



### **MiniCase1**

#### **Names**

Dora Guadalupe Valencia Zempoalteca

Azucena Méndez López

Rubén Uriel Flores Bello

Erick Orlando Carcaño Estévez

Mariana Guevara Munive

#### **Student Registration**

A01734100

A01737791

A01737098

A01737840

A01733708

#### **Career**

Business

#### **Semester \_ 2ro**

#### **Teacher**

Hesus García Cobos

## Instructions

Please follow the instructions below to complete the assignment:

1. Read the Whiskas case description provided.
2. Run the provided code snippets, WhiskasModel1 and WhiskasModel2, related to the Whiskas problem.

```
[13] !pip install pulp

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting pulp
  Downloading PuLP-2.7.0-py3-none-any.whl (14.3 MB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 14.3/14.3 MB 42.9 MB/s eta 0:00:00
Installing collected packages: pulp
Successfully installed pulp-2.7.0
```

```
# Import PuLP modeler functions
from pulp import *

# Create the 'prob' variable to contain the problem data
prob = LpProblem("The Whiskas Problem", LpMinimize)

# The 2 variables Beef and Chicken are created with a lower limit of zero
x1 = LpVariable("ChickenPercent", 0, None, LpInteger)
x2 = LpVariable("BeefPercent", 0)

# The objective function is added to 'prob' first
prob += 0.013 * x1 + 0.008 * x2, "Total Cost of Ingredients per can"

# The five constraints are entered
prob += x1 + x2 == 100, "PercentagesSum"
prob += 0.100 * x1 + 0.200 * x2 >= 8.0, "ProteinRequirement"
prob += 0.080 * x1 + 0.100 * x2 >= 6.0, "FatRequirement"
prob += 0.001 * x1 + 0.005 * x2 <= 2.0, "FibreRequirement"
prob += 0.002 * x1 + 0.005 * x2 <= 0.4, "SaltRequirement"

# The problem data is written to an .lp file
prob.writeLP("WhiskasModel.lp")

# 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])

# Each of the variables is printed with it's resolved optimum value
for v in prob.variables():
    print(v.name, "=", v.varValue)

# The optimised objective function value is printed to the screen
print("Total Cost of Ingredients per can = ", value(prob.objective))
```

```
Status: Optimal
BeefPercent = 66.0
ChickenPercent = 34.0
Total Cost of Ingredients per can = 0.97
/usr/local/lib/python3.10/dist-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 '_'")
```

3. Answer the questions based on the code snippets and the Whiskas case.

## Whiskas Case Description

Whiskas cat food, shown above, is manufactured by Uncle Ben's. Uncle Ben's wants to produce their cat food products as cheaply as possible while ensuring they meet the stated nutritional analysis requirements shown on the cans. The main ingredients of Whiskas cat food are chicken, beef, mutton, rice, wheat, and gel. Each ingredient contributes to the total weight of protein, fat, fiber, and salt in the final product.

**The costs of the ingredients per gram are as follows:**

**Chicken:** \$0.013

**Beef:** \$0.008

**Mutton:** \$0.010

**Rice:** \$0.002

**Wheat:** \$0.005

**Gel:** \$0.001

Ingrediente	Proteína	Gordo	Fibra	Sal
Pollo	0.100	0.080	0.001	0.002
Carne de res	0.200	0.100	0.005	0.005
Carne de cordero	0.150	0.110	0.003	0.007
Arroz	0.000	0.010	0.100	0.002
Trigo	0.040	0.010	0.150	0.008
Gel	0.000	0.000	0.000	0.000

WhiskasModel1 - Simplified Whiskas Model

The WhiskasModel1 snippet provides a simplified formulation of the Whiskas problem using the PuLP Modeller. This code defines the decision variables, objective function, and constraints, and solves the problem to find the optimal solution. It also displays the resolved optimum values for the variables and the total cost of ingredients per can.

WhiskasModel2 - Full Whiskas Model

The WhiskasModel2 snippet presents a more comprehensive formulation of the Whiskas problem using the PuLP Modeller. This code represents the ingredients, costs, and nutritional percentages using dictionaries. It then defines the decision variables, objective function, and constraints. The problem is solved, and the optimal solution, variable values, and total cost of ingredients per can are displayed.

**1. The LpProblem function implements two parameters:**

- **The whiskas problem:** This parameter is used to provide a name or identifier for the problem. It is an optional parameter.
- **LpMinimize:** This parameter is used to specify the sense of the objective function, i.e., whether it is a minimization problem or a maximization problem.

## 2. No, it is not mandatory to name the prob variable as "prob."?

It can be named differently according to the user's preference. In the provided code snippets, "prob" is used as a common convention for representing the problem.

## 3. In PuLP, LpContinuous and LpInteger are used to define the type of decision variables.

- **LpContinuous (BeefPercent):** This variable type is used to represent continuous decision variables, i.e., variables that can take any real value within a specified range.
- **LpInteger (ChickenPercent):** This variable type is used to represent integer decision variables, i.e., variables that can only take integer values.

