Optimizing a weekly diet plan

Fannisa Fahmi (1006182) Tee Yue Wan (1006146)

Tong Zhen Chung (1006097) Vivek Sebastian Thomas (1006016)

April 23, 2023

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

1 Problem Statement

A nutritionist often has to create a healthy diet plan for a client. The client, in particular, has a limited budget but needs to ingest a certain amount of nutrients every day. The diet plan will take into account specific requirements of the client, such as weight maintenance or weight loss, as well as any other medical conditions, such as heart disease or diabetes.

The nutritionist must figure out the ideal quantity of each food to include in the diet plan in order to address this issue. To construct a balanced diet that satisfies the client's needs, the nutritionist must combine the available food products as efficiently as possible to deliver the required nutrients while minimising the cost of the diet.

2 Constraints

The constraints for each nutrient may vary based on individual factors, but for this problem, we will be considering daily intake of **calories, protein, carbohydrates, saturated fats, sodium intake cholesterol and sugar**.

$$m_i$$

is a binary variable representing the serving amount of food selected for each particular food, out of a data set containing 551 accessible and readily available foods with varying nutrition specifications.

$$m_i = \begin{cases} 1, & \text{for 1 serving (food is selected)} \\ 0, & \text{for 0 servings(food is not selected)} \end{cases}$$

2.1

As on average, the daily intake of meals for an individual is 3 meals per day. Hence we begin with the following constraint.

$$\sum_{i=1}^{n} m_i \geq 3$$

2.2

The recommended maximum daily **calorie** intake depends on the individual's weight, exercise level, and other factors. This can be calculated using a BMI calculator.

$$\sum_{i=1}^{n} ca_i m_i \leq RecommendedCalorieIntake$$

On the other hand, base on research, the recommended minimum daily calorie intake is 1200 kcal.

$$\sum_{i=1}^{n} ca_i m_i \geq 1200$$

2.3

The recommended daily intake of **carbohydrates** should be between 45 % to 65 % of the total daily calorie intake in grams. 1 Kcal = 0.129595 grams

$$\sum_{i=1}^{n} cb_i m_i \leq 0.65 RecommendedCalorieIntakeinGrams$$

$$\sum_{i=1}^{n} cb_i m_i \geq 0.45 RecommendedCalorieIntakeinGrams$$

2.4

For **saturated fats**, men should limit their intake to no more than 30 grams per day, while women should limit their intake to no more than 20 grams per day. Sodium intake will be expressed in grams per day

$$w_i = \begin{cases} 1, \text{ if woman } i \\ 0, \text{ if man } i \ . \end{cases} \qquad w_i \in \{0,1\} \quad \forall i = 1, \ldots, n$$

$$\sum_{i=1}^{n} s_i m_i (1 - w_i) \leq 30$$

$$\sum_{i=1}^{n} s_i m_i w_i \leq 20$$

2.5

For **cholesterol**, individuals with risk factors for heart disease should limit their intake to no more than 0.2g per day. Those without risk factors should aim to consume no more than 0.3g per day.

$$f_i = \begin{cases} 1, \text{ for risk of heart disease } i \\ 0, \text{ otherwise } i \ . \end{cases} \qquad f_i \in \{0,1\} \quad \forall i = 1, \ldots,$$

$$\sum_{i=1}^{n} ch_i m_i (1 - f_i) \leq 0.3$$

$$\sum_{i=1}^{n} ch_i m_i f_i \leq 0.2$$

2.6

As a general guideline, the recommended daily intake of **protein** is expressed in grams per kilogram of body weight. For healthy individuals, protein intake is 0.8 grams of protein per kg and for individuals trying to lose weight protein intake is equal to 0.075 of the recommended day calorie intake

$$z_i = \begin{cases} 1, \text{ for individuals trying to lose weight } i \\ 0, \text{ otherwise } i \ . \end{cases} \qquad z_i \in \{0,1\} \quad \forall i = 1, \ldots, n$$

$$\sum_{i=1}^{n} pn_i m_i (1 - z_i) \leq 0.8 Weight$$

$$\sum_{i=1}^{n} pn_i m_i z_i \leq 0.075 RecommendedCalorieIntakeinGrams$$

2.7

For **sodium**, individuals with high blood pressure should limit their intake to no more than 1.5 grams per day. Those without risk factors should aim to consume no more than 2.3 grams per day.

$$hb_i = \begin{cases} 1, \text{ for individuals with high blood pressure } i \\ 0, \text{ otherwise } i \ . \end{cases} \qquad hb_i \in \{0,1\} \quad \forall i = 1, \ldots, n$$

$$\sum_{i=1}^{n} na_i m_i (1 - hb_i) \leq 2.3$$

$$\sum_{i=1}^{n} na_i m_i hb_i \leq 1.5$$

2.8

For **sugar**, individuals with diabetes should limit their intake to no more than 25 grams per day. Those without diabetes should aim to consume no more than 50 grams per day.

$$d_i = \begin{cases} 1, & \text{for individuals with diabetes } i \\ 0, & \text{otherwise } i \text{ is not aired .} \end{cases} \qquad d_i \in \{0,1\} \quad \forall i = 1, \ldots, n$$

$$\sum_{i=1}^{n} gl_i(1-d)_i \leq 50$$

$$\sum_{i=1}^{n} gl_i d_i \leq 25$$

3 Objective function

We want to **minimize total cost** , which is the amount of money needed to fund the healthy diet, i.e.

$$\sum_{i=1}^{n} p_i m_i$$

Then our MILP is

$$\text{minimize} \sum_{i=1}^{n} p_i m_i$$

subject to

$$m_i = \begin{cases} 1, & \text{for 1 serving (food is selected)} \\ 0, & \text{for 0 servings(food is not selected)} \end{cases}$$

$$\sum_{i=1}^{n} m_i \geq 3$$

$$\sum_{i=1}^{n} ca_i m_i \leq RecommendedCalorieIntake$$

$$\sum_{i=1}^{n} ca_i m_i \geq 1200$$

$$\sum_{i=1}^{n} cb_i m_i \leq 0.65 RecommendedCalorieIntakeinGrams$$

$$\sum_{i=1}^{n} cb_i m_i \geq 0.45 RecommendedCalorieIntakeinGrams$$

$$w_i = \begin{cases} 1, & \text{if woman } i \\ 0, & \text{if man } i \text{ .} \end{cases} \qquad w_i \in \{0,1\} \quad \forall i = 1, \ldots, n$$

$$\sum_{i=1}^{n} s_i m_i (1 - w_i) \leq 30$$

$$\sum_{i=1}^{n} s_i m_i w_i \leq 20$$

$$f_i = \begin{cases} 1, \text{ for risk of heart disease } i \\ 0, \text{ otherwise } i \ . \end{cases} \qquad f_i \in \{0,1\} \quad \forall i = 1, \ldots,$$

