1. **Project Title:** Electricity Cost Calculator using Python
2. **Overview:**

   This Python application calculates electricity costs based on user input. It offers two modes:
   - Input meter readings (previous and current)
   - Manually input the number of units

   It uses '**tkinter**' & '**customtkinter**' for a modern GUI experience.
3. **Setup Requirements:**

   pip install customtkinter
   pip install tkinter
4. **UI Layout**

   - **Title Frame**: Displays the app title
   - **Main Frame**: Contains radio buttons, entry fields, and buttons
   - **Summary Frame**: Displays calculated total cost

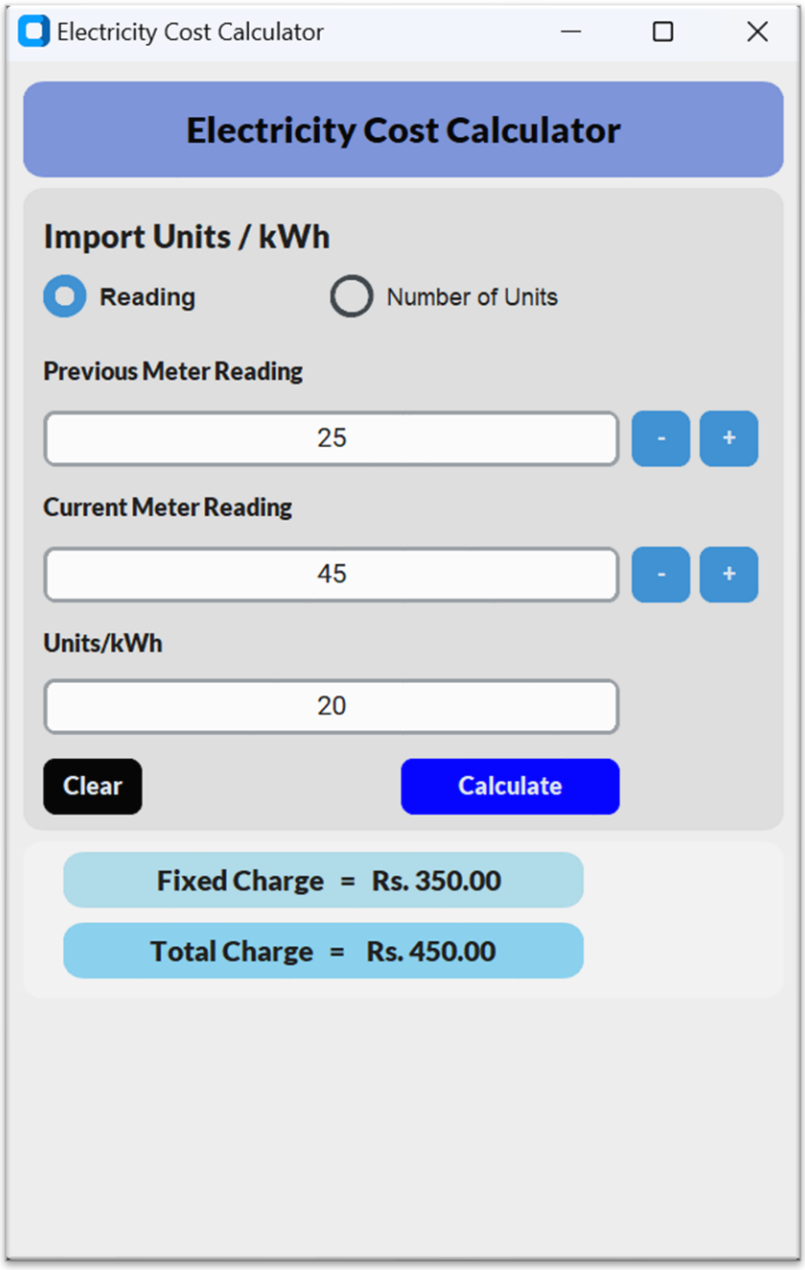
5. **User Interface Components**

   - Frames
   - Labels
   - Entry
   - Radio Buttons
   - Buttons
6. **Major Components Brief Description**

