# 

# **TEMPERATURE CONVERTER USING TKINTER**

# **A GUI based python application Created**

# **with Tkinter**

# N Karthikeya 122210701134

# S Srikanth 122210701152

# M Dhanush 122210701129

|  |  |  |
| --- | --- | --- |
| S.NO | Chapter Name | Page No |
| 1. | Introduction | 3 |
| 2. | Abstract | 4 |
| 3. | Advantages | 5 |
| 4. | Disadvantages | 6 |
| 5. | Installation Guide | 7 |
| 6. | Code Structure | 8-11 |
| 7. | GUI Components & Structure | 12 |
| 8. | Functionallity Overview | 13-14 |
| 9. | Conversion Logic | 15 |
| 10. | Input Validation | 16-17 |
| 11. | Color-coding temperature changes | 17-19 |
| 12. | Error Handling | 20 |
| 13. | Trouble Shooting | 22-23 |
| 14. | Conclusion | 24 |

# 1. Introduction

The Temperature Converter Application is a graphical user interface (GUI) tool built with Python's Tkinter library. It allows users to seamlessly convert temperatures between Celsius, Fahrenheit, and Kelvin using an intuitive and user-friendly interface. Designed for accuracy and ease of The Temperature Converter Application is a graphical user interface (GUI) tool built with Python's Tkinter library. Its primary purpose is to provide a simple and intuitive way for users to convert temperatures between different units, specifically Celsius (°C), Fahrenheit (°F), and Kelvin (K). The application was designed with ease of use in mind, ensuring that users can quickly perform temperature conversions with minimal effort. This tool is ideal for anyone needing to convert temperatures in scientific, educational, or everyday settings.use, the application features real-time error handling and color-coded results for better readability.

2. Abstract

The Python-based temperature converter utilizes the Tkinter library to create a graphical user interface (GUI). It allows users to convert temperatures between Celsius, Fahrenheit, and Kelvin. The conversion process is user-friendly, requiring users to input a temperature value and select a conversion option from radio buttons. Upon clicking the "Convert" button, the application performs the necessary calculation and displays the converted temperature. To enhance usability, the program color-codes results:

- \*Red\* for high temperatures

- \*Orange\* for moderate temperatures

- \*Green\* for low temperatures

Additionally, error handling ensures users receive appropriate messages in case of invalid inputs, such as non-numeric values or unselected conversions.

# 3. Advantages

1. \*User-Friendly Interface\* – The Tkinter-based GUI simplifies interactions, making the application accessible even to users without programming knowledge.

2. \*Multi-Conversion Options\* – The tool supports multiple temperature conversions, enhancing flexibility:

- Celsius to Fahrenheit

- Fahrenheit to Celsius

- Celsius to Kelvin

- And other standard conversions

3. \*Error Handling\* – Basic error handling ensures users are informed about invalid inputs or missing selections, improving reliability.

4. \*Color-Coded Results\* – Visual indicators help users quickly interpret temperature ranges, making the tool more intuitive.

# 4. Disadvantages

1. \*Limited to Basic Conversions\* – The application only supports predefined temperature conversions, preventing more complex or user-defined conversions.

2. \*No Input Validation for Extreme Values\* – While it checks for non-numeric input, the program does not handle edge cases, such as extremely high or low temperatures, which could cause unrealistic results.

3. \*Basic GUI Design\* – The interface, while functional, lacks advanced features like tooltips, animations, or modern styling, which could improve user experience.

4. \*Single-Functionality Tool\* – The application focuses solely on temperature conversion and does not support other unit conversions like length, weight, or volume.

# 5. Installation Guide

1. Download and install Python from the official website.
2. Clone or download the Temperature Converter repository.
3. Navigate to the project directory in a terminal or command prompt.
4. Execute python temperature\_converter.py to run the application.