$$\sum_{i=1}^{n} ch_i m_i (1 - f_i) \leq 0.3$$

$$\sum_{i=1}^{n} ch_i m_i f_i \leq 0.2$$

$$z_i = \begin{cases} 1, \text{ for individuals trying to lose weight } i \\ 0, \text{ otherwise } i \ . \end{cases} \qquad z_i \in \{0,1\} \quad \forall i = 1, \ldots, n$$

$$\sum_{i=1}^{n} pn_i m_i (1 - z_i) \leq 0.8 Weight$$

$$\sum_{i=1}^{n} pn_i m_i z_i \leq 0.075 RecommendedCalorieIntakeinGrams$$

$$hb_i = \begin{cases} 1, \text{ for individuals with high blood pressure } i \\ 0, \text{ otherwise } i \ . \end{cases} \qquad hb_i \in \{0,1\} \quad \forall i = 1, \ldots, n$$

$$\sum_{i=1}^{n} na_i m_i (1 - hb_i) \leq 2.3$$

$$\sum_{i=1}^{n} na_i m_i hb_i \leq 1.5$$

$$d_i = \begin{cases} 1, \text{ for individuals with diabetes } i \\ 0, \text{ otherwise } i \text{ is not aired } . \end{cases} \qquad d_i \in \{0,1\} \quad \forall i = 1, \ldots, n$$

$$\sum_{i=1}^{n} gl_i (1 - d)_i \leq 50g$$

$$\sum_{i=1}^{n} gl_i d_i \leq 25$$

4 Assumptions

The following assumptions are made:

1)The datasets for nutritional values of the foods that are offered are accurate

2)The prices of the costs of the accessible foods are accurate as of 2021.

3)The client's dietary requirements and restrictions are accurately specified.

4)The customer does not have any other dietary restrictions or food allergies.

5)Main focus of the problem will be to minimise the cost of diet regardless of taste or variety.

6)The average individual consumes at least 3 meals/day

5 Correctness of Model

Upon decreasing Emily's maximum daily calorie intake, the cost of her diet increases. This is in line with findings from ers.usda.gov, which suggest that healthier foods typically come with a higher price tag.

Refer to Figure 1, Figure 2 in Appendix

6 Solving

To test our model, we've used the profile of a female office worker Emily, who is 25 years old, weighs 65kg, and is 162cm tall. She engages in light exercise daily and aims to lose 5 % to 10 % of her body weight over 6 to 12 months by reducing caloric intake. Using a Calorie Calculator, her recommended daily caloric intake is up to 1800 kcal and she has a risk of diabetes. Based on Emily's nutritional requirements, we recommend Emily to limit her protein intake to 42.9g, carbohydrate intake to be in the range of 760.95g - 1099.15g, and a saturated fat intake of less than 20g per day. Her sodium intake should be less than 2.3g per day, and her cholesterol intake should not exceed 300 milligrams. Taking into account her risk of diabetes, she should not exceed 30g of sugar intake per day. Therefore, we came up with a model to determine her diet for the week.

7 Results of Optimization Model

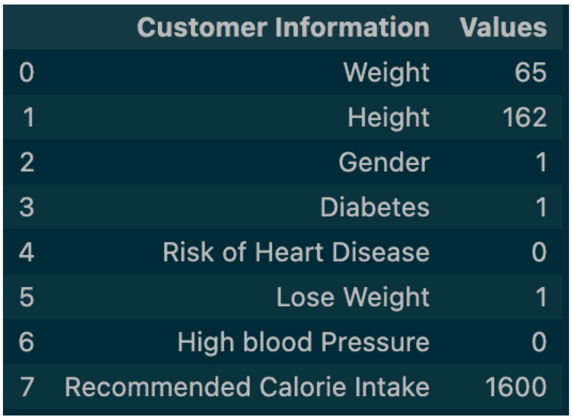
The full code can be found in appendix. **Refer to Figure 3, Figure 4**

8 Interpretation of the results

According to our model, the average predicted cost for a meal in Singapore is **$3.73**. The food dataset is dated back in 2021 and considering inflation, and base on a recent research about food prices in Singapore which showed that average cost of meal is about $5, $3.73 per meal is reasonable cost for a meal. Based on the results, it appears that the majority of the foods recommended for consumption are unsalted and primarily made from wheat, with cooking methods that are generally considered healthier, such as boiling rather than frying. As a result, it can be concluded that **our model is reasonably reliable.**

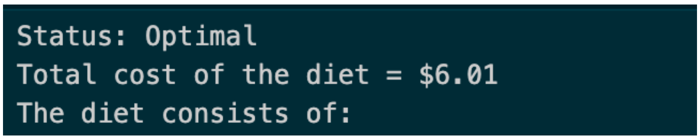
9 Appendix

9.1 Python Code



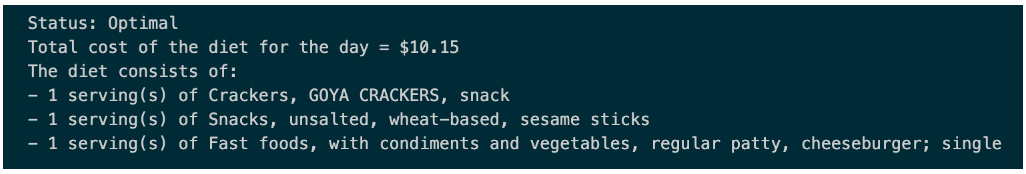
	Customer Information	Values
0	Weight	65
1	Height	162
2	Gender	1
3	Diabetes	1
4	Risk of Heart Disease	0
5	Lose Weight	1
6	High blood Pressure	0
7	Recommended Calorie Intake	1600

