import json

import os

import shutil

import csv

import tkinter as tk

from tkinter import ttk, filedialog

from datetime import datetime

import re

import exifread

def obtener\_coordenadas\_gps(imagen\_path):

    with open(imagen\_path, 'rb') as f:

        tags = exifread.process\_file(f, stop\_tag="GPS GPSLongitudeRef")

        if 'GPS GPSLatitude' in tags and 'GPS GPSLongitude' in tags:

            latitud = tags['GPS GPSLatitude'].values

            longitud = tags['GPS GPSLongitude'].values

            lat\_ref = tags['GPS GPSLatitudeRef'].values

            long\_ref = tags['GPS GPSLongitudeRef'].values

            return latitud, longitud, lat\_ref, long\_ref

    return None, None, None, None

def obtener\_fecha(nombre\_original):

    try:

        # Expresión regular para el formato YYYYMMDD\_HHMMSS

        regex\_yyyymmdd\_hhmmss = r'(\d{4})(\d{2})(\d{2})\_(\d{2})(\d{2})(\d{2})'

        # Expresión regular para el formato DJI\_YYYYMMDDHHMMSS

        regex\_dji\_yyyymmddhhmmss = r'DJI\_(\d{4})(\d{2})(\d{2})(\d{2})(\d{2})(\d{2})'

        # Intentar hacer coincidir con el formato YYYYMMDD\_HHMMSS

        match\_yyyymmdd\_hhmmss = re.match(regex\_yyyymmdd\_hhmmss, nombre\_original)

        if match\_yyyymmdd\_hhmmss:

            # Extraer la fecha y la hora

            year, month, day, hour, minute, second = match\_yyyymmdd\_hhmmss.groups()

            fecha\_hora = datetime(int(year), int(month), int(day), int(hour), int(minute), int(second))

        else:

            # Intentar hacer coincidir con el formato DJI\_YYYYMMDDHHMMSS

            match\_dji\_yyyymmddhhmmss = re.match(regex\_dji\_yyyymmddhhmmss, nombre\_original)

            if match\_dji\_yyyymmddhhmmss:

                # Extraer la fecha y la hora

                year, month, day, hour, minute, second = match\_dji\_yyyymmddhhmmss.groups()

                fecha\_hora = datetime(int(year), int(month), int(day), int(hour), int(minute), int(second))

            else:

                raise ValueError("Formato de fecha no reconocido")

        # Formatear la fecha y la hora

        return fecha\_hora.strftime('%y%m%d%H%M%S')

    except ValueError as e:

        print(f"No se pudo convertir la fecha en el nombre original '{nombre\_original}'. Se utilizará la fecha actual.")

        return datetime.now().strftime('%y%m%d%H%M%S')

def obtener\_valor():

    pergola\_volada = entry\_pergola\_volada.get().upper()

    mesa = combo\_mesa.get()

    fila = combo\_fila.get()

    if pergola\_volada and mesa and directorio\_originales.get() and fila:

        # Obtener el valor de la mesa indicada

        valor = inversores[pergola\_volada][mesa]

        resultado\_texto = f"El valor en la mesa '{mesa}' es: {valor}\n"

        # Obtener valores de las mesas en orden inverso al abecedario desde la mesa indicada hasta "A"  o "False" para que el valor no se inverso

        mesas\_invertidas = sorted(inversores[pergola\_volada].keys(), reverse=False)

        for mesa\_invertida in mesas\_invertidas:

            resultado\_texto += f"{mesa\_invertida}={inversores[pergola\_volada][mesa\_invertida]}, "

        resultado\_label.config(text=resultado\_texto)

        # Renombrar carpetas

        renombrar\_carpetas(pergola\_volada, directorio\_originales.get(), fila)

    else:

        resultado\_label.config(text="Asegúrate de completar todos los campos.")

def seleccionar\_directorio():

    # Pedir al usuario que seleccione el directorio de las imágenes originales

    directorio = filedialog.askdirectory(title="Seleccionar directorio de imágenes originales")

    if directorio:

        directorio\_originales.set(directorio)

def renombrar\_carpetas(pergola\_volada, directorio\_originales, fila):

    if pergola\_volada not in inversores:

        print(f"Pergola '{pergola\_volada}' no encontrada en el JSON.")

        return

    # Obtener la mesa seleccionada

    mesa\_seleccionada = combo\_mesa.get()

    # Obtener el índice de la mesa seleccionada en la lista de columnas

    indice\_mesa\_seleccionada = columnas.index(mesa\_seleccionada)

    # Obtener las mesas en el orden adecuado según la selección

    if mesa\_seleccionada == 'A':

        mesas = columnas[:indice\_mesa\_seleccionada+1]

    elif mesa\_seleccionada == 'E':

        mesas = columnas[indice\_mesa\_seleccionada:]

    else:

        mesas = columnas[indice\_mesa\_seleccionada::-1]

    # Inicializar el contador de imágenes

    contador\_imagenes = 0

    for mesa in mesas:

        valor\_mesa = inversores[pergola\_volada][mesa]

        if valor\_mesa <= 0:

            mensaje\_label.config(text=f"No hay imágenes para la pergola '{pergola\_volada}', mesa '{mesa}'.")

            continue

        # Crear el directorio de destino si no existe

        directorio\_destino\_por\_mesa = os.path.join(directorio\_originales, f"{pergola\_volada}\_{mesa}")

        os.makedirs(directorio\_destino\_por\_mesa, exist\_ok=True)

        # Obtener la lista de archivos en el directorio de imágenes originales

        archivos = [archivo for archivo in os.listdir(directorio\_originales) if os.path.isfile(os.path.join(directorio\_originales, archivo))]

