Question 15.2

1. Formulate an optimization model (a linear program) to find the cheapest diet that satisfies the maximum and minimum daily nutrition constraints, and solve it using PuLP. Turn in your code and the solution. ¶

(The optimal solution should be a diet of air-popped popcorn, poached eggs, oranges, raw iceberg lettuce, raw celery, and frozen broccoli. UGH!)

In [626]:

```
pip install pulp
```

```
Requirement already satisfied: pulp in c:\users\chintan\anaconda3\lib\site-pa
ckages (2.1)
Requirement already satisfied: pyparsing>=2.0.1 in c:\users\chintan\anaconda3
\lib\site-packages (from pulp) (2.4.6)
Note: you may need to restart the kernel to use updated packages.
```

In [813]:

```
from pulp import *
import numpy as np
```

In [814]:

```
import pandas as pd
```

In [815]:

```
df = pd.read_excel('/Users/chintan/Downloads/6501/diet.xls')
df.head()
```

Out[815]:

	Foods	Price/ Serving	Serving Size	Calories	Cholesterol mg	Total_Fat g	Sodium mg	Carbohydrat
0	Frozen Broccoli	0.16	10 Oz Pkg	73.8	0.0	0.8	68.2	13
1	Carrots,Raw	0.07	1/2 Cup Shredded	23.7	0.0	0.1	19.2	5
2	Celery, Raw	0.04	1 Stalk	6.4	0.0	0.1	34.8	1
3	Frozen Corn	0.18	1/2 Cup	72.2	0.0	0.6	2.5	17
4	Lettuce,Iceberg,Raw	0.02	1 Leaf	2.6	0.0	0.0	1.8	0

In [816]:

```
#The data seems pretty clean except the last three rows also confired that by
 looking at tge excel. Hence, removing those rows.
data1 = df[0:64]
data1.tail()
```

Out[816]:

	Foods	Price/ Serving	Serving Size	Calories	Cholesterol mg	Total_Fat g	Sodium mg	Carbohydrates g
59	Neweng Clamchwd	0.75	1 C (8 Fl Oz)	175.7	10.0	5.0	1864.9	21.8
60	Tomato Soup	0.39	1 C (8 Fl Oz)	170.7	0.0	3.8	1744.4	33.2
61	New E Clamchwd,W/Mlk	0.99	1 C (8 Fl Oz)	163.7	22.3	6.6	992.0	16.6
62	Crm Mshrm Soup,W/Mlk	0.65	1 C (8 Fl Oz)	203.4	19.8	13.6	1076.3	15.0
63	Beanbacn Soup,W/Watr	0.67	1 C (8 Fl Oz)	172.0	2.5	5.9	951.3	22.8

In [817]:

```
#Let's convert the data frame into a list
data=data1.values.tolist()
```

In [818]:

```
# create dctionaries for all of the columns.....

cost = dict([(x[0], float(x[1])) for x in data])
foods = [x[0] for x in data]
calories = dict([(x[0], float(x[3])) for x in data])
cholesterol = dict([(x[0], float(x[4])) for x in data])
totalFat = dict([(x[0], float(x[5])) for x in data])
sodium = dict([(x[0], float(x[6])) for x in data])
carbs = dict([(x[0], float(x[7])) for x in data])
fiber = dict([(x[0], float(x[8])) for x in data])
protien = dict([(x[0], float(x[9])) for x in data])
vitaminA = dict([(x[0], float(x[10])) for x in data])
vitaminC = dict([(x[0], float(x[11])) for x in data])
calcium = dict([(x[0], float(x[12])) for x in data])
iron = dict([(x[0], float(x[13])) for x in data])
```

In [819]:

```
# let's define the optimization problem using PuLP....

prob = LpProblem('DietProblem', LpMinimize)
```

In [820]:

```
# Let's define the variable

amountvars = LpVariable.dicts("foods", foods,0)
x = LpVariable.dicts("x", foods, 0)
```

In [821]:

```
#Objective function = cost* Food

prob += lpSum([cost[i] * amountvars[i] for i in foods])
```

In [822]:

```
#  Here is the list of all min and max for each food and we can create a for l
oop function to iterate over them...
intakemin = [1500, 30, 20, 800, 130, 125, 60, 1000, 400, 700, 10]
intakemax = [2500, 240, 70, 2000, 450, 250, 100, 10000, 5000, 1500, 40]
intake = []
for j in range(0,11):
    intake.append(dict([(x[0], float(x[j+3])) for x in data]))
```

