In [1]:

```
from pulp import *
import pandas as pd
import os
```

In [4]:

```
# Import the data
diet = pd.read_excel('diet.xls')
```

In [5]:

```
# Exploring data - Columns
diet.columns
```

Out[5]:

```
Index(['Foods', 'Price/ Serving', 'Serving Size', 'Calories', 'Cho
lesterol mg',
       'Total_Fat g', 'Sodium mg', 'Carbohydrates g', 'Dietary_Fib
er g',
       'Protein g', 'Vit_A IU', 'Vit_C IU', 'Calcium mg', 'Iron m
g'],
      dtype='object')
```

In [6]:

```
# Exploring data - Header
diet.head()
```

Out[6]:

	Foods	Price/ Serving	Serving Size	Calories	Cholesterol mg	Total_Fat g	Sodium mg
0	Frozen Broccoli	0.16	10 Oz Pkg	73.8	0.0	0.8	68.2
1	Carrots,Raw	0.07	1/2 Cup Shredded	23.7	0.0	0.1	19.2
2	Celery, Raw	0.04	1 Stalk	6.4	0.0	0.1	34.8
3	Frozen Corn	0.18	1/2 Cup	72.2	0.0	0.6	2.5
4	Lettuce,Iceberg,Raw	0.02	1 Leaf	2.6	0.0	0.0	1.8

In [7]:

```
# Exploring data- Tail
diet.tail()
```

Out[7]:

	Foods	Price/ Serving	Serving Size	Calories	Cholesterol mg	Total_Fat g	Sodium mg	Car
62	Crm Mshrm Soup,W/Mlk	0.65	1 C (8 Fl Oz)	203.4	19.8	13.6	1076.3	
63	Beanbacn Soup,W/Watr	0.67	1 C (8 Fl Oz)	172.0	2.5	5.9	951.3	
64	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
65	NaN	NaN	Minimum daily intake	1500.0	30.0	20.0	800.0	
66	NaN	NaN	Maximum daily intake	2500.0	240.0	70.0	2000.0	

In [8]:

```
# Since it is clear that the bottom of file has missing rows, let's consider cl
ean data for this analysis
diet_clean = diet[0:64]
```

In [9]:

```
# converting data to list
diet_list = diet_clean.values.tolist()
```

In [10]:

```
# Creating dictionaries

foods = [x[0] for x in diet_list]
cost =  dict([(x[0], float(x[1])) for x in diet_list])
calories = dict([(x[0], float(x[3])) for x in diet_list])
cholesterol = dict([(x[0], float(x[4])) for x in diet_list])
totalFat = dict([(x[0], float(x[5])) for x in diet_list])
sodium = dict([(x[0], float(x[6])) for x in diet_list])
carbohydrates = dict([(x[0], float(x[7])) for x in diet_list])
dietryfiber = dict([(x[0], float(x[8])) for x in diet_list])
protien = dict([(x[0], float(x[9])) for x in diet_list])
vit_A = dict([(x[0], float(x[10])) for x in diet_list])
vit_C = dict([(x[0], float(x[11])) for x in diet_list])
calcium = dict([(x[0], float(x[12])) for x in diet_list])
iron = dict([(x[0], float(x[13])) for x in diet_list])
```

In [11]:

```
# Initializing the problem

diet_prob = LpProblem(name = "Diet_problem", sense = LpMinimize)

# Initializing the initial food variable as continuous

amount_vars = LpVariable.dicts("amounts", foods,0 )

# Initializing the initial chosen variable as Binary
set_var = LpVariable.dicts("Set_Var", foods, lowBound = 0, upBound = 1, cat =
"Binary")
```

In [12]:

```
# Setting the objective function (minimize the total cost for diet)
diet_prob += lpSum([cost[i] * amount_vars[i] for i in foods]), 'total cost'
```

