Poo.py Documentation

1. Data Sources:

Data	**Attributes**	**Source**	**URL**
City	Cities rank by population	.xlsx	https://www2.census.gov/programs-surveys/popest/tables/2010-2019/cities/totals/SUB-IP-EST2019-ANNRNK.xlsx
	State Abbreviation	.xls	https://www3.epa.gov/ttnairs1/airsaqsORIG/manuals/State%20and%20County%20Codes.xls
	Coordinates	API	https://developer.mapquest.com/documentation/open/
	Zip Code	CSV	https://simplemaps.com/data/us-zips
Environment (take average)	Air quality index - PM2.5	API	https://docs.airnowapi.org/
	Safe drinking water violation %	CSV	https://www.sustainabledevelopment.report/reports/2019-us-cities-sustainable-development-report/
Job	Job Opportunity	Web scraping	https://wallethub.com/edu/best-cities-for-jobs/2173
Education	Education index	Web scraping	https://wallethub.com/edu/e/most-and-least-educated-cities/6656
Safety	Safety score	Web scraping	https://wallethub.com/edu/safest-cities-in-america/41926
Leisure & Accessibility	Walk Score	API	https://www.walkscore.com/professional/walk-score-apis.php

(take the average)	Bike Score	API	https://www.walkscore.com/professional/walk-score-apis.php
	Parkscore	PDF & CSV	https://parkserve.tpl.org/mapping/historic/2020_ParkScoreRank.pdf
Affordability	Cost of Living Index	Web scraping	https://www.numbeo.com/cost-of-living/country_result.jsp?country=United+States

2. Libraries Used (details in source code)

- Requests - an HTTP library for page download
- Json - API
- BeautifulSoup4 - help pulling data from scraping static web pages
- Pandas - csv, excel, dataframe
- Numpy -
- Folium - for map visualization
- Ipywidgets - to construct sliders for weight selection
- Plotly - Radar Plot

3. “Behind the Scene” Workflow

For your reference (in case you want to run all our code from scratch)

Note: since we have used data sources with web scraping, APIs, etc. the content and format of such data sources may change in the future and be different from our results. Certain websites may block IP addresses using web scraping or API calls that reach a maximum limit, and hence cause errors after running the code.
* You will need to upgrade your Python to 3.9 to run the code.

Go to “Final Source Code” folder > ‘Thomas’ folder

1. Run ‘city_state_geocode_walkscore.py’ **[clean and merge the data]**
    a. The first part of this script read the city data from 2 excel files, one with city full names and other with city’s state abbreviation
        1. The output is ‘final_citystate.csv’
    b. The second part of this script is to use the city name from the first output ‘final_citystate.csv’ to get the city’s coordinates from a website using API **[collect, clean and merge the data]**

        1. The output is 'final_city_state_geocode.csv'
    c. The last part of this script is to use the city's coordinates from the second output to get the walk score and bike score from a website using API **[collect, clean and merge the data]**
        1. The output is 'final_city_state_geocode.csv'

Go to "Final Source Code" folder > 'Yvette' folder

1. Run 'cost_of_living.py' **[collect the data]**
    a. The script scrapes the data for the cost of living from the web page.
    b. The data was previously stored in '<table>' in HTML codes
    c. Its output is a .csv file named 'yvette.csv'.
2. Run 'ranked_index.ipynb' **[process the data]**
    a. The program will export the ranking numbers of each city in terms of different variables
    b. The only parameter of the program will determine if cities are ranked in an ascending way or descending way.

Go to "Final Source Code" folder > 'Jamie' folder

1. Run 'edu_score_webscraping.py'. **[web scrape data]**
    a. This script scrapes data for the education score from the web page
    b. It produces an output called 'edu_score.csv'
2. Run 'job_score_webscraping.py'. **[web scrape data]**
    a. This script scrapes data for the job score from the web page
    b. It produces an output called 'job_score.csv'
3. Run 'safety_score_webscraping.py'. **[web scrape data]**
    a. This script scrapes data for the safety score from the web page
    b. It produces an output called 'safety_score.csv'

    ** web scraping scripts may not run if your IP is blocked from the web page. In this case, use csv output files provided in the same folder
4. Run 'score_cleaning.py'. **[clean score data]**
    a. This script cleans education/job/safety score data from step 3
    b. It produces outputs called 'edu_score_cleaned.csv', 'job_score_cleaned.csv', 'safety_score_cleaned.csv'
5. Run 'merge_job_edu_safety_cities.py'. **[merge score data]**
    a. This script merges cleaned score data from step 4 with city names & abbreviations data ('city_cleaned_with_abbr.csv')
    b. It produces an output called 'jamie.csv'

Go to "Final Source Code" folder > 'Tina' folder

1. Run 'city_cleaned_with_zip_tina.py'. **[collect and clean the data]**

    a. This script uses ‘uszips.csv’ and ‘final_city_state_geocode_thomas.csv’ (you can find them in the folder)
    b. It produces an output called ‘city_cleaned_with_zip_tina.csv’

2. Run ‘aqi_scraped_tina.py’. **[collect and clean the data]**
    a. This script uses ‘city_cleaned_with_zip_tina.csv’ from step 1.
    b. It produces an output called 'aqi_scraped_tina.csv'
    (Note: since this API has a limit on how many times it can be accessed by each account, it is possible that when you run the code, it fails to return the results. Therefore, I include the output ‘aqi_scraped_tina.csv’ in the folder as well for your reference)
3. Run ‘aqi+park_score_tina.py’. **[collect and clean the data]**
    a. This script uses ‘ParkScore_2021.csv’ (you can find it in the folder) and ‘aqi_scraped_tina.csv' from step 2
    b. It produces an output called 'aqi+park_score_tina.csv'
4. Run ‘aqi+park_score+water_tina.py’ **[collect and clean the data]**
    a. This script uses '2019USCitiesIndexResults.csv' (you can find it in the folder) and 'aqi+park_score_tina.csv' from step 3
    b. It produces an output called ‘tina.csv’
5. Run ‘merge.ipynb’ **[merge the data]**
    a. This script uses ‘tina.csv’, ‘yvette.csv’, ‘final_walkscore_thomas.csv’,’jamie.csv’
    b. Except ‘tina.csv’ produced from step 4, the other csv’s can be found in ‘Yvette’, ‘Jamie’, and ‘Thomas’ folders respectively (you may check their code workflow to see how to get those csv’s)
    c. It produces an output called ‘merged_final.csv’ to be used for the final product.
    This csv is included in ‘Final Source Code’ folder > ‘User Product’ folder in order to run the final visualization code.