

0. Imports and Setting up Anthropic API Client

```
from google.colab import drive
```

```
drive.mount('/content/drive')
```

Mounted at /content/drive

```
!pip install python-dotenv
```

```
import os
import dotenv
```

```
dotenv.load_dotenv('/content/drive/MyDrive/.env')
```

Collecting python-dotenv
 Downloading python_dotenv-1.0.1-py3-none-any.whl (19 kB)
 Installing collected packages: python-dotenv
 Successfully installed python-dotenv-1.0.1
 True

```
# Load Prompts and Problem Description
```

```
# Variables Prompt
```

```
prompt11_path = '/content/drive/MyDrive/Thesis/Prompts/Prompt11_MathematicalModel.txt'
```

```
# Objective Prompt
```

```
prompt12_path = '/content/drive/MyDrive/Thesis/Prompts/Prompt12_MathematicalModel.txt'
```

```
# Constraint Prompt
```

```
prompt13_path = '/content/drive/MyDrive/Thesis/Prompts/Prompt13_MathematicalModel.txt'
```

```
# Code Prompt
```

```
prompt2_path = '/content/drive/MyDrive/Thesis/Prompts/Prompt2_PyomoCode.txt'
```

```
problem_desc_path = '/content/drive/MyDrive/Thesis/ProblemDescriptions/LP/LP4.txt'
```

```
prompt11_file = open(prompt11_path, "r")
prompt12_file = open(prompt12_path, "r")
prompt13_file = open(prompt13_path, "r")
prompt2_file = open(prompt2_path, "r")
problem_desc_file = open(problem_desc_path, "r")
```

```
prompt11 = prompt11_file.read()
print("Prompt 1.1 (Variables):\n", prompt11)
```

```
prompt12 = prompt12_file.read()
print("Prompt 1.2 (Objective):\n", prompt12)
```

```
prompt13 = prompt13_file.read()
print("Prompt 1.3 (Constraints):\n", prompt13)
```

```
prompt2 = prompt2_file.read()
print("Prompt 2:\n", prompt2)
```

```
problem_desc = problem_desc_file.read()
print("Problem Description:\n", problem_desc)
```

Prompt 1.1 (Variables):
 Please formulate only the variables for this mathematical optimization problem.
 Prompt 1.2 (Objective):
 Please formulate only the objective function for this mathematical optimization problem.
 Prompt 1.3 (Constraints):
 Please formulate only the constraints for this mathematical optimization problem.
 Prompt 2:
 Please write a python pyomo code for this optimization problem.
 Use sample data where needed.
 Indicate where you use sample data.
 Problem Description:
 Consider a farmer who specializes in raising wheat, corn, and sugar beets on her 500 acres of land. During the winter, The farmer knows that at least 200 tons (T) of wheat and 240 T of corn are needed for cattle feed. These amounts can be Any production in excess of the feeding requirement would be sold.
 Over the last decade, mean selling prices have been \$170 and \$150 per ton of wheat and corn, respectively. The purchase prices are 40% more than this due to the wholesaler's margin and transportation costs.
 Another profitable crop is sugar beet, which she expects to sell at \$36/T. However, the European Commission imposes a qu Based on past experience, the farmer knows that the mean yield on her land is roughly 2.5 T, 3 T, and 20 T per acre for The farmer can decide how much of her land she will use to grow each of the three products: wheat, corn and sugar beets.

```
!pip install anthropic
```

 Show hidden output

```
# Importing Anthropic & Setting Headers
import anthropic
```


```
client = anthropic.Anthropic(
    # defaults to os.environ.get("ANTHROPIC_API_KEY")
    api_key=os.environ.get("ANTHROPIC_API_KEY"),
)
```

✓ 1. Prompt 1.1: Create Variables for Mathematical Model

```
message11 = client.messages.create(
    model="claude-3-opus-20240229",
    system=prompt11,
    max_tokens=4096,
    messages=[
        {"role": "user", "content": problem_desc}
    ]
)
```

```
response_p11 = message11.content[0].text
```

```
# Print response
print(response_p11)
```

 Great! Let's define the variables for this mathematical optimization problem:

Decision Variables:

- Let 'w' be the number of acres of land devoted to wheat production.
- Let 'c' be the number of acres of land devoted to corn production.
- Let 's' be the number of acres of land devoted to sugar beet production.

Additional Variables:

- Let 'W' be the number of tons of wheat produced.
- Let 'C' be the number of tons of corn produced.
- Let 'S' be the number of tons of sugar beets produced.
- Let 'Wp' be the number of tons of wheat purchased from the wholesaler.
- Let 'Cp' be the number of tons of corn purchased from the wholesaler.
- Let 'Ws' be the number of tons of wheat sold.
- Let 'Cs' be the number of tons of corn sold.
- Let 'Sq' be the number of tons of sugar beets sold at the quota price.
- Let 'Se' be the number of tons of sugar beets sold at the excess price.