# 6. Code Structure

The application follows a structured modular approach, including:

* convert\_temperature(): Manages temperature conversion logic.
* setup\_gui(): Initializes and designs the graphical interface.
* validate\_input(): Ensures only numerical values are entered.
* import tkinter as tk
* from tkinter import messagebox
* def convert\_temperature():
* try:
* temp = float(entry.get())
* selected\_option = var.get()
* result = None
* if selected\_option == "C to F":
* result = (temp \* 9/5) + 32
* result\_text = f"{temp:.2f}°C = {result:.2f}°F"
* elif selected\_option == "F to C":
* result = (temp - 32) \* 5/9
* result\_text = f"{temp:.2f}°F = {result:.2f}°C"
* elif selected\_option == "C to K":
* result = temp + 273.15
* result\_text = f"{temp:.2f}°C = {result:.2f}K"
* elif selected\_option == "K to C":
* result = temp - 273.15
* result\_text = f"{temp:.2f}K = {result:.2f}°C"
* elif selected\_option == "F to K":
* celsius = (temp - 32) \* 5/9
* result = celsius + 273.15
* result\_text = f"{temp:.2f}°F = {result:.2f}K"
* elif selected\_option == "K to F":
* celsius = temp - 273.15
* result = (celsius \* 9/5) + 32
* result\_text = f"{temp:.2f}K = {result:.2f}°F"
* else:
* messagebox.showerror("Error", "Please select a conversion type.")
* return
* # Determine temperature range and color
* if result is not None:
* if result > 85: # High temperature
* result\_color = "red"
* elif 60 <= result <= 85: # Moderate temperature
* result\_color = "orange"
* else: # Good temperature
* result\_color = "green"
* result\_label.config(text=result\_text, fg=result\_color)
* except ValueError:
* messagebox.showerror("Invalid Input", "Please enter a valid number.")
* # GUI setup
* root = tk.Tk()
* root.title("Temperature Converter")
* # Styling
* root.config(bg="lightblue")
* # Input Section
* tk.Label(root, text="Enter Temperature:", bg="lightblue", font=("Arial", 12)).grid(row=0, column=0, padx=10, pady=10)
* entry = tk.Entry(root, font=("Arial", 12))
* entry.grid(row=0, column=1, padx=10, pady=10)
* # Conversion Options
* var = tk.StringVar(value="C to F")
* options = ["C to F", "F to C", "C to K", "K to C", "F to K", "K to F"]
* for i, option in enumerate(options):
* tk.Radiobutton(
* root, text=option, variable=var, value=option, bg="lightblue", font=("Arial", 10)
* ).grid(row=i+1, column=0, columnspan=2, sticky="w", padx=10)
* # Convert Button
* convert\_button = tk.Button(root, text="Convert", command=convert\_temperature, bg="blue", fg="white", font=("Arial", 12))
* convert\_button.grid(row=len(options)+1, column=0, columnspan=2, pady=10)
* # Result Section
* result\_label = tk.Label(root, text="", font=("Arial", 14), bg="lightblue")
* result\_label.grid(row=len(options)+2, column=0, columnspan=2, pady=10)
* # Run the application
* root.mainloop()

# 7. GUI Components and Design

The graphical interface consists of the following components:

* **Labels**: Displaying instructions and results.
* **Entry Widget**: Accepting user input.
* **Radio Buttons**: Allowing users to choose the conversion type.
* **Button**: Executing the conversion upon user command.
* **Result Label**: Displaying the converted temperature in a color-coded format.

The GUI consists of several key elements that make it easy for users to interact with the application:

* **Temperature Input Field**: A text box (entry widget) where users can enter the temperature value they wish to convert.
* **Radio Buttons**: A set of radio buttons allows users to select the type of conversion they wish to perform. The available options are:
  + Celsius to Fahrenheit
  + Fahrenheit to Celsius
  + Celsius to Kelvin
  + Kelvin to Celsius
  + Fahrenheit to Kelvin
  + Kelvin to Fahrenheit
* **Convert Button**: A button that triggers the conversion when clicked.
* **Result Label**: A label where the result of the conversion is displayed. The text color is dynamically changed based on the result's value (red, orange, or green).