In [823]:

```
for i in range(0,11):
    prob += pulp.lpSum([intake[i][j] * amountvars[j] for j in foods])>=intakem
in[i]
    prob += pulp.lpSum([intake[i][j] * amountvars[j] for j in foods])<=intakem
ax[i]
```

In [824]:

```
prob.solve()
```

Out[824]:

```
1
```

In [825]:

```
print("Status:", LpStatus[prob.status])
```

```
Status: Optimal
```

Let's print the list of optimal foods with servings size ...

In [827]:

```
for var in prob.variables():
    if var.varValue !=0:
        print(var.name,"=",var.varValue)
        #print(str(var.varValue) + " units of " + str(var))
```

```
foods_Celery,_Raw = 52.64371
foods_Frozen_Broccoli = 0.25960653
foods_Lettuce,Iceberg,Raw = 63.988506
foods_Oranges = 2.2929389
foods_Poached_Eggs = 0.14184397
foods_Popcorn,Air_Popped = 13.869322
```

Just trying to convert the above list into Data Frame so that we can have price/serving as well....

In [829]:

```
var = prob.variables()
df_optimal = pd.DataFrame(
    {'Foods':[v.name.replace("_", " ").replace("foods ","") for v in var],
     'Optimal Servings 1': [v.varValue for v in var]},
)

newdf = pd.merge(df_optimal, data1, on = "Foods",how = "left")
```

In [830]:

```
df1 = newdf.loc[newdf['Optimal Servings 1'] > 0 ][['Foods', 'Price/ Serving',
'Optimal Servings 1']]
df1
```

Out[830]:

	Foods	Price/ Serving	Optimal Servings 1
11	Celery, Raw	0.04	52.643710
20	Frozen Broccoli	0.16	0.259607
28	Lettuce,Iceberg,Raw	0.02	63.988506
35	Oranges	0.15	2.292939
39	Poached Eggs	0.08	0.141844
40	Popcorn,Air Popped	NaN	13.869322

In [812]:

```
np.sum(df1['Price/ Serving'] * df1['Optimal Servings 1'])
```

Out[812]:

```
3.7823439173999995
```

it's looks like the Popcorn price is not showing up. Adding that price manually 13.869322*0.04 = 0.554769,

adding that to total cost

Total Cost = 4.33711

or

In [839]:

```
print("Total cost of food = $%.2f" % value(prob.objective))
```

```
Total cost of food = $4.34
```

2. Please add to your model the following constraints

(which might require adding more variables) and solve the new model: Here we are going to combine part a,b and c of the 2nd question..

a. If a food is selected, then a minimum of 1/10 serving must be chosen.

In [861]:

```
prob4 = LpProblem('Dietproblem2', LpMinimize)
```

In [862]:

```
prob4 += lpSum([cost[f] * amountvars[f] for f in foods])
```

In [843]:

```
# Binary variable
BinVars = LpVariable.dicts("Chosen",foods,0,1,"Binary")
```

In [844]:

```
# If any of a food is eaten, its binary variable must be 1
for i in foods:
    prob4 += amountvars[i] >= 0.1*BinVars[i]
    prob4 += amountvars[i] <= 10000*BinVars[i]
```

In [845]:

```
for i in range(0,11):
    prob4 += pulp.lpSum([intake[i][j] * amountvars[j] for j in foods])>=intakemin[i]
    prob4 += pulp.lpSum([intake[i][j] * amountvars[j] for j in foods])<=intakemax[i]
```

b.Many people dislike celery and frozen broccoli. So at most one, but not both, can be selected.

In [847]:

```
prob4 += BinVars['Frozen Broccoli'] + BinVars['Celery, Raw'] <= 1
```

c.To get day-to-day variety in protein, at least 3 kinds of meat/poultry/fish/eggs must be selected.

