Blending Problem.

Copied from https://coin-or.github.io/pulp/CaseStudies/a_blending_problem.html#problem-description

The_Whiskas_Problem:

MINIMIZE

0.018 *BEEF + 0.3* CHICKEN + 0.011 *GEL + 0.01* MUTTON + 0.02 *RICE + 0.05* WHEAT

SUBJECT TO

PercentagesSum: BEEF + CHICKEN + GEL + MUTTON + RICE + WHEAT = 100

ProteinRequirement: 0.2 BEEF + 0.1 Ingr_CHICKEN + 0.15 MUTTON + 0.04 WHEAT >= 8

FatRequirement: 0.1 BEEF + 0.08 Ingr_CHICKEN + 0.11 MUTTON + 0.01 RICE + 0.01 WHEAT >= 6

FibreRequirement: 0.005 BEEF + 0.001 CHICKEN + 0.003 MUTTON + 0.1 RICE + 0.15 WHEAT <= 2

SaltRequirement: 0.005 BEEF + 0.002 CHICKEN + 0.007 MUTTON + 0.002 RICE + 0.008 WHEAT <= 0.4

Nonnegativity: BEEF, CHICKEN, MUTTON, RICE, WHEAT >= 0

Steps For Installing PuLP

In [1]:

```
import sys
!{sys.executable} -m pip install pulp
```

```
Requirement already satisfied: pulp in c:\users\muntakim\appdata\local\programs\python\p
ython38-32\lib\site-packages (2.4)
Requirement already satisfied: amply>=0.1.2 in c:\users\muntakim\appdata\local\programs
\python\python38-32\lib\site-packages (from pulp) (0.1.4)
Requirement already satisfied: docutils>=0.3 in c:\users\muntakim\appdata\local\programs
\python\python38-32\lib\site-packages (from amply>=0.1.2->pulp) (0.17.1)
Requirement already satisfied: pyparsing in c:\users\muntakim\appdata\local\programs\pyt
hon\python38-32\lib\site-packages (from amply>=0.1.2->pulp) (2.4.7)
```

In [2]:

```
import pulp
```

In [3]:

```
"""
The Full Whiskas Model Python Formulation for the PuLP Modeller

Authors: Antony Phillips, Dr Stuart Mitchell  2007
"""

# Import PuLP modeler functions.
```

```
# Here because of * we will not put `pulp' before each pulp command; e.g. instead of pu

from pulp import *
```

Steps For Decision Variables.

In [4]:

```
# Creates a list of the Ingredients.
Ingredients = ['CHICKEN', 'BEEF', 'MUTTON', 'RICE', 'WHEAT', 'GEL']
# This gives the names for the indexes in the vector.

# A dictionary of the costs of each of the Ingredients is created. They give vector val
costs = {'CHICKEN': 0.30, # originally 0.013
         'BEEF': 0.018, # originally 0.008
         'MUTTON': 0.010, # originally 0.010
         'RICE': 0.02, # originally 0.002
         'WHEAT': 0.05, # originally 0.005
         'GEL': 0.011}  # originally 0.001

# A dictionary of the protein percent in each of the Ingredients is created.
proteinPercent = {'CHICKEN': 0.100,
                  'BEEF': 0.200,
                  'MUTTON': 0.150,
                  'RICE': 0.000,
                  'WHEAT': 0.040,
                  'GEL': 0.000}

# A dictionary of the fat percent in each of the Ingredients is created.
fatPercent = {'CHICKEN': 0.080,
              'BEEF': 0.100,
              'MUTTON': 0.110,
              'RICE': 0.010,
              'WHEAT': 0.010,
              'GEL': 0.000}

# A dictionary of the fibre percent in each of the Ingredients is created.
fibrePercent = {'CHICKEN': 0.001,
                'BEEF': 0.005,
                'MUTTON': 0.003,
                'RICE': 0.100,
                'WHEAT': 0.150,
                'GEL': 0.000}

# A dictionary of the salt percent in each of the Ingredients is created.
saltPercent = {'CHICKEN': 0.002,
               'BEEF': 0.005,
               'MUTTON': 0.007,
               'RICE': 0.002,
               'WHEAT': 0.008,
               'GEL': 0.000}
```

In [5]:

```
# Create the 'prob' variable to contain the problem data.
prob = LpProblem("The_Whiskas_Problem", LpMinimize)
```

In [6]:

```
# A dictionary called 'ingredient_vars' is created to contain the referenced Variables.
ingredient_vars = LpVariable.dicts("Ingr", Ingredients, 0)
```

```
# Here the last value '0' gives the lower bound for the variable.
# Here "Ingr" is what appears when we print its name; e.g. Ingr_Beef. In the code, `ing
# We use the `dicts' command to use the previously given list `Ingredient'.

print(ingredient_vars)
```

```
{'CHICKEN': Ingr_CHICKEN, 'BEEF': Ingr_BEEF, 'MUTTON': Ingr_MUTTON, 'RICE': Ingr_RICE,
'WHEAT': Ingr_WHEAT, 'GEL': Ingr_GEL}
```