Figure 1: Illustrating correctness of model



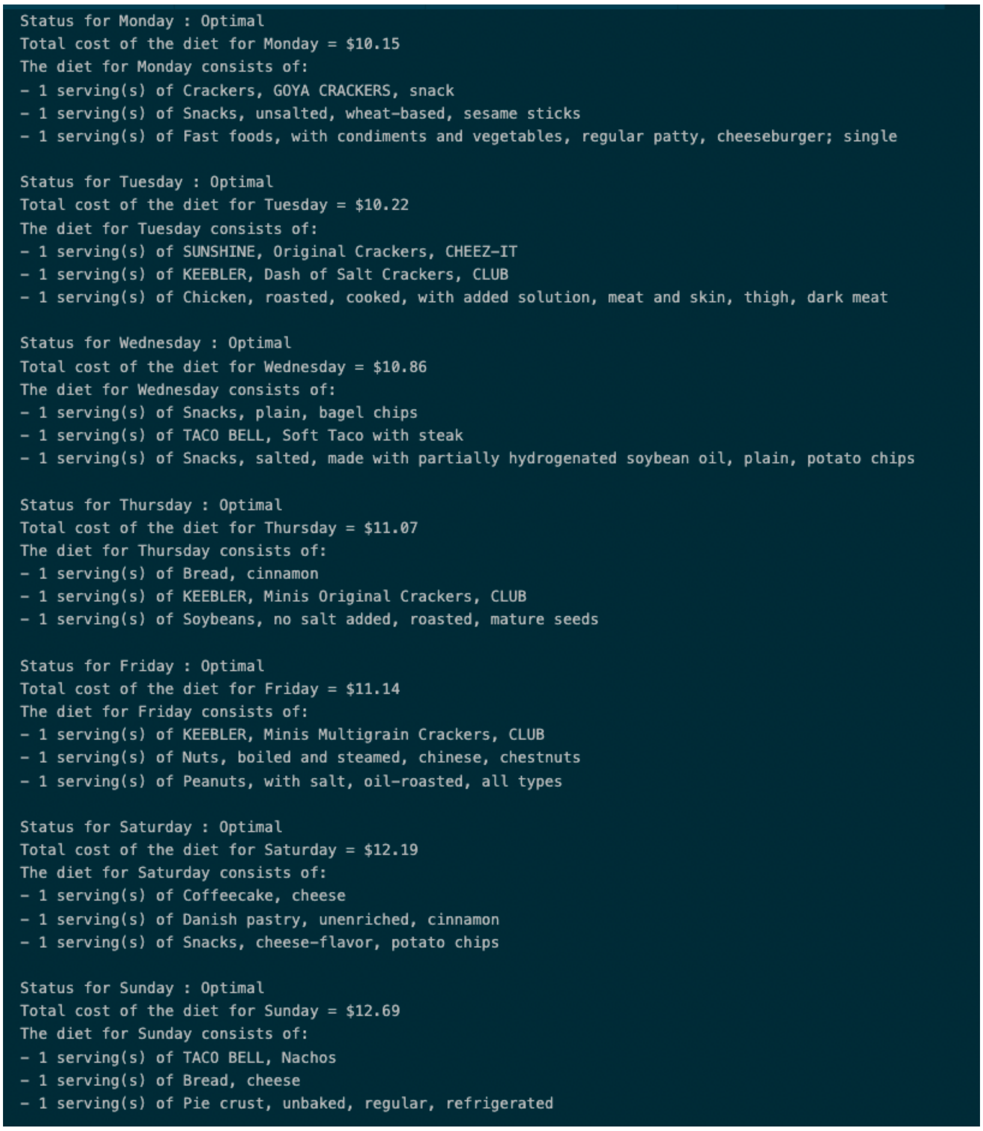
```
Status: Optimal
Total cost of the diet = $6.01
The diet consists of:
```

Figure 2: Illustrating correctness of model



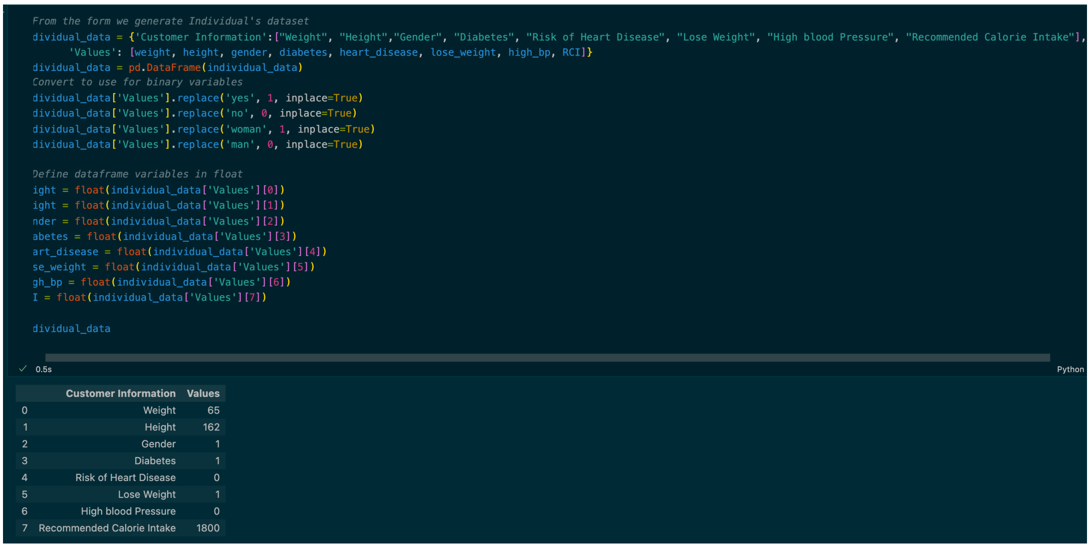
```
Status: Optimal
Total cost of the diet for the day = $10.15
The diet consists of:
- 1 serving(s) of Crackers, GOYA CRACKERS, snack
- 1 serving(s) of Snacks, unsalted, wheat-based, sesame sticks
- 1 serving(s) of Fast foods, with condiments and vegetables, regular patty, cheeseburger; single
```

Figure 3: Emily's Daily Diet plan(one day)



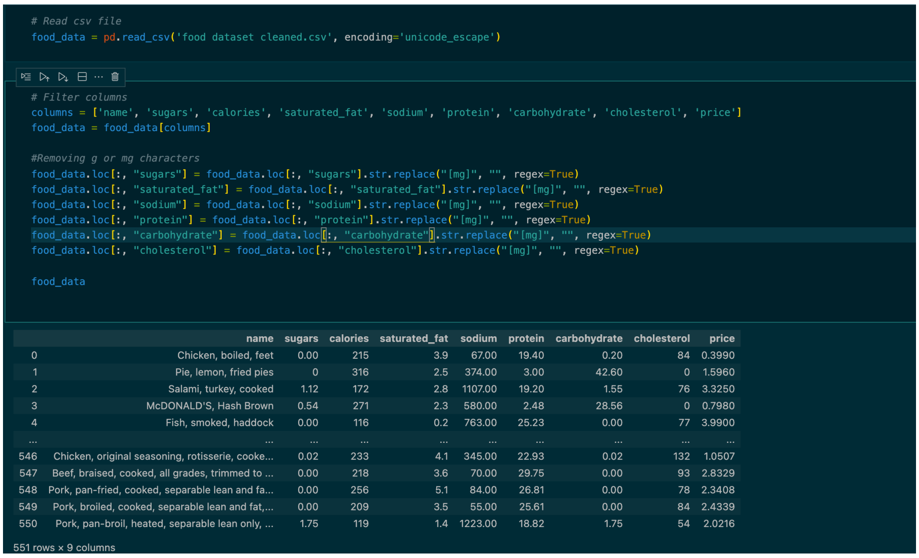
```
Status for Monday : Optimal
Total cost of the diet for Monday = $10.15
The diet for Monday consists of:
- 1 serving(s) of Crackers, GOYA CRACKERS, snack
- 1 serving(s) of Snacks, unsalted, wheat-based, sesame sticks
- 1 serving(s) of Fast foods, with condiments and vegetables, regular patty, cheeseburger; single

Status for Tuesday : Optimal
Total cost of the diet for Tuesday = $10.22
The diet for Tuesday consists of:
- 1 serving(s) of SUNSHINE, Original Crackers, CHEEZ-IT
- 1 serving(s) of KEEBLER, Dash of Salt Crackers, CLUB
- 1 serving(s) of Chicken, roasted, cooked, with added solution, meat and skin, thigh, dark meat

Status for Wednesday : Optimal
Total cost of the diet for Wednesday = $10.86
The diet for Wednesday consists of:
- 1 serving(s) of Snacks, plain, bagel chips
- 1 serving(s) of TACO BELL, Soft Taco with steak
- 1 serving(s) of Snacks, salted, made with partially hydrogenated soybean oil, plain, potato chips

Status for Thursday : Optimal
Total cost of the diet for Thursday = $11.07
The diet for Thursday consists of:
- 1 serving(s) of Bread, cinnamon
- 1 serving(s) of KEEBLER, Minis Original Crackers, CLUB
- 1 serving(s) of Soybeans, no salt added, roasted, mature seeds

Status for Friday : Optimal
Total cost of the diet for Friday = $11.14
The diet for Friday consists of:
- 1 serving(s) of KEEBLER, Minis Multigrain Crackers, CLUB
- 1 serving(s) of Nuts, boiled and steamed, chinese, chestnuts
- 1 serving(s) of Peanuts, with salt, oil-roasted, all types

Status for Saturday : Optimal
Total cost of the diet for Saturday = $12.19
The diet for Saturday consists of:
- 1 serving(s) of Coffeecake, cheese
- 1 serving(s) of Danish pastry, unenriched, cinnamon
- 1 serving(s) of Snacks, cheese-flavor, potato chips

Status for Sunday : Optimal
Total cost of the diet for Sunday = $12.69
The diet for Sunday consists of:
- 1 serving(s) of TACO BELL, Nachos
- 1 serving(s) of Bread, cheese
- 1 serving(s) of Pie crust, unbaked, regular, refrigerated
```

