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/LP/LP3.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:
 The PRODA, S.A. industrial products firm has to face the problem of scheduling
the weekly production of its three products (P1, P2 and P3). These products are
sold to large industrial firms and PRODA, S.A. wishes to supply its products in
quantities that are more profitable for it.

Each product entails three operations contributing to the costs: smelting; mechanisation; assembly and
packaging. The smelting operations for products P1 and P2 could be subcontracted, but the smelting operation for product
preventing the use of subcontracts. PRODA also want to know, how much they should subcontract.

For product P1 the direct unit costs of all possible operations are:
- smelting at PRODA: 0.30$
- subcontracted smelting: 0.50$
- mechanisation: 0.20$
- Assembly and packaging: 0.3$
The unit sales price is 1.50$.

For product P2 the direct unit costs of all possible operations are:
- smelting at PRODA: 0.50$
- subcontracted smelting: 0.60$
- mechanisation: 0.10$
- Assembly and packaging: 0.20$
The unit sales price is 1.80$.

For product P3 the direct unit costs of all possible operations are:
- smelting at PRODA: 0.40$
- mechanisation: 0.27$
- Assembly and packaging: 0.20$
The unit sales price is 1.97$.

Each unit of product P1 requires 6 min of smelting time (if performed at PRODA, S.A.), 6 min of mechanisation time and 3
The objective is to maximize weekly profits.
```

```
!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}
    ]
)

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

```
# Print response
print(response_p1)
```

```
To solve this problem, we can formulate a linear programming model. Let's define the decision variables, parameters, obj

Decision Variables:
- Let x1 be the number of units of product P1 produced with smelting at PRODA.
- Let y1 be the number of units of product P1 produced with subcontracted smelting.
- Let x2 be the number of units of product P2 produced with smelting at PRODA.
- Let y2 be the number of units of product P2 produced with subcontracted smelting.
- Let x3 be the number of units of product P3 produced.

Parameters:
- Let p1, p2, and p3 be the unit sales prices of products P1, P2, and P3, respectively.
- Let c1, c2, and c3 be the unit costs of smelting at PRODA for products P1, P2, and P3, respectively.
- Let s1 and s2 be the unit costs of subcontracted smelting for products P1 and P2, respectively.
- Let m1, m2, and m3 be the unit costs of mechanisation for products P1, P2, and P3, respectively.
- Let a1, a2, and a3 be the unit costs of assembly and packaging for products P1, P2, and P3, respectively.
- Let t1, t2, and t3 be the smelting times (in minutes) for products P1, P2, and P3, respectively.
- Let u1, u2, and u3 be the mechanisation times (in minutes) for products P1, P2, and P3, respectively.
- Let v1, v2, and v3 be the assembly and packaging times (in minutes) for products P1, P2, and P3, respectively.
- Let S, M, and A be the weekly capacities (in minutes) for smelting, mechanisation, and assembly and packaging, respect

Objective Function:
- Maximize: (p1 - c1 - m1 - a1) * x1 + (p1 - s1 - m1 - a1) * y1 + (p2 - c2 - m2 - a2) * x2 + (p2 - s2 - m2 - a2) * y2 +

Constraints:
- Smelting capacity: t1 * x1 + t2 * x2 + t3 * x3 ≤ S
- Mechanisation capacity: u1 * (x1 + y1) + u2 * (x2 + y2) + u3 * x3 ≤ M
- Assembly and packaging capacity: v1 * (x1 + y1) + v2 * (x2 + y2) + v3 * x3 ≤ A
- Non-negativity: x1, y1, x2, y2, x3 ≥ 0

The objective function maximizes the total weekly profit by considering the profit per unit of each product and the quan
```

2. Prompt 2: Write Code based on Mathematical Model

```
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 Pyomo code for the optimization problem:

```python
```
```

```
from pyomo.environ import *

# Create the model
model = ConcreteModel()

# Define the decision variables
model.x1 = Var(within=NonNegativeReals)
model.y1 = Var(within=NonNegativeReals)
model.x2 = Var(within=NonNegativeReals)
model.y2 = Var(within=NonNegativeReals)
model.x3 = Var(within=NonNegativeReals)

# Define the parameters (sample data)
p1, p2, p3 = 100, 120, 90  # Unit sales prices
c1, c2, c3 = 40, 50, 35    # Unit costs of smelting at PRODA
s1, s2 = 50, 60            # Unit costs of subcontracted smelting
m1, m2, m3 = 10, 15, 12    # Unit costs of mechanisation
a1, a2, a3 = 20, 25, 18    # Unit costs of assembly and packaging
t1, t2, t3 = 5, 6, 4       # Smelting times (in minutes)
u1, u2, u3 = 2, 3, 2       # Mechanisation times (in minutes)
v1, v2, v3 = 3, 4, 3       # Assembly and packaging times (in minutes)
S = 2400                   # Weekly smelting capacity (in minutes)
M = 1200                   # Weekly mechanisation capacity (in minutes)
A = 1800                   # Weekly assembly and packaging capacity (in minutes)

# Define the objective function
model.profit = Objective(expr=(p1 - c1 - m1 - a1) * model.x1 + (p1 - s1 - m1 - a1) * model.y1 +
                              (p2 - c2 - m2 - a2) * model.x2 + (p2 - s2 - m2 - a2) * model.y2 +
                              (p3 - c3 - m3 - a3) * model.x3, sense=maximize)

# Define the constraints
model.smelting_capacity = Constraint(expr=t1 * model.x1 + t2 * model.x2 + t3 * model.x3 <= S)
model.mechanisation_capacity = Constraint(expr=u1 * (model.x1 + model.y1) + u2 * (model.x2 + model.y2) + u3 * model.x3 <
model.assembly_capacity = Constraint(expr=v1 * (model.x1 + model.y1) + v2 * (model.x2 + model.y2) + v3 * model.x3 <= A)

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

# Print the results
print("Optimal Profit:", model.profit())
print("x1 =", model.x1())
print("y1 =", model.y1())
print("x2 =", model.x2())
print("y2 =", model.y2())
print("x3 =", model.x3())
```

Note: The code uses sample data for the parameters, as indicated by the comments. You can replace these values with the

To run the code, make sure you have Pyomo and a solver (e.g., GLPK) installed. You can install Pyomo using `pip install

The code creates a Pyomo model, defines the decision variables and parameters, sets up the objective function and constr

4. Input Problem Data and Test Model Code

```
# Installing pyomo nd solver
!pip install -q pyomo
!apt-get install -y -qq glpk-utils
!pip install glpk
```

Show hidden output

```
from pyomo.environ import *

# Create the model
model = ConcreteModel()

# Define the decision variables
model.x1 = Var(within=NonNegativeReals)
model.y1 = Var(within=NonNegativeReals)
model.x2 = Var(within=NonNegativeReals)
model.y2 = Var(within=NonNegativeReals)
model.x3 = Var(within=NonNegativeReals)

# Define the parameters (sample data)
p1, p2, p3 = 1.50, 1.80, 1.97  # Unit sales prices
c1, c2, c3 = 0.3, 0.5, 0.4    # Unit costs of smelting at PRODA
s1, s2 = 0.5, 0.6            # Unit costs of subcontracted smelting
m1, m2, m3 = 0.2, 0.1, 0.27    # Unit costs of mechanisation
a1, a2, a3 = 0.3, 0.2, 0.2    # Unit costs of assembly and packaging
t1, t2, t3 = 6, 10, 8       # Smelting times (in minutes)
u1, u2, u3 = 6, 3, 8       # Mechanisation times (in minutes)
```

```
v1, v2, v3 = 3, 2, 2       # Assembly and packaging times (in minutes)
S = 8000                   # Weekly smelting capacity (in minutes)
M = 12000                   # Weekly mechanisation capacity (in minutes)
A = 10000                   # Weekly assembly and packaging capacity (in minutes)

# Define the objective function
model.profit = Objective(expr=(p1 - c1 - m1 - a1) * model.x1 + (p1 - s1 - m1 - a1) * model.y1 +
                              (p2 - c2 - m2 - a2) * model.x2 + (p2 - s2 - m2 - a2) * model.y2 +
                              (p3 - c3 - m3 - a3) * model.x3, sense=maximize)

# Define the constraints
model.smelting_capacity = Constraint(expr=t1 * model.x1 + t2 * model.x2 + t3 * model.x3 <= S)
model.mechanisation_capacity = Constraint(expr=u1 * (model.x1 + model.y1) + u2 * (model.x2 + model.y2) + u3 * model.x3 <= M)
model.assembly_capacity = Constraint(expr=v1 * (model.x1 + model.y1) + v2 * (model.x2 + model.y2) + v3 * model.x3 <= A)

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

# Print the results
print("Optimal Profit:", model.profit())
print("x1 =", model.x1())
print("y1 =", model.y1())
print("x2 =", model.x2())
print("y2 =", model.y2())
print("x3 =", model.x3())
```

```
Optimal Profit: 3680.0000000000005
x1 = 0.0
y1 = 0.0
x2 = 800.0
y2 = 3200.0
x3 = 0.0
```

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