These variables will be used to formulate the objective function and constraints of the optimization problem, which aims

✓ 1. Prompt 1.2: Create Objective for Mathematical Model

```
message12 = client.messages.create(
    model="claude-3-opus-20240229",
    system=prompt12,
    max_tokens=4096,
    messages=[
        {"role": "user", "content": (problem_desc + response_p11)}
    ]
)
```

```
response_p12 = message12.content[0].text
```

```
# Print response
print(response_p12)
```

 Objective Function:

Minimize:

$$150w + 230c + 260s + 238Wp + 210Cp - 170Ws - 150Cs - 36Sq - 10Se$$

This objective function represents the total costs the farmer wants to minimize. It includes the planting costs for each

The negative signs before the selling terms indicate that the revenue is subtracted from the total costs, as selling the

✓ 1. Prompt 1.3: Create Constraints for Mathematical Model

```
message13 = client.messages.create(
    model="claude-3-opus-20240229",
    system=prompt13,
    max_tokens=4096,
    messages=[
        {"role": "user", "content": (problem_desc + response_p11 + response_p12)}
    ]
)

response_p13 = message13.content[0].text

# Print response
print(response_p13)
```

↩ Here are the constraints for this optimization problem:

1. Land constraint:
 $w + c + s \leq 500$
 (The total acres used for wheat, corn, and sugar beets cannot exceed the available 500 acres)
2. Wheat production and usage:
 $W = 2.5w$
 $W + Wp \geq 200$
 $Ws = W + Wp - 200$
 (Wheat produced equals 2.5 times the acres used for wheat; total wheat available must be at least 200 tons; wheat sold equals total wheat available minus wheat sold)
3. Corn production and usage:
 $C = 3c$
 $C + Cp \geq 240$
 $Cs = C + Cp - 240$
 (Corn produced equals 3 times the acres used for corn; total corn available must be at least 240 tons; corn sold equals total corn available minus corn sold)
4. Sugar beet production and quota:
 $S = 20s$
 $Sq + Se = S$
 $Sq \leq 6000$
 (Sugar beets produced equals 20 times the acres used for sugar beets; sugar beets sold at quota price plus excess price)
5. Non-negativity constraints:
 $w, c, s, W, C, S, Wp, Cp, Ws, Cs, Sq, Se \geq 0$
 (All variables must be non-negative)

These constraints, along with the objective function you provided earlier, complete the formulation of the mathematical

✓ 2. Prompt 2: Write Code based on Mathematical Model

In case more than the component of each step was generated, use "last instance"/ "newest version" of the components.

Example: Prompt 1.1 (Variables): Model generates only variables Prompt 1.2 (Objective): Model generates objective and constraints Prompt 1.3 (Constraints): Model generates only constraints

Use Variables from step 1.1, use objective from 1.2 and use constraints from 1.3 (since constraints version 1.3 is more recent than constraints version 1.2).

This set up allows for potential self-correction of the model throughout the three prompts.

Correctness of each component is still generated in the respective step, so self-correction is independent of this.

```
message2 = client.messages.create(
    model="claude-3-opus-20240229",
    system=prompt2,
    max_tokens=4096,
    messages=[
        {"role": "user", "content": (response_p11 + response_p12 + response_p13)}
    ]
)

response_p2 = message2.content[0].text

# Print response
print(response_p2)
```



```

model.C = Var(domain=NonNegativeReals) # Tons of corn produced
model.S = Var(domain=NonNegativeReals) # Tons of sugar beets produced
model.Wp = Var(domain=NonNegativeReals) # Tons of wheat purchased from wholesaler
model.Cp = Var(domain=NonNegativeReals) # Tons of corn purchased from wholesaler
model.Ws = Var(domain=NonNegativeReals) # Tons of wheat sold
model.Cs = Var(domain=NonNegativeReals) # Tons of corn sold
model.Sq = Var(domain=NonNegativeReals) # Tons of sugar beets sold at quota price
model.Se = Var(domain=NonNegativeReals) # Tons of sugar beets sold at excess price

# Define objective function
model.obj = Objective(expr=150 * model.w + 230 * model.c + 260 * model.s +
                      238 * model.Wp + 210 * model.Cp -
                      170 * model.Ws - 150 * model.Cs -
                      36 * model.Sq - 10 * model.Se,
                      sense=minimize)

# Define constraints
model.land_constraint = Constraint(expr=model.w + model.c + model.s <= 500)

model.wheat_production = Constraint(expr=model.W == 2.5 * model.w)
model.wheat_usage = Constraint(expr=model.W + model.Wp >= 200)
model.wheat_sold = Constraint(expr=model.Ws == model.W + model.Wp - 200)

model.corn_production = Constraint(expr=model.C == 3 * model.c)
model.corn_usage = Constraint(expr=model.C + model.Cp >= 240)
model.corn_sold = Constraint(expr=model.Cs == model.C + model.Cp - 240)

model.sugar_beet_production = Constraint(expr=model.S == 20 * model.s)
model.sugar_beet_sold = Constraint(expr=model.Sq + model.Se == model.S)
model.sugar_beet_quota = Constraint(expr=model.Sq <= 6000)

# Solve the optimization problem
solver = SolverFactory('glpk') # Specify the solver (e.g., 'glpk', 'gurobi', 'cplex')
results = solver.solve(model)

# Print the results
print("Optimization Results:")
print("Objective Function Value:", model.obj())
print("Acres of Wheat:", model.w())
print("Acres of Corn:", model.c())
print("Acres of Sugar Beets:", model.s())
print("Tons of Wheat Produced:", model.W())
print("Tons of Corn Produced:", model.C())
print("Tons of Sugar Beets Produced:", model.S())
print("Tons of Wheat Purchased:", model.Wp())
print("Tons of Corn Purchased:", model.Cp())
print("Tons of Wheat Sold:", model.Ws())
print("Tons of Corn Sold:", model.Cs())
print("Tons of Sugar Beets Sold at Quota Price:", model.Sq())
print("Tons of Sugar Beets Sold at Excess Price:", model.Se())
'''

This code defines the optimization problem using Pyomo, including the decision variables, objective function, and cons

