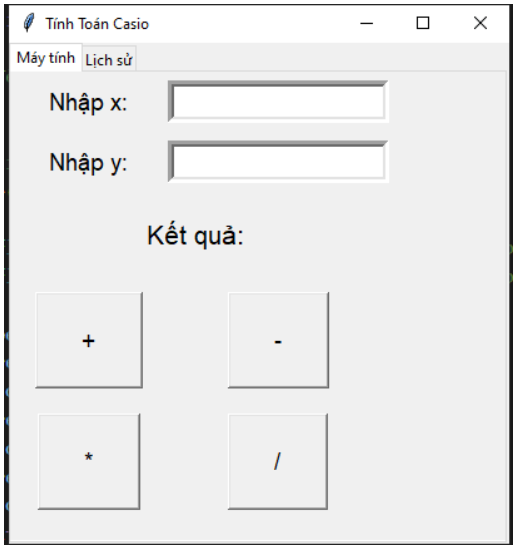
BÀI BÁO CÁO

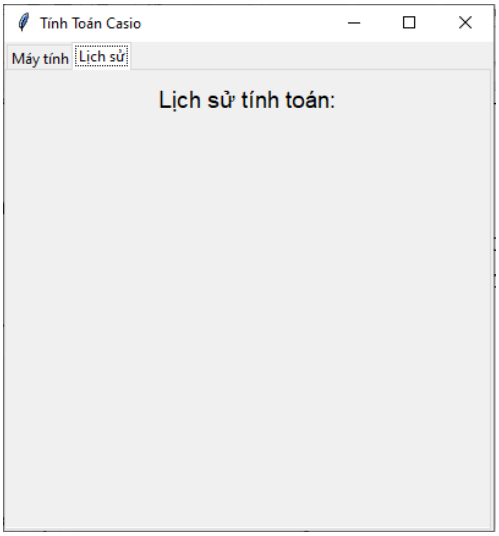
TÊN BÀI: TẠO GIAO DIỆN GUI

MSSV: 2274802010083 **HỌ TÊN: ĐOÀN TRUNG CƯỜNG**

Nội dung:

1. Giao diện:
   giao diện của em muốn tạo là giao diện tính toán cơ bản như cộng, trừ, nhân, chia. Mục đích tạo ra là muốn mọi người có thể tính toán một cách tiện lợi hơn nhiều.





2. Chức năng:
   Cửa sổ chính:
   - root = tk.Tk(): Tạo cửa sổ chính cho ứng dụng.
   - root.title("Tính Toán Casio"): Đặt tiêu đề của cửa sổ là "Tính Toán Casio".
   - root.geometry("400x400"): Thiết lập kích thước của cửa sổ chính là 400x400 pixel.

   Tabs (Tab Control)
   tab_control = ttk.Notebook(root): Tạo một Notebook (một kiểu Widget cho các tab) để chứa các tab trong cửa sổ.

   Tab 1 (Máy tính):
   - tab1 = ttk.Frame(tab_control): Tạo một tab mới cho tính toán.
   - tab_control.add(tab1, text='Máy tính'): Đặt tên cho tab là "Máy tính".

   - Tab này chứa các ô nhập liệu, nút bấm cho phép tính toán và khung hiển thị kết quả.

   Tab 2 (Lịch sử):
   - tab2 = ttk.Frame(tab_control): Tạo một tab thứ hai để hiển thị lịch sử tính toán.
   - tab_control.add(tab2, text='Lịch sử'): Đặt tên cho tab là "Lịch sử".

Lịch sử: Hiển thị các phép tính và kết quả đã thực hiện trước đó.

Ô nhập liệu

- entry_x và entry_y: Đây là hai ô nhập liệu cho các biến x và y, nơi người dùng nhập các giá trị cần tính toán.
- entry_x = tk.Entry(tab1): Tạo ô nhập cho giá trị x.
- entry_y = tk.Entry(tab1): Tạo ô nhập cho giá trị y.

Nút tính toán:
Các nút tính toán tương ứng với các phép tính:

- **+: Phép cộng.**
- **-: Phép trừ.**
- ***: Phép nhân.**
- **/: Phép chia**.

