Converter Project

Assignment 05

28/02/2025

Group 03
Team Members:

- Youzhou Xu
- Chathurika Madurangi Mohottige
- Chamithu Dewnith Jayathilake Panditharanthna Wijesuriya
- Radin Dinsara Gunarathne
- Hussein Suleiman
- Pham Hong Tan

Group 03 - Trello Board Project link : https://trello.com/b/nJgjNTbn/group-03-computer-science

GitHub Converter - Project repository link : https://github.com/radindinsara/Converter-Project

---

Project Description:

This project involves building a Unit Converter application using Tkinter in Python. The converter will allow users to convert between different units of measurement, such as:

```
1.Temperature (Celsius ↔ Fahrenheit)
2.Length (Miles ↔ Kilometers)
3.Weight (Pounds ↔ Kilograms)
```

Task and Goal of the Project:

The goal of this project is to create a graphical user interface (GUI) that allows users to convert between different units. The user will input a value, select the type of conversion (Temperature, Length, or Weight), and view the result in a display area.

The application will dynamically adjust the available conversion types based on the user's selection. Additionally, the result display will change its background color depending on the selected task:

```
Temperature: Light orange
Length: White
Weight: Light blue
```

Tools :
- **Python** for development
- **Tkinter** for UI
- **Jupyter Notebook** for testing
- **Git & GitHub** for version control
- **VS Code** as the primary IDE.

Main Window Setup (UI Components)

In [13]:
```python
import tkinter as tk
from tkinter import ttk, messagebox

# Conversion Functions
# Convert Fahrenheit to Celsius
def fahrenheit_to_celsius(f):
    return (f - 32) * 5/9

def celsius_to_fahrenheit(c):
    return c * 9/5 + 32

def miles_to_km(miles):
    return miles * 1.60934

def km_to_miles(km):
    return km / 1.60934
```

```python
def pounds_to_kg(pounds):
    return pounds * 0.453592

def kg_to_pounds(kg):
    return kg / 0.453592


def show_result():
    try:
        value = float(value_entry.get())
        conversion = conversion_dropdown.get()

        if task_dropdown.get() == "Temperature":
            if conversion == "Fahrenheit to Celsius":
                result = fahrenheit_to_celsius(value)
                result_text = f"{value}°F = {result:.2f}°C"
            else:
                result = celsius_to_fahrenheit(value)
                result_text = f"{value}°C = {result:.2f}°F"

        elif task_dropdown.get() == "Length":
            if conversion == "Miles to Kilometers":
                result = miles_to_km(value)
                result_text = f"{value} miles = {result:.2f} km"
            else:
                result = km_to_miles(value)
                result_text = f"{value} km = {result:.2f} miles"

        elif task_dropdown.get() == "Weight":
            if conversion == "Pounds to Kilograms":
                result = pounds_to_kg(value)
                result_text = f"{value} pounds = {result:.2f} kg"
            else:
                result = kg_to_pounds(value)
                result_text = f"{value} kg = {result:.2f} pounds"


        result_window.config(state="normal")
        result_window.delete(1.0, tk.END)
        result_window.insert(tk.END, result_text)
        result_window.config(state="disabled")


        if task_dropdown.get() == "Temperature":
            result_window.config(bg="#FFDDC1")  # Light orange
        elif task_dropdown.get() == "Length":
            result_window.config(bg="#FFFFFF")  # White
        elif task_dropdown.get() == "Weight":
            result_window.config(bg="#ADD8E6")  # Light blue

    except ValueError:
        messagebox.showerror("Input Error", "Please enter a valid number.")


def go_back():
    result_window.config(state="normal")
    result_window.delete(1.0, tk.END)
    result_window.config(state="disabled")
    value_entry.delete(0, tk.END)
    task_dropdown.set(task_options[0])
    conversion_dropdown.set(conversion_options[0])


root = tk.Tk()
root.title("Unit Converter")
root.geometry("450x350")
root.resizable(False, False)


style = ttk.Style()
style.configure("TLabel", font=("Arial", 12))
style.configure("TButton", font=("Arial", 12), padding=6)
style.configure("TCombobox", font=("Arial", 12), padding=6)


task_label = ttk.Label(root, text="Select Task:")
task_label.grid(row=0, column=0, padx=10, pady=10)

task_options = ['Temperature', 'Length', 'Weight']
task_dropdown = ttk.Combobox(root, values=task_options, state="readonly")
task_dropdown.grid(row=0, column=1, padx=10, pady=10)
task_dropdown.set(task_options[0])
```

```
conversion_label = ttk.Label(root, text="Select Conversion:")
conversion_label.grid(row=1, column=0, padx=10, pady=10)

conversion_options = []
conversion_dropdown = ttk.Combobox(root, values=conversion_options, state="readonly")
conversion_dropdown.grid(row=1, column=1, padx=10, pady=10)


def update_conversion_options(event):
    selected_task = task_dropdown.get()

    if selected_task == "Temperature":
        conversion_options = ['Fahrenheit to Celsius', 'Celsius to Fahrenheit']
    elif selected_task == "Length":
        conversion_options = ['Miles to Kilometers', 'Kilometers to Miles']
    elif selected_task == "Weight":
        conversion_options = ['Pounds to Kilograms', 'Kilograms to Pounds']

    conversion_dropdown['values'] = conversion_options
    conversion_dropdown.set(conversion_options[0])

task_dropdown.bind("<<ComboboxSelected>>", update_conversion_options)


value_label = ttk.Label(root, text="Enter Value:")
value_label.grid(row=2, column=0, padx=10, pady=10)

value_entry = ttk.Entry(root, font=("Arial", 12))
value_entry.grid(row=2, column=1, padx=10, pady=10)


result_button = ttk.Button(root, text="Show Result", command=show_result)
result_button.grid(row=3, column=0, columnspan=2, pady=10)


result_window = tk.Text(root, height=4, width=35, font=("Arial", 12), wrap=tk.WORD, state="disabled", bg="#FFFF
result_window.grid(row=4, column=0, columnspan=2, padx=10, pady=10)


back_button = ttk.Button(root, text="Back", command=go_back)
back_button.grid(row=5, column=0, columnspan=2, pady=10)


group_label = ttk.Label(root, text="Group 03 Project", font=("Arial", 10, "italic"))
group_label.grid(row=6, column=0, columnspan=2, pady=10)


root.mainloop()
```

