Antivirasfrontend.py

import tkinter as tk

from tkinter import filedialog

from tkinter import messagebox

from MalwareDetector import MalwareDetector

from lists import Lists

from prosess import Process

import result

class AntivirusFrontend:

    def \_\_init\_\_(self):

        self.root = tk.Tk()

        self.root.title("Malware Detection")

        self.root.geometry("1300x700")

        self.root.configure(bg="#F0F0F0")

        # Header Label

        self.header\_label = tk.Label(self.root, text="Malware Detection Applycation", font=("Arial", 40, "bold"), fg="#fa0505", bg="#F0F0F0")

        self.header\_label.place(relx=0.5, rely=0.1, anchor="center")

        # Left Label

        self.left\_label = tk.Label(self.root, text="Select if you need to search location:", font=("Arial", 18),fg= "#2196F3", bg="#F0F0F0")

        self.left\_label.place(relx=0.05, rely=0.2)

        # Enter Location Button

        self.enter\_location\_button = tk.Button(self.root, text="Choose Location", font=("Arial", 14),   bg="#2196F3", fg="#FFFFFF", command=self.search\_location)

        self.enter\_location\_button.place(relx=0.37, rely=0.2)

        # Search Result Frame

        self.result\_frame = tk.Frame(self.root, width=1200, height=400, bg="#FFFFFF")

        self.result\_frame.place(relx=0.5, rely=0.6, anchor="center")

# Delete Malware Button

        self.delete\_malware\_button = tk.Button(self.root, text="Delete Malware", font=("Arial", 12), fg="#ffffff", bg="#ff5252", activebackground="#ff5252", activeforeground="#ffffff", padx=10, pady=5, borderwidth=0, command=self.delete\_malware)

        self.delete\_malware\_button.place(relx=0.4, rely=0.9, anchor="center")

# More Info Malware Button

        self.more\_info\_button = tk.Button(self.root, text="More Info", font=("Arial", 12), fg="#ffffff", bg="#2196F3", activebackground="#388E3C", activeforeground="#ffffff", padx=10, pady=5, borderwidth=0, command=self.more\_info)

        self.more\_info\_button.place(relx=0.5, rely=0.9, anchor="center")

# Clear All Data Button

        self.clear\_button = tk.Button(self.root, text="Clear All Data", font=("Arial", 12), fg="#ffffff", bg="#2196F3", activebackground="#1976D2", activeforeground="#ffffff", padx=10, pady=5, borderwidth=0, command=self.clear\_all\_data)

        self.clear\_button.place(relx=0.6, rely=0.9, anchor="center")

        # Attach the frame to the bottom of the main frame

        self.result\_frame.pack\_propagate(False)

        # Call function to show search results

        # Creating instances of backend classes

        self.malware\_detector = MalwareDetector()

        self.lists = Lists()

        self.processor = Process()

    def clear\_tree(self, node):

        if node is None:

            return

        node.children = {}

        for child in node.children.values():

            self.clear\_tree(child)

    def clear\_all\_data(self):

        # Clear stored data

        self.processor.file\_path = None

        self.processor.searchingLocation = []

        self.processor.searchResult = []

        self.processor.filePathtoDelete = []

        # Clear TreeNode data structure

        self.clear\_tree(self.malware\_detector.root)

        # Show success notification

        messagebox.showinfo("Success", "Successfully cleared data tree diagram and list.")

    def search\_location(self):

        # Function to handle opening a directory selection dialog

        search\_path = filedialog.askdirectory()

        if search\_path:  # If a location is selected

            self.process\_file(search\_path)

    def process\_file(self, search\_path):

        # Function to process the selected file/location

        self.processor.setFilePath(search\_path)

        self.processor.main()  # Call the main method in the Process class

        self.print\_search\_location()

    def print\_search\_location(self):

        # Print search results

        search\_results = self.processor.searchResult

        if search\_results:

            header\_label = tk.Label(self.result\_frame, text="Malware detected", font=("Arial", 15,"bold"), fg="blue", bg="#FFFFFF")

            header\_label.pack(anchor='center')

            for lo in search\_results:

                file\_label = tk.Label(self.result\_frame, text=lo, font=("Arial", 11 ,"bold"), fg="red", bg="#FFFFFF")

                file\_label.pack(anchor='w')

        else:

            header\_label = tk.Label(self.result\_frame, text="No malware detected. You are safe.", font=("Arial", 16,"bold"), fg="green",bg="#F0F0F0")

            header\_label.pack(anchor='center')

