README

This zip file: **L\_Group6,** contains all files needed for our project from scraping data, cleaning data to UI designing.

You can find the instruction video here: +*youtube link*

**Requirements:**

1. scrapy - spider modules
2. Workbook – the tool to open excel, add this line in the code and install openpyxl module:

from openpyxl.workbook import Workbook

This situation may occur in MAC, for windows you may not need to add this

3. selenium - Browser simulator for scraping javascript rendered website

\*Checking your Chrome browser version from Chrome setting

\*Download corresponding browser drivers from http://npm.taobao.org/mirrors/chromedriver/

\*Put the downloaded browser driver into corresponding Anaconda environment. (e.g. my path is: E:\Anaconda3\envs\pythonProject)

4. Numpy - for data processing

5. Pandas - for dataframe and data processing

6. pyqt5 (This one has to be installed through pip in the corresponding environment terminal)

\*Referrence: https://pypi.org/project/PyQt5/

\*pip install PyQt5

7. pyqt5-tools (This one also has to be install through pip in the corresponding environment terminal)

\*Referrence: https://pypi.org/project/pyqt5-tools/

\*pip install pyqt5-tools

8. fuzz module - in the corresponding environment by “pip install fuzzywuzzy” in cmd(win)/terminal(MacOS)

9.requests – requests modules

**L\_Group6 contains one README file and 6 folders:**

1. **Target**

This folder contains all files needed to product final cleaned target data, and a pyqt5 ui file which allows user to quickly calculate calories of a list of foods by sending a post request to Nutrition API.

1. **targetProject - Scrape Target Websites**

The folder targetProject contains all file needed to scrape data from target website using scrapy.

- codes.py - containing a constant storing a list of target grocery websites' identifiers

- items.py - defined an item scraped from target

- middlewares.py - contains Selenium Module to simulate real person visiting the website, so we can scrape data from javascript rendered result.

- settings.py - setting file for scrapy spider

- pipelines.py - process results after finish scraping

- run.py - **START** the scrapy spider program

- /spiders

- target.py - codes for the spider to scrape data from various target websites

**2) other files in Target**

- clean\_clean\_target.xlsx - final cleaned target data

* clean\_clean\_target\_data.py - the python program used to create the final target clean data
* clean\_target\_data.py - first attempt to clean the raw target file and product target\_clean.xlsx
* target\_clean.xlsx - result file of first clean attempt
* target.csv - raw target data scraped from its website

1. **Yelp**

This folder contains all files needed to product final cleaned yelp data with restaurant detailed information:

1. **Py files**

All the py files are inside the Yelp folder and outside the data\_completed, data\_scrape folders

- yelpScrape.py – contains Selenium module and chrome webdriver, the codes to scrape restaurant list page with restaurant names and detail page links. Used for detailPage.py

- detailPage.py – contains Selenium module and chrome webdriver, codes to scrape restaurant details using links scraped before.

- yelpmenuScrape.py – contains Selenium module and chrome webdriver, codes to scrape restaurant menu(dishes and recipe) details using links scraped before. [Please run restaurantClean.py before run yelpmenuScrape.py]

- restaurantClean.py – clean the yelp data (detailPage.csv)

- menuCombine.py – combine the data in menulist.csv which means combine menu name & recipe data in menu page together if they belong to the same restaurant.

getRaw.py – combine res\_list.csv, menulist.csv and detailPage.csv together

getClean.py – combine res\_list.csv, all\_menudata.csv and detailPage\_clean.csv together

**2) CSV files**

- /data\_scrape

the empty folder to store the output csv files in case of changing the existed outcome tables

* /data\_completed

The folder stores all copies of existed outcomes, please do not change the folder. This folder is used as input path in case of long-time scraping process

* res\_list.csv – contains all restaurant names and links as an outcome table of yelpScrape.py
* detailPage.csv - contains all restaurant details information as an outcome table of detailPage.py
* menulist.csv – contains all menu details ( name & recipe) as an outcome table of yelpmenuScrape.py
* all\_menudata.csv – contains cleaned data with one restaurant one row of menu detail information as an outcome table of menuCombine.py
* detailPage\_clean.csv – contains cleaned data of restaurant information as an outcome table of restaurantClean.py
* raw\_data.csv – contains all raw data of yelp as an outcome table of getRaw.py
* restaurant\_clean.csv – contains all clean data of yelp as an outcome table of getClean.py

**3. Recipe**

This folder contains two python program, no other packages needed to install to run the program. The program will produce two .csv files and one .txt file.