Temperature Conversion

In [8]:
```
def fahrenheit_to_celsius(f):
    return (f - 32) * 5/9

def celsius_to_fahrenheit(c):
    return c * 9/5 + 32
```

Length Conversion

In [9]:
```
def miles_to_km(miles):
    return miles * 1.60934

def km_to_miles(km):
    return km / 1.60934
```

Weight Conversion

In [10]:
```
def pounds_to_kg(pounds):
    return pounds * 0.453592

def kg_to_pounds(kg):
    return kg / 0.453592
```

Result Display and Dynamic Background Color Change

In [11]:
```
def show_result():
    try:
        value = float(value_entry.get())
        conversion = conversion_dropdown.get()
        if task_dropdown.get() == "Temperature":
            if conversion == "Fahrenheit to Celsius":
                result = fahrenheit_to_celsius(value)
                result_text = f"{value}°F = {result:.2f}°C"
            else:
                result = celsius_to_fahrenheit(value)
                result_text = f"{value}°C = {result:.2f}°F"

        elif task_dropdown.get() == "Length":
            if conversion == "Miles to Kilometers":
                result = miles_to_km(value)
                result_text = f"{value} miles = {result:.2f} km"
            else:
                result = km_to_miles(value)
                result_text = f"{value} km = {result:.2f} miles"

        elif task_dropdown.get() == "Weight":
            if conversion == "Pounds to Kilograms":
                result = pounds_to_kg(value)
                result_text = f"{value} pounds = {result:.2f} kg"
            else:
                result = kg_to_pounds(value)
                result_text = f"{value} kg = {result:.2f} pounds"


        result_window.config(state="normal")
        result_window.delete(1.0, tk.END)
        result_window.insert(tk.END, result_text)
        result_window.config(state="disabled")

        # Change background color based on selected task
        if task_dropdown.get() == "Temperature":
            result_window.config(bg="#FFDDC1")  # Light orange
        elif task_dropdown.get() == "Length":
            result_window.config(bg="#FFFFFF")  # White
        elif task_dropdown.get() == "Weight":
            result_window.config(bg="#ADD8E6")  # Light blue

    except ValueError:
        messagebox.showerror("Input Error", "Please enter a valid number.")
```

Back Button Function

In [12]:
```
def go_back():
    result_window.config(state="normal")
    result_window.delete(1.0, tk.END)
    result_window.config(state="disabled")
    value_entry.delete(0, tk.END)
    task_dropdown.set(task_options[0])
    conversion_dropdown.set(conversion_options[0])
```

Conclusion:

By breaking down the project into separate components like temperature, length, and weight conversion logic, the program becomes modular and easy to maintain.
Each component is responsible for a single task and is held accountable for a single conversion operation.
This makes the code more readable, modular, and extensible too.

Unit Converter Project - Enhancements and Updates

New Features and Enhancements

This section outlines the new features and improvements that have been added to the Unit Converter project to make it more powerful, user-friendly, and feature-rich.