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

    def more\_info(self):

        serching\_locations =self.processor.searchingLocation

        search\_results = self.processor.searchResult

        if search\_results :

            result.show\_data(serching\_locations,search\_results)  # Show search results using result.py

        else:

               self.header\_label = tk.Label(self.root, text="Not selected location ", font=("Arial", 15, "bold"), fg="green", bg="#F0F0F0")

               self.header\_label.place(relx=0.5, rely=0.95, anchor="center")

    def delete\_malware(self):

        search\_results = self.processor.filePathtoDelete

        if search\_results:

           num=0

           for path in search\_results:

               self.processor.delete\_file(path)

               num+=1

           if num>0:

               self.header\_label = tk.Label(self.root, text="Malware Deleted successfully, now you are safe", font=("Arial", 15, "bold"), fg="green", bg="#F0F0F0")

               self.header\_label.place(relx=0.5, rely=0.95, anchor="center")

           else:

               self.header\_label = tk.Label(self.root, text="Malware not Deleted , now not you  are safe", font=("Arial", 15, "bold"), fg="red", bg="#F0F0F0")

               self.header\_label.place(relx=0.5, rely=0.95, anchor="center")

        else:

               self.header\_label = tk.Label(self.root, text="Not selected Malware ", font=("Arial", 15, "bold"), fg="green", bg="#F0F0F0")

               self.header\_label.place(relx=0.5, rely=0.95, anchor="center")

    def run(self):

        self.root.mainloop()

if \_\_name\_\_ == "\_\_main\_\_":

    frontend = AntivirusFrontend()

    frontend.run()

full scan.py

import tkinter as tk

from tkinter import filedialog

from tkinter import messagebox

from MalwareDetector import MalwareDetector

from lists import Lists

from prosess import Process

import result

class FullScan:

     def \_\_init\_\_(self):

        self.root = tk.Tk()

        self.root.title("Malware Detection")

        self.root.geometry("1300x700")

        self.root.configure(bg="#F0F0F0")

        # Header Label

        self.header\_label = tk.Label(self.root, text="Malware Detection Applycation", font=("Arial", 40, "bold"), fg="#fa0505", bg="#F0F0F0")

        self.header\_label.place(relx=0.5, rely=0.1, anchor="center")

        # Left Label

        self.left\_label = tk.Label(self.root, text="Select if you need to search full pc:", font=("Arial", 18),fg= "#2196F3", bg="#F0F0F0")

        self.left\_label.place(relx=0.05, rely=0.2)

        # Enter Location Button

        self.enter\_location\_button = tk.Button(self.root, text="Full PC Scan", font=("Arial", 14),   bg="#2196F3", fg="#FFFFFF")

        self.enter\_location\_button.place(relx=0.37, rely=0.2)

        # Search Result Frame

        self.result\_frame = tk.Frame(self.root, width=1200, height=400, bg="#FFFFFF")

        self.result\_frame.place(relx=0.5, rely=0.6, anchor="center")

# Delete Malware Button

        self.delete\_malware\_button = tk.Button(self.root, text="Delete Malware", font=("Arial", 12), fg="#ffffff", bg="#ff5252", activebackground="#ff5252", activeforeground="#ffffff", padx=10, pady=5, borderwidth=0)

        self.delete\_malware\_button.place(relx=0.4, rely=0.9, anchor="center")

# More Info Malware Button

        self.more\_info\_button = tk.Button(self.root, text="More Info", font=("Arial", 12), fg="#ffffff", bg="#2196F3", activebackground="#388E3C", activeforeground="#ffffff", padx=10, pady=5, borderwidth=0)

        self.more\_info\_button.place(relx=0.5, rely=0.9, anchor="center")