* scrape\_method.py

This file is to scrape the recipe instruction and ingredients with the amount and unit form website 101cookbook. This program will create three files named "reciperaw.txt"(stores the raw data scraped from the website),

"recipe\_instruction\_try.csv"(contains recipe name and instruction),

and "cleaned\_ingredient\_try.csv"(contains recipe name, amount, unit and ingredients)

* recipename\_clean.py

This file read the ingredients and further clean the content. The program will generate one file: 'recipe\_clean\_try.csv' (which is recipe\_clean.csv in python\_ui to avoid overrride)

* recipe\_clean.csv: contains recipe name, amount, unit and ingredients
* recipe\_instruction.csv: contains recipe name and instruction
* cleaned\_ingridients.csv: contains recipe name, amount, unit and ingredients

**4. food\_conflict**

This folder contains all files needed to product final cleaned food conflict data:

1. **Py files**

- clean\_food\_conflicts.py – clean the downloaded data from CDC to remove irrational data

- getRaw\_fc.py – combine raw data food\_conflicts.csv and FoodData.csv together

- getClean\_fc.py – combine food\_cdc.csv and cleaned allergy data(FoodData.csv) together

**2) CSV files**

- /data\_downloaded

the empty folder to store the output csv files in case of changing the existed outcome tables

* /data\_completed

The folder stores all copies of existed outcomes, please do not change the folder. This folder is used as input path in case of some unexpected situation:

* food-conflicts.csv – contains all downloaded CDC food born diseases report
* FoodData.csv - contains all allergy information from Kaggle
* food\_cdc.csv – contains all cleaned CDC data as an outcome table of clean\_food\_conflicts.py
* FoodDataFinal.csv – contains all raw data as an outcome of getRaw\_fc.py
* FoodDataFinalClean.csv – contains all clean data as an outcome of getClean\_fc.py

**5. MatchRecipe**

The folder contains files to product match results between ingredients’ prices and recipes

- match\_recipe\_clean.py: the file to link target price and recipe ingredients together, to get each recipes’ price. We do not use real-time matching is because the matching process is about 3 hours due to the data size. So we run it before, and use final results in UI. The code will generate recipe\_match.csv.

- recipe\_clean.csv: the file contains cleaned recipe data copy from Recipe folder

- target\_clean.csv: the file contains cleaned target data copy from Target folder

**6. python\_ui**

This folder includes all python files to set up the UI windows and necessary functions to implement some data, as well as some files created when running python files (EXCLUDING files generated from other programs).

**1) Py files**

- index.py: This file is used to set up the first window “Input Body Data”, where you can input age/height/weight/activity level/gender(click on the pic), then calculate corresponding calories. This program requires inputting two png files in the folder, named”woman\_posture.png” and “man\_posture.png”

- ingredient\_to\_recipe.py: this program contains functions used to match ingredients with recipes, find prices of certain recipes and find the list of instructions of recipes. The program requires inputting files including “recipe\_clean.csv”, “recipe\_match.csv” and “recipe\_instruction.csv”

- main1.py: The program is used to design the first window’s UI furthermore and pass input text to later windows, and designed to check if the input ingredients could cause possible illness or allergy. The program requires to import index.py, nutrition\_api.py and “FoodDataFinalClean.csv”.

- nutrition\_api.py: This file is included in target.zip, I didn’t delete it in python\_ui.zip, please check and delete this one

- nutrition\_main.py: This file is included in target.zip, I didn’t delete it in python\_ui.zip, please check and delete this one

- python\_ui.py: This program is used to set up UI of the third window “Recipes and REstaurants Recommendation”, especially for setting up two tableviews. The program requires two input files named “restaurant\_clean.csv” and “recipe\_clean.csv”. The program also imports UI\_new.py and recipe\_match\_instruction.py

- recipe\_match\_instruction.py: This program is used to read input ingredients and find all recipes that contain all these ingredients from recipe\_clean.csv, then match instructions and prices to these recipes, write the result into “table.csv”. The program imports ingredient\_to\_recipe.py

- UI\_new.py: This program is used to read input restaurant data and find restaurants that have recipe / menu name contains user input ingredients. It will also show restaurant other information like address, price, tags and open time(based on searching date)

**2) png files**

- woman\_posture.png: a woman picture used for UI

- man\_posture.png: a man picture used for UI

**3) csv files**

- table.csv: outcome table of recipe\_match\_instruction.py

- FoodDataFinalClean.csv: a copy data from food\_conflicts

- recipe\_clean.csv: a copy data from Recipe

- recipe\_instruction.csv: a copy data from Recipe

- recipe\_match: the copy of final generation of match\_recipe\_clean.py in MatchRecipe folder

- restaurant\_clean: a copy data from Yelp

**4) ui file**

- nutrition\_api.ui: automatic generated file of UI

**Hope this instruction helps!**

**Our contact information**

Name | Andrew ID | Email

Xueting Meng | xuetingm | xuetingm@andrew.cmu.edu

Ruidi Chang | ruidic | ruidic@andrew.cmu.edu

Simon Zhang | hanzez | hanzez@andrew.cmu.edu

Renjie Zhong | renjiez | renjiez@andrew.cmu.edu

Junfeng Lin | junfengl | junfengl@andrew.cmu.edu

Chenxu Wang | chenxuw | chenxuw@andrew.cmu.edu