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 (Objctive):\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 (Objctive):
 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
```

```
Collecting anthropic
  Downloading anthropic-0.28.0-py3-none-any.whl (862 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 862.7/862.7 kB 9.1 MB/s eta 0:00:00
Requirement already satisfied: anyio<5,>=3.5.0 in /usr/local/lib/python3.10/dist-packages (from anthropic) (3.7.1)
Requirement already satisfied: distro<2,>=1.7.0 in /usr/lib/python3/dist-packages (from anthropic) (1.7.0)
Collecting httpx<1,>=0.23.0 (from anthropic)
  Downloading httpx-0.27.0-py3-none-any.whl (75 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 75.6/75.6 kB 9.3 MB/s eta 0:00:00
Collecting jiter<1,>=0.4.0 (from anthropic)
  Downloading jiter-0.4.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (328 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 328.3/328.3 kB 33.9 MB/s eta 0:00:00
Requirement already satisfied: pydantic<3,>=1.9.0 in /usr/local/lib/python3.10/dist-packages (from anthropic) (2.7.3)
Requirement already satisfied: sniffio in /usr/local/lib/python3.10/dist-packages (from anthropic) (1.3.1)
Requirement already satisfied: tokenizers>=0.13.0 in /usr/local/lib/python3.10/dist-packages (from anthropic) (0.19.1)
Requirement already satisfied: typing-extensions<5,>=4.7 in /usr/local/lib/python3.10/dist-packages (from anthropic) (4.
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.10/dist-packages (from anyio<5,>=3.5.0->anthropic) (3
Requirement already satisfied: exceptiongroup in /usr/local/lib/python3.10/dist-packages (from anyio<5,>=3.5.0->anthropi
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages (from httpx<1,>=0.23.0->anthropic) (20
Collecting httpcore==1.* (from httpx<1,>=0.23.0->anthropic)
  Downloading httpcore-1.0.5-py3-none-any.whl (77 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 77.9/77.9 kB 10.5 MB/s eta 0:00:00
Collecting h11<0.15,>=0.13 (from httpcore==1.*->httpx<1,>=0.23.0->anthropic)
  Downloading h11-0.14.0-py3-none-any.whl (58 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 58.3/58.3 kB 7.6 MB/s eta 0:00:00
Requirement already satisfied: annotated-types>=0.4.0 in /usr/local/lib/python3.10/dist-packages (from pydantic<3,>=1.9.
Requirement already satisfied: pydantic-core==2.18.4 in /usr/local/lib/python3.10/dist-packages (from pydantic<3,>=1.9.0
Requirement already satisfied: huggingface-hub<1.0,>=0.16.4 in /usr/local/lib/python3.10/dist-packages (from tokenizers>
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.16.4->t
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0
Requirement already satisfied: packaging>=20.9 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.16.4
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.16.4->t
Requirement already satisfied: tqdm>=4.42.1 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.16.
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->huggi
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->huggingface
Installing collected packages: jiter, h11, httpcore, httpx, anthropic
Successfully installed anthropic-0.28.0 h11-0.14.0 httpcore-1.0.5 httpx-0.27.0 jiter-0.4.1
```

```
# 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)
```

```
Let's define the following variables for this 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 Ws be the number of tons of wheat sold
- Let Cs be the number of tons of corn sold
- Let Wb be the number of tons of wheat bought from the wholesaler
- Let Cb be the number of tons of corn bought from the wholesaler
- Let Ss be the number of tons of sugar beets sold at the regular price of $36/T
- Let Se be the number of tons of sugar beets sold at the excess price of $10/T
```

```
The goal is to minimize the total costs, considering planting costs, buying costs, and sales revenues.
```

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)
```

```
To formulate the objective function for this optimization problem, we need to consider all the costs and revenues associ

Objective Function:
Minimize:
150W + 230C + 260S + 238Wb + 210Cb - 170Ws - 150Cs - 36Ss - 10Se

Explanation:
- 150W represents the planting cost for wheat acres
- 230C represents the planting cost for corn acres
- 260S represents the planting cost for sugar beet acres
- 238Wb represents the cost of buying wheat from the wholesaler (40% more than the selling price of $170/T)
- 210Cb represents the cost of buying corn from the wholesaler (40% more than the selling price of $150/T)
- 170Ws represents the revenue from selling wheat (negative cost)
- 150Cs represents the revenue from selling corn (negative cost)
- 36Ss represents the revenue from selling sugar beets at the regular price (negative cost)
- 10Se represents the revenue from selling sugar beets at the excess price (negative cost)

The objective function aims to minimize the total costs, considering all the planting costs, buying costs, and sales rev
```

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 the optimization problem:

1. Land constraint:
   W + C + S ≤ 500

2. Wheat production and purchase constraint:
   2.5W + Wb - Ws ≥ 200

3. Corn production and purchase constraint:
   3C + Cb - Cs ≥ 240

4. Sugar beet production constraint:
   20S - Ss - Se = 0

5. Sugar beet quota constraint:
   Ss ≤ 6000

6. Non-negativity constraints:
   W, C, S, Ws, Cs, Wb, Cb, Ss, Se ≥ 0

Explanation of the constraints:

1. The total land used for wheat, corn, and sugar beets cannot exceed the available 500 acres.

2. The amount of wheat produced (yield per acre multiplied by acres) plus the amount bought minus the amount sold should
```

```
3. The amount of corn produced (yield per acre multiplied by acres) plus the amount bought minus the amount sold should