   - App Class
     Handles the layout and GUI logic using customtkinter. CTk
   - radioBtnEvent( )
     Enables/disables fields based on selected radio button.
   - valueIncrease( ) / valueDecreaser( )
     Increments/decrements numeric fields.
   - unitsCal( )
     Calculates units used (current reading – previous reading
   - clear_entries ( )
     Clear the entered value on fields
   - calculate_cost ( )
     Compute the total cost

- calculate_and_update ( )
  Display the total cost on screen

7. User Interface Display



8. Code Explainer

- Firstly, import relevant libraries first.

```
import tkinter
import customtkinter
```

- Create a class named **App**.
  - This class inherits from customtkinter.CTk, which is a modern version of a Tkinter Tk window provided by the customtkinter library. It's used to create the **main window** of application.
  - Initialize **__init__** method: it initializes the application window and sets up its layout and frames.
  - **super().__init__() –** call the constructor of the parent class (setup the window)
  - **self.title_Frame() -** Creates the top section with a title label
  - **self.main_Frame() -** Contains the main content
  - **self.summary_Frame() –** Contains output display labels

```
class App(customtkinter.CTk):
    def __init__(self):
        super().__init__()

        # Set System Settings
        self.title("Electricity Cost Calculator") # Set window title
        self.geometry("400x600")                  # Set window size
        self.grid_rowconfigure(1, weight=1)       # Configure the layout more responsive
        self.grid_columnconfigure(0, weight=1)

        #Frame methods
        self.title_Frame()
        self.main_Frame()
        self.summary_Frame()
```

- Inside the class create a method called **title_Frame()**
  - Firstly create a frame called **titleFrame = customtkinter.CTkFrame(...)**
  - Then, create a label called **titleLabel** ("Electricity Cost Calculator") inside the **titleFrame** as below and set the grid layout it in all direction inside the frame setting **sticky="nsew"**

```
def title_Frame(self):
    # Create the frame
    titleFrame = customtkinter.CTkFrame(self,
                    height=60, corner_radius=10,
                    fg_color="#7a91d7")
    titleFrame.grid(row=0, column=0, padx=10, pady=(10, 0), sticky="ew")
    titleFrame.columnconfigure(0,weight=1)

    # Create the label
    titleLabel = customtkinter.CTkLabel(
        titleFrame,
        text="Electricity Cost Calculator",
        font=("Lato Black", 18, "bold"),
        fg_color="transparent",
        text_color="black",
        anchor="center"
    )
    titleLabel.grid(row=0, column=0, padx=10, pady=10, sticky="nsew")
```

- Inside the class, create another method called **main_Frame()**
  - Firstly create a frame called **mainFrame = customtkinter.CTkFrame(...)**
  - Then, create a label called **subTitleLabel (** "Import Units / kWh" **)** inside the **mainFrame** as below and set the grid layout
  - Also,create two radio buttons and set inside the mainFrame (**" Reading " & "Number of Units"**)
  - Two radio buttons are used to choose the input method.
  - **"Reading"** radio button enables to enter the previous meter reading and current meter reading (Value = 1)
  - "**Number of Units**" radio button enables to enter the no of units manually. (Value = 2)
  - Both buttons call the **radioBtnEvent(...)** function to enable/disable the right input fields. (Explained the function below)
  - Previous Meter Reading Section : Using same approach design a label and entry field for previous reading.