---

1. Add More Conversion Types

We've expanded the conversion options to support more types of units:

Currency Conversion

- Convert between various currencies (e.g., USD to EUR, GBP to USD).

Time Conversion

- Convert between different time units (seconds, minutes, hours, days).

Area Conversion

- Convert between different units of area (square meters to square kilometers, acres to square feet, etc.).

Speed Conversion

- Convert between different speed units (miles per hour to kilometers per hour, meters per second to miles per hour, etc.).

---

2. Add a History Feature

Now, you can keep track of your previous conversions with a **conversion history** feature:

- **History List**: Keep the last 5 conversions in a list, which will be displayed in a history window.
- **Quick Access**: Users can click on any historical conversion to quickly input the value and conversion type again.

---

3. Improve Error Handling

Error handling has been enhanced to ensure the app provides helpful feedback:

Input Validation

- Ensure the user enters a valid number in the input field (handle non-numeric values, empty fields, etc.).
- Display error messages for invalid inputs (e.g., a message like "Please enter a valid number").

Unit Compatibility Check

- Ensure that the selected units are compatible for conversion (e.g., you cannot convert Fahrenheit to kilometers).
- Provide an error message or a notification when incompatible units are selected.

---

4. Enhance the User Interface (UI)

We've improved the overall UI for a more modern and customizable experience:

Dark Mode/Light Mode Responsive Design Tooltips and Help Section

---

5. Add Multi-language Support

To make the application accessible to a global audience, we've added multi-language support:

- **Internationalization**: The application can now display content in different languages (e.g., English, Spanish, French, etc.).
- **Language Dropdown**: Users can select their preferred language from a dropdown menu.

---

6. Unit Converter for Large Numbers

The converter now handles large and small numbers with ease:

Scientific Notation

- For very large or very small numbers, display the results in scientific notation (e.g., `1.23e+5` for large numbers).

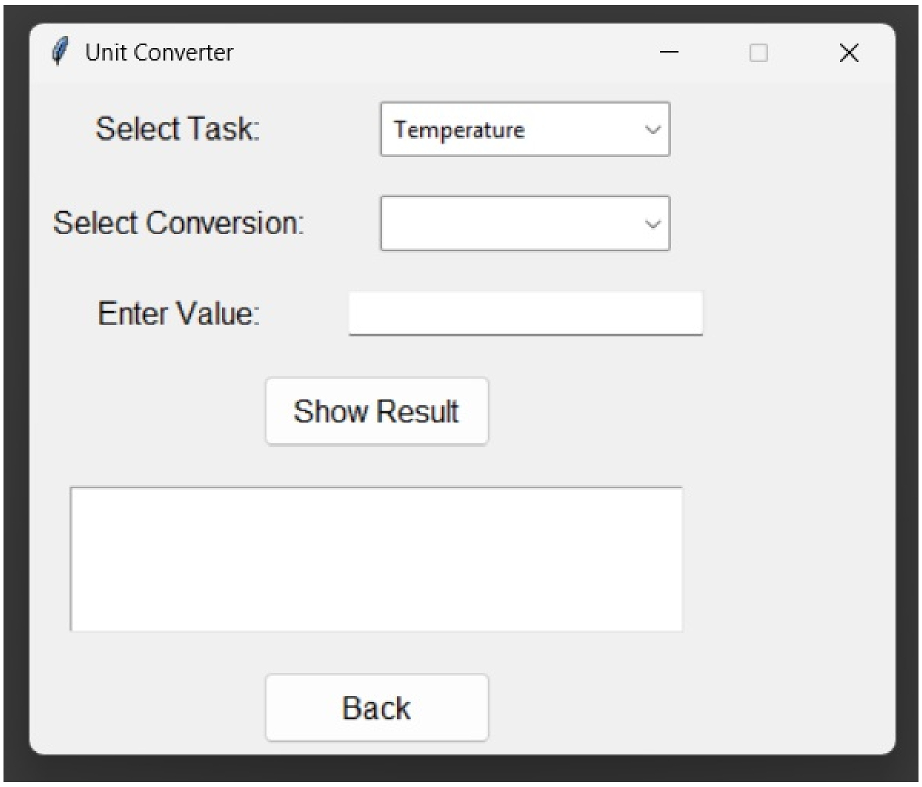
Custom Precision

- Allow users to set the precision for results (e.g., number of decimal places).

Example Implementation:

- Use Python's string formatting to display large numbers in scientific notation.
- Add a setting or input field where the user can specify the number of decimal places for conversion results.

Sample Output



```
In [ ]:
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js