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/NL/NL1.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:
 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 1.1: 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)
```

```
To formulate the variables for this mathematical optimization problem, we need to identify the decision variables, objec

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

    These are the main variables that the firm needs to determine to optimize its daily profits.
```

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:
    The objective is to maximize the daily profit. The daily income is given by the function:
    49000X1 - X1^2 + 30X2 - 2X2^2

    The cost of buying and processing each hectolitre of syrup is $150. Since each hectolitre produces 40 units of COLI (X1)
    150 × (X1/40 + X2/20)

    Therefore, the objective function to maximize the daily profit (income minus cost) is:

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

    This can be simplified to:
    Maximize: 49000X1 - X1^2 + 30X2 - 2X2^2 - 3.75X1 - 7.5X2

    So, the complete objective function is:
    Maximize: 48996.25X1 - X1^2 + 22.5X2 - 2X2^2
```

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)
```

```
Great! Now, let's focus on formulating the constraints for this optimization problem.

    Constraint 1: Production capacity
    The S1 packaging line has a net capacity of producing 7100 1.5 l product units every hour. The firm works 5 days a week
    X1 + X2 ≤ 7100 × 8
    X1 + X2 ≤ 56800

    Constraint 2: Minimum PEPSA production
    The firm is committed to producing at least half the amount of PEPSA compared to COLI due to its weekly target coverage.
    X2 ≥ 0.5 × X1

    Constraint 3: Non-negativity
    The decision variables X1 and X2 represent the number of units produced, which cannot be negative. Therefore, the non-ne
    X1 ≥ 0
    X2 ≥ 0

    In summary, the constraints for this optimization problem are:

    1. X1 + X2 ≤ 56800 (Production capacity)
    2. X2 ≥ 0.5 × X1 (Minimum PEPSA production)
    3. X1 ≥ 0 (Non-negativity)
    4. X2 ≥ 0 (Non-negativity)

    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 (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)
```

```
Here's the Python Pyomo code for the given optimization problem:

    ```python
    from pyomo.environ import *

    # Create a Pyomo model
    model = ConcreteModel()

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

    # Define the objective function
    model.profit = Objective(expr=48996.25 * model.X1 - model.X1**2 + 22.5 * model.X2 - 2 * model.X2**2, sense=maximize)
```
```

```
# Define the constraints
model.production_capacity = Constraint(expr=model.X1 + model.X2 <= 56800)
model.minimum_pepsa = Constraint(expr=model.X2 >= 0.5 * model.X1)

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

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

Explanation:
1. We import the necessary components from the `pyomo.environ` module.
2. We create a `ConcreteModel` named `model` to represent our optimization problem.
3. We define the decision variables `X1` and `X2` using `Var()` and specify their domains as non-negative real numbers u
4. We define the objective function using `Objective()` and provide the expression for the daily profit. The `sense` par
5. We define the constraints using `Constraint()`:
   - The production capacity constraint: `X1 + X2 <= 56800`
   - The minimum PEPSA production constraint: `X2 >= 0.5 * X1`
   Note that the non-negativity constraints are already handled by specifying the domain of the decision variables as `N
6. We create an instance of the solver using `SolverFactory('ipopt')`. You can use any suitable solver available in Pyom
7. We solve the optimization problem using `solver.solve(model)`.
8. Finally, we print the optimal values of the decision variables `X1` and `X2`, as well as the optimal daily profit.

Note: This code assumes that you have Pyomo and a suitable solver (e.g., IPOPT) installed in your Python environment.

In this example, we did not use any sample data as all the necessary data was provided in the problem description.

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 20.5 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 49.6/49.6 kB 2.2 MB/s eta 0:00:00
```

```
from pyomo.environ import *

# Create a Pyomo model
model = ConcreteModel()

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

# Define the objective function
model.profit = Objective(expr=48996.25 * model.X1 - model.X1**2 + 22.5 * model.X2 - 2 * model.X2**2, sense=maximize)

# Define the constraints
model.production_capacity = Constraint(expr=model.X1 + model.X2 <= 56800)
model.minimum_pepsa = Constraint(expr=model.X2 >= 0.5 * model.X1)

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

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

```
Optimal solution:
X1 = 16335.833333340004
X2 = 8167.91666666004
Daily profit = 400289176.04199183
```

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