```

✓ 4. Input Problem Data and Test Model Code

```

# Installing pyomo and solver
!pip install -q pyomo
!pip install pandas
!apt-get install -y -qq glpk-utils
!pip install glpk

```

 [Show hidden output](#)

```

from pyomo.environ import *

# Create a Concrete Model
model = ConcreteModel()

# Define decision variables
model.w = Var(domain=NonNegativeReals) # Acres of land devoted to wheat
model.c = Var(domain=NonNegativeReals) # Acres of land devoted to corn
model.s = Var(domain=NonNegativeReals) # Acres of land devoted to sugar beets

# Define additional variables
model.W = Var(domain=NonNegativeReals) # Tons of wheat produced
model.C = Var(domain=NonNegativeReals) # Tons of corn produced
model.S = Var(domain=NonNegativeReals) # Tons of sugar beets produced
model.Wp = Var(domain=NonNegativeReals) # Tons of wheat purchased from wholesaler
model.Cp = Var(domain=NonNegativeReals) # Tons of corn purchased from wholesaler
model.Ws = Var(domain=NonNegativeReals) # Tons of wheat sold
model.Cs = Var(domain=NonNegativeReals) # Tons of corn sold

```

```

model.Sq = Var(domain=NonNegativeReals) # Tons of sugar beets sold at quota price
model.Se = Var(domain=NonNegativeReals) # Tons of sugar beets sold at excess price

# Define objective function
model.obj = Objective(expr=150 * model.w + 230 * model.c + 260 * model.s +
                      238 * model.Wp + 210 * model.Cp -
                      170 * model.Ws - 150 * model.Cs -
                      36 * model.Sq - 10 * model.Se,
                      sense=minimize)

# Define constraints
model.land_constraint = Constraint(expr=model.w + model.c + model.s <= 500)

model.wheat_production = Constraint(expr=model.W == 2.5 * model.w)
model.wheat_usage = Constraint(expr=model.W + model.Wp >= 200)
model.wheat_sold = Constraint(expr=model.Ws == model.W + model.Wp - 200)

model.corn_production = Constraint(expr=model.C == 3 * model.c)
model.corn_usage = Constraint(expr=model.C + model.Cp >= 240)
model.corn_sold = Constraint(expr=model.Cs == model.C + model.Cp - 240)

model.sugar_beet_production = Constraint(expr=model.S == 20 * model.s)
model.sugar_beet_sold = Constraint(expr=model.Sq + model.Se == model.S)
model.sugar_beet_quota = Constraint(expr=model.Sq <= 6000)

# Solve the optimization problem
solver = SolverFactory('glpk') # Specify the solver (e.g., 'glpk', 'gurobi', 'cplex')
results = solver.solve(model)

# Print the results
print("Optimization Results:")
print("Objective Function Value:", model.obj())
print("Acres of Wheat:", model.w())
print("Acres of Corn:", model.c())
print("Acres of Sugar Beets:", model.s())
print("Tons of Wheat Produced:", model.W())
print("Tons of Corn Produced:", model.C())
print("Tons of Sugar Beets Produced:", model.S())
print("Tons of Wheat Purchased:", model.Wp())
print("Tons of Corn Purchased:", model.Cp())
print("Tons of Wheat Sold:", model.Ws())
print("Tons of Corn Sold:", model.Cs())
print("Tons of Sugar Beets Sold at Quota Price:", model.Sq())
print("Tons of Sugar Beets Sold at Excess Price:", model.Se())

```

```

↗ Optimization Results:
Objective Function Value: -118600.0
Acres of Wheat: 120.0
Acres of Corn: 80.0
Acres of Sugar Beets: 300.0
Tons of Wheat Produced: 300.0
Tons of Corn Produced: 240.0
Tons of Sugar Beets Produced: 6000.0
Tons of Wheat Purchased: 0.0
Tons of Corn Purchased: 0.0
Tons of Wheat Sold: 100.0
Tons of Corn Sold: 0.0
Tons of Sugar Beets Sold at Quota Price: 6000.0
Tons of Sugar Beets Sold at Excess Price: 0.0

```

✓ 5. Correct The Model Code to Test Mathematical Model (if applicable)

