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/LP/LP2.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:
 You are in charge of the supply purchasing of a company that produces two kinds of drugs.
The drugs contain a specific active agent, which is extracted from two different kinds of raw materials that should be p

The goal is to maximize the total profit obtained from producing the drugs, which means minimizing purchasing costs for

You are given a budget for purchasing raw materials and operating the production process of the drugs which cannot be ex
Additionally, you need to keep the capacity constraints for the production of the drugs in mind as there is only a limit
Your company can also only store a limited amount of raw materials.
Most importantly, the amount of active ingredient extracted from the raw materials you purchase needs to satisfy the req
Importantly, your experience with the production of the drugs has shown the the amount of active ingredient your process
During purchasing, you need to make sure that no matter how much the amount of active ingredient varies, the required am
```

```
!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)
```

```
Here are the variables for the mathematical optimization problem:

Decision Variables:
- x1: Amount of raw material 1 purchased (in kg)
- x2: Amount of raw material 2 purchased (in kg)
- y1: Amount of drug 1 produced (in units)
- y2: Amount of drug 2 produced (in units)

Parameters:
- c1: Cost per kg of raw material 1
- c2: Cost per kg of raw material 2
- p1: Price per unit of drug 1
- p2: Price per unit of drug 2
- B: Total budget for purchasing raw materials and operating the production process
- M: Available manpower hours
- E: Available equipment hours
- S1: Storage capacity for raw material 1 (in kg)
- S2: Storage capacity for raw material 2 (in kg)
- a1: Amount of active ingredient required per unit of drug 1
- a2: Amount of active ingredient required per unit of drug 2
- r11: Amount of active ingredient extracted from 1 kg of raw material 1 (lower bound)
- r12: Amount of active ingredient extracted from 1 kg of raw material 1 (upper bound)
- r21: Amount of active ingredient extracted from 1 kg of raw material 2 (lower bound)
- r22: Amount of active ingredient extracted from 1 kg of raw material 2 (upper bound)
- m1: Manpower hours required to produce one unit of drug 1
- m2: Manpower hours required to produce one unit of drug 2
- e1: Equipment hours required to produce one unit of drug 1
- e2: Equipment hours required to produce one unit of drug 2

The objective is to maximize the total profit, which is the revenue from selling the drugs minus the costs for purchasin
```

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)
```

```
The objective function for this mathematical optimization problem can be formulated as follows:

Maximize: p1 * y1 + p2 * y2 - (c1 * x1 + c2 * x2)

Where:
- p1 * y1 represents the revenue from selling drug 1
- p2 * y2 represents the revenue from selling drug 2
- c1 * x1 represents the cost of purchasing raw material 1
- c2 * x2 represents the cost of purchasing raw material 2

The objective function aims to maximize the total profit by maximizing the revenue from selling the drugs while minimizi
```

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)
```

```
Constraints:

1. Budget constraint:
   c1 * x1 + c2 * x2 <= B

2. Manpower constraint:
   m1 * y1 + m2 * y2 <= M

3. Equipment constraint:
   e1 * y1 + e2 * y2 <= E

4. Storage capacity constraints:
   x1 <= S1
   x2 <= S2

5. Active ingredient constraints (worst-case scenarios):
   r11 * x1 + r21 * x2 >= a1 * y1 + a2 * y2
   r12 * x1 + r22 * x2 >= a1 * y1 + a2 * y2

6. Non-negativity constraints:
   x1, x2, y1, y2 >= 0

Explanation:
1. The budget constraint ensures that the total cost of purchasing raw materials does not exceed the available budget B.

2. The manpower constraint ensures that the total manpower hours required for producing the drugs do not exceed the avai

3. The equipment constraint ensures that the total equipment hours required for producing the drugs do not exceed the av

4. The storage capacity constraints ensure that the amount of each raw material purchased does not exceed the respective

5. The active ingredient constraints ensure that the amount of active ingredient extracted from the purchased raw materi

6. The non-negativity constraints ensure that the decision variables (x1, x2, y1, y2) take non-negative values.

These constraints, along with the objective function, form the complete mathematical optimization problem for maximizing
```

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)
```

```
# Define the parameters (sample data)
model.c1 = Param(initialize=10)  # Cost per kg of raw material 1
model.c2 = Param(initialize=15)  # Cost per kg of raw material 2
model.p1 = Param(initialize=50)  # Price per unit of drug 1
model.p2 = Param(initialize=60)  # Price per unit of drug 2
model.B = Param(initialize=1000)  # Total budget for purchasing raw materials and operating the production process
model.M = Param(initialize=200)  # Available manpower hours
model.E = Param(initialize=150)  # Available equipment hours
model.S1 = Param(initialize=100)  # Storage capacity for raw material 1 (in kg)
model.S2 = Param(initialize=80)  # Storage capacity for raw material 2 (in kg)
model.a1 = Param(initialize=0.5)  # Amount of active ingredient required per unit of drug 1
model.a2 = Param(initialize=0.6)  # Amount of active ingredient required per unit of drug 2
model.r11 = Param(initialize=0.8)  # Amount of active ingredient extracted from 1 kg of raw material 1 (lower bound)
model.r12 = Param(initialize=0.9)  # Amount of active ingredient extracted from 1 kg of raw material 1 (upper bound)
model.r21 = Param(initialize=0.7)  # Amount of active ingredient extracted from 1 kg of raw material 2 (lower bound)
model.r22 = Param(initialize=0.8)  # Amount of active ingredient extracted from 1 kg of raw material 2 (upper bound)
model.m1 = Param(initialize=1.5)  # Manpower hours required to produce one unit of drug 1
model.m2 = Param(initialize=2)  # Manpower hours required to produce one unit of drug 2
model.e1 = Param(initialize=1)  # Equipment hours required to produce one unit of drug 1
model.e2 = Param(initialize=1.2)  # Equipment hours required to produce one unit of drug 2

# Define the objective function
model.profit = Objective(expr=model.p1 * model.y1 + model.p2 * model.y2 - (model.c1 * model.x1 + model.c2 * model.x2),

# Define the constraints
model.budget_constraint = Constraint(expr=model.c1 * model.x1 + model.c2 * model.x2 <= model.B)
model.manpower_constraint = Constraint(expr=model.m1 * model.y1 + model.m2 * model.y2 <= model.M)
model.equipment_constraint = Constraint(expr=model.e1 * model.y1 + model.e2 * model.y2 <= model.E)
model.storage_constraint1 = Constraint(expr=model.x1 <= model.S1)
model.storage_constraint2 = Constraint(expr=model.x2 <= model.S2)
model.ingredient_constraint1 = Constraint(expr=model.r11 * model.x1 + model.r21 * model.x2 >= model.