

## 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 the PRODA, S.A. production scheduling problem, we can formulate a linear programming model. Let's define the de

Decision Variables:

- Let  $x_1$ ,  $x_2$ , and  $x_3$  be the number of units produced for products P1, P2, and P3, respectively.
- Let  $s_1$  and  $s_2$  be the number of units subcontracted for smelting operation for products P1 and P2, respectively.

Parameters:

- Unit costs for smelting, mechanisation, and assembly and packaging for each product.
- Unit sales prices for each product.
- Time requirements for each operation per unit of product.
- Weekly capacities for each operation.

Objective Function:

Maximize weekly profit = (Revenue from sales) - (Production costs) - (Subcontracting costs)

Max  $Z = 1.50x_1 + 1.80x_2 + 1.97x_3 - (0.30(x_1-s_1) + 0.20x_1 + 0.30x_1) - (0.50(x_2-s_2) + 0.10x_2 + 0.20x_2) - (0.40x_3 + 0.27x_3)$

Constraints:

1. Smelting capacity:  $6(x_1-s_1) + 10(x_2-s_2) + 8x_3 \leq 8000$
2. Mechanisation capacity:  $6x_1 + 3x_2 + 8x_3 \leq 12000$
3. Assembly and packaging capacity:  $3x_1 + 2x_2 + 2x_3 \leq 10000$
4. Non-negativity constraints:  $x_1, x_2, x_3, s_1, s_2 \geq 0$
5. Subcontracting constraints:  $s_1 \leq x_1$  and  $s_2 \leq x_2$  (Subcontracted units cannot exceed the total units produced for each product)

The objective function maximizes the weekly profit by considering the revenue from sales and subtracting the production costs.

The constraints ensure that the weekly capacities for each operation are not exceeded, the decision variables are non-negative, and subcontracted units do not exceed produced units.

By solving this linear programming model, PRODA, S.A. can determine the optimal production quantities for each product (P1, P2, and P3).

## ✓ 2. Prompt 2: Write Code based on Mathematical Model

```
message2 = client.messages.create(
    model="claude-3-opus-20240229",
    system=prompt2,
    max_tokens=4096,
    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 PRODA, S.A. production scheduling problem:

```
```python
from pyomo.environ import *
```

```

# Create the model
model = ConcreteModel()

# Define the decision variables
model.x1 = Var(domain=NonNegativeReals)
model.x2 = Var(domain=NonNegativeReals)
model.x3 = Var(domain=NonNegativeReals)
model.s1 = Var(domain=NonNegativeReals)
model.s2 = Var(domain=NonNegativeReals)

# Define the objective function
model.profit = Objective(expr=1.50*model.x1 + 1.80*model.x2 + 1.97*model.x3 - (0.30*(model.x1-model.s1) + 0.20*model.x1

# Define the constraints
model.smelting_capacity = Constraint(expr=6*(model.x1-model.s1) + 10*(model.x2-model.s2) + 8*model.x3 <= 8000)
model.mechanisation_capacity = Constraint(expr=6*model.x1 + 3*model.x2 + 8*model.x3 <= 12000)
model.assembly_capacity = Constraint(expr=3*model.x1 + 2*model.x2 + 2*model.x3 <= 10000)
model.subcontracting_constraint1 = Constraint(expr=model.s1 <= model.x1)
model.subcontracting_constraint2 = Constraint(expr=model.s2 <= model.x2)

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

# Print the results
print("Optimal solution:")
print("x1 =", model.x1.value)
print("x2 =", model.x2.value)
print("x3 =", model.x3.value)
print("s1 =", model.s1.value)
print("s2 =", model.s2.value)
print("Weekly profit =", model.profit.expr())

```

In this code, we define the decision variables `x1`, `x2`, `x3`, `s1`, and `s2` using the `Var` function from Pyomo. We The objective function is defined using the `Objective` function, where we maximize the weekly profit by calculating the The constraints are defined using the `Constraint` function. We have constraints for smelting capacity, mechanisation ca To solve the model, we create an instance of the solver using `SolverFactory('glpk')`. You can replace `glpk` with the Finally, we print the optimal solution, including the values of the decision variables (`x1`, `x2`, `x3`, `s1`, `s2`) an Note that this code uses sample data directly in the objective function and constraints. In a real-world scenario, you w

## 4. Input Problem Data and Test Model Code

```

# Installing pyomo and 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(domain=NonNegativeReals)
model.x2 = Var(domain=NonNegativeReals)
model.x3 = Var(domain=NonNegativeReals)
model.s1 = Var(domain=NonNegativeReals)
model.s2 = Var(domain=NonNegativeReals)

# Define the objective function
model.profit = Objective(expr=1.50*model.x1 + 1.80*model.x2 + 1.97*model.x3 - (0.30*(model.x1-model.s1) + 0.20*model.x1 + 0.3

# Define the constraints
model.smelting_capacity = Constraint(expr=6*(model.x1-model.s1) + 10*(model.x2-model.s2) + 8*model.x3 <= 8000)
model.mechanisation_capacity = Constraint(expr=6*model.x1 + 3*model.x2 + 8*model.x3 <= 12000)
model.assembly_capacity = Constraint(expr=3*model.x1 + 2*model.x2 + 2*model.x3 <= 10000)
model.subcontracting_constraint1 = Constraint(expr=model.s1 <= model.x1)
model.subcontracting_constraint2 = Constraint(expr=model.s2 <= model.x2)

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

```

```
# Print the results
print("Optimal solution:")
print("x1 =", model.x1.value)
print("x2 =", model.x2.value)
print("x3 =", model.x3.value)
print("s1 =", model.s1.value)
print("s2 =", model.s2.value)
print("Weekly profit =", model.profit.expr())
```

```
↗ Optimal solution:
x1 = 0.0
x2 = 4000.0
x3 = 0.0
s1 = 0.0
s2 = 3200.0
Weekly profit = 3680.0
```

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