# Clear All Data Button

        self.clear\_button = tk.Button(self.root, text="Clear All Data", font=("Arial", 12), fg="#ffffff", bg="#2196F3", activebackground="#1976D2", activeforeground="#ffffff", padx=10, pady=5, borderwidth=0)

        self.clear\_button.place(relx=0.6, rely=0.9, anchor="center")

        # Attach the frame to the bottom of the main frame

        self.result\_frame.pack\_propagate(False)

if \_\_name\_\_ == "\_\_main\_\_":

    frontend = FullScan()

    frontend.run()

list.py

class Lists:

    def \_\_init\_\_(self):

        self.searchingLocation = []

        self.searchResult = []

        self.filePathtoDelete = []

        self.file\_path = None

malware dectector.py

from TreeNode import TreeNode

class MalwareDetector:

    def \_\_init\_\_(self):

# --creates a root node for the tree, initialized with None value.

        self.root = TreeNode(None)

#---adds a malware signature to the tree

# <while>--signature perameter is // one line  get books signatures

    def add\_signature(self, signature):

        node = self.root

        for char in signature:

            found\_child = None

            for child in node.children:

                if child.value == char:

                    found\_child = child

                    break

            if found\_child is None:

                new\_node = TreeNode(char)

                node.children.append(new\_node)

                node = new\_node

            else:

                node = found\_child

        node.children.append(TreeNode('\*'))  # Mark the end of a signature

    def is\_infected(self, content):

        node = self.root

        for char in content:

            found\_child = None

            for child in node.children:

                if child.value == char:

                    found\_child = child

                    break

            if found\_child is None:

                node = self.root  # Start over if not a part of a signature

            else:

                node = found\_child

                for child in node.children:

                    if child.value == '\*':

                        return True  # Found a malware signature

        return False

menu.py

import tkinter as tk

from PIL import Image, ImageTk

from antivirasFrontend import AntivirusFrontend

from fullScann import FullScan

from settings import Settings

from updates import Updates

def full\_scan():

    fullscan=FullScan()

    fullscan.run()

def location\_scan():

    frontend = AntivirusFrontend()

    frontend.run()

def updates():

    up=Updates()

    up.run()

def settings():

    st=Settings()

    st.run()

# Create Tkinter window

root = tk.Tk()

root.title("Malware Detection Menu")

# Load the image

image = Image.open("abc.jpg")  # Replace "abc.jpg" with your image file

photo = ImageTk.PhotoImage(image)

# Get the width and height of the image

width, height = image.size

# Create a frame with the same size as the image

frame = tk.Frame(root, width=width, height=height)

# Add a label to the frame to display the image

label = tk.Label(frame, image=photo)

label.pack(fill=tk.BOTH, expand=tk.YES)

# Place the frame

frame.pack(fill=tk.BOTH, expand=tk.YES)

# Create a label for "Malware"

malware\_label = tk.Label(text="Malware Dtaction", font=("Arial", 50, "bold"), fg="red", bg='black')

malware\_label.place(relx=0.5, rely=0.3, anchor=tk.CENTER)

# Create buttons

def on\_enter(e):

    e.widget.config(bg="#32a8a4", fg="black", font=("Arial", 21, "bold"),  borderwidth=5,  highlightbackground="#32a8a4")

def on\_leave(e):

    e.widget.config(bg="black", fg="#32a8a4", font=("Arial", 20), borderwidth=5,  highlightbackground="#32a8a4")

full\_scan\_button = tk.Button(frame, text="Full Scan", bg="black", fg="#32a8a4", font=("Arial", 20), bd=0, command=full\_scan, highlightthickness=0, borderwidth=0, highlightbackground="black")

full\_scan\_button.place(relx=0.1, rely=0.45)

full\_scan\_button.bind("<Enter>", on\_enter)

full\_scan\_button.bind("<Leave>", on\_leave)

location\_scan\_button = tk.Button(frame, text="Location Scan", bg="black", fg="#32a8a4", font=("Arial", 20), bd=0, command=location\_scan, highlightthickness=0, borderwidth=0, highlightbackground="black")

location\_scan\_button.place(relx=0.1, rely=0.55)

location\_scan\_button.bind("<Enter>", on\_enter)

location\_scan\_button.bind("<Leave>", on\_leave)