        # Filtrar solo las imágenes (puedes ajustar la extensión según tus necesidades)

        imagenes = [archivo for archivo in archivos if archivo.lower().endswith(('.png', '.jpg', '.jpeg'))]

        # Obtener el número de imágenes para esta mesa

        cantidad\_imagenes = min(valor\_mesa, len(imagenes) - contador\_imagenes)

        # Obtener el nombre de la carpeta de destino (sin la ruta)

        nombre\_carpeta\_destino = os.path.basename(directorio\_destino\_por\_mesa)

        # Crear un archivo CSV en la carpeta de destino con el nombre de la carpeta

        csv\_path = os.path.join(directorio\_destino\_por\_mesa, f"{nombre\_carpeta\_destino}\_F{fila}.csv")

        with open(csv\_path, mode='w', newline='') as csv\_file:

            csv\_writer = csv.writer(csv\_file)

            # Reiniciar el contador de columnas para cada mesa

            contador\_columnas = 1

            # Renombrar y mover las imágenes al directorio de destino

            for i in range(contador\_imagenes, contador\_imagenes + cantidad\_imagenes):

                # Obtener la fecha y hora del nombre original

                fecha\_original = obtener\_fecha(imagenes[i])

                # Crear el nuevo nombre de la imagen con el número de columna reiniciado para cada mesa

                nuevo\_nombre\_base = f"{pergola\_volada}\_{mesa}"

                nuevo\_nombre = f"{fecha\_original}\_{nuevo\_nombre\_base}-F{fila}-C{contador\_columnas}"

                # Construir las rutas de origen y destino

                viejo\_ruta = os.path.join(directorio\_originales, imagenes[i])

                nuevo\_ruta = os.path.join(directorio\_destino\_por\_mesa, f"{nuevo\_nombre}{os.path.splitext(imagenes[i])[1]}")

                # Copiar la imagen al directorio de destino con el nuevo nombre

                shutil.copy(viejo\_ruta, nuevo\_ruta)

                # Obtener las coordenadas GPS de la imagen

                latitud, longitud, lat\_ref, long\_ref = obtener\_coordenadas\_gps(viejo\_ruta)

                # Convertir las coordenadas a formato decimal

                latitud\_decimal = float(latitud[0]) + float(latitud[1])/60 + float(latitud[2])/3600

                longitud\_decimal = float(longitud[0]) + float(longitud[1])/60 + float(longitud[2])/3600

                # Ajustar los valores según la referencia

                if lat\_ref == 'S':

                    latitud\_decimal \*= -1

                if long\_ref == 'W':

                    longitud\_decimal \*= -1

                # Agregar el nuevo nombre al archivo CSV

                csv\_writer.writerow([f"{nuevo\_nombre}{os.path.splitext(imagenes[i])[1]}", latitud\_decimal, longitud\_decimal])

                print(f"Se ha renombrado y copiado {imagenes[i]} para la pergola '{pergola\_volada}', mesa '{mesa}' a '{directorio\_destino\_por\_mesa}' con la fecha original '{fecha\_original}'.")

                # Incrementar el contador de columnas para la próxima imagen

                contador\_columnas += 1

        # Actualizar el contador de imágenes para la próxima mesa

        contador\_imagenes += cantidad\_imagenes

        print(f"Se ha creado el archivo CSV en: {csv\_path}")

# Cargar el JSON

with open('castelnau.json', 'r') as json\_file:

    inversores = json.load(json\_file)

# Crear la interfaz

root = tk.Tk()

root.title("Obtener Valor y Renombrar Carpetas")

# Variables globales para los directorios

directorio\_originales = tk.StringVar()

# Widgets

label\_pergola = ttk.Label(root, text="Pergola volada:")

entry\_pergola\_volada = ttk.Entry(root)

label\_directorio = ttk.Label(root, text="Directorio:")

entry\_directorio = ttk.Entry(root, state="readonly", textvariable=directorio\_originales)

button\_seleccionar\_directorio = ttk.Button(root, text="Seleccionar Directorio", command=seleccionar\_directorio)

label\_fila = ttk.Label(root, text="Fila:")

filas = [ "1", "2", "3", "4", "5", "6", "7", "8", "9", "10", "11", "12", "13"]

combo\_fila = ttk.Combobox(root, values=filas)

label\_mesa = ttk.Label(root, text="Mesa:")

columnas = ["A", "B", "C", "D", "E"]

combo\_mesa = ttk.Combobox(root, values=columnas)

button\_obtener\_valor = ttk.Button(root, text="Renombrar", command=obtener\_valor)

resultado\_label = ttk.Label(root, text="")

mensaje\_label = ttk.Label(root, text="")

# Posicionamiento de widgets

label\_pergola.grid(row=1, column=0, padx=5, pady=5)

entry\_pergola\_volada.grid(row=1, column=1, padx=5, pady=5)

label\_directorio.grid(row=0, column=0, padx=5, pady=5)

entry\_directorio.grid(row=0, column=1, padx=5, pady=5)

button\_seleccionar\_directorio.grid(row=0, column=2, padx=5, pady=5)

label\_fila.grid(row=3, column=0, padx=5, pady=5)

combo\_fila.grid(row=3, column=1, padx=5, pady=5)

label\_mesa.grid(row=2, column=0, padx=5, pady=5)

combo\_mesa.grid(row=2, column=1, padx=5, pady=5)

button\_obtener\_valor.grid(row=5, column=1, padx=5, pady=10)

resultado\_label.grid(row=6, column=0, columnspan=2, pady=5)

mensaje\_label.grid(row=7, column=0, columnspan=2, pady=5)

# Iniciar la interfaz

root.mainloop()