

**Name: Hamza Ahmad**

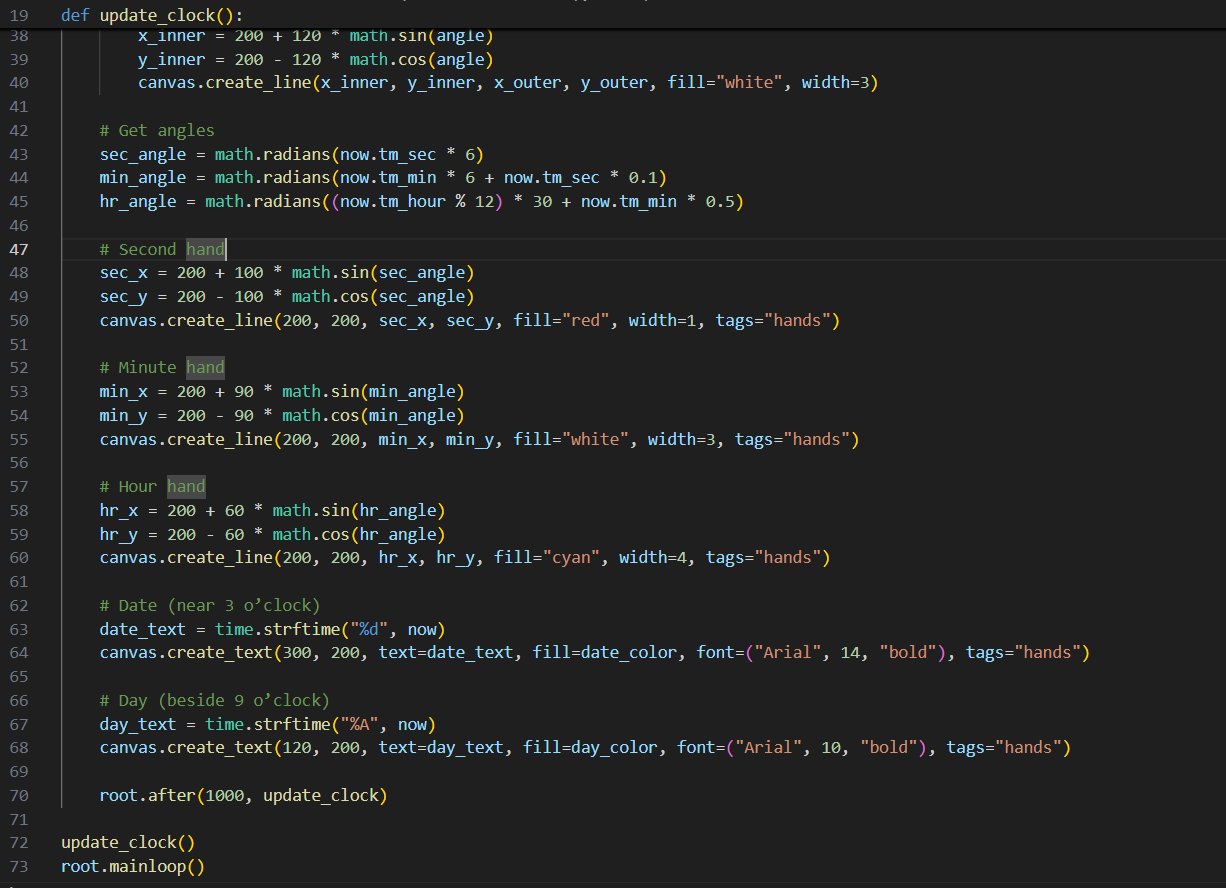
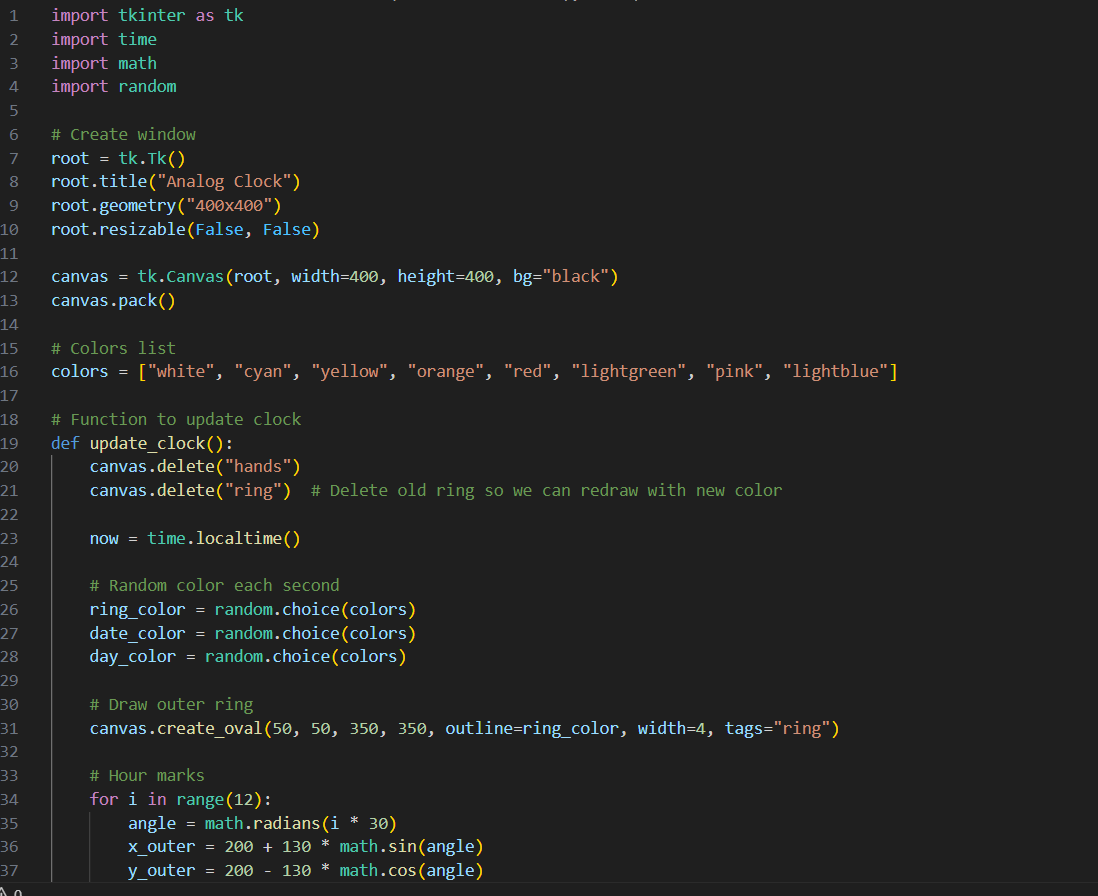
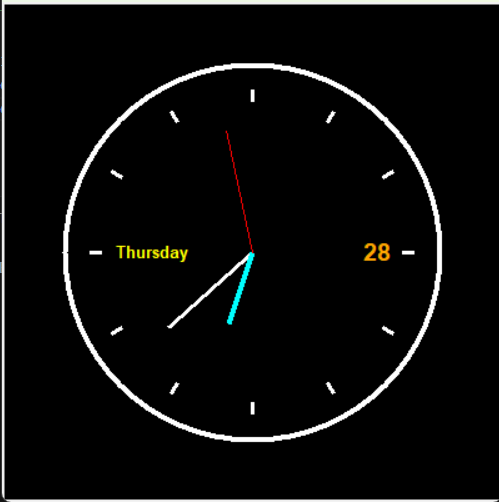
**Intern ID: TN/IN02/PY/008**