updates\_button = tk.Button(frame, text="Updates", bg="black", fg="#32a8a4", font=("Arial", 20), bd=0, command=updates, highlightthickness=0, borderwidth=0, highlightbackground="black")

updates\_button.place(relx=0.85, rely=0.45)

updates\_button.bind("<Enter>", on\_enter)

updates\_button.bind("<Leave>", on\_leave)

settings\_button = tk.Button(frame, text="Settings", bg="black", fg="#32a8a4", font=("Arial", 20), bd=0, command=settings, highlightthickness=0, borderwidth=0, highlightbackground="black")

settings\_button.place(relx=0.85, rely=0.55)

settings\_button.bind("<Enter>", on\_enter)

settings\_button.bind("<Leave>", on\_leave)

# Run the Tkinter event loop

root.mainloop()

prosess.py

import os

import fitz  # PyMuPDF

from pptx import Presentation

from docx import Document

import tkinter as tk

from tkinter import filedialog

from MalwareDetector import MalwareDetector

from lists import Lists

class Process:

    def \_\_init\_\_(self):

        self.file\_path = None  # Initialize file\_path attribute to None  --> to get user select file path

        self.searchingLocation = []  # Initialize an empty list to ----->store search locations

        self.searchResult=[]            #----->store search result

        self.filePathtoDelete = []

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_get user select file path   and    change name ---> file\_path

        # Define the setFilePath method

    def setFilePath(self, path):

        self.file\_path = path

        #print("aa.py Selected path:", self.file\_path)

# \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ file\_paths --> "blackbook.txt", "blackbook2.txt"  data line one by one pass MalwareDetector.add\_signature()

    @staticmethod

    def read\_signatures(file\_paths):

 # Read malware signatures from files and add them to the detector

         malware\_detector = MalwareDetector()

         for file\_path in file\_paths:

            try:

                with open(file\_path, 'r') as f:

                    for line in f:

                        signature = line.strip()

# \_\_\_  line by line get  for "blackbook.txt", "blackbook2.txt"  and pass line by line --->  malware\_detector.add\_signature(signature)

                        malware\_detector.add\_signature(signature)

            except Exception as e:

                print(  f"Error reading signature file {file\_path}: {e}")

         return malware\_detector

#@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

# \_\_\_\_\_\_.extents methord with \*\*os\*\*   +each file paths and   +malweare signatures   to  compaire

    @staticmethod

    def scan\_text(content, malware\_detector):

        #  scan text content for malware

        return malware\_detector.is\_infected(content)

#@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

# \_\_\_\_\_\_.extents methord with \*\*os\*\* each file paths and malweare signatures   to  compaire

#\_\_\_\_\_\_\_ get .pdf text and return Process.\*\*\*\*\*\*scan\_text(text, malware\_detector)\*\*\*\*\*\*

    @staticmethod

    def scan\_pdf(file\_path, malware\_detector):

        # Scan a PDF file for malware

        try:

            doc = fitz.open(file\_path)

            text = ""

            for page in doc:

                text += page.get\_text()

            return Process.scan\_text(text, malware\_detector)

        except Exception as e:

            print(f"Error scanning PDF {file\_path}: {e}")

            return False

# \_\_\_\_\_\_.extents methord with \*\*os\*\* each file paths and malweare signatures   to  compaire

#\_\_\_\_\_\_\_ get .pptx text and return Process.\*\*\*\*\*\*scan\_text(text, malware\_detector)\*\*\*\*\*\*

    @staticmethod

    def scan\_pptx(file\_path, malware\_detector):

        # Scan a PPTX file for malware

        try:

            prs = Presentation(file\_path)

            text = ""

            for slide in prs.slides:

                for shape in slide.shapes:

                    if hasattr(shape, "text"):

                        text += shape.text

            return Process.scan\_text(text, malware\_detector)

        except Exception as e:

            print(f"Error scanning PPTX {file\_path}: {e}")

            return False

# \_\_\_\_\_\_.extents methord with \*\*os\*\* each file paths and malweare signatures   to  compaire

#\_\_\_\_\_\_\_ get .docx text and return Process.\*\*\*\*\*\*scan\_text(text, malware\_detector)\*\*\*\*\*\*