Mỗi nút được liên kết với hàm calculate(operator) để thực hiện phép tính khi người dùng bấm nút.

Hàm calculate(operator):
Hàm này được gọi khi người dùng bấm một trong các nút phép tính.
Chức năng:

- Lấy giá trị từ hai ô nhập x và y.
- Thực hiện phép tính dựa trên nút mà người dùng đã bấm.
- Hiển thị kết quả trong nhãn result_label.
- Nếu nhập sai (ví dụ nhập chữ vào ô số), sẽ hiện thông báo lỗi.
-

Lưu lại kết quả trong danh sách history và cập nhật lịch sử trên Tab 2.

Hàm update_history():

- Hàm này xóa tất cả các widget cũ trên Tab 2 và hiển thị lại lịch sử tính toán từ danh sách history.
- Sau mỗi phép tính, danh sách lịch sử sẽ được cập nhật và hiển thị trong Tab 2.

Nhãn hiển thị kết quả: result_label: Đây là một nhãn (Label) để hiển thị kết quả tính toán sau khi người dùng bấm nút phép tính.

Vòng lặp chính: root.mainloop(): Bắt đầu vòng lặp sự kiện chính của Tkinter. Vòng lặp này sẽ liên tục lắng nghe các hành động từ người dùng (như bấm nút hoặc nhập liệu) và phản hồi lại.

3. Mã chương trình:
import tkinter as tk: Import thư viện tkinter để tạo giao diện người dùng trong Python.

from tkinter import ttk, messagebox: Import các thành phần ttk để tạo các widget kiểu tab và messagebox để hiển thị thông báo lỗi.

history = []: Khởi tạo danh sách trống để lưu lịch sử các phép tính đã thực hiện.

def calculate(operator): Định nghĩa hàm calculate để thực hiện phép tính dựa trên phép toán được chọn.

x = float(entry_x.get())
y = float(entry_y.get()) :Lấy giá trị từ ô nhập liệu x,y chuyển thành số thực (float).

if operator == '+': Nếu phép tính là cộng, thực hiện phép cộng.
elif operator == '-': Nếu phép tính là trừ, thực hiện phép trừ.
elif operator == '*': Nếu phép tính là nhân, thực hiện phép nhân.
elif operator == '/': Nếu phép tính là chia, thực hiện phép chia.

result = x + y: Tính tổng của x và y.
result = x – y: Tính hiệu của x và y.
result = x * y: Tính tích của x và y.
result = x / y: Tính thương của x và y.

if y != 0: Kiểm tra xem y có khác 0 không (để tránh chia cho 0).

else: messagebox.showerror("Lỗi", "Không thể chia cho 0")
Chú thích: Hiển thị thông báo lỗi nếu cố gắng chia cho 0.

result = round(result, 2)
Chú thích: Làm tròn kết quả tính toán tới 2 chữ số thập phân.

result_label.config(text=f"Kết quả: {result}")
Chú thích: Hiển thị kết quả phép tính trong nhãn (Label) trên giao diện.

history.append(f"{x} {operator} {y} = {result}")
Chú thích: Lưu phép tính vừa thực hiện vào danh sách lịch sử (history).

update_history()
Chú thích: Gọi hàm update_history để cập nhật lịch sử các phép tính trên tab lịch sử.

except ValueError:
Chú thích: Bắt lỗi nếu giá trị nhập không hợp lệ (ví dụ: không phải là số).

messagebox.showerror("Lỗi", "Vui lòng nhập số hợp lệ vào x và y.")
Chú thích: Hiển thị thông báo lỗi nếu giá trị nhập không hợp lệ.

def update_history():
Chú thích: Định nghĩa hàm để cập nhật lịch sử tính toán và hiển thị trên tab 2.

for widget in tab2.winfo_children():
Chú thích: Lặp qua tất cả các widget trên tab 2 và xóa widget cũ để làm mới nội dung.

widget.destroy()
Chú thích: Xóa widget cũ trong tab 2 để chuẩn bị hiển thị lịch sử mới.

tk.Label(tab2, text="Lịch sử tính toán:", font=('Arial', 14)).pack(pady=10)
Chú thích: Tạo và hiển thị nhãn "Lịch sử tính toán" trong tab 2.