In [ ]:

```
# List of necessary protein
prob4 += BinVars['Roasted Chicken'] + BinVars['Poached Eggs'] + BinVars['Scrambled Eggs'] + BinVars['Frankfurter, Beef'] + BinVars['Kielbasa,Prk'] + BinVars['Hamburger W/Toppings'] + BinVars['Hotdog, Plain'] + BinVars['Pork'] + BinVars['Bologna,Turkey'] + BinVars['Ham,Sliced,Extralean'] + BinVars['White Tuna in Water']>= 3
```

In [850]:

```
prob4.solve()
```

Out[850]:

```
1
```

In [851]:

```
for var in prob4.variables():
    if var.varValue != 0:
        if str(var).find('Chosen'):
            print(var.name,"=",var.varValue)
```

```
foods_Celery,_Raw = 42.399358
foods_Kielbasa,Prk = 0.1
foods_Lettuce,Iceberg,Raw = 82.802586
foods_Oranges = 3.0771841
foods_Peanut_Butter = 1.9429716
foods_Poached_Eggs = 0.1
foods_Popcorn,Air_Popped = 13.223294
foods_Scrambled_Eggs = 0.1
```

In [852]:

```
newvar = []
for var in prob4.variables():
    if var.varValue != 0:
        if str(var).find('Chosen'):
            newvar.append(var)
newvar
```

Out[852]:

```
[foods_Celery,_Raw,
 foods_Kielbasa,Prk,
 foods_Lettuce,Iceberg,Raw,
 foods_Oranges,
 foods_Peanut_Butter,
 foods_Poached_Eggs,
 foods_Popcorn,Air_Popped,
 foods_Scrambled_Eggs]
```

In [869]:

```
# Dataframe of optimal values

df_optimal4 = pd.DataFrame(
    {'Foods':[v.name.replace("foods_","").replace("_", " ") for v in newvar],
     'Optimal Servings 2': [v.varValue for v in newvar]},
)
```

In [855]:

```
df_final = pd.merge(df_optimal4,newdf,on="Foods")
df_final
df2 = df_final.loc[df_final['Optimal Servings 2'] >0][['Foods', 'Price/ Serving', 'Optimal Servings 2']]
df2
```

Out[855]:

	Foods	Price/ Serving	Optimal Servings 2
0	Celery, Raw	0.04	42.399358
1	Kielbasa,Prk	0.15	0.100000
2	Lettuce,Iceberg,Raw	0.02	82.802586
3	Oranges	0.15	3.077184
4	Peanut Butter	0.07	1.942972
5	Poached Eggs	0.08	0.100000
6	Popcorn,Air Popped	NaN	13.223294
7	Scrambled Eggs	0.11	0.100000

In [856]:

```
np.sum(df2['Price/ Serving'] * df2['Optimal Servings 2'])
```

Out[856]:

```
3.9836116670000004
```

it's looks like the Popcorn price is not showing up.Just adding that price manually

13.869322*0.04 = 0.554769, adding that to total cost

Total Cost = 4.5383

or

```
In [858]: print("Total cost of food = $%.2f" % value(prob4.objective))
```

```
Total cost of food = $4.51
```

Summary

For problem one, the cost of cheapest diet that satisfies the maximum and minimum daily nutrition is $4.34

```
In [868]: Food                Price/ Serving  Optimal Servings 1
          Celery, Raw         0.04            52.643710
          Frozen Broccoli     0.16             0.259607
          Lettuce,Iceberg,Raw 0.02            63.988506
          Oranges             0.15             2.292939
          Poached Eggs        0.08             0.141844
          Popcorn,Air Popped  0.04            13.869322
```

```
  File "<ipython-input-868-f8d21a15c575>", line 1
    Food                Price/ Serving  Optimal Servings 1
                                    ^
SyntaxError: invalid syntax
```

For problem two, with below mention constraints the cost of cheapest diet that satisfies the maximum and minimum daily nutrition is $4.51

Constraints

- maximum and minimum daily values of each nutrient
- chosen foods bounded between .1 and M(large constant)
- only one of broccoli and celery can be in the optimal diet
- at least three proteins must be selected in the optimal diet

In [ ]:

```
Foods        Price/Serving      Optimal Servings 2
Celery, Raw          0.04       42.399358
Kielbasa,Prk         0.15        0.100000
Lettuce,Iceberg,Raw  0.02       82.802586
Oranges              0.15        3.077184
Peanut Butter        0.07        1.942972
Poached Eggs         0.08        0.100000
Popcorn,Air Popped   0.04       13.223294
Scrambled Eggs       0.11        0.100000
```

In [ ]: