# Explaining Scraper Workflow

## Motivation

In deciding which business category to choose, we read on the Portland overview page saying that the city is famous for its gastronomy, and so the best choice goes to the "restaurants" business. Specifically, the information intended to be scraped can be described in two parts:

### Business Information

With regards to the business attributes, we set out to examine the relationship between the overall average rating and the number of reviews written, the number of photos displayed, whether the "Highlights" section has shown on the business page. That is, considering restaurants within a specific price range, whether an increase in the number of reviews, the number of photos, and paying the "Highlights" service will help boost the rating.

### Review Information

With regards to the review attributes, we set out to examine the relationship between the reviewers' rating and their number of friends, number of reviews written, number of photos taken, length of the review text. That is, whether an increase in the specific independent variable, holding all else equal, is associated with an increase in the rating.

## Scraper Workflow

### Packages Import

The scraper, as usual, starts with the standard imports. Note that, instead of using "requests" to get the HTML document, we apply a client API key to avoid getting blocked from Yelp.

### Mother List: Businesses

- The second part of the scraper loops over multiple search results pages (30 results per page * 10 pages). For each iteration, we insert a piece of "if-else" statement to have the correct URL, use the get method to acquire the HTML document, and extract the desired information. The nested loop is to, within the main page, select each business name and business link and append them to the corresponding list.
- The mother URL list and the associated business are then put into a DataFrame (`object_df`) structure, which contains 300 rows (30 results per page * 10 pages) x 2 columns. However, we notice that there are some duplicated values (12), so we take an extra step to drop duplicates and end up with a total of 288 (300 - 12) observations.

**Sub List: Business Review Information**

Next, we dedicate the chunk of code to defining the `crawl_review_page` function to extract several reviews information in one go (`reviews_info`), including reviewer's name (`reviewer_name`), reviewer's location (`reviewer_location`), reviewer's individual rating and rating date (`reviewer_ratings, reviewer_date`), reviewer's number of friends (`reviewer_friends`), reviewer's number of photos (`reviewer_photos`), reviewer's number of written reviews (`reviewer_reviews`), and reviewer's review texts (`reviewer_text`). As expected from the function output, it returns a list of tuples, with each tuple containing one reviewer's 8 pieces of information.

**Put Together**

The next chunk of code puts together all the essential elements:

- First block (One-To-one mapping): For each business, we extract its overall rating (`overall_ratings`), cuisine styles (`styles`), price range (`price`), the number of reviews (`num_reviews`), the number of photos (`num_photos`), and whether it pays Yelp to have highlights displayed (`high`).
- Second block (One-To-Many mapping): For each business (20 reviews per page * 3 pages), we again insert a piece of "if-else" statement to have the correct URL, and then retrieve the reviews information defined in the `crawl_review_page` function.

## Sidenote: Programming Attempts + File Versions

### Tree Navigation

In order to successfully scrape the reviews information, we look up the BeautifulSoup documentation and seek out the methods to navigate the HTML structure.

### Error Handling

The "if-else" statements included are meant to handle the missing values, so that the whole scraper can run smoothly without spitting out errors somewhere in the middle.

### File Versions

The file, `Review.csv`, has two versions. The previous version is stored at the same run with `Business.csv` and `Rating.csv`. However, we later found out that we hadn't included the rating dates, and so instead of running the whole scraper again, we ran only the review part to preserve some time (`Review_Updated.csv`). For consistency, we use the previous version to make data analysis, so that it aligns with the business order and rating information.