- Current Meter Reading Section : Using the same approach design a label and entry field for current reading
- Also, design four buttons for increase and decrease the entered values by one on **preReadingEntry** and **curReadingeEntry.** Design functions to maintain the increasing and decreasing mechanism. (Explained the function below)
- Those buttons create inside frames such as **"preBtnFrame"** & **"curBtnFrame"**
- To enter or display the no of units, design an entry field with a label called **"Units/kWh"**
- To clear the user inputs on entries, create a button called "Clear". To clear the user inputs, define a function called **"clear_entries(...) "** (Explained the function below)
- To calculate the total cost , create a button called "Calculate". To define the how the cost should be calculated, define a function called **"calculate_and_update(...) "**
- When one of the above mentioned button is clicke, to call the function used command as below.
    - **command=lambda:clear_entries(...)**
    - **command=lambda:calculate_and_update(...)**

```
def main_Frame(self):
        # Create a frame
        mainFrame  = customtkinter.CTkFrame(self,height=200,corner_radius=10)
        mainFrame.grid(row=1, column=0, padx=10, pady=(5, 5), sticky="nsew")
        mainFrame.columnconfigure((0,1),weight=1)#Expand equally between col0 & 1
        mainFrame.grid_propagate(False)

        # Create a label
        subTitleLabel = customtkinter.CTkLabel(
            mainFrame,
            text="Import Units / kWh",
            font=("Lato Black", 16, "bold"),
            fg_color="transparent",
            anchor="w"
        )
        subTitleLabel.grid(row=0, column=0, columnspan=2, padx=10,
                           pady=(10, 5), sticky="w")

        self.radio_var = tkinter.IntVar(value=0)

        #Radio Button 01 : "Reading"
        self.radio_btn1 = customtkinter.CTkRadioButton(
            mainFrame,
            text="Reading",
            font=("Lato", 12),
            command=lambda: radioBtnEvent(
                self.radio_var,
                self.radio_btn1,
                self.radio_btn2,
                self.preReadingEntry,
                self.curReadingEntry,
                self.noOfUnitsEntry
            ),
            variable=self.radio_var,
            value=1
        )
        self.radio_btn1.grid(row=1, column=0, padx=(10, 5),
                             pady=(0, 10), sticky="w")
```

```
        #Radio Button 02 : "Number of Units"
        self.radio_btn2 = customtkinter.CTkRadioButton(
            mainFrame,
            text="Number of Units",
            font=("Lato", 12),
            command=lambda: radioBtnEvent(
                self.radio_var,
                self.radio_btn1,
                self.radio_btn2,
                self.preReadingEntry,
                self.curReadingEntry,
                self.noOfUnitsEntry
            ),
            variable=self.radio_var,
            value=2
        )

        self.radio_btn2.grid(row=1, column=1, padx=(10, 5),
                                 pady=(0, 10), sticky="w")

        # Pre Meter Reading
        preReadingLabel = customtkinter.CTkLabel(
            mainFrame,
            text="Previous Meter Reading",
            font=("Lato Black", 12, "bold"),
            fg_color="transparent",
            anchor="w"
        )
        preReadingLabel.grid(row=2, column=0, columnspan=2, padx=10,
                                            pady=(2, 2),    sticky="w")

        # Pre Reading Input Entry
        self.preReadingEntry = customtkinter.CTkEntry(mainFrame,
                               width = 350,justify ="center")

        self.preReadingEntry.grid(row=3,column=0,columnspan=2,
                                 padx=(10, 2),pady=(2,2),sticky="w")

        # Pre reading value increse or decrease frame
        preBtnFrame = customtkinter.CTkFrame(mainFrame,fg_color="transparent")
        preBtnFrame.grid(row=3, column=2, columnspan=2, padx=(2,10), pady=(2, 2), sticky="w")
```

```
        # Pre meter reading decreaser
        pre_Dec_Btn = customtkinter.CTkButton(preBtnFrame,
                      text="-",
                      font=("Lato Black", 12, "bold"),
                      width=30,
                      command=lambda:valueDecreaser(self.preReadingEntry))

        pre_Dec_Btn.grid(row=0, column=0, padx=2, pady=(2, 2), sticky="w")

        # Pre meter reading increaser
        pre_Inc_Btn = customtkinter.CTkButton(preBtnFrame,
                      text="+",
                      font=("Lato Black", 12, "bold"),
                      width=30,
                      command=lambda:valueIncreaser(self.preReadingEntry))
        pre_Inc_Btn.grid(row=0, column=1, padx=2, pady=(2, 2), sticky="w")

        # Current Meter Reading
        curReadingLabel = customtkinter.CTkLabel(
            mainFrame,
            text="Current Meter Reading",
            font=("Lato Black", 12, "bold"),
            fg_color="transparent",
            anchor="w"
        )
        curReadingLabel.grid(row=4, column=0, columnspan=2, padx=10, pady=(2, 2), sticky="w")

        # Current Reading Input Entry
        self.curReadingEntry = customtkinter.CTkEntry(mainFrame,
                               width = 350,
                               justify ="center")

        self.curReadingEntry.grid(row=5,column=0,
                                 columnspan=2,padx=(10, 2),pady=(2,2),sticky="w")


        # Current reading value increse or decrease frame
        curBtnFrame = customtkinter.CTkFrame(mainFrame,fg_color="transparent")
        curBtnFrame.grid(row=5, column=2, columnspan=2, padx=(2,10), pady=(2, 2), sticky="w")
```

```
        # Current meter reading decreaser
        cur_Dec_Btn = customtkinter.CTkButton(curBtnFrame,
                        text="-",
                        font=("Lato Black", 12, "bold"),
                        width=30,
                        command=lambda:valueDecreaser(self.curReadingEntry))
        cur_Dec_Btn.grid(row=0, column=0, padx=2, pady=(2, 2), sticky="w")

        # Current meter reading increaser
        cur_Inc_Btn = customtkinter.CTkButton(curBtnFrame,
                        text="+",
                        font=("Lato Black", 12, "bold"),
                        width=30,
                        command=lambda:valueIncreaser(self.curReadingEntry))
        cur_Inc_Btn.grid(row=0, column=1, padx=2, pady=(2, 2), sticky="w")

        # Units/kWh
        noOfUnitsLabel = customtkinter.CTkLabel(
            mainFrame,
            text="Units/kWh",
            font=("Lato Black", 12, "bold"),
            fg_color="transparent",
            anchor="w"
        )
        noOfUnitsLabel.grid(row=6, column=0,
                            columnspan=2,
                            padx=10,
                            pady=(2, 2), sticky="w")

        # Units/kWh Reading Input Entry
        self.noOfUnitsEntry = customtkinter.CTkEntry(mainFrame,width = 350,justify ="center")
        self.noOfUnitsEntry.insert(0, "0")
        self.noOfUnitsEntry.grid(row=7,
                                 column=0,
                                 columnspan=2,
                                 padx=(10, 2),
                                 pady=(2,2),sticky="w")
```

```
        # Clear the entry content
        clearBtn = customtkinter.CTkButton(mainFrame,
                                           text="Clear",
                                           width=50,
                                           font=("Lato Black", 12, "bold"),
                                           fg_color="black",
                                           border_color="black",
                                           border_width=1,
                                           command=lambda:
                                   clear_entries(self.preReadingEntry,
                                                 self.curReadingEntry,
                                                 self.noOfUnitsEntry,
                                                 self.totalCostLabel)
                                           )
        clearBtn.grid(row=8,column=0,padx=(10, 5),pady=(10,10),sticky="w")

        # Calculate the total cost
        calculateBtn = customtkinter.CTkButton(mainFrame,
                                               text="Calculate",
                                               width=110,
                                               font=("Lato Black", 12, "bold"),
                                               fg_color="blue",
                                               border_color="black",
                                               command=lambda:calculate_and_update(
                                                   self.noOfUnitsEntry,
                                                   self.totalCostLabel,
                                                   self.radio_var,
                                                   self.preReadingEntry,
                                                   self.curReadingEntry
                                                   )
                                               )
        calculateBtn.grid(row=8,column=1,columnspan=8,padx=(45, 30),pady=(10,10),sticky="w")
```