**Email ID: hamzaahmad3632@gmail.com**

**Task no: Final Project.**

**Internship Domain: Python**

**Instructor Name: Mr. Hassan Ali**

****

# ****Project Report****

## **Title:** Analog Clock using Python Tkinter

## **1. Introduction**

This project is a **graphical analog clock** developed using **Python’s Tkinter library**. The application displays the **current time**, **day**, and **date** in a visually appealing manner. The clock features a **dynamic outer ring** and uses **random colors** to enhance the interface, providing an attractive and interactive experience.

## **2. Objectives**

* To design a **GUI-based analog clock** using Python.
* To demonstrate **Tkinter canvas drawing** and **real-time updates**.
* To learn about **event-driven programming** in Python.
* To implement **math-based angle calculations** for clock hands.

## **3. Uses of the Project**

### **A. Educational Uses**

* **Learning Tkinter:** Helps beginners understand how to create GUIs using Tkinter.
* **Understanding Trigonometry:** Uses sine and cosine functions to calculate angles for clock hands.
* **Real-time Programming:** Demonstrates how to update UI elements dynamically using the after() method.

### **B. Practical Uses**

* **Desktop Clock Application:** Can be used as a **real-time analog clock** on personal systems.
* **Customization:** Users can modify colors, designs, and themes as per their preferences.
* **Integration with Other Systems:** The project can be integrated into other applications such as **dashboards** or **productivity tools**.

## **4. Features Implemented**

* Real-time **hour**, **minute**, and **second** hands.
* **Dynamic colors** for the clock’s outer ring, date, and day.
* Displays **current date** and **day name**.
* Smooth updates every **1 second** using Tkinter’s after() method.
* Attractive UI with a **black background** and **colorful visuals**.

## **5. Challenges Faced During Development**

### **A. Logical Challenges**

1. **Calculating Hand Positions**
   * Used trigonometric formulas to convert seconds, minutes, and hours into angles.
   * Required correct conversion from degrees to radians:

Angle=Value×π180\text{Angle} = \text{Value} \times \frac{\pi}{180}Angle=Value×180π​

* + Example: For seconds → sec\_angle = math.radians(now.tm\_sec \* 6)

1. **Updating the Clock in Real Time**
   * Needed to refresh the canvas every **1000 ms** without freezing the GUI.
   * Solved using root.after(1000, update\_clock) instead of using time.sleep(), which would block the main thread.
2. **Positioning Text (Date & Day)**
   * Adjusting the exact positions near **3 o’clock** and **9 o’clock** for better UI alignment.

### **B. Technical Challenges**

1. **Color Flickering**
   * Since random colors are updated every second, the rapid changes sometimes cause visual distraction.
   * Solution: Limited color choices to a predefined list for a balanced design.
2. **Canvas Redrawing**
   * Old clock hands and rings overlapped with the new ones.
   * Solved by **deleting previous shapes** using:
   * canvas.delete("hands")
   * canvas.delete("ring")
3. **Window Size & Responsiveness**
   * Tkinter is **not responsive by default**; had to fix dimensions to maintain proper design.

## **6. Future Enhancements**

* Add **digital clock support** alongside the analog design.
* Allow **user customization** of colors and themes.
* Make the UI **responsive** for different screen sizes.
* Add an **alarm feature** with sound notifications.

## **7. Conclusion**

This project successfully demonstrates how to create a **real-time analog clock** using **Python Tkinter**. It covers essential concepts such as GUI design, event-driven programming, trigonometric calculations, and dynamic canvas updates. Despite some challenges, the final application is **functional, visually appealing, and educational**.