Figure 4: Emily's Weekly Diet plan(one week)



```
From the form we generate Individual's dataset
dividual_data = {'Customer Information':["Weight", "Height","Gender", "Diabetes", "Risk of Heart Disease", "Lose Weight", "High blood Pressure", "Recommended Calorie Intake"],
      'Values': [weight, height, gender, diabetes, heart_disease, lose_weight, high_bp, RCI]}
dividual_data = pd.DataFrame(individual_data)
Convert to use for binary variables
dividual_data['Values'].replace('yes', 1, inplace=True)
dividual_data['Values'].replace('no', 0, inplace=True)
dividual_data['Values'].replace('woman', 1, inplace=True)
dividual_data['Values'].replace('man', 0, inplace=True)

Define dataframe variables in float
ight = float(individual_data['Values'][0])
ight = float(individual_data['Values'][1])
nder = float(individual_data['Values'][2])
abetes = float(individual_data['Values'][3])
art_disease = float(individual_data['Values'][4])
se_weight = float(individual_data['Values'][5])
gh_bp = float(individual_data['Values'][6])
I = float(individual_data['Values'][7])

dividual_data
```

	Customer Information	Values
0	Weight	65
1	Height	162
2	Gender	1
3	Diabetes	1
4	Risk of Heart Disease	0
5	Lose Weight	1
6	High blood Pressure	0
7	Recommended Calorie Intake	1800

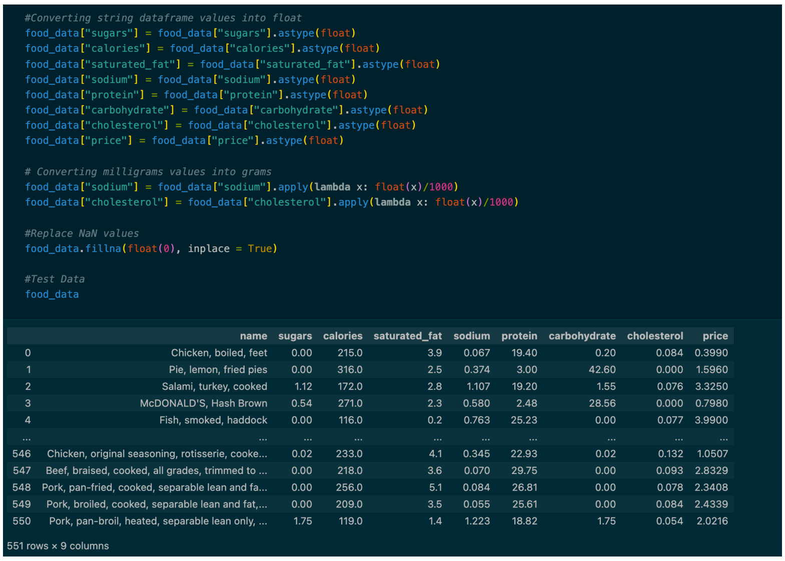


```
# Read csv file
food_data = pd.read_csv('food dataset cleaned.csv', encoding='unicode_escape')
```

```
# Filter columns
columns = ['name', 'sugars', 'calories', 'saturated_fat', 'sodium', 'protein', 'carbohydrate', 'cholesterol', 'price']
food_data = food_data[columns]

#Removing g or mg characters
food_data.loc[:, "sugars"] = food_data.loc[:, "sugars"].str.replace("[mg]", "", regex=True)
food_data.loc[:, "saturated_fat"] = food_data.loc[:, "saturated_fat"].str.replace("[mg]", "", regex=True)
food_data.loc[:, "sodium"] = food_data.loc[:, "sodium"].str.replace("[mg]", "", regex=True)
food_data.loc[:, "protein"] = food_data.loc[:, "protein"].str.replace("[mg]", "", regex=True)
food_data.loc[:, "carbohydrate"] = food_data.loc[:, "carbohydrate"].str.replace("[mg]", "", regex=True)
food_data.loc[:, "cholesterol"] = food_data.loc[:, "cholesterol"].str.replace("[mg]", "", regex=True)

food_data
```

	name	sugars	calories	saturated_fat	sodium	protein	carbohydrate	cholesterol	price
0	Chicken, boiled, feet	0.00	215	3.9	67.00	19.40	0.20	84	0.3990
1	Pie, lemon, fried pies	0	316	2.5	374.00	3.00	42.60	0	1.5960
2	Salami, turkey, cooked	1.12	172	2.8	1107.00	19.20	1.55	76	3.3250
3	McDONALD'S, Hash Brown	0.54	271	2.3	580.00	2.48	28.56	0	0.7980
4	Fish, smoked, haddock	0.00	116	0.2	763.00	25.23	0.00	77	3.9900
...	...	...	...	...	...	...	...	...	...
546	Chicken, original seasoning, rotisserie, cooke...	0.02	233	4.1	345.00	22.93	0.02	132	1.0507
547	Beef, braised, cooked, all grades, trimmed to ...	0.00	218	3.6	70.00	29.75	0.00	93	2.8329
548	Pork, pan-fried, cooked, separable lean and fa...	0.00	256	5.1	84.00	26.81	0.00	78	2.3408
549	Pork, broiled, cooked, separable lean and fat,...	0.00	209	3.5	55.00	25.61	0.00	84	2.4339
550	Pork, pan-broil, heated, separable lean only, ...	1.75	119	1.4	1223.00	18.82	1.75	54	2.0216