# 8. Functionality Overview

The application provides:

* Easy numerical input for temperature values.
* Multiple unit conversion options.
* Interactive UI with error handling and color-coded results.
* Clear visual representation of the converted values.
* **User Input Field (Entry Widget)**:
* The application provides an input field, allowing users to easily enter numerical temperature values.
* This field is specifically designed to accept numerical input, including integers and floating-point values. For example, a user can input 36.5 to convert a temperature with a decimal value or simply input 25 for a whole number temperature.
* **User Interaction:**
* The user simply types the temperature value into the input box, ensuring that the application is ready to process that value. This interaction is intuitive and ensures ease of use, allowing even users with minimal technical knowledge to navigate the application.
* The Entry widget is configured to only accept valid numeric inputs. If the user enters any non-numeric characters, the application triggers error handling, guiding the user to enter valid data.
* **Benefits:**
* **Accuracy**: The ability to input precise temperature values, including decimal places, ensures the application handles both real-world data and scientific data efficiently.
* **Simplicity**: The input mechanism is straightforward, requiring only basic knowledge to use.
* **Flexibility**: The input accepts any valid numeric temperature value, allowing for conversions across various temperature scales.

# 9. Conversion Logic

The program utilizes standard temperature conversion formulas:

* Celsius to Fahrenheit:

 (°F)=(Celsius (°C)×59​)+32

* Fahrenheit to Celsius:

(°C)=(Fahrenheit (°F)−32)×95​

* Celsius to Kelvin:

(K)=Celsius (°C)+273.15

* Kelvin to Celsius:

(°C)=Kelvin (K)−273.15

* Fahrenheit to Kelvin:

(K)=((Fahrenheit (°F)−32)×95​)+273.15

* Kelvin to Fahrenheit:

(°F)=((Kelvin (K)−273.15)×59​)+32

# 10. Input Validation

The primary purposes of input validation in the application are:

* **Accuracy**: To ensure that only valid data (i.e., numeric values) are used in the conversion process. Invalid data would result in incorrect or failed conversions.
* **Error Prevention**: To catch errors before they occur, preventing the application from processing erroneous input that could cause crashes or unexpected behaviors.
* **User Guidance**: To help users understand the format of expected input, ensuring they know how to provide data that the system can work with (e.g., numerical temperatures, selecting a conversion type).

**Validating the Temperature Input**

* Ensuring that the temperature entered by the user is a valid numerical value.

**Validating the Conversion Type**

* Ensuring that the user selects a valid conversion type (Celsius to Fahrenheit, Fahrenheit to Celsius, etc.).
* **Types of Invalid Inputs:**
* **Textual Input**: Letters, words, or phrases like abc, hello, or temperature.
* **Special Characters**: Symbols or punctuation marks such as @, #, !, or $.
* **Empty Input**: When the user leaves the input field empty or blank.
* **Examples of Invalid Input:**
* abc → Error message: "Please enter a valid number."
* !@# → Error message: "Please enter a valid number."
* (empty input) → Error message: "Please enter a valid number."