    @staticmethod

    def scan\_docx(file\_path, malware\_detector):

        # Scan a DOCX file for malware

        try:

            doc = Document(file\_path)

            text = ""

            for paragraph in doc.paragraphs:

                text += paragraph.text

            return Process.scan\_text(text, malware\_detector)

        except Exception as e:

            print(f"Error scanning DOCX {file\_path}: {e}")

            return False

# \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_pass the malweare signatures and \*\*os\*\* each file paths

#  try to find OS each directly .extents  and pass the values in .extents methord with \*\*os\*\* each file paths and malweare signatures

    @staticmethod

    def scan\_file(malware\_detector, file\_path):

        # Scan a file for malware based on its extension

        \_, file\_extension = os.path.splitext(file\_path)

        if file\_extension == ".pdf":

            return Process.scan\_pdf(file\_path, malware\_detector)

        elif file\_extension == ".pptx":

            return Process.scan\_pptx(file\_path, malware\_detector)

        elif file\_extension == ".docx":

            return Process.scan\_docx(file\_path, malware\_detector)

        else:

            try:

                with open(file\_path, 'r') as f:

                    content = f.read()

                return Process.scan\_text(content, malware\_detector)

            except Exception as e:

                print(f"Error scanning file {file\_path}: {e}")

                return False

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_main pass need to scan location , tree add done all signatores

    def scan\_directory(self, directory, malware\_detector):

        # Scan a directory for malware

        malware\_files = []

        try:

            print("Scanning files in progress:")

            for root, \_, files in os.walk(directory):

                for idx, file in enumerate(files, start=1):

                    file\_path = os.path.join(root, file)

                    print(f"Scanning file {idx}: {file\_path}")

                    # Append the file\_path to searchingLocation list

                    self.searchingLocation.append(file\_path)

#  -\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* os path one by one  send

                    if Process.scan\_file(malware\_detector, file\_path):

                        # ---\*\*\*\*\*  append malweare file path to ----> malware\_files[]

                        malware\_files.append(file\_path)

        except Exception as e:

            print(f"Error scanning directory {directory}: {e}")

        return malware\_files

# --------- frontend give the parths array []  to deleted

    def delete\_file(self, file\_path):

        # Delete a file

        try:

            os.unlink(file\_path)

            print(f"Deleted: {file\_path}")

        except Exception as e:

            print(f"Error deleting {file\_path}: {e}")

#\_\_\_\_ change name ---> file\_path  after call --> main

    def main(self):

        if self.file\_path:

            signature\_files = ["blackbook.txt", "blackbook2.txt"]

#\_\_\_\_\_ send the signature\_files to the       ---->  read\_signatures(signature\_files)

            malware\_detector = self.read\_signatures(signature\_files)

            print("\nScanning directory:", self.file\_path)

            malware\_files = self.scan\_directory(self.file\_path, malware\_detector)

            if malware\_files:

                print("\nMalware detected:")

                print("{:<70}{:<40}{:<10}".format("Location", "Name", "Size"))

                print("-" \* 120)

                for file\_path in malware\_files:

                    file\_name = os.path.basename(file\_path)

                    file\_size = os.path.getsize(file\_path)

                    print("{:<70}{:<50}{:<20}".format(file\_path, file\_name, f"{file\_size} bytes"))

                    self.searchResult.append(f"{file\_path}      {file\_name}     {file\_size} bytes")

                    self.filePathtoDelete.append(file\_path)

                print("-" \* 120)

            else:

                print("No malware detected. You are safe.")

        else:

            print("No file path selected.")

if \_\_name\_\_ == "\_\_main\_\_":

    process = Process()

    process.main()

result.py

import tkinter as tk

from tkinter import ttk

def show\_data(serching\_locations, cd):

    root = tk.Tk()

    root.title("More Info")

    window\_width = 800

    window\_height = 600

    root.geometry(f"{window\_width}x{window\_height}")

    scrollable\_frame = ttk.Frame(root)

    scrollable\_frame.pack(fill='both', expand=True)

    scrollbar = ttk.Scrollbar(scrollable\_frame, orient='vertical')

    scrollbar.pack(side='right', fill='y')

    canvas = tk.Canvas(scrollable\_frame, yscrollcommand=scrollbar.set)

