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
prompt1_path = '/content/drive/MyDrive/Thesis/Prompts/Prompt1_MathematicalModel.txt'
prompt2_path = '/content/drive/MyDrive/Thesis/Prompts/Prompt2_PyomoCode.txt'
problem_desc_path = '/content/drive/MyDrive/Thesis/ProblemDescriptions/NL/NL1.txt'

prompt1_file = open(prompt1_path, "r")
prompt2_file = open(prompt2_path, "r")
problem_desc_file = open(problem_desc_path, "r")

prompt1 = prompt1_file.read()
print("Prompt 1:\n", prompt1)

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

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

```
Prompt 1:
 Please write a mathematical optimization model for this problem. Include parameters, decision variables, the objective
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:
 A firm that packs refreshments and beers, situated in the province of Valencia (Spain) employs the same syrup to produc
49000 times X1 minus X1 squared plus 30 times X2 minus two times X2 squared.
It costs 150 dollars to buy and process each hectolitre of syrup. The S1 packaging line has a net capacity of producing
```

```
!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: Create Mathematical Model

```
message = client.messages.create(
    model="claude-3-opus-20240229",
    system=prompt1,
    max_tokens=4096,
    messages=[
        {"role": "user", "content": problem_desc}
    ]
)
```

```python
response_p1 = message.content[0].text


# Print response
print(response_p1)
```

```
To solve this problem, we need to formulate a mathematical optimization model that maximizes the firm's daily profits wh

Parameters:
- syrup_ratio_COLI = 40 units of 1.5 l COLI per hectolitre of syrup
- syrup_ratio_PEPSA = 20 units of 1.5 l PEPSA per hectolitre of syrup
- cost_syrup = $150 per hectolitre of syrup
- production_capacity = 7100 units per hour
- working_hours = 8 hours per day

Decision Variables:
- X1: units of 1.5 l COLI produced daily
- X2: units of 1.5 l PEPSA produced daily

Objective Function:
Maximize daily profit (Z):
Z = 49000X1 - X1^2 + 30X2 - 2X2^2 - cost_syrup * (X1/syrup_ratio_COLI + X2/syrup_ratio_PEPSA)

Constraints:
1. Production capacity constraint:
   X1 + X2 <= production_capacity * working_hours

2. PEPSA production constraint:
   X2 >= 0.5 * X1

3. Non-negativity constraints:
   X1 >= 0
   X2 >= 0

The complete mathematical optimization model:

Maximize:
Z = 49000X1 - X1^2 + 30X2 - 2X2^2 - 150 * (X1/40 + X2/20)

Subject to:
1. X1 + X2 <= 56800
2. X2 >= 0.5 * X1
3. X1 >= 0
4. X2 >= 0

In this model, the objective function represents the daily profit, which includes the income from selling COLI and PEPSA

Solving this optimization model will provide the optimal daily production quantities for COLI (X1) and PEPSA (X2) that m
```

2. Prompt 2: Write Code based on Mathematical Model

```python
message2 = client.messages.create(
    model="claude-3-opus-20240229",
    system=prompt2,
    max_tokens=4069,
    messages=[
        {"role": "user", "content": response_p1}
    ]
)

response_p2 = message2.content[0].text


# Print response
print(response_p2)
```

````
Here's the Python code using Pyomo to solve the optimization problem:

```python
from pyomo.environ import *

# Create the model
model = ConcreteModel()

# Define the parameters
model.syrup_ratio_COLI = 40
model.syrup_ratio_PEPSA = 20
model.cost_syrup = 150
model.production_capacity = 7100
model.working_hours = 8

# Define the decision variables
model.X1 = Var(domain=NonNegativeReals)
model.X2 = Var(domain=NonNegativeReals)
````
````

```
# Define the objective function
model.profit = Objective(expr=49000 * model.X1 - model.X1**2 + 30 * model.X2 - 2 * model.X2**2 -
                         model.cost_syrup * (model.X1 / model.syrup_ratio_COLI + model.X2 / model.syrup_ratio_PEPSA),
                         sense=maximize)

# Define the constraints
model.capacity_constraint = Constraint(expr=model.X1 + model.X2 <= model.production_capacity * model.working_hours)
model.pepsa_constraint = Constraint(expr=model.X2 >= 0.5 * model.X1)

# Solve the model
solver = SolverFactory('ipopt')
solver.solve(model)

# Print the optimal solution
print("Optimal Solution:")
print("X1 (COLI):", model.X1.value)
print("X2 (PEPSA):", model.X2.value)
print("Daily Profit:", model.profit.expr())
```

```
In this code:
- We import the necessary components from the `pyomo.environ` module.
- We create a `ConcreteModel` object named `model` to represent our optimization problem.
- We define the parameters using the values provided in the problem statement.
- We define the decision variables `X1` and `X2` as non-negative real variables.
- We define the objective function `profit` using the expression for the daily profit, which includes the income from
- We define the constraints:
  - `capacity_constraint` ensures that the total production does not exceed the daily production capacity.
  - `pepsa_constraint` ensures that the firm produces at least half the amount of PEPSA compared to COLI.
- We create an instance of the solver (in this case, 'ipopt') and solve the model.
- We print the optimal solution, including the values of `X1` (COLI production), `X2` (PEPSA production), and the dail

Note that this code uses the parameter values provided in the problem statement, so no sample data is used here.

To run this code, make sure you have Pyomo and a suitable solver (e.g., IPOPT) installed. You can install Pyomo and IP

```
pip install pyomo
```

4. Input Problem Data and Test Model Code

```
# Installing pyomo nd solver
!pip install -q pyomo
!wget -N -q "https://matematica.unipv.it/gualandi/solvers/ipopt-linux64.zip"
!unzip -o -q ipopt-linux64
```

```
12.8/12.8 MB 81.8 MB/s eta 0:00:00
49.6/49.6 kB 6.2 MB/s eta 0:00:00
```

```
from pyomo.environ import *

# Create the model
model = ConcreteModel()

# Define the parameters
model.syrup_ratio_COLI = 40
model.syrup_ratio_PEPSA = 20
model.cost_syrup = 150
model.production_capacity = 7100
model.working_hours = 8

# Define the decision variables
model.X1 = Var(domain=NonNegativeReals)
model.X2 = Var(domain=NonNegativeReals)

# Define the objective function
model.profit = Objective(expr=49000 * model.X1 - model.X1**2 + 30 * model.X2 - 2 * model.X2**2 -
                         model.cost_syrup * (model.X1 / model.syrup_ratio_COLI + model.X2 / model.syrup_ratio_PEPSA),
                         sense=maximize)

# Define the constraints
model.capacity_constraint = Constraint(expr=model.X1 + model.X2 <= model.production_capacity * model.working_hours)
model.pepsa_constraint = Constraint(expr=model.X2 >= 0.5 * model.X1)

# Solve the model
solver = SolverFactory('ipopt')
solver.solve(model)

# Print the optimal solution
print("Optimal Solution:")
print("X1 (COLI):", model.X1.value)
print("X2 (PEPSA):", model.X2.value)
```
```

```
print("Daily Profit:", model.profit.expr())
```

```
Optimal Solution:
X1 (COLI): 16335.833333340004
X2 (PEPSA): 8167.91666666004
Daily Profit: 400289176.04199195
```

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