada\_pplant.get()

            pconfig = entrada\_pconfig.get()

            pini = entrada\_pini.get()

            pfin = entrada\_pfin.get()

            if muestra and especie and pplant and pconfig and pini and pfin:

                conn = sqlite3.connect("BD\_neumatron.db")

                c = conn.cursor()

                c.execute(

                    """UPDATE mediciones SET

                                Muestra = ?,

                                Especie = ?,

                                Pplant = ?,

                                Pconfig = ?,

                                Pini = ?,

                                Pfin = ?

                            WHERE ID = ?""",

                    (muestra, especie, pplant, pconfig, pini, pfin, datos[0]),

                )

                conn.commit()

                conn.close()

                mostrar\_registros()

                ventana\_modificar.destroy()

            else:

                messagebox.showerror("Error", "Por favor, complete todos los campos.")

        ventana\_modificar = tk.Toplevel(root)

        ventana\_modificar.title("Modificar Registro")

        etiqueta\_muestra = ttk.Label(ventana\_modificar, text="Muestra:")

        etiqueta\_muestra.pack()

        entrada\_muestra = ttk.Entry(ventana\_modificar)

        entrada\_muestra.pack()

        entrada\_muestra.insert(0, datos[1])

        etiqueta\_especie = ttk.Label(ventana\_modificar, text="Especie:")

        etiqueta\_especie.pack()

        entrada\_especie = ttk.Entry(ventana\_modificar)

        entrada\_especie.pack()

        entrada\_especie.insert(0, datos[2])

        etiqueta\_pplant = ttk.Label(ventana\_modificar, text="Pplant:")

        etiqueta\_pplant.pack()

        entrada\_pplant = ttk.Entry(ventana\_modificar)

        entrada\_pplant.pack()

        entrada\_pplant.insert(0, datos[3])

        etiqueta\_pconfig = ttk.Label(ventana\_modificar, text="Pconfig:")

        etiqueta\_pconfig.pack()

        entrada\_pconfig = ttk.Entry(ventana\_modificar)

        entrada\_pconfig.pack()

        entrada\_pconfig.insert(0, datos[4])

        etiqueta\_pini = ttk.Label(ventana\_modificar, text="Pini:")

        etiqueta\_pini.pack()

        entrada\_pini = ttk.Entry(ventana\_modificar)

        entrada\_pini.pack()

        entrada\_pini.insert(0, datos[5])

        etiqueta\_pfin = ttk.Label(ventana\_modificar, text="Pfin:")

        etiqueta\_pfin.pack()

        entrada\_pfin = ttk.Entry(ventana\_modificar)

        entrada\_pfin.pack()

        entrada\_pfin.insert(0, datos[6])

        boton\_guardar = ttk.Button(

            ventana\_modificar, text="Guardar", command=guardar\_modificacion

        )

        boton\_guardar.pack()

    else:

        messagebox.showwarning("Advertencia", "Seleccione un registro para modificar.")

### Función eliminar\_registro

def eliminar\_registro():

    seleccionado = tabla.focus()

    if seleccionado:

        if messagebox.askyesno(

            "Confirmar", "¿Está seguro que quiere eliminar este registro?"

        ):

            datos = tabla.item(seleccionado)["values"]

            id\_registro = datos[0]

            conn = sqlite3.connect("BD\_neumatron.db")

            c = conn.cursor()

            c.execute("DELETE FROM mediciones WHERE ID = ?", (id\_registro,))

            conn.commit()

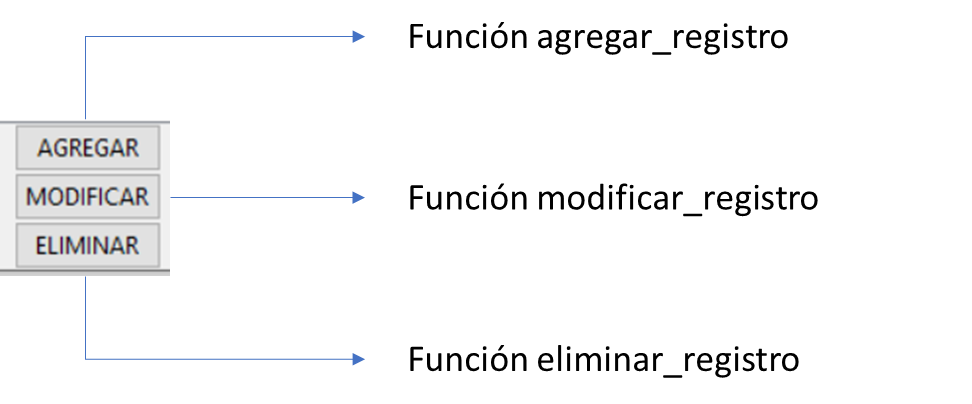
            conn.close()

            mostrar\_registros()

    else:

        messagebox.showwarning("Advertencia", "Seleccione un registro para eliminar.")

Cada función está asociada a un botón que dispara dichas funciones.



La función guardar\_registro se asocia al botón “Guardar”, presente en la ventana para agregar o modificar un registro.

