```python
# -*- coding: utf-8 -*-
"""
Created on Tue Aug 26 17:36:53 2025

@author: ugims
"""

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# === READ AND CLEAN OF CSV FILE ===
file_path = r"C:/Users/ugims/inegi.up.pt/Teses & EstГЎgios - Teses_EstГЎgios - Miguel AntГinio Costa - Teses_EstГЎgios - Miguel AntГinio Costa/3. RepositГirio do Migue

df = pd.read_csv(
    file_path,
    skiprows=8,
    sep=',',
    skip_blank_lines=True,
    engine='python',
    encoding='windows-1252'
)

# Clean column names
df.columns = df.columns.str.strip()

# remove column "Components" (opcional, hardly used)
if 'Components' in df.columns:
    df.drop(columns=['Components'], inplace=True)

# convert main columns to numericals
for col in ['Node', 'Value (Celsius)', 'X (mm)', 'Y (mm)', 'Z (mm)']:
    df[col] = pd.to_numeric(df[col], errors='coerce')

# clean invalid lines
df.dropna(inplace=True)

print("вњ… Data uploaded successefuly.")
print("Data pre-screen:\n", df.head())

# === CALCULATION FOR GLOBAL THERMIC UNIFORMITY ===
temperaturas = df['Value (Celsius)']
media_global = temperaturas.mean()
desvio_padrao = temperaturas.std()
cv_percentual = (desvio_padrao / media_global) * 100
delta_T = temperaturas.max() - temperaturas.min()

print("\n=== Global indicators of thermal uniformity ===")
print(f"Medium global temperature: {media_global:.2f} B°C")
print(f"Standard deviation: {desvio_padrao:.2f} B°C")
print(f"variation cofficient (CV): {cv_percentual:.2f}%")
print(f"О”T (mГЎx - mГn): {delta_T:.2f} B°C")

# Export global results (opcional)
pd.DataFrame([{
    'Temperatura mГ©dia global (B°C)': media_global,
    'Desvio-padrГЈo (B°C)': desvio_padrao,
    'Coeficiente de variaГ§ГЈo (%)': cv_percentual,
    'О”T (mГЎx - mГn) (B°C)': delta_T
}]).to_csv("resultado_uniformidade_molde1.csv", sep=';', index=False)

# === DEFINITION OF ZONES ALONG AXIS X ===
# Automatic division of interval [xmin, xmax] in 3 equal zones
xmin = df['X (mm)'].min()
xmax = df['X (mm)'].max()
L = xmax - xmin

# Bins: [xmin, xmin+L/3), [xmin+L/3, xmin+2L/3), [xmin+2L/3, xmax]
edges = [xmin, xmin + L/3, xmin + 2*L/3, xmax + 1e-9]  # +epsilon to include last point

df['Zona'] = pd.cut(
    df['X (mm)'],
    bins=edges,
    labels=['Z1', 'Z2', 'Z3'],
    include_lowest=True,
    right=False  # left-inclusive, right exclusive (avoid boundary holes)
)

# All point classication to be validated
if df['Zona'].isna().any():
    n_na = df['Zona'].isna().sum()
    print(f"вљ пёЏ WARNING: {n_na} dots not set. Verify X limits.")

# === ZONE METRICS ===
agg = df.groupby('Zona')['Value (Celsius)'].agg(['count','mean','std','min','max']).rename(
    columns={'count':'N','mean':'T_mean','std':'T_std','min':'T_min','max':'T_max'}
)
agg['CV_%'] = (agg['T_std'] / agg['T_mean']) * 100
agg['DeltaT'] = agg['T_max'] - agg['T_min']

print("\n=== Zone Metrics ===")
print(agg)

# Export zone metrics (opcional)
agg.to_csv("metricas_por_zona.csv", sep=';')

# === ISOLATION INDICE (tipical case: heat only Z2) ===
# Pondered average of Z1 and Z3 (ponderada por N de pontos), devided for average of Z2
# In case of missing zones, avoid division by 0.
def safe_get(zone, col):
    return agg.loc[zone, col] if zone in agg.index else np.nan

N1, N2, N3 = safe_get('Z1','N'), safe_get('Z2','N'), safe_get('Z3','N')
T1, T2, T3 = safe_get('Z1','T_mean'), safe_get('Z2','T_mean'), safe_get('Z3','T_mean')

if pd.notna(N1) and pd.notna(N2) and pd.notna(N3) and N2 > 0:
    # MГ©dia ponderada lateral (Z1+Z3)
    T13 = (T1*N1 + T3*N3) / (N1 + N3) if (N1 + N3) > 0 else np.nan
    isolamento_percent = (T13 / T2) * 100 if pd.notna(T13) and pd.notna(T2) and T2 != 0 else np.nan
    print("\n=== Isolation Indice (apt only when Z2 is heated) ===")
    print(f"Average Z1+Z3 (ponderada): {T13:.2f} B°C")
    print(f"Average Z2: {T2:.2f} B°C")
    print(f"Isolation (Z1+Z3 / Z2 Г— 100): {isolamento_percent:.2f} %")
else:
    print("\nвљ пёЏ WARNING: Isolation Indice calculation was not possible, please verify your zones and data.")

# === GRAPHIC 3D ===
x = df['X (mm)']
y = df['Z (mm)']  # width of CSV
temperatura = df['Value (Celsius)']

fig = plt.figure(figsize=(10, 8))
```

```
ax = fig.add_subplot(111, projection='3d')
scatter = ax.scatter(x, y, temperatura, c=temperatura, cmap='plasma')

ax.set_xlabel('X (mm) - Comprimento')
ax.set_ylabel('Z (mm) - Largura')
ax.set_zlabel('Temperatura (B°C)')
ax.set_title('DistribuiΓ§ΓJo de Temperatura na SuperfΓ-cie Superior do Molde')

# Mark boundary zones of axis X (opcional, visual)
for xb in edges[1:-1]:
    ax.plot([xb, xb], [y.min(), y.max()], zs=temperatura.mean(), zdir='y', alpha=0.0)  # "invisible" guide to keep it all simple

cbar = plt.colorbar(scatter, ax=ax, pad=0.1)
cbar.set_label('Temperatura (B°C)')

plt.tight_layout()
plt.show()
```