- Inside the class, create a method called **summary_Frame()** to display the total cost.
  - In here firstly create a frame as above called "**summaryFrame**"
  - Inside the frame, create a label to display the fixed cost naming "**fixedCostLabel**"
  - Also create a label to display the total charge that need to pay as "**totalCostLabel**"

```
# Summary Frame (below mainFrame)
def summary_Frame(self):
    summaryFrame = customtkinter.CTkFrame(self,
                    corner_radius=10, fg_color="#f0f0f0",width=100)
    summaryFrame.grid(row=2, column=0, padx=10, pady=(0, 10), sticky="nsew")
    self.grid_rowconfigure(3, weight=1)
    self.columnconfigure(0,weight=1)

    self.grid_columnconfigure(0, weight=1)


    self.fixedCostLabel = customtkinter.CTkLabel(
       summaryFrame,
       text="  Fixed Charge   =   Rs. 350.00",
       font=("Lato Black", 14, "bold"),
       anchor="center",
       fg_color="light blue",
       corner_radius=10,
       width=260
    )
    self.fixedCostLabel.grid(row=3, column=0,padx=20, pady=(5,2),sticky="ew")


    self.totalCostLabel = customtkinter.CTkLabel(
       summaryFrame,
       text="  Total Charge   =  Rs. 0.00",
       font=("Lato Black", 14, "bold"),
       anchor="center",
       fg_color="sky blue",
       corner_radius=10,
       width=260
    )
    self.totalCostLabel.grid(row=5, column=0,padx=20, pady=(5,10),sticky="ew")
```