for record in history:
Chú thích: Lặp qua từng phép tính trong danh sách history.

tk.Label(tab2, text=record, font=('Arial', 12)).pack()
Chú thích: Tạo nhãn và hiển thị từng phép tính đã thực hiện trên tab 2.

root = tk.Tk()
Chú thích: Tạo cửa sổ chính của ứng dụng.

root.title("Tính Toán Casio")
Chú thích: Đặt tiêu đề của cửa sổ là "Tính Toán Casio".

root.geometry("400x400")
Chú thích: Thiết lập kích thước của cửa sổ chính là 400x400 pixel.

tab_control = ttk.Notebook(root)
Chú thích: Tạo Notebook (giao diện với các tab) cho cửa sổ chính.

tab1 = ttk.Frame(tab_control)
Chú thích: Tạo khung (Frame) cho tab 1, nơi sẽ chứa các thành phần của máy tính.

tab_control.add(tab1, text='Máy tính')
Chú thích: Thêm tab "Máy tính" vào giao diện với tiêu đề "Máy tính".

tab2 = ttk.Frame(tab_control)
Chú thích: Tạo khung (Frame) cho tab 2, nơi sẽ chứa lịch sử tính toán.

tab_control.add(tab2, text='Lịch sử')
Chú thích: Thêm tab "Lịch sử" vào giao diện với tiêu đề "Lịch sử".

tab_control.pack(expand=1, fill="both")
Chú thích: Đặt tab vào cửa sổ chính và cho phép nó mở rộng để lấp đầy toàn bộ cửa sổ.

tk.Label(tab1, text="Nhập x:", font=('Arial', 14)).grid(row=0, column=0, padx=10, pady=10)
Chú thích: Tạo nhãn "Nhập x" và đặt nó vào tab 1, nơi người dùng sẽ nhập giá trị x.

entry_x = tk.Entry(tab1, width=15, borderwidth=5, font=('Arial', 14))
Chú thích: Tạo ô nhập liệu cho biến x trong tab 1.

entry_x.grid(row=0, column=1)
Chú thích: Đặt ô nhập liệu cho biến x vào lưới giao diện (ở hàng 0, cột 1).

tk.Label(tab1, text="Nhập y:", font=('Arial', 14)).grid(row=1, column=0, padx=10, pady=10)
Chú thích: Tạo nhãn "Nhập y" và đặt nó vào tab 1, nơi người dùng sẽ nhập giá trị y.

entry_y = tk.Entry(tab1, width=15, borderwidth=5, font=('Arial', 14))
Chú thích: Tạo ô nhập liệu cho biến y trong tab 1.

entry_y.grid(row=1, column=1)
Chú thích: Đặt ô nhập liệu cho biến y vào lưới giao diện (ở hàng 1, cột 1).

result_label = tk.Label(tab1, text="Kết quả: ", font=('Arial', 16))
Chú thích: Tạo nhãn để hiển thị kết quả phép tính trong tab 1.

result_label.grid(row=2, column=0, columnspan=2, pady=20)
Chú thích: Đặt nhãn hiển thị kết quả vào lưới giao diện, trải dài qua hai cột.

operators = ['+', '-', '*', '/']
Chú thích: Tạo danh sách chứa các phép tính (+, -, *, /).

for i, operator in enumerate(operators):
Chú thích: Duyệt qua từng phép tính trong danh sách operators.

button = tk.Button(tab1, text=operator, padx=30, pady=20, font=('Arial', 14), command=lambda op=operator: calculate(op))
Chú thích: Tạo nút cho từng phép tính và liên kết nó với hàm calculate.

button.grid(row=3 + i // 2, column=i % 2, padx=20, pady=10)
Chú thích: Đặt nút vào lưới giao diện, sắp xếp nó trong hai hàng với khoảng cách giữa các nút.

tk.Label(tab2, text="Lịch sử tính toán:", font=('Arial', 14)).pack(pady=10)
Chú thích: Tạo nhãn "Lịch sử tính toán" trong tab 2

4. Github:
   Link GitHub: https://github.com/Cuong2k42004/BaiCuoiKy_PythonNangCao/tree/main