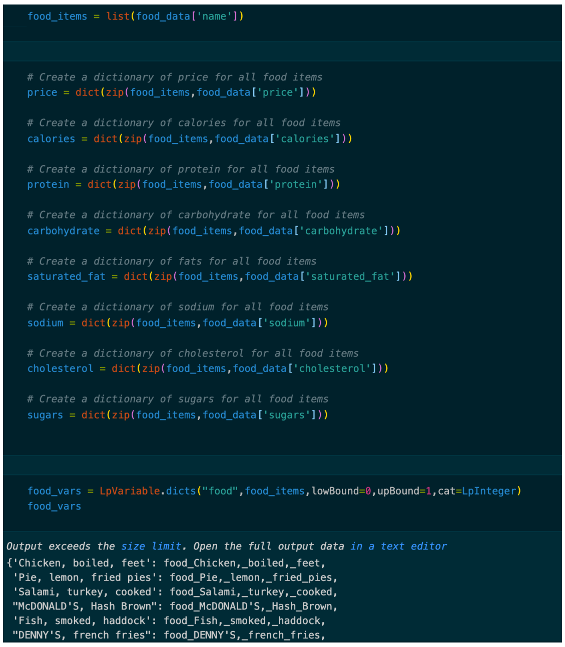
551 rows × 9 columns



```
#Converting string dataframe values into float
food_data["sugars"] = food_data["sugars"].astype(float)
food_data["calories"] = food_data["calories"].astype(float)
food_data["saturated_fat"] = food_data["saturated_fat"].astype(float)
food_data["sodium"] = food_data["sodium"].astype(float)
food_data["protein"] = food_data["protein"].astype(float)
food_data["carbohydrate"] = food_data["carbohydrate"].astype(float)
food_data["cholesterol"] = food_data["cholesterol"].astype(float)
food_data["price"] = food_data["price"].astype(float)

# Converting milligrams values into grams
food_data["sodium"] = food_data["sodium"].apply(lambda x: float(x)/1000)
food_data["cholesterol"] = food_data["cholesterol"].apply(lambda x: float(x)/1000)

#Replace NaN values
food_data.fillna(float(0), inplace = True)

#Test Data
food_data
```

	name	sugars	calories	saturated_fat	sodium	protein	carbohydrate	cholesterol	price
0	Chicken, boiled, feet	0.00	215.0	3.9	0.067	19.40	0.20	0.084	0.3990
1	Pie, lemon, fried pies	0.00	316.0	2.5	0.374	3.00	42.60	0.000	1.5960
2	Salami, turkey, cooked	1.12	172.0	2.8	1.107	19.20	1.55	0.076	3.3250
3	McDONALD'S, Hash Brown	0.54	271.0	2.3	0.580	2.48	28.56	0.000	0.7980
4	Fish, smoked, haddock	0.00	116.0	0.2	0.763	25.23	0.00	0.077	3.9900
...	...	...	...	...	...	...	...	...	...
546	Chicken, original seasoning, rotisserie, cooke...	0.02	233.0	4.1	0.345	22.93	0.02	0.132	1.0507
547	Beef, braised, cooked, all grades, trimmed to ...	0.00	218.0	3.6	0.070	29.75	0.00	0.093	2.8329
548	Pork, pan-fried, cooked, separable lean and fa...	0.00	256.0	5.1	0.084	26.81	0.00	0.078	2.3408
549	Pork, broiled, cooked, separable lean and fat,...	0.00	209.0	3.5	0.055	25.61	0.00	0.084	2.4339
550	Pork, pan-broil, heated, separable lean only, ...	1.75	119.0	1.4	1.223	18.82	1.75	0.054	2.0216

551 rows × 9 columns

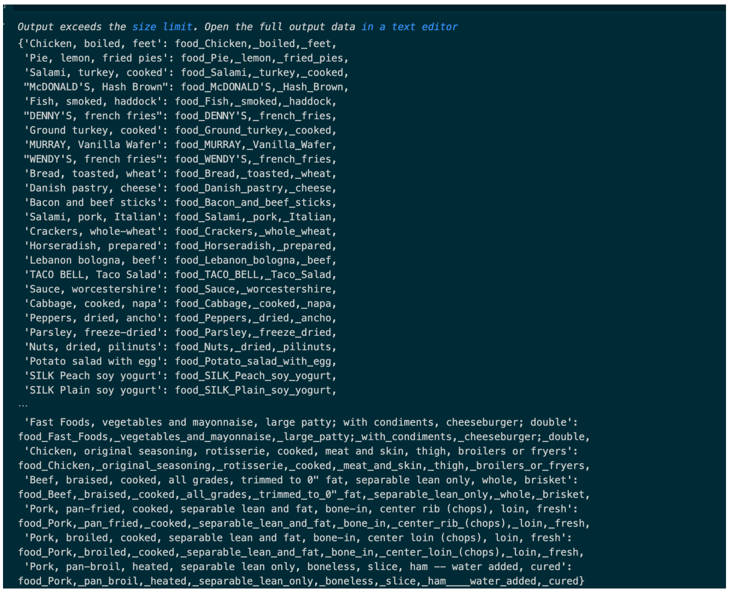


```
food_items = list(food_data['name'])
```

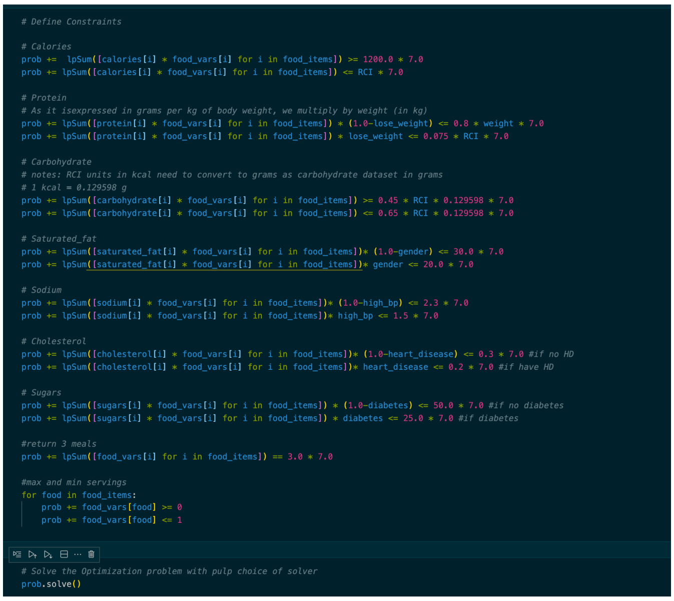
```
# Create a dictionary of price for all food items
price = dict(zip(food_items,food_data['price']))

# Create a dictionary of calories for all food items
calories = dict(zip(food_items,food_data['calories']))

# Create a dictionary of protein for all food items
protein = dict(zip(food_items,food_data['protein']))

# Create a dictionary of carbohydrate for all food items
carbohydrate = dict(zip(food_items,food_data['carbohydrate']))

# Create a dictionary of fats for all food items
saturated_fat = dict(zip(food_items,food_data['saturated_fat']))

# Create a dictionary of sodium for all food items
sodium = dict(zip(food_items,food_data['sodium']))

# Create a dictionary of cholesterol for all food items
cholesterol = dict(zip(food_items,food_data['cholesterol']))

# Create a dictionary of sugars for all food items
sugars = dict(zip(food_items,food_data['sugars']))
```

```
food_vars = LpVariable.dicts("food",food_items,lowBound=0,upBound=1,cat=LpInteger)
food_vars
```

```
Output exceeds the size limit. Open the full output data in a text editor
{'Chicken, boiled, feet': food_Chicken,_boiled,_feet,
 'Pie, lemon, fried pies': food_Pie,_lemon,_fried_pies,
 'Salami, turkey, cooked': food_Salami,_turkey,_cooked,
 "McDONALD'S, Hash Brown": food_McDONALD'S,_Hash_Brown,
 'Fish, smoked, haddock': food_Fish,_smoked,_haddock,
 "DENNY'S, french fries": food_DENNY'S,_french_fries,
```