## 4. The section of code that defines the objective function in WhiskasModel1.py is as follows:

```
prob += 0.013 * x1 + 0.008 * x2, "Total Cost of Ingredients per can"
```

This line of code defines the objective function of the linear programming problem.

prob: It refers to the problem variable created using LpProblem earlier. It represents the linear programming problem to which we are adding the objective function.

0.013 \* x1 + 0.008 \* x2: This expression represents the cost of ingredients per can.

"Total Cost of Ingredients per can": It helps to provide a descriptive name to identify the objective when analyzing the solution.

## 5. The section of code that defines the constraints in WhiskasModel1.py is as follows:

```
prob += x1 + x2 == 100, "PercentagesSum"
prob += 0.100 * x1 + 0.200 * x2 >= 8.0, "ProteinRequirement"
prob += 0.080 * x1 + 0.100 * x2 >= 6.0, "FatRequirement"
prob += 0.001 * x1 + 0.005 * x2 <= 2.0, "FibreRequirement"
prob += 0.002 * x1 + 0.005 * x2 <= 0.4, "SaltRequirement"
```

Explanation

`prob += x1 + x2 == 100, "PercentagesSum"`: This line adds a constraint to the problem (prob). It states that the sum of x1 and x2 variables should be equal to 100. The "PercentagesSum" is an optional name or label given to this constraint.

`prob += 0.100 * x1 + 0.200 * x2 >= 8.0, "ProteinRequirement"`: This line adds another constraint to the problem. It sets a lower bound on the expression  $0.100 * x1 + 0.200 * x2$ . It states that the expression should be greater than or equal to 8.0. It represents a protein requirement constraint. The "ProteinRequirement" is the label given to this constraint.

`prob += 0.080 * x1 + 0.100 * x2 >= 6.0, "FatRequirement"`: This line adds a constraint similar to the previous one but for fat requirement. It sets a lower bound on the expression  $0.080 * x1 + 0.100 * x2$ , requiring it to be greater than or equal to 6.0.

`prob += 0.001 * x1 + 0.005 * x2 <= 2.0, "FibreRequirement"`: This line adds a constraint for the fiber requirement. It sets an upper bound on the expression  $0.001 * x1 + 0.005 * x2$ , ensuring it doesn't exceed 2.0.

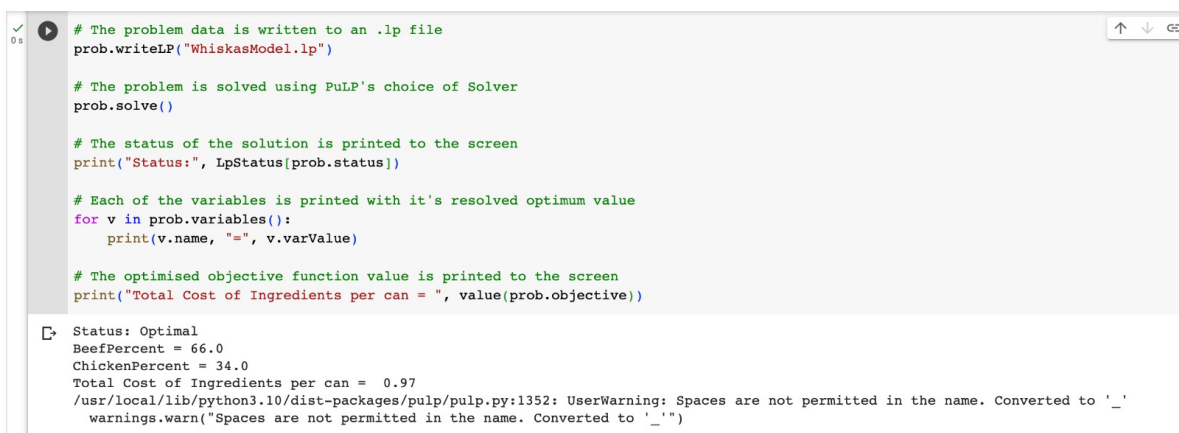
`prob += 0.002 * x1 + 0.005 * x2 <= 0.4, "SaltRequirement"`: This line adds a constraint for the salt requirement. It sets an upper bound on the expression  $0.002 * x1 + 0.005 * x2$ , limiting it to a maximum of 0.4.

## 6. Is this a minimization or maximization problem?

Minimization

## 7. Running the WhiskasModel1.py code without making any changes, the values of the following variables are displayed:

```
Status: Optimal
BeefPercent = 66.0
ChickenPercent = 34.0
Total Cost of Ingredients per can = 0.97
```



```
# The problem data is written to an .lp file
prob.writeLP("WhiskasModel1.lp")

# 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])

# Each of the variables is printed with it's resolved optimum value
for v in prob.variables():
    print(v.name, "=", v.varValue)

# The optimised objective function value is printed to the screen
print("Total Cost of Ingredients per can = ", value(prob.objective))

Status: Optimal
BeefPercent = 66.0
ChickenPercent = 34.0
Total Cost of Ingredients per can = 0.97
/usr/local/lib/python3.10/dist-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 '_'")
```