- To execute the code below code is added. Also App.mainloop() will keep the window open and responsive until user close it.
- Also define a color theme and keep the same window size disabling resizing option.

```
if __name__ == "__main__":
    customtkinter.set_default_color_theme("blue")
    app = App()
    app.resizable(False, False)
    app.mainloop()
```

9. **Functions Used:**

- **radioBtnEvent(...) :** This function is used to handle what happens when the user toggles between two radio buttons:
  - **"Reading"** mode (enter previous + current meter readings)
  - **"Number of Units"** mode (directly enter the kWh)
  - It enables/disables the correct fields and updates the UI appearance, ensuring only the relevant inputs are editable.

```
#Radio Button Function
def radioBtnEvent(radio_var,radio_btn1,radio_btn2,pre_Entry,cur_Entry,units_entry):
    print("Radio Btn Toggled. currentValue : ",radio_var.get())
    # Reset both fonts
    radio_btn1.configure(font=("Lato", 12))
    radio_btn2.configure(font=("Lato", 12))

    # Reading
    if radio_var.get() == 1:
        radio_btn1.configure(font=("Lato", 12, "bold"))
        pre_Entry.configure(state="normal")
        cur_Entry.configure(state="normal")
        units_entry.configure(state="readonly")
        unitsCal(pre_Entry, cur_Entry, units_entry)  # Auto-calculate

    # No of Units
    elif radio_var.get() == 2:
        radio_btn2.configure(font=("Lato", 12, "bold"))
        pre_Entry.configure(state="disabled")
        cur_Entry.configure(state="disabled")
        units_entry.configure(state="normal")
        units_entry.delete(0, "end")
```

- **valueIncreaser(...) :** Use to increase the preEntryReading and curEntryReading entries quantity by one.
- **valueDecreaser(...) :** Use to decrease the preEntryReading and curEntryReading entries quantity by one.

```
def valueIncreaser(entryValue):
    try:
        value = int(entryValue.get())
        entryValue.delete(0, "end")
        entryValue.insert(0,str(value+1))
    except ValueError:
        entryValue.delete(0, "end")
        entryValue.insert(0, "0")

def valueDecreaser(entryValue):
    try:
        value = int(entryValue.get())
        entryValue.delete(0, "end")
        entryValue.insert(0,str(max(0, value - 1)))
    except ValueError:
        entryValue.delete(0, "end")
        entryValue.insert(0, "0")
```

- **unitsCal(...) :** To calculate the number of units from previous meter reading and current meter reading, design a function called **unitsCal(...).** In here use a try except to validate the user input. If the user input is less than zero, it will print as ‘ **Invalid Input** ’ . Then will configure the output on the screen.
    - Number of units used = Current Meter Reading – Previous Meter Reading

```
def unitsCal(pre_Entry, cur_Entry, result_Entry):
    try:
        pre_val = int(pre_Entry.get())
        cur_val = int(cur_Entry.get())
        no_Of_Units = cur_val - pre_val
        if no_Of_Units < 0:
            print("Invalid Input")
            result_Entry.insert(0,"Invalid Input")
```

```
        else:
            print("No of Units:", no_Of_Units)
            result_Entry.configure(state="normal")
            result_Entry.delete(0, "end")
            result_Entry.insert(0, str(no_Of_Units))
            result_Entry.configure(state="readonly")
    except ValueError:
        result_Entry.configure(state="normal")
        result_Entry.delete(0, "end")
        result_Entry.configure(state="readonly")
```

- **clear_Entries(...)** : To clear all input fields (meter readings and units), reset states, and set the cost label back to default. It's triggered by a "Clear" button in the UI.

  - **Parameters:**
    - **pre_Entry:** The Previous Meter Reading input field.
    - **cur_Entry:** The Current Meter Reading input field.
    - **units_entry:** The Units/kWh input field.
    - **costLabel:** The label that displays the calculated total cost.
  - Also, enable all entry fields, clear the input fields and reset the cost label to zero.

```
def clear_entries(pre_Entry,cur_Entry,units_entry,costLabel):
    pre_Entry.configure(state="normal")
    cur_Entry.configure(state="normal")
    units_entry.configure(state="normal")

    pre_Entry.delete(0, "end")
    cur_Entry.delete(0, "end")
    units_entry.delete(0, "end")

    costLabel.configure(text= "Total Charge = Rs.  0.00")
```

- **calculate_cost(...)** : Calculate the total cost for entered no of units and will return the calculated total cost. In here use a try except to validate the user input.