4. The amount of sugar beets produced (yield per acre multiplied by acres) should be equal to the sum of the amount sold

5. The amount of sugar beets sold at the regular price should be less than or equal to the quota of 6000 tons.

6. All decision variables and additional variables should be non-negative, as negative values would not make sense in th

These constraints, along with the objective function, complete the formulation of the optimization problem for the farme
```

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 (Objectvie): 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=4069,
    messages=[
        {"role": "user", "content": (response_p11 + response_p12 + response_p13)}
    ]
)

response_p2 = message2.content[0].text


# Print response
print(response_p2)
```

~~~
```python
from pyomo.environ import *

# Create a Pyomo model
model = ConcreteModel()

# Define decision variables
model.W = Var(domain=NonNegativeReals)  # Acres of land devoted to wheat production
model.C = Var(domain=NonNegativeReals)  # Acres of land devoted to corn production
model.S = Var(domain=NonNegativeReals)  # Acres of land devoted to sugar beet production

# Define additional variables
model.Ws = Var(domain=NonNegativeReals)  # Tons of wheat sold
model.Cs = Var(domain=NonNegativeReals)  # Tons of corn sold
model.Wb = Var(domain=NonNegativeReals)  # Tons of wheat bought from the wholesaler
model.Cb = Var(domain=NonNegativeReals)  # Tons of corn bought from the wholesaler
model.Ss = Var(domain=NonNegativeReals)  # Tons of sugar beets sold at the regular price
model.Se = Var(domain=NonNegativeReals)  # Tons of sugar beets sold at the excess price

# Define the objective function
model.obj = Objective(expr=150*model.W + 230*model.C + 260*model.S + 238*model.Wb + 210*model.Cb - 170*model.Ws - 150*

# Define constraints
model.land_constraint = Constraint(expr=model.W + model.C + model.S <= 500)
model.wheat_constraint = Constraint(expr=2.5*model.W + model.Wb - model.Ws >= 200)
model.corn_constraint = Constraint(expr=3*model.C + model.Cb - model.Cs >= 240)
model.sugar_beet_constraint = Constraint(expr=20*model.S - model.Ss - model.Se == 0)
model.sugar_beet_quota_constraint = Constraint(expr=model.Ss <= 6000)

# Solve the optimization problem
solver = SolverFactory('glpk')  # You can use any solver available in Pyomo
results = solver.solve(model)

# Print the results
print("Objective value:", model.obj())
print("Acres of land devoted to wheat:", model.W())
print("Acres of land devoted to corn:", model.C())
print("Acres of land devoted to sugar beets:", model.S())
print("Tons of wheat sold:", model.Ws())
print("Tons of corn sold:", model.Cs())
print("Tons of wheat bought from the wholesaler:", model.Wb())
print("Tons of corn bought from the wholesaler:", model.Cb())
~~~

```
print("Tons of sugar beets sold at the excess price:", model.Se())
```

This code defines the decision variables, additional variables, objective function, and constraints as discussed in th

Note: Make sure you have Pyomo and a solver (such as GLPK) installed to run this code. You can install them using pip:

```
pip install pyomo
pip install glpk
```

The code uses sample data in the sense that it assumes a specific objective function and constraint coefficients based

4. Input Problem Data and Test Model Code

```
# Installing pyomo nd 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 Pyomo model
model = ConcreteModel()

# Define decision variables
model.W = Var(domain=NonNegativeReals)  # Acres of land devoted to wheat production
model.C = Var(domain=NonNegativeReals)  # Acres of land devoted to corn production
model.S = Var(domain=NonNegativeReals)  # Acres of land devoted to sugar beet production

# Define additional variables
model.Ws = Var(domain=NonNegativeReals)  # Tons of wheat sold
model.Cs = Var(domain=NonNegativeReals)  # Tons of corn sold
model.Wb = Var(domain=NonNegativeReals)  # Tons of wheat bought from the wholesaler
model.Cb = Var(domain=NonNegativeReals)  # Tons of corn bought from the wholesaler
model.Ss = Var(domain=NonNegativeReals)  # Tons of sugar beets sold at the regular price
model.Se = Var(domain=NonNegativeReals)  # Tons of sugar beets sold at the excess price

# Define the objective function
model.obj = Objective(expr=150*model.W + 230*model.C + 260*model.S + 238*model.Wb + 210*model.Cb - 170*model.Ws - 150*model.C

# Define constraints
model.land_constraint = Constraint(expr=model.W + model.C + model.S <= 500)
model.wheat_constraint = Constraint(expr=2.5*model.W + model.Wb - model.Ws >= 200)
model.corn_constraint = Constraint(expr=3*model.C + model.Cb - model.Cs >= 240)
model.sugar_beet_constraint = Constraint(expr=20*model.S - model.Ss - model.Se == 0)
model.sugar_beet_quota_constraint = Constraint(expr=model.Ss <= 6000)

# Solve the optimization problem
solver = SolverFactory('glpk')  # You can use any solver available in Pyomo
results = solver.solve(model)

# Print the results
print("Objective value:", model.obj())
print("Acres of land devoted to wheat:", model.W())
print("Acres of land devoted to corn:", model.C())
print("Acres of land devoted to sugar beets:", model.S())
print("Tons of wheat sold:", model.Ws())
print("Tons of corn sold:", model.Cs())
print("Tons of wheat bought from the wholesaler:", model.Wb())
print("Tons of corn bought from the wholesaler:", model.Cb())
print("Tons of sugar beets sold at the regular price:", model.Ss())
print("Tons of sugar beets sold at the excess price:", model.Se())
```

```
Objective value: -118599.99999999999
Acres of land devoted to wheat: 120.0
Acres of land devoted to corn: 80.0
Acres of land devoted to sugar beets: 300.0
Tons of wheat sold: 99.9999999999999
Tons of corn sold: 0.0
Tons of wheat bought from the wholesaler: 0.0
Tons of corn bought from the wholesaler: 0.0
Tons of sugar beets sold at the regular price: 6000.0
Tons of sugar beets sold at the excess price: 0.0
```

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