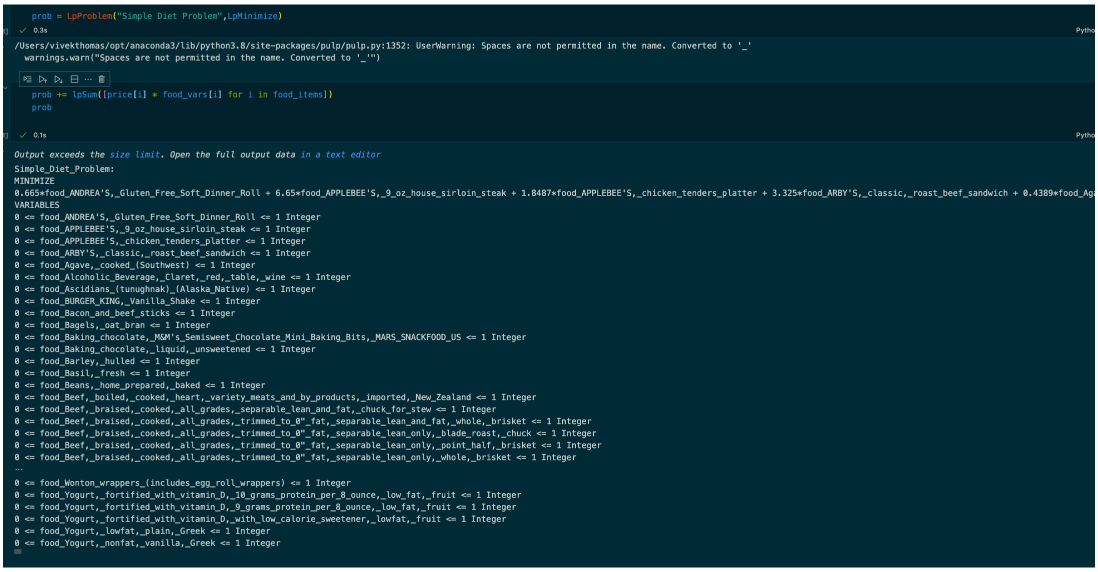


```
Output exceeds the size limit. Open the full output data in a text editor
{'Chicken, boiled, feet': food_Chicken,_boiled,_feet,
 'Pie, lemon, fried pies': food_Pie,_lemon,_fried_pies,
 'Salami, turkey, cooked': food_Salami,_turkey,_cooked,
 "McDONALD'S, Hash Brown": food_McDONALD'S,_Hash_Brown,
 'Fish, smoked, haddock': food_Fish,_smoked,_haddock,
 "DENNY'S, french fries": food_DENNY'S,_french_fries,
 'Ground turkey, cooked': food_Ground_turkey,_cooked,
 'MURRAY, Vanilla Wafer': food_MURRAY,_Vanilla_Wafer,
 "WENDY'S, french fries": food_WENDY'S,_french_fries,
 'Bread, toasted, wheat': food_Bread,_toasted,_wheat,
 'Danish pastry, cheese': food_Danish_pastry,_cheese,
 'Bacon and beef sticks': food_Bacon_and_beef_sticks,
 'Salami, pork, Italian': food_Salami,_pork,_Italian,
 'Crackers, whole-wheat': food_Crackers,_whole_wheat,
 'Horseradish, prepared': food_Horseradish,_prepared,
 'Lebanon bologna, beef': food_Lebanon_bologna,_beef,
 'TACO BELL, Taco Salad': food_TACO_BELL,_Taco_Salad,
 'Sauce, worcestershire': food_Sauce,_worcestershire,
 'Cabbage, cooked, napa': food_Cabbage,_cooked,_napa,
 'Peppers, dried, ancho': food_Peppers,_dried,_ancho,
 'Parsley, freeze-dried': food_Parsley,_freeze_dried,
 'Nuts, dried, pilinuts': food_Nuts,_dried,_pilinuts,
 'Potato salad with egg': food_Potato_salad_with_egg,
 'SILK Peach soy yogurt': food_SILK_Peach_soy_yogurt,
 'SILK Plain soy yogurt': food_SILK_Plain_soy_yogurt,
…
 'Fast Foods, vegetables and mayonnaise, large patty; with condiments, cheeseburger; double':
food_Fast_Foods,_vegetables_and_mayonnaise,_large_patty;_with_condiments,_cheeseburger;_double,
 'Chicken, original seasoning, rotisserie, cooked, meat and skin, thigh, broilers or fryers':
food_Chicken,_original_seasoning,_rotisserie,_cooked,_meat_and_skin,_thigh,_broilers_or_fryers,
 'Beef, braised, cooked, all grades, trimmed to 0" fat, separable lean only, whole, brisket':
food_Beef,_braised,_cooked,_all_grades,_trimmed_to_0"_fat,_separable_lean_only,_whole,_brisket,
 'Pork, pan-fried, cooked, separable lean and fat, bone-in, center rib (chops), loin, fresh':
food_Pork,_pan_fried,_cooked,_separable_lean_and_fat,_bone_in,_center_rib_(chops),_loin,_fresh,
 'Pork, broiled, cooked, separable lean and fat, bone-in, center loin (chops), loin, fresh':
food_Pork,_broiled,_cooked,_separable_lean_and_fat,_bone_in,_center_loin_(chops),_loin,_fresh,
 'Pork, pan-broil, heated, separable lean only, boneless, slice, ham -- water added, cured':
food_Pork,_pan_broil,_heated,_separable_lean_only,_boneless,_slice,_ham____water_added,_cured}
```



```
# Define Constraints

# Calories
prob +=  lpSum([calories[i] * food_vars[i] for i in food_items]) >= 1200.0 * 7.0
prob += lpSum([calories[i] * food_vars[i] for i in food_items]) <= RCI * 7.0

# Protein
# As it isexpressed in grams per kg of body weight, we multiply by weight (in kg)
prob += lpSum([protein[i] * food_vars[i] for i in food_items]) * (1.0-lose_weight) <= 0.8 * weight * 7.0
prob += lpSum([protein[i] * food_vars[i] for i in food_items]) * lose_weight <= 0.075 * RCI * 7.0

# Carbohydrate
# notes: RCI units in kcal need to convert to grams as carbohydrate dataset in grams
# 1 kcal = 0.129598 g
prob += lpSum([carbohydrate[i] * food_vars[i] for i in food_items]) >= 0.45 * RCI * 0.129598 * 7.0
prob += lpSum([carbohydrate[i] * food_vars[i] for i in food_items]) <= 0.65 * RCI * 0.129598 * 7.0

# Saturated_fat
prob += lpSum([saturated_fat[i] * food_vars[i] for i in food_items])* (1.0-gender) <= 30.0 * 7.0
prob += lpSum([saturated_fat[i] * food_vars[i] for i in food_items])* gender <= 20.0 * 7.0

# Sodium
prob += lpSum([sodium[i] * food_vars[i] for i in food_items])* (1.0-high_bp) <= 2.3 * 7.0
prob += lpSum([sodium[i] * food_vars[i] for i in food_items])* high_bp <= 1.5 * 7.0

# Cholesterol
prob += lpSum([cholesterol[i] * food_vars[i] for i in food_items])* (1.0-heart_disease) <= 0.3 * 7.0 #if no HD
prob += lpSum([cholesterol[i] * food_vars[i] for i in food_items])* heart_disease <= 0.2 * 7.0 #if have HD

# Sugars
prob += lpSum([sugars[i] * food_vars[i] for i in food_items]) * (1.0-diabetes) <= 50.0 * 7.0 #if no diabetes
prob += lpSum([sugars[i] * food_vars[i] for i in food_items]) * diabetes <= 25.0 * 7.0 #if diabetes

#return 3 meals
prob += lpSum([food_vars[i] for i in food_items]) == 3.0 * 7.0

#max and min servings
for food in food_items:
    prob += food_vars[food] >= 0
    prob += food_vars[food] <= 1
```

```
# Solve the Optimization problem with pulp choice of solver
prob.solve()
```



```
prob = LpProblem("Simple Diet Problem",LpMinimize)
```

