0. Import Rename notebook ng 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 = anthrop
    # defaults to os.environ.get("ANTHROPIC_API_KEY")
    api_key=os.environ.get("ANTHROPIC_API_KEY"),
)
```

Rename notebook

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 variables for this mathematical optimization problem:

X1: The number of units of the 1.5 l COLI product produced daily.
X2: The number of units of the 1.5 l PEPSA product produced daily.

These are the main decision variables that the firm needs to determine in order to maximize their 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)
```

```
Based on the given information, the objective function for this mathematical optimization problem can be formulated as f

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

This objective function represents the daily profits in dollars that the firm aims to maximize. The terms in the objecti

1. 49000X1 - X1^2: Revenue generated from selling X1 units of the 1.5 l COLI product, considering a quadratic revenue fu
2. 30X2 - 2X2^2: Revenue generated from selling X2 units of the 1.5 l PEPSA product, also considering a quadratic revenu
3. 150(X1/40 + X2/20): Cost of buying and processing the syrup required to produce X1 units of COLI and X2 units of PEPS

The firm's goal is to find the optimal values of X1 and X2 that maximize this objective function, subject to the given c
```

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)
```

Rename notebook

```
Here are th                or this mathematical optimization problem:

1. Syrup processing constraint:
   X1/40 + X2/20 ≤ Total available syrup (in hectolitres) per day

2. Production capacity constraint:
   X1 + X2 ≤ 7100 * 8
   (The S1 packaging line can produce 7100 units per hour, and the firm works 8 hours per day)

3. Production ratio constraint:
   X2 ≥ 0.5 * X1
   (The firm is committed to producing at least half the amount of PEPSA compared to COLI)

4. Non-negativity constraints:
   X1 ≥ 0
   X2 ≥ 0
   (The number of units produced for each product cannot be negative)

These constraints ensure that the optimal solution satisfies the production capacity, syrup availability, production rat
```

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 a Python code using Pyomo to solve the optimization problem:

```python
from pyomo.environ import *

# Create a Pyomo model
model = ConcreteModel()

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

# Define the 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 the constraints
model.syrup_constraint = Constraint(expr=model.X1/40 + model.X2/20 <= 100)  # Sample data: 100 hectolitres of syrup avai
model.capacity_constraint = Constraint(expr=model.X1 + model.X2 <= 7100 * 8)
model.ratio_constraint = Constraint(expr=model.X2 >= 0.5 * model.X1)

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

# Print the results
print("Optimal solution:")
print(f"X1 (COLI) = {model.X1.value:.2f} units")
print(f"X2 (PEPSA) = {model.X2.value:.2f} units")
print(f"Maximum daily profit = ${model.profit.expr():.2f}")
```
````

```
In this code:

1. We impor[illegible] components from the Pyomo library.
2. We creat[illegible] using `ConcreteModel()`.
3. We define the decision variables `X1` and `X2` as non-negative real numbers using `Var(within=NonNegativeReals)`.
4. We define the objective function `profit` using the given quadratic expressions and the cost of syrup.
5. We define the constraints:
   - `syrup_constraint`: Limiting the total syrup usage to 100 hectolitres per day (sample data).
   - `capacity_constraint`: Limiting the total production to the capacity of the S1 packaging line (7100 units per hour
   - `ratio_constraint`: Ensuring that the production of PEPSA is at least half the production of COLI.
6. We create an instance of the IPOPT solver using `SolverFactory('ipopt')` and solve the optimization problem.
7. We print the optimal values of `X1` and `X2`, as well as the maximum daily profit.

Note: In the `syrup_constraint`, we used sample data of 100 hectolitres of syrup available per day. You can replace this

This code will solve the optimization problem and provide the optimal daily production quantities for COLI and PEPSA, as
```

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 21.2 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 49.6/49.6 kB 4.0 MB/s eta 0:00:00
```

```
from pyomo.environ import *

# Create a Pyomo model
model = ConcreteModel()

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

# Define the 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 the constraints
model.syrup_constraint = Constraint(expr=model.X1/40 + model.X2/20 <= 1000)  # Sample data: 100 hectolitres of syrup availabl
model.capacity_constraint = Constraint(expr=model.X1 + model.X2 <= 7100 * 8)
model.ratio_constraint = Constraint(expr=model.X2 >= 0.5 * model.X1)

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

# Print the results
print("Optimal solution:")
print(f"X1 (COLI) = {model.X1.value:.2f} units")
print(f"X2 (PEPSA) = {model.X2.value:.2f} units")
print(f"Maximum daily profit = ${model.profit.expr():.2f}")
```

```
Optimal solution:
X1 (COLI) = 16335.83 units
X2 (PEPSA) = 8167.92 units
Maximum daily profit = $400289176.04
```

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

Rename notebook