5.0 - External sources

Often data is fetched from remote sources, such as in this example "https://towardsdev.com/create-an-etl-pipeline-in-python-with-pandas-in-10-minutes-6be436483ec9"
In this article a movie record is transformed so that it is easier to identify different genres in the table.

For this exercise we are going to do a similar etl process, but related to the weather instead of movies, and using real-live data.

5.1 - OpenWeather

https://openweathermap.org/
Openweather is an API that allows anybody to request live data from its servers.

A. Register an account on the website
B. Navigate to your profile (upper right corner), press "My API keys"
C. Write a new at "Create key" and generate the key

Assuming the registration went fine you could verify that the key is working by entering the following url (**replace <api_key>** with your generated key(!))

https://api.openweathermap.org/data/2.5/weather?lat=57.7&lon=12&appid=**<api_key>**

The lon and lat parameters in the url may be found by entering google maps https://www.google.com/maps and then right click on the desired location.

5.2 - OpenWeather Requests

The key that you received from *5.1 - OpenWeather* should remain private to you, which is why you now shall put it into a config file in your working directory.

5.2.1 - Hidden configuration data

A. Download the zip file called "weather_data" from canvas
B. In the folder, locate a file called "config.ini"
    a. In this file, type the following (replace <api_key> with your key)

```
[DEV]
API_KEY = <api_key>
```

5.2.2 - weather_data_requests.py

The file contains detailed documentation of the working flow, however, if you wish to modify the amount of cities you will be needing to change the directory geo_locations. You may do

this by entering google maps https://www.google.com/maps, right click on desired location and copy the coordinates. They are written in lon first and lat second.

5.2.3 - weather_etl.py

The file contains one main function, *transform_weather* which at present time converts the different earlier samples from the requests into a single table with the city and name values.

5.3 - Task, add more attributes

The weather data is compiled in weather_etl.py into final_weather.csv per the description of a weather_entry in the aforementioned file.

```
weather_entry = {
    "city": os.path.basename(path)[:-4], # Remove ".csv" at the end
    "tempature": entry["main.temp"] - 273, # Convert to celcius from kelvin
}
```

As a first task you will add the missing weather attributes:

- Weather
- Weather description
- Cloudy
- Humid

In order to solve this task you need to figure out the reference for these attributes.

A. Inspect any of the csv files in data
B. Find column name matching the attributes above
C. In the weather_entry dictionary (in the picture above) add the identified attribute name, e.g. “weather” followed by the value entry[column_name] corresponding to the identified column in step B.

   E.g. for weather it would look like
   “weather”: entry[“weather.main”],

```
weather_entry = {
    "city": os.path.basename(path)[:-4], # Remove ".csv" at the end
    "tempature": entry["main.temp"] - 273, # Convert to celcius from kelvin
    "weather": entry["weather.main"],
}
```

After adding all the missing attributes you should generate a file looking similar to *target/final_weather.csv*

```
city,tempature,weather,weather_desc,cloudy,humid
luleå,20.20999999999998,Clear,clear sky,0,64
uppsala,18.069999999999993,Clouds,overcast clouds,86,88
östersund,15.100000000000023,Clouds,broken clouds,75,84
göteborg,14.0,Rain,light rain,75,90
trelleborg,18.420000000000016,Clear,clear sky,0,63
```