Group 5: Pied Pipers
Claraestela Torres Rivero
Fernando Granados Ruiz
Jessica Moo Young
Rosa Johnson

Final Project Documentation - Intermediate Python

The purpose of this document is to provide a description of the major functions and packages used in `mood menu_graphic.py`, our main Python application program. Also please note that substantial comments have been added throughout the code and each of the functions to aid in the documentation efforts and explain what we've done.

The `mood menu_graphic.py` program imports the following Python packages:

`tkinter`, the standard graphical user interface (GUI) library. We used the `*` import command, as well as the `messagebox` widget to display a warning message box and the `ttk` widget to configure a scroll bar for content display.

`os.path`, the file processing and directories library. We used the `exists()` method to check if a specified path exists.

The `pandas`, `numpy`, `requests`, `json`, `matplotlib`, and `BeautifulSoup` libraries were introduced and used in class prior to being used for this project.

Additionally, the application program imports the following programs written by our team:

`Yelp_Scraping`, this program will scrape the top 10 restaurants from Yelp based on a cuisine and retrieve them alongside their ratings. Two functions are encompassed in this program: `rating()` and `top_10_rest()`. The function `rating()` receives the name of a restaurant and then scrapes the restaurant rating, given on a scale of one to five stars, from the Yelp webpage. The `top_10_rest()` function searches the top ten restaurants in Pittsburgh based on Yelp's restaurant categories, such as Japanese or Mexican; it then constructs a dataframe composed of restaurant and rating pairs that is sent back to the main program.

On certain occasions the YELP website will suspect that we are scrapping their site and attempt to block the IP with a captcha challenge. Loading the following URL on a browser (https://www.yelp.com/search?find_desc=Indian&find_loc=Pittsburgh%2C+PA) and resolving the captcha prompt will fix the issue. While this is not a suitable long term solution for a production application, it is enough for the scope and purposes of this project.

`Recipe` defines the function `recipe()` to load data from the 'Food Ingredients and Recipe Dataset with Image Name Mapping.csv' file and remove NaN values. We decided to completely remove empty rows since they had no value.

`imdb`, which defines the function `imdb()` to grab and format data from the IMDB API and convert it to json. The `imdb()` function searches for movies based on IMDB genres, such as comedies or action movies. We had to register and obtain a key from IMDB to use it's API, our key is: **k_wpnry5q2.**

`combo_generator`, which defines the function `generate_combo()` to pick one row at random from each of the DataFrames corresponding to Yelp restaurants, recipes, and IMDB movies and produce a single table containing the three-item combination. This final dataframe also stores the mood and the date it was created in order to display this information back to the user in the saved_combos() function.

In the `mood menu_graphic.py` main program, we've also defined the following functions to provide the main functionality of the application and to navigate the GUI:

`save()`, which saves code corresponding to 3-item combinations to a CSV file named: combinationSaved.csv that will be later used to display the saved combinations to the user.

`saved_combos()`, which creates a GUI frame containing previously saved combinations and displays them to the user in a carousel like format. The function also offers the option to generate a new combination.

`build_combo_display()`, which retrieves the information from the recommendation dataframe and displays it in a nicely formatted GUI frame. Depending on the source screen that called for this function it will display additional buttons at the bottom of the screen.

`display_combo()`, which displays the 3-item combination in a readable format. It relies on "build_combo_display()" to generate the output and inserts it into a frame containing a scroll bar

`process_selection()`, which ensures that a mood has been selected and if so displays feedback to the user that his/her request is being processed.

`get_recommendation()`, which links mood selection to combination items based on mood-related food and movie categories. Mood-food-movie relationships were determined arbitrarily. For example, the "flirty" mood has been associated with Italian restaurants and romance movies based on our team's whim. This function will call on all other programs written by our team: Yelp_Scrapping, combo_generator, imdb and Recipe to retrieve the appropriate information from each of the sources and generate a random recommendation to the user that will be displayed later in other functions

`new_combo()`, which allows the user to select a new mood and click on a button to get a recommendation (which is then created by other functions). This function is the initial menu option called by the main program to start the GUI application flow and functionality.