# 11. Color-Coding Temperature Ranges

* **Red: High Temperatures**
* **Threshold**: Temperatures **greater than 85°C / 185°F**.
* **Associated Color**: **Red**
* **Meaning**: Red is typically associated with hot, extreme, or dangerous temperatures. In this context, temperatures above 85°C (185°F) are considered high and likely indicative of extreme heat, such as hot weather or boiling liquids.
* **Examples**:
* **Celsius**: 90°C, 100°C (boiling point of water).
* **Fahrenheit**: 194°F, 212°F (boiling point of water).
* **Kelvin**: 363.15K, 373.15K (boiling point of water in Kelvin).
* **Reasoning**: Temperatures in this range are typically uncomfortable, dangerous, or extreme, requiring extra caution (e.g., in industrial, scientific, or weather applications).
* **Orange: Moderate Temperatures**
* **Threshold**: Temperatures **between 60°C / 140°F and 85°C / 185°F**.
* **Associated Color**: **Orange**
* **Meaning**: Orange is used for moderate temperatures, which are warm but not extreme. This range might represent typical room temperature, warm weather, or the comfortable heat experienced in various environments.
* **Examples**:
* **Celsius**: 25°C (comfortable room temperature), 60°C (a very hot day).
* **Fahrenheit**: 77°F (comfortable room temperature), 140°F (very hot day).
* **Kelvin**: 298.15K (room temperature), 333.15K (a very hot day).
* **Reasoning**: These are temperatures you might encounter in everyday situations, such as warm weather or hot food. While they are not extreme, they are noticeable and should prompt attention (e.g., be cautious of hot surfaces or weather conditions).
* **Green: Low Temperatures**
* **Threshold**: Temperatures **below 60°C / 140°F**.
* **Associated Color**: **Green**
* **Meaning**: Green represents cooler temperatures that are often considered comfortable, pleasant, or cold. This range is generally associated with moderate to cold weather, such as cool days or freezing temperatures.
* **Examples**:
* **Celsius**: 15°C (cool day), 0°C (freezing point), -10°C (cold winter day).
* **Fahrenheit**: 59°F (cool day), 32°F (freezing point), 14°F (cold winter day).
* **Kelvin**: 273.15K (freezing point), 263.15K (cold winter day).
* **Reasoning**: Green is used to represent lower temperatures, which are often pleasant (cool spring days) or extremely cold (winter temperatures). This range typically represents temperatures that are still manageable or comfortable for most people.

# 12. Error Handling

* There are several types of errors that can occur in this application, and each one is handled in a way that prevents the application from failing while also informing the user of what went wrong. Below are the key types of errors the application is designed to handle:
* **Invalid Input Format**
  + This occurs when the user enters a value that is not a valid number, such as text or symbols. Since the temperature field expects a numeric input, any non-numeric value will cause a ValueError.
* **Missing Conversion Selection**
  + If the user does not select any conversion option (radio button) and tries to convert the temperature, the application should notify them that they need to select a conversion type.
* **Unexpected Errors**
  + Any unforeseen errors that might occur, such as issues with the system’s performance or bugs in the application, are caught using general error handling mechanisms.

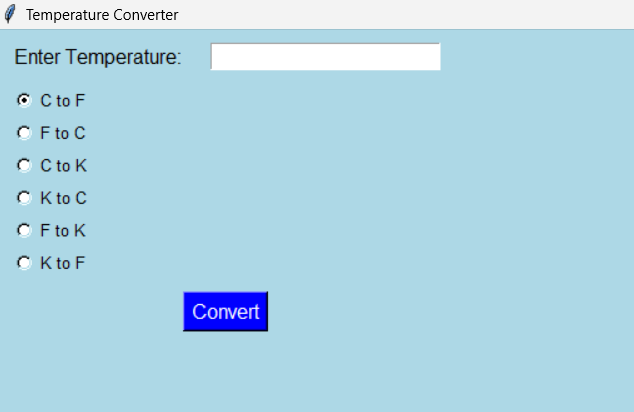
## 13. Troubleshooting

|  |  |
| --- | --- |
| **Issue** | **Possible Solution** |
| Application does not start | Ensure Python 3 is installed and Tkinter is available. |
| Incorrect conversion results | Verify that the correct conversion type is selected. |
| UI does not respond | Close and restart the application. |
| Error message displayed | Check that the input is a valid number. |

|  |
| --- |
|  |

## .

## Output Format



### **Conclusion**

This temperature converter project demonstrates how Python and Tkinter can be used to create an interactive and user-friendly application. It allows users to seamlessly convert temperatures between Celsius, Fahrenheit, and Kelvin, providing real-time results with intuitive color-coded feedback based on temperature ranges. The program ensures error handling for invalid inputs, enhancing user experience and reliability.

This project can be further expanded by adding features like additional unit conversions, a history log for previous conversions, or even integration with an API for real-time weather temperature comparisons. Overall, this serves as a great foundational project for beginners learning GUI development in Python.