✓ 0.3s

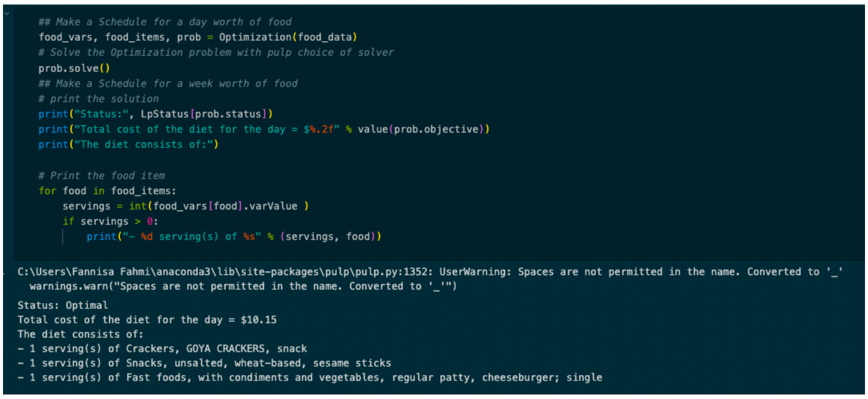
```
/Users/vivekthomas/opt/anaconda3/lib/python3.8/site-packages/pulp/pulp.py:1352: UserWarning: Spaces are not permitted in the name. Converted to '_'
  warnings.warn("Spaces are not permitted in the name. Converted to '_'")
```

```
prob += lpSum([price[i] * food_vars[i] for i in food_items])
prob
```