    canvas.pack(side='left', fill='both', expand=True)

    scrollbar.config(command=canvas.yview)

    inner\_frame = ttk.Frame(canvas)

    canvas.create\_window((0, 0), window=inner\_frame, anchor='nw')

    scanning\_label = ttk.Label(inner\_frame, text="Scanning Locations ...", foreground='green', font=("Arial", 20, "bold"))

    scanning\_label.grid(row=0, column=0, padx=10, pady=5, sticky='ew')

    for idx, item in enumerate(serching\_locations):

        label = ttk.Label(inner\_frame, text=item, foreground='blue')

        label.grid(row=idx+1, column=0, padx=10, pady=5, sticky='w')

    malware\_label = ttk.Label(inner\_frame, text="Malware Detected", foreground='red', font=("Arial", 20, "bold"))

    malware\_label.grid(row=len(serching\_locations)+2, column=0, padx=10, pady=5, sticky='ew')

    for idx, item in enumerate(cd):

        label = ttk.Label(inner\_frame, text=item, foreground='red')

        label.grid(row=idx+len(serching\_locations)+3, column=0, padx=10, pady=5, sticky='w')

    inner\_frame.update\_idletasks()

    canvas.config(scrollregion=canvas.bbox('all'))

    root.mainloop()

setting.py

import tkinter as tk

from tkinter import ttk, filedialog, messagebox

from MalwareDetector import MalwareDetector

from lists import Lists

from prosess import Process

import result

class Settings:

    def \_\_init\_\_(self):

        self.root = tk.Tk()

        self.root.title("Antivirus Settings")

        self.root.geometry("800x600")

        self.root.configure(bg="#F0F0F0")

        # Header Label

        self.header\_label = tk.Label(self.root, text="Antivirus Settings", font=("Arial", 30, "bold"), fg="black", bg="#F0F0F0")

        self.header\_label.place(relx=0.5, rely=0.1, anchor="center")

        # Style

        self.style = ttk.Style()

        self.style.configure('TButton', font=('Arial', 14), background='#4CAF50', foreground='white')

        # Add More Signatures Button

        self.add\_signatures\_button = ttk.Button(self.root, text="Add More Signatures")

        self.add\_signatures\_button.place(relx=0.5, rely=0.3, anchor="center")

        # Contact Developers Button

        self.contact\_developers\_button = ttk.Button(self.root, text="Contact Developers")

        self.contact\_developers\_button.place(relx=0.5, rely=0.4, anchor="center")

        # Firewall & Network Protection Button

        self.firewall\_button = ttk.Button(self.root, text="Firewall & Network Protection")

        self.firewall\_button.place(relx=0.5, rely=0.5, anchor="center")

        # Device Security Button

        self.device\_security\_button = ttk.Button(self.root, text="Device Security")

        self.device\_security\_button.place(relx=0.5, rely=0.6, anchor="center")

        # Exit Button

        self.exit\_button = ttk.Button(self.root, text="Exit")

        self.exit\_button.place(relx=0.5, rely=0.9, anchor="center")

if \_\_name\_\_ == "\_\_main\_\_":

    frontend = Settings()

    frontend.root.mainloop()

treenode.py

# class TreeNode:

#     def \_\_init\_\_(self, value):

#         self.value = value

#         self.children = {}

class TreeNode:

    def \_\_init\_\_(self, value):

        self.value = value

        self.children = []

updates.py

import tkinter as tk

from tkinter import filedialog

from tkinter import messagebox

from MalwareDetector import MalwareDetector

from lists import Lists

from prosess import Process

import result

class Updates:

    def \_\_init\_\_(self):

      self.root = tk.Tk()

      self.root.title("Malware Detection")

      self.root.geometry("1300x700")

      self.root.configure(bg="#F0F0F0")

            # Header Label

      self.header\_label = tk.Label(self.root, text="Updates funtion is Devaloping", font=("Arial", 30, "bold"), fg="black", bg="#F0F0F0")

      self.header\_label.place(relx=0.5, rely=0.1, anchor="center")

if \_\_name\_\_ == "\_\_main\_\_":

    frontend = Updates()

    frontend.run()