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

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

response_p1 = message.content[0].text


# Print response
print(response_p1)
```

```
To formulate the mathematical optimization model for this problem, we need to define the parameters, decision variables,

Parameters:
- 1 hectolitre of syrup produces 40 units of 1.5 l COLI and 20 units of 1.5 l PEPSA
- Cost of buying and processing 1 hectolitre of syrup: $150
- S1 packaging line capacity: 7100 units/hour
- Working hours: 8 hours/day, 5 days/week

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

Objective Function:
- Maximize daily profit (Z):
Z = 49000X1 - X1^2 + 30X2 - 2X2^2 - 150(X1/40 + X2/20)

Constraints:
1. Production capacity constraint:
X1 + X2 ≤ 7100 × 8

2. PEPSA production should be at least half of COLI production:
X2 ≥ 0.5X1

3. Non-negativity constraints:
X1 ≥ 0
X2 ≥ 0

Complete Optimization Model:
Maximize Z = 49000X1 - X1^2 + 30X2 - 2X2^2 - 150(X1/40 + X2/20)
Subject to:
X1 + X2 ≤ 56800
X2 ≥ 0.5X1
X1 ≥ 0
X2 ≥ 0

The objective is to maximize the daily profit (Z) by determining the optimal number of 1.5 l COLI (X1) and 1.5 l PEPSA (
```

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 pyomo code for the given optimization problem:

```python
from pyomo.environ import *

# Create a concrete model
model = ConcreteModel()

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

# Define objective function
model.profit = Objective(expr=49000*model.X1 - model.X1**2 + 30*model.X2 - 2*model.X2**2 - 150*(model.X1/40 + model.X2/2
````
````

```
# Define constraints
model.capacity = Constraint(expr=model.X1 + model.X2 <= 7100 * 8)
model.pepsa_coli_ratio = Constraint(expr=model.X2 >= 0.5 * model.X1)

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

# Print the results
print("Optimization Results:")
print("X1 (1.5 l COLI):", model.X1.value)
print("X2 (1.5 l PEPSA):", model.X2.value)
print("Daily Profit:", model.profit.expr())
```

```
This code uses the Pyomo optimization modeling library to formulate and solve the problem. Here's a breakdown of the cod

1. We import the necessary components from the pyomo.environ module.
2. We create a concrete model using `ConcreteModel()`.
3. We define the decision variables `X1` and `X2` using `Var()` and specify that they should be non-negative using `with
4. We define the objective function using `Objective()` and set the `sense` to `maximize` to indicate that we want to ma
5. We define the constraints using `Constraint()`:
   - The production capacity constraint: `model.X1 + model.X2 <= 7100 * 8`
   - The PEPSA-COLI production ratio constraint: `model.X2 >= 0.5 * model.X1`
6. We create a solver using `SolverFactory('ipopt')` and solve the optimization problem using `solver.solve(model)`.
7. Finally, we print the results, including the optimal values of `X1` and `X2`, and the daily profit.

Note: This code assumes that you have the Pyomo library and the IPOPT solver installed. If you don't have IPOPT, you can

In this code, we directly use the given parameters and constraints from the problem statement, so there is no need for s
```

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 78.9 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 49.6/49.6 kB 5.9 MB/s eta 0:00:00
```

```
from pyomo.environ import *

# Create a concrete model
model = ConcreteModel()

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

# Define objective function
model.profit = Objective(expr=49000*model.X1 - model.X1**2 + 30*model.X2 - 2*model.X2**2 - 150*(model.X1/40 + model.X2/20), s

# Define constraints
model.capacity = Constraint(expr=model.X1 + model.X2 <= 7100 * 8)
model.pepsa_coli_ratio = Constraint(expr=model.X2 >= 0.5 * model.X1)

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

# Print the results
print("Optimization Results:")
print("X1 (1.5 l COLI):", model.X1.value)
print("X2 (1.5 l PEPSA):", model.X2.value)
print("Daily Profit:", model.profit.expr())
```

```
Optimization Results:
X1 (1.5 l COLI): 16335.833333340004
X2 (1.5 l PEPSA): 8167.91666666004
Daily Profit: 400289176.04199195
```

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