✓ 0.1s

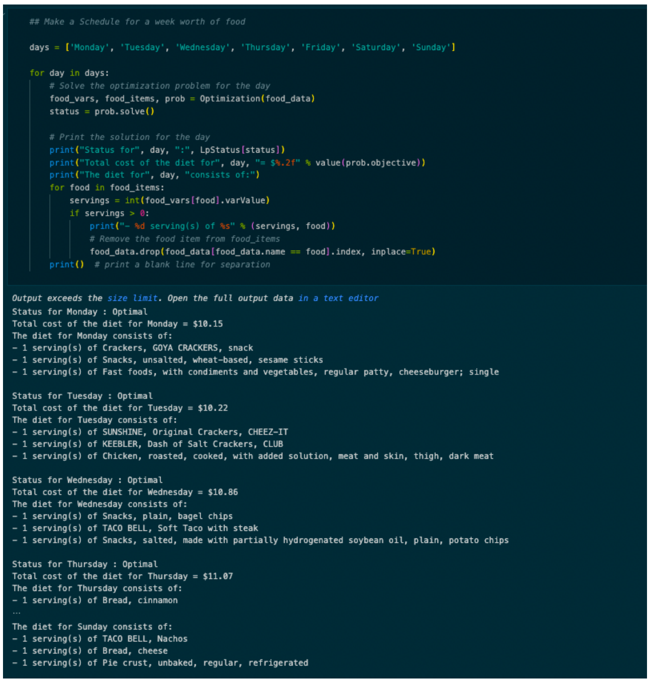
Output exceeds the size limit. Open the full output data in a text editor

```
Simple_Diet_Problem:
MINIMIZE
0.665*food_ANDREA'S,_Gluten_Free_Soft_Dinner_Roll + 6.65*food_APPLEBEE'S,_9_oz_house_sirloin_steak + 1.8487*food_APPLEBEE'S,_chicken_tenders_platter + 3.325*food_ARBY'S,_classic,_roast_beef_sandwich + 0.4389*food_Aga
VARIABLES
0 <= food_ANDREA'S,_Gluten_Free_Soft_Dinner_Roll <= 1 Integer
0 <= food_APPLEBEE'S,_9_oz_house_sirloin_steak <= 1 Integer
0 <= food_APPLEBEE'S,_chicken_tenders_platter <= 1 Integer
0 <= food_ARBY'S,_classic,_roast_beef_sandwich <= 1 Integer
0 <= food_Agave,_cooked_(Southwest) <= 1 Integer
0 <= food_Alcoholic_Beverage,_Claret,_red,_table,_wine <= 1 Integer
0 <= food_Ascidians_(tunughnak)_(Alaska_Native) <= 1 Integer
0 <= food_BURGER_KING,_Vanilla_Shake <= 1 Integer
0 <= food_Bacon_and_beef_sticks <= 1 Integer
0 <= food_Bagels,_oat_bran <= 1 Integer
0 <= food_Baking_chocolate,_M&M's_Semisweet_Chocolate_Mini_Baking_Bits,_MARS_SNACKFOOD_US <= 1 Integer
0 <= food_Baking_chocolate,_liquid,_unsweetened <= 1 Integer
0 <= food_Barley,_hulled <= 1 Integer
0 <= food_Basil,_fresh <= 1 Integer
0 <= food_Beans,_home_prepared,_baked <= 1 Integer
0 <= food_Beef,_boiled,_cooked,_heart,_variety_meats_and_by_products,_imported,_New_Zealand <= 1 Integer
0 <= food_Beef,_braised,_cooked,_all_grades,_separable_lean_and_fat,_chuck_for_stew <= 1 Integer
0 <= food_Beef,_braised,_cooked,_all_grades,_trimmed_to_0"_fat,_separable_lean_and_fat,_whole,_brisket <= 1 Integer
0 <= food_Beef,_braised,_cooked,_all_grades,_trimmed_to_0"_fat,_separable_lean_only,_blade_roast,_chuck <= 1 Integer
0 <= food_Beef,_braised,_cooked,_all_grades,_trimmed_to_0"_fat,_separable_lean_only,_point_half,_brisket <= 1 Integer
0 <= food_Beef,_braised,_cooked,_all_grades,_trimmed_to_0"_fat,_separable_lean_only,_whole,_brisket <= 1 Integer
...
0 <= food_Wonton_wrappers_(includes_egg_roll_wrappers) <= 1 Integer
0 <= food_Yogurt,_fortified_with_vitamin_D,_10_grams_protein_per_8_ounce,_low_fat,_fruit <= 1 Integer
0 <= food_Yogurt,_fortified_with_vitamin_D,_9_grams_protein_per_8_ounce,_low_fat,_fruit <= 1 Integer
0 <= food_Yogurt,_fortified_with_vitamin_D,_with_low_calorie_sweetener,_lowfat,_fruit <= 1 Integer
0 <= food_Yogurt,_lowfat,_plain,_Greek <= 1 Integer
0 <= food_Yogurt,_nonfat,_vanilla,_Greek <= 1 Integer
```



```
## Make a Schedule for a day worth of food
food_vars, food_items, prob = Optimization(food_data)
# Solve the Optimization problem with pulp choice of solver
prob.solve()
## Make a Schedule for a week worth of food
# print the solution
print("Status:", LpStatus[prob.status])
print("Total cost of the diet for the day = $%.2f" % value(prob.objective))
print("The diet consists of:")

# Print the food item
for food in food_items:
    servings = int(food_vars[food].varValue )
    if servings > 0:
        print("- %d serving(s) of %s" % (servings, food))
```

```
C:\Users\Fannisa Fahmi\anaconda3\lib\site-packages\pulp\pulp.py:1352: UserWarning: Spaces are not permitted in the name. Converted to '_'
  warnings.warn("Spaces are not permitted in the name. Converted to '_'")
Status: Optimal
Total cost of the diet for the day = $10.15
The diet consists of:
- 1 serving(s) of Crackers, GOYA CRACKERS, snack
- 1 serving(s) of Snacks, unsalted, wheat-based, sesame sticks
- 1 serving(s) of Fast foods, with condiments and vegetables, regular patty, cheeseburger; single
```



```python
## Make a Schedule for a week worth of food

days = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']

for day in days:
    # Solve the optimization problem for the day
    food_vars, food_items, prob = Optimization(food_data)
    status = prob.solve()

    # Print the solution for the day
    print("Status for", day, ":", LpStatus[status])
    print("Total cost of the diet for", day, "= $%.2f" % value(prob.objective))
    print("The diet for", day, "consists of:")
    for food in food_items:
        servings = int(food_vars[food].varValue)
        if servings > 0:
            print("- %d serving(s) of %s" % (servings, food))
            # Remove the food item from food_items
            food_data.drop(food_data[food_data.name == food].index, inplace=True)
    print()  # print a blank line for separation
```

```
Output exceeds the size limit. Open the full output data in a text editor
Status for Monday : Optimal
Total cost of the diet for Monday = $10.15
The diet for Monday consists of:
- 1 serving(s) of Crackers, GOYA CRACKERS, snack
- 1 serving(s) of Snacks, unsalted, wheat-based, sesame sticks
- 1 serving(s) of Fast foods, with condiments and vegetables, regular patty, cheeseburger; single

Status for Tuesday : Optimal
Total cost of the diet for Tuesday = $10.22
The diet for Tuesday consists of:
- 1 serving(s) of SUNSHINE, Original Crackers, CHEEZ-IT
- 1 serving(s) of KEEBLER, Dash of Salt Crackers, CLUB
- 1 serving(s) of Chicken, roasted, cooked, with added solution, meat and skin, thigh, dark meat

Status for Wednesday : Optimal
Total cost of the diet for Wednesday = $10.86
The diet for Wednesday consists of:
- 1 serving(s) of Snacks, plain, bagel chips
- 1 serving(s) of TACO BELL, Soft Taco with steak
- 1 serving(s) of Snacks, salted, made with partially hydrogenated soybean oil, plain, potato chips

Status for Thursday : Optimal
Total cost of the diet for Thursday = $11.07
The diet for Thursday consists of:
- 1 serving(s) of Bread, cinnamon
...
The diet for Sunday consists of:
- 1 serving(s) of TACO BELL, Nachos
- 1 serving(s) of Bread, cheese
- 1 serving(s) of Pie crust, unbaked, regular, refrigerated
```

9.2 Constraint reference

https://www.healthhub.sg/programmes/191/nutrition-hub/tools-and-resourcescalorie-calculator https://www.nhs.uk/live-well/eat-well types/different-fats-nutrition/: :text=The%20government%20recommends%20that%3A,of%20saturated%20fat%20a%20day https://ww content-of-foods: :text=If%20you%20have%20risk%20factors,than%20300%20milligrams%20a%20day. https://www.healthline.com/nut much-protein-per-day https://www.healthline.com/nutrition/sodium-per-dayrecommendations https://www.healthhub.sg/programmes hub/eat-less: :text=How%20much%20sugar%20should%20we,a%202000%2Ddaily%20calorie%20intake https://www.who.int/news/item 03-2015-who-calls-on-countries-to-reduce-sugars-intake-among-adults-and-children https://www.inchcalculator.com/convert/calorie-bur to-gram/ https://nap.nationalacademies.org/catalog/10490/dietary-reference-intakes-for-energy-carbohydrate-fiber-fat-fatty-acids-chole protein-and-amino-acids https://www.hematology.org/education/patients/anemia/iron-deficiency: :text=Most%20people%20with%20ir

9.3 Code reference

https://machinelearninggeek.com/solving-balanced-diet-problem-in-python-using-pulp https://neos-guide.org/case-studies/om/the-diet problem/ https://towardsdatascience.com/linear-programming-and-discrete-optimization-with-python-using-pulp-449f3c5f6e99 https:// of-nutritional-constraints-included-in-the-linear-programming-optimization-models$_tbl4_2$77817604

9.4 Dataset

https://databank.worldbank.org/source/food-prices-for-nutrition/ https://www.kaggle.com/datasets/trolukovich/nutritional-values-for-common-foods-and-products https://www.ars.usda.gov/is/np/NutritiveValueofFoods/NutritiveValueofFoods.pdf https://www.world

9.5 Correctness of model

https://www.ers.usda.gov/webdocs/publications/44678/19980$_{e}ib96.pdf https : //blog.seedly.sg/cheapest - places - to - eat - out -$ $singapore - based - on - neighbourhood$