Trabajo Final | Ciencia de Datos

Andrés Felipe Jaramillo Tamayo

José Antonio Trillos Paredes

Juan Pablo Aguirre Martínez

Código

- *main.py*

```
import os
import pandas as pd
from geopy.distance import geodesic
from components import extract_send
from components import processdata
from components import print
from components import conec_sqlite3

#Create paths to find the data
gpx_directory = os.path.dirname(__file__)
gpx_subdirectory = os.path.join(gpx_directory,'data')
gpx_files = [f for f in os.listdir(gpx_subdirectory) if f.endswith
(".gpx")]
DBMS_Path = os.path.join(gpx_directory, 'data', 'DBMS.db')
Resulth_Path = os.path.join(gpx_directory, 'data',
'ResultGPXprocessrun_{}.xlsx')

# Read each GPX file and calculate distance, slope, speed, and acc
eleration
if __name__ == '__main__':
    for indice, gpx_file in enumerate(gpx_files):
        # Get the data object
        gpx = extract_send.read_data(gpx_subdirectory, gpx_file)
        # Create a list of data from the GPX data
        data = extract_send.get_data(gpx)
        # Create a dataframe from the data
        df = pd.DataFrame(data, columns=["lat", "lon", "ele",
"time"])

# Calculate the distance between each point and the next point in
 the route
        df["Distance"] = processdata.distance(df)
        df["DistanceTotal"] = df["Distance"].cumsum()

# Calculate the slope between each point and the next point in the
route
        df["Slope"] = processdata.slope(df)

# Calculate the speed between each point and the next point in the
route
        df["Speed"] = processdata.speed(df)

# Calculate the acceleration between each point and the next point
in the route
        df["Acceleration"] = processdata.acceleration(df)
        # Rename column names
        df.rename(columns={"lat": "Latitude","lon":"Longitude",
"ele":"Elevation","time":"Time"}, inplace=True)
        # Change time column
        df["Time"] = pd.to_datetime(df["Time"]).dt.strftime(
"%H:%M:%S")
        # Print results
        print.print_results(df)
        # Create database
        BD = conec_sqlite3.database(df, indice, DBMS_Path)

# Save the dataframe as a CSV file with the same name as the GPX f
ile
        extract_send.send_data(df,Resulth_Path.format(indice+1))
```

- *extract_send.py*

```python
import os
import gpxpy
from geopy.distance import geodesic
import xlsxwriter
import pandas as pd

# Read the data from the gpx_file
def read_data(gpx_subdirectory,gpx_file):
    try:
        print('Get the data object')
        with open(os.path.join(gpx_subdirectory, gpx_file), "r") as f:
            gpx = gpxpy.parse(f)
    except Exception as e:
        print("Error",e)
        print('Error reading the file')
    return gpx

# Create a list of data from the GPX data
def get_data(gpx):
    data = []
    for track in gpx.tracks:
        for segment in track.segments:
            for point in segment.points:
                data.append([point.latitude, point.longitude, point.elevation, point.time])
    return data

# Send the result data
def send_data(df,file_path):
    try:
        writer = pd.ExcelWriter(file_path,engine='xlsxwriter')
        df_result = df.to_excel(writer,index=False)
        writer.close()
        print('Result file created')
    except Exception as e:
        print("Error",e)
        print('Result file not created')
    return df_result
```

- processdata.py

```python
import pandas as pd
from geopy.distance import geodesic

# Calculate the distance between each point and the next point in the route
def distance(df):
    distances = []
    for index, row in df.iterrows():
        if index < len(df) - 1:
            point1 = (row["lat"], row["lon"])
            point2 = (df.iloc[index+1]["lat"], df.iloc[index+1]["lon"])
            distances.append(geodesic(point1, point2).meters)
        else:
            distances.append(0)
    return distances

# Calculate the slope between each point and the next point in the route
def slope(df):
    slopes = []
    distances = df["Distance"]
    for index, row in df.iterrows():
        if index < len(df) - 1:
            if distances[index] == 0:
                slopes.append(0)
            else:
                slope = (df.iloc[index+1]["ele"] - row["ele"]) / distances[index]
                slopes.append(slope)
        else:
            slopes.append(0)
    return slopes

# Calculate the speed between each point and the next point in the route
def speed(df):
    speeds = []
    distances = df["Distance"]
    for index, row in df.iterrows():
        if index < len(df) - 1:
            time_diff = (pd.to_datetime(df.iloc[index+1]["time"]) - pd.to_datetime(row["time"])).total_seconds()
            speed = distances[index] / time_diff
            speeds.append(speed)
        else:
            speeds.append(0)
    return speeds

# Calculate the acceleration between each point and the next point in the route
def acceleration(df):
    accelerations = []
    speeds = df["Speed"]
    for index, row in df.iterrows():
        if index < len(df) - 1:
            time_diff = (pd.to_datetime(df.iloc[index+1]["time"]) - pd.to_datetime(row["time"])).total_seconds()
            speed_diff = speeds[index+1] - speeds[index]
            acceleration = speed_diff / time_diff
            accelerations.append(acceleration)
        else:
            accelerations.append(0)
    return accelerations
```

- conec_sqlite3.py

```python
import sqlite3
import pandas as pd

# Create database
def database(df,indice, DBMS_Path):
    BD = []
    try:
        conn = sqlite3.connect(DBMS_Path)
        BD = df.to_sql('BDrun{}'.format(indice+1), conn, if_exists='replace', index=False)
        conn.close()
    except Exception as e:
        print(e)
    return BD
```

Estructura de los archivos