In [13]:

```
# Adding constraints for all foods

diet_prob += lpSum([calories[i] * amount_vars[i] for i in foods]) >= 1500, 'min
_calories'
diet_prob += lpSum([calories[i] * amount_vars[i] for i in foods]) <= 2500, 'max
_calories'

# For Cholestrol
diet_prob += lpSum([cholesterol[i] * amount_vars[i] for i in foods]) >= 30, 'mi
n_cholestrol'
diet_prob += lpSum([cholesterol[i] * amount_vars[i] for i in foods]) <= 240, 'm
ax_cholesterol'

# For total Fat
diet_prob += lpSum([totalFat[i] * amount_vars[i] for i in foods]) >= 20, 'min_t
otalFat'
diet_prob += lpSum([totalFat[i] * amount_vars[i] for i in foods]) <= 70, 'max_t
otalFat'

# For sodium
diet_prob += lpSum([sodium[i] * amount_vars[i] for i in foods]) >= 800, 'min_so
dium'
diet_prob += lpSum([sodium[i] * amount_vars[i] for i in foods]) <= 2000, 'max_s
odium'

# For carbohydrates
diet_prob += lpSum([carbohydrates[i] * amount_vars[i] for i in foods]) >= 130,
'min_carbohydrates'
diet_prob += lpSum([carbohydrates[i] * amount_vars[i] for i in foods]) <= 450,
'max_carbohydrates'

# For dietry fiber
diet_prob += lpSum([dietryfiber[i] * amount_vars[i] for i in foods]) >= 125, 'm
in_dietryfiber'
diet_prob += lpSum([dietryfiber[i] * amount_vars[i] for i in foods]) <= 250, 'm
ax_dietryfiber'

# For protien
diet_prob += lpSum([protien[i] * amount_vars[i] for i in foods]) >= 60, 'min_pr
otien'
diet_prob += lpSum([protien[i] * amount_vars[i] for i in foods]) <= 100, 'max_p
rotien'

# For Vit_A
diet_prob += lpSum([vit_A[i] * amount_vars[i] for i in foods]) >= 1000, 'min_vi
t_A'
diet_prob += lpSum([vit_A[i] * amount_vars[i] for i in foods]) <= 10000, 'max_v
it_A'

# For Vit_C
diet_prob += lpSum([vit_C[i] * amount_vars[i] for i in foods]) >= 400, 'min_vit
_C'
diet_prob += lpSum([vit_C[i] * amount_vars[i] for i in foods]) <= 5000, 'max_vi
```

```
t_C'

# For Calcium

diet_prob += lpSum([calcium[i] * amount_vars[i] for i in foods]) >= 700, 'min_c
alcium'
diet_prob += lpSum([calcium[i] * amount_vars[i] for i in foods]) <= 1500, 'max_
calcium'

# For Iron

diet_prob += lpSum([iron[i] * amount_vars[i] for i in foods]) >= 10, 'min_iron'
diet_prob += lpSum([iron[i] * amount_vars[i] for i in foods]) <= 40, 'max_iron'
```

In [14]:

```
# Solving the problem for optimization
diet_prob.solve()
```

Out[14]:

```
1
```

In [16]:

```
# Print the optimized diet
print("Diet Optimization Solution:", LpStatus[diet_prob.status])
for j in diet_prob.variables():
    if j.varValue > 0:
        print(j.name, "=", j.varValue)
```

```
Diet Optimization: Optimal
amounts_Celery,_Raw = 52.64371
amounts_Frozen_Broccoli = 0.25960653
amounts_Lettuce,Iceberg,Raw = 63.988506
amounts_Oranges = 2.2929389
amounts_Poached_Eggs = 0.14184397
amounts_Popcorn,Air_Popped = 13.869322
```

In [17]:

```
# Print the cost of optimized diet
print ("Total Cost of food = $%.2f" % value(diet_prob.objective))
```

```
Total Cost of food = $4.34
```

15.2.2: Adding Constraints and updating model

In [19]:

```
# 15.2.2.a Adding Contraint for serving size (1/10) if selected

for n in foods:
    diet_prob += amount_vars[n] <= 10000* set_var[n]
    diet_prob += amount_vars[n] >= .1*set_var[n]
```

In [20]:

```
# 15.2.2.b Adding Contraint for celery and frozen broccoli

diet_prob += set_var['Frozen Broccoli'] + set_var['Celery, Raw'] <= 1
```

In [21]:

```
# 15.2.2.c Adding Constraint for variety in Protien

diet_prob += set_var['Roasted Chicken'] + set_var['White Tuna in Water'] + \
                set_var['Pork'] + set_var['Hamburger W/Toppings'] +\
                set_var['Hamburger W/Toppings'] + set_var['Tofu'] +\
                set_var['Scrambled Eggs'] + set_var['Frankfurter, Beef'] +\
                set_var['Bologna,Turkey'] +set_var['Pizza W/Pepperoni'] +\
                set_var['Spaghetti W/ Sauce'] >=3
```

In [22]:

```
# Solving the problem for optimization with contraints set in place

diet_prob.solve()
```

Out[22]:

```
1
```

In [23]:

```
# Print the optimized diet with constraints
print("Diet Optimization Solution with Constraints:", LpStatus[diet_prob.status
])
for d in diet_prob.variables():
    if d.varValue >0:
        print(d.name, "=", d.varValue)
```

```
Diet Optimization Solution with Constraints: Optimal
Set_Var_Bologna,Turkey = 1.0
Set_Var_Celery,_Raw = 1.0
Set_Var_Lettuce,Iceberg,Raw = 1.0
Set_Var_Oranges = 1.0
Set_Var_Peanut_Butter = 1.0
Set_Var_Popcorn,Air_Popped = 1.0
Set_Var_Scrambled_Eggs = 1.0
Set_Var_Tofu = 1.0
amounts_Bologna,Turkey = 0.1
amounts_Celery,_Raw = 42.784493
amounts_Lettuce,Iceberg,Raw = 81.603764
amounts_Oranges = 3.0831784
amounts_Peanut_Butter = 1.9444189
amounts_Popcorn,Air_Popped = 13.20665
amounts_Scrambled_Eggs = 0.12874053
amounts_Tofu = 0.1
```

In [25]:

```
# Print the cost of optimized diet with constraints
print ("Total Cost of optimized food with constraints = $%.2f" % value(diet_pro
b.objective))
```

```
Total Cost of optimized food with constraints = $4.53
```

When comparing both models, the cost doesn't change drastically.

With simpler constraints such as intake, it is set at 4.34.

Afterwards with added constraints on serving portion, vegetable preference and protien variety it goes up slightly to 4.53.