README
Arroyo Rivera Juan José 416053223
Versión Beta

Ejecución del programa:

Instalar python3
En la carpeta /src ejecutar: *python webService.py*

Nota: Faltan tests y manejo de excepciones para la peticion a la API, pues algunas veces la API_KEY no esta online y no se ejecuta la query

1. *Definición del problema:* Crear un programa que procese un Csv con informacion de vuelos (ciudad de origen y destino) y para cada uno de estos vuelos determinar la temperatura actual en ambas ciudades
2. *Análisis del problema*: Se requiere procesar el archivo csv de entrada para extraer la información referente a: Número de ciudades diferentes con su latitud y longitud y Ciudad de Origen - Destino asociada a cada fila del archivo. A partir de esta extracción de información se pueden hacer consultas a la API de OpenWeather utilizando la longitud y latitud de cada ciudad. Posteriormente se guarda esta información en un diccionario y se hacen consultas en este diccionario con el nombre de la ciudad como llave para cada ciudad en cada vuelo, y las temperaturas asociadas se agregan a la información inicial. Finalmente la información actualizada se escribe en un archivo csv de salida.
3. *Selección de la mejor alternativa:* Conviene extraer las ciudades diferentes y obtener su latitud y longitud, a partir de estos datos hacer las consultas a la API por cada ciudad y guardar las temperaturas en un diccionario. Tras guardar estos resultados ya solo se hacen consultas en el diccionario creado para agregar la información de la temperatura de cada una de las ciudades por cada vuelo.
4. *Pseudocódigo*:

main():
Se instancia un objeto webService
Se llama a *readCsv()*
Se llama a *updateTemperatureRecords()*
Se llama a *writeOutputCsv()*

readCsv(filename):
Se instancia un csv.reader con el filename
Se omite la primer fila (header del csv)
for row en csvreader:
Se agregan ambas ciudades en la fila (2 iteraciones) a self.citiesCoordinates[ciudad] = (latitud, longitud)
Se agrega una lista en self.destinationOriginForFlights de la forma [ciudad_origen, ciudad_destino]

updateTemperatureRecords():
for ciudad en self.citiesCoordinates:
Si la ciudad no esta en self.citiesTemperature

(no se ha obtenido su temperatura):
self.citiesTemperature[ciudad] = self.getTemperature(ciudad)

getTemperature(ciudad):

Se obtienen las coordenadas de la ciudad desde self.citiesCoordinates[ciudad]

Se hace una peticion construyendo la Url para OpenWeather utilizando las coordenadas obtenidas

writeOutputCsv(output)

for listaCiudadOrigenDestino en self.destinationOriginForFlights:

Se anexa a la lista [ciudadOrigen, ciudadDestino] los valores de las temperaturas de acuerdo al diccionario self.citiesTemperature

Y termina de la forma [ciudadOrigen, ciudadDestino, temperaturaCiudadOrigen. temperaturaCiudadDestino]

Se reescribe esta lista de listas en el archivo *output.csv*