Hand Gesture Calculator with UI in Python - Step-by-Step Guide

# Tools & Libraries Required

- Python  
- OpenCV – for hand tracking  
- MediaPipe – for hand gesture detection  
- Tkinter or PyQt – for UI  
- NumPy – for basic array operations

# Step 1: Install Required Libraries

Install dependencies using pip:  
  
```  
pip install opencv-python mediapipe numpy  
pip install PyQt5 # If using PyQt5 for UI  
```

# Step 2: Set Up Hand Detection with MediaPipe

```python  
import cv2  
import mediapipe as mp  
  
mp\_hands = mp.solutions.hands  
hands = mp\_hands.Hands(max\_num\_hands=1)  
mp\_draw = mp.solutions.drawing\_utils  
  
cap = cv2.VideoCapture(0)  
  
while True:  
 success, img = cap.read()  
 img\_rgb = cv2.cvtColor(img, cv2.COLOR\_BGR2RGB)  
 results = hands.process(img\_rgb)  
  
 if results.multi\_hand\_landmarks:  
 for hand\_lms in results.multi\_hand\_landmarks:  
 mp\_draw.draw\_landmarks(img, hand\_lms, mp\_hands.HAND\_CONNECTIONS)  
  
 cv2.imshow("Hand Tracker", img)  
 if cv2.waitKey(1) & 0xFF == ord('q'):  
 break  
  
cap.release()  
cv2.destroyAllWindows()  
```

# Step 3: Detect Finger Count for Gestures

```python  
def count\_fingers(hand\_landmarks):  
 tips = [4, 8, 12, 16, 20]  
 fingers = []  
  
 # Thumb  
 if hand\_landmarks.landmark[tips[0]].x < hand\_landmarks.landmark[tips[0] - 1].x:  
 fingers.append(1)  
 else:  
 fingers.append(0)  
  
 # Other fingers  
 for i in range(1, 5):  
 if hand\_landmarks.landmark[tips[i]].y < hand\_landmarks.landmark[tips[i] - 2].y:  
 fingers.append(1)  
 else:  
 fingers.append(0)  
  
 return sum(fingers)  
```

# Step 4: Map Gestures to Digits or Operators

| Fingers | Action |  
|---------|---------|  
| 1 | '1' |  
| 2 | '2' |  
| 3 | '3' |  
| 4 | '+' |  
| 5 | '=' |

# Step 5: Create the Calculator Logic

```python  
expression = ""  
  
if gesture == 1:  
 expression += "1"  
elif gesture == 4:  
 expression += "+"  
elif gesture == 5:  
 try:  
 result = str(eval(expression))  
 expression = result  
 except:  
 expression = "Error"  
```

# Step 6: Build the GUI

```python  
import tkinter as tk  
  
def update\_display(text):  
 display\_var.set(text)  
  
root = tk.Tk()  
root.title("Gesture Calculator")  
  
display\_var = tk.StringVar()  
display = tk.Entry(root, textvariable=display\_var, font=("Arial", 24), justify="right")  
display.pack(fill='both', expand=True)  
  
root.after(100, lambda: update\_display(expression)) # refresh display  
  
root.mainloop()  
```

# Step 7: Integrate Webcam & GUI

```python  
import threading  
  
def gesture\_loop():  
 global expression  
 # Your OpenCV + gesture recognition code  
 # call update\_display(expression) inside the loop  
  
thread = threading.Thread(target=gesture\_loop)  
thread.daemon = True  
thread.start()  
root.mainloop()  
```

# Final Notes

- You can customize gestures using machine learning for more accurate detection.  
- You can add a clear gesture (e.g., fist) to reset the expression.  
- Test thoroughly with different lighting and background.