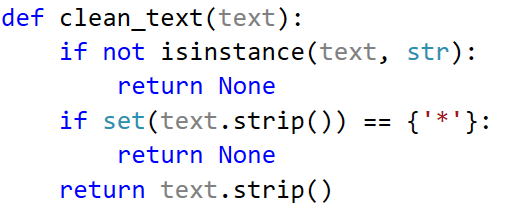
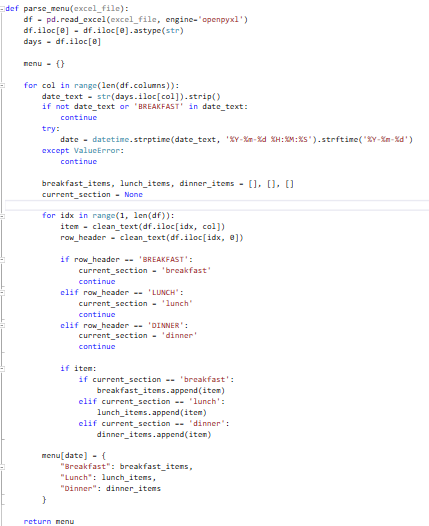


Here we are importing three libraries- pandas, json and datetime. Datetime is imported to convert date strings into a proper format (YYYY-MM-DD).



* This function ensures that: 1) Non-string values (like NaN) are removed. 2) Rows containing only \*\*\*\*\*\*\*\* are ignored. 3) Extra spaces are removed.
* Firstly, it checks if the input is a string. If it’s not a string, the function will return none. This is done so that the function does not operate on non-string values.
* Secondly, we have used the strip function in this code. So strip function basically removes any leading or trailing whitespace from the text.
* After stripping, the set(text.strip()) converts the string into a set of unique characters. If the set contains only the asterisk ({'\*'}), the text is considered invalid so the function returns None. This ensures that strings consisting only of asterisks are excluded.
* If the previous checks pass, the function will return the text after removing any leading or trailing spaces using the strip() method.



* The first line reads the CSV file into a Pandas DataFrame df. This makes it easy to manipulate the data in tabular form.
* **df =pd.read\_excel(excel\_file,engine='openpyxl')** 1)This line reads an Excel file into a Pandas DataFrame (df) 2)engine='openpyxl' tells Pandas to use the OpenPyXL library to handle .xlsx files (since

* df.iloc[0] selects the first row (index 0) of the DataFrame,  containing the column headers or days for each day.
* The menu dictionary will hold the final result, where each key is a date.
* for col in range(len(df.columns)): 1) The loop iterates through each column.

2) date\_text = dates.iloc[col] retrieves the text for each date from the first row.

* **for col in range(len(df.columns)): date\_text = str(days.iloc[col]).strip() if not date\_text or 'BREAKFAST' in date\_text: continue try: date = datetime.strptime(date\_text, '%Y-%m-%d %H:%M:%S').strftime('%Y-%m-%d') except ValueError: continue** 1)We first loop through each column in the dataframe. 2) We then ensure taht it’s a string and apply the strip function on it to remove any extra spaces. 3) If the value in date\_text is not a string or contains the word BREAKFAST(because only breakfast is in the first column), the loop will skip to the next column. This prevents processing columns that don't represent valid dates or sections.
* This code attempts to convert date\_text from the format YYYY-MM-DD HH:MM:S into YYYY-MM-DD. If the conversion fails due to an invalid format, the except ValueError block ensures the loop skips that entry without breaking the program.
* Three lists are initialized- breakfast\_items, lunch\_items, dinner\_items and a variable- current\_section to track the current meal section being processed.
* for idx in range(1, len(df)):

            item = clean\_text(df.iloc[idx, col])

            row\_header = clean\_text(df.iloc[idx, 0])

            if row\_header == 'BREAKFAST':

                current\_section = 'breakfast'

                continue

            elif row\_header == 'LUNCH':

                current\_section = 'lunch'

                continue

            elif row\_header == 'DINNER':

                current\_section = 'dinner'

                continue

            if item:

                if current\_section == 'breakfast':

                    breakfast\_items.append(item)

                elif current\_section == 'lunch':

                    lunch\_items.append(item)

                elif current\_section == 'dinner':

                    dinner\_items.append(item)

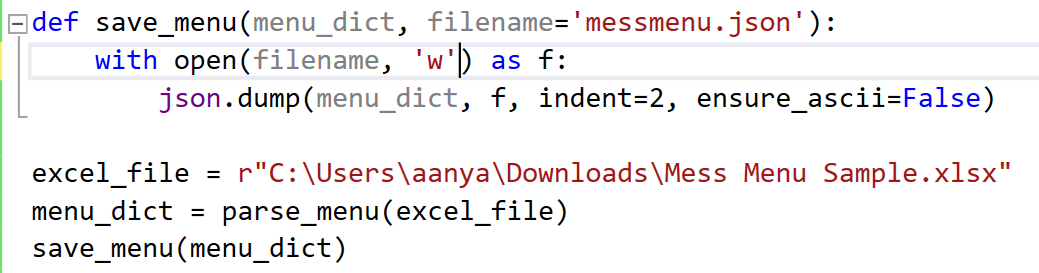
Explanation- 1)The loop iterates through each row in the DataFrame (excluding the first row of days)

2)item = clean\_text(df.iloc[idx, col]) retrieves the menu item for that row and column, cleaning it using the clean\_text function defined earlier. The cleaned text (if valid) will be stored in item.

3)row\_header = clean\_text(df.iloc[idx, 0]) retrieves the values like BREAKFAST, LUNCH, or DINNER.

4)We then append the menu items accordingly.

* Once all columns (dates) are processed, the function returns the menu dictionary, which contains structured data for each date with meal sections (Breakfast, Lunch, Dinner).



* save\_menu(menu\_dict, filename='messmenu.json'): This function saves the parsed menu data into a JSON file with the filename saved as messmenu.json.
* with open(filename, 'w') as f:This function opens the file in write mode.

json.dump(menu\_dict, f, indent=2, ensure\_ascii=False

1)Converts menu\_dict to a JSON-formatted string and writes it to the file.2)indent=2 ensures the JSON file is properly formatted with indentation (for readability).3)ensure\_ascii=False allows Unicode characters to be stored correctly instead of being converted to escape sequences.

* csv\_file = r"C:\Users\aanya\Downloads\Mess Menu Sample.xlsx"- This stores the file path of the CSV file to be processed.
* menu\_dict = parse\_menu(csv\_file): Calls the previously defined parse\_menu() function to process the csv file.