  - **Parameters:**
    - **units_Entry :** The Units/kWh input field
  - Total cost is calculated as below.
    - If the no of units used is less than or equal to 20 ,

      Total cost = Fixed cost + no of units * Rs. 5.00
    - If the no of units used is less than or equal to 50 ,

      Total cost = Fixed cost + 20 * Rs. 5.00 + (no of units – 20) * Rs. 5.00
    - If the no of units is greater than 50,

      Total cost = Fixed cost + 20* Rs. 5.00 + 30 * Rs. 7.00 +

      (no of units – 50) * Rs. 10.00
    - If the no of units is less than zero, will raise a value error as "**value cannot be negative**"

```python
def calculate_cost(units_entry):
    try:
        units = int(units_entry.get())
        total_cost = 0.0

        if units < 0:
            raise ValueError("Units cannot be negative.")

        if units <= 20:
            total_cost = 350.00 + units * 5.00
        elif units <= 50:
            total_cost = 350.00 + 20 * 5.00 + (units - 20) * 7.00
        else:
            total_cost = 350.00 + 20 * 5.00 + 30 * 7.00 + (units - 50) * 10.00

        print(f"Total Cost: Rs. {total_cost:.2f}")

        return total_cost
```

```
    except ValueError:
        print("Invalid input for units.")
        return None
```

- **calculate_and_update(...) :** Used to call the calculate function to calculate the total cost and update the relevant labels with the output. It is triggered by "Calculate" button in the UI.
    - **Parameters :**
        - **units_entry :** The Units/kWh input field
        - **totalLabel :** The label that displays the calculated total cost.
        - **radio_var :** receive the which radio button is set
        - **pre_Entry :** The Previous Meter Reading input field
        - **cur_Entry :** The Current Meter Reading input field
    - Firstly, create a if condition to check whether "**Reading**" radio button (Value = 1) is selected. If that condition is true, inside a try except action to take the previous and current meter reading and to check the no of units is less than zero.
    - If that condition is met, configure the label as below.
    - Also, if no of units are greater than zero then, update total cost displaying label with the calculated cost.
    - If the **radio_var** is not equal to 1 means , **"number of units"** buttons is selected. So, **(value = 2)** , then calculate the total cost against entered no of units and update the total cost displayed label.
    - In above both two ways used **calculate_cost(...)** function to calculate the total cost inside the **calculate_and_update(...)**

```
def calculate_and_update(units_entry, totalLabel, radio_var, pre_Entry, cur_Entry):
    if radio_var.get() == 1:
        try:
            pre_val =int(pre_Entry.get())
            cur_val =int(cur_Entry.get())
            units = cur_val-pre_val
```

```
            if units < 0:
                units_entry.configure(state="normal")
                units_entry.delete(0,"end")
                units_entry.insert(0,"Invalid")
                units_entry.configure(state="readonly")
                totalLabel.configure(text="Total Charge   : Invalid Reading")
                return

            units_entry.configure(state="normal")
            units_entry.delete(0,"end")
            units_entry.insert(0,str(units))
            units_entry.configure(state="readonly")

            cost = calculate_cost(units_entry)

            if cost is not None:
                totalLabel.configure(text=f"Total Charge   =    Rs. {cost:.2f}")

        except ValueError:
            units_entry.configure(state="normal")
            units_entry.delete(0,"end")
            units_entry.insert(0,"Invalid")
            units_entry.configure(state="readonly")
            totalLabel.configure(text="Total Charge  =  Invalid Input")

    else:
        cost = calculate_cost(units_entry)
        if cost is not None:
            totalLabel.configure(text=f"Total Charge   =    Rs. {cost:.2f}")

        pre_Entry.configure(state="normal")
        cur_Entry.configure(state="normal")
        pre_Entry.delete(0,"end")
        cur_Entry.delete(0,"end")
        pre_Entry.insert(0,"-")
        cur_Entry.insert(0,"-")
```

Date : 06/19/2025

Name : Sandarenu Dassanayaka