Objective Function

In [7]:

```
# The objective function is added to 'prob' first.
prob += lpSum([costs[i]*ingredient_vars[i] for i in Ingredients]), "Total Cost of Ingre
# Here "Total Cost of Ingredients per can" gives an explanation comment. Do not forget
```

Constraints

In [8]:

```
# The five constraints are added to 'prob'.
prob += lpSum([ingredient_vars[i] for i in Ingredients]) == 100, "PercentagesSum"
prob += lpSum([proteinPercent[i] * ingredient_vars[i] for i in Ingredients]) >= 8.0, "P
prob += lpSum([fatPercent[i] * ingredient_vars[i] for i in Ingredients]) >= 6.0, "FatRe
prob += lpSum([fibrePercent[i] * ingredient_vars[i] for i in Ingredients]) <= 2.0, "Fib
prob += lpSum([saltPercent[i] * ingredient_vars[i] for i in Ingredients]) <= 0.4, "Salt
```

Notice that we did not add the condition that the ingridients are >=0, as it was given in ingredient_vars = LpVariable.dicts("Ingr",Ingredients, 0) by adding the `0` .

If we did not add `0` there, we can instead add the contraints in the code as

```
for i in Ingredients:
    prob += ingredient_vars[i] >= 0
```

Show The LP problem.

In [9]:

```
# You can write the problem to an .lp file.
prob.writeLP("WhiskasModel.lp")
```

Out[9]:

```
[Ingr_BEEF, Ingr_CHICKEN, Ingr_GEL, Ingr_MUTTON, Ingr_RICE, Ingr_WHEAT]
```

In [10]:

```
# Or you can directly display the problem here.

print(prob)
```

```
The_Whiskas_Problem:
MINIMIZE
0.018*Ingr_BEEF + 0.3*Ingr_CHICKEN + 0.011*Ingr_GEL + 0.01*Ingr_MUTTON + 0.02*Ingr_RICE
+ 0.05*Ingr_WHEAT + 0.0
SUBJECT TO
PercentagesSum: Ingr_BEEF + Ingr_CHICKEN + Ingr_GEL + Ingr_MUTTON + Ingr_RICE
 + Ingr_WHEAT = 100
```

```
ProteinRequirement: 0.2 Ingr_BEEF + 0.1 Ingr_CHICKEN + 0.15 Ingr_MUTTON
 + 0.04 Ingr_WHEAT >= 8

FatRequirement: 0.1 Ingr_BEEF + 0.08 Ingr_CHICKEN + 0.11 Ingr_MUTTON
 + 0.01 Ingr_RICE + 0.01 Ingr_WHEAT >= 6

FibreRequirement: 0.005 Ingr_BEEF + 0.001 Ingr_CHICKEN + 0.003 Ingr_MUTTON
 + 0.1 Ingr_RICE + 0.15 Ingr_WHEAT <= 2

SaltRequirement: 0.005 Ingr_BEEF + 0.002 Ingr_CHICKEN + 0.007 Ingr_MUTTON
 + 0.002 Ingr_RICE + 0.008 Ingr_WHEAT <= 0.4

VARIABLES
Ingr_BEEF Continuous
Ingr_CHICKEN Continuous
Ingr_GEL Continuous
Ingr_MUTTON Continuous
Ingr_RICE Continuous
Ingr_WHEAT Continuous
```

Notice that the lower bound >=0 for the variable is not shown, as it is the default condition. If you had changed the lowerbound to something else, then it will show up here.

Solve the LP.

In [11]:

```
# The problem is solved using PuLP's choice of Solver.
prob.solve()
# The status of the solution is printed to the screen.
print("Status:", LpStatus[prob.status])
```

```
Status: Optimal
```

Each of the variables is printed with its resolved optimum value.

In [12]:

```
for a in prob.variables():
    print(a.name, "=", a.varValue)
```

```
Ingr_BEEF = 0.0
Ingr_CHICKEN = 0.0
Ingr_GEL = 42.857143
Ingr_MUTTON = 57.142857
Ingr_RICE = 0.0
Ingr_WHEAT = 0.0
```

In [13]:

```
print("Total Cost of Ingredients per can = ", value(prob.objective))
```

```
Total Cost of Ingredients per can =  1.042857143
```

Other way to write the final results.

In [14]:

```
print(LpStatus[prob.status])
for i in prob.variables():
    print("Variable {0} = {1}".format(i.name, i.varValue))
print("Objective function z = {0}".format(value(prob.objective)))
```

```
Optimal
Variable Ingr_BEEF = 0.0
Variable Ingr_CHICKEN = 0.0
Variable Ingr_GEL = 42.857143
Variable Ingr_MUTTON = 57.142857
Variable Ingr_RICE = 0.0
Variable Ingr_WHEAT = 0.0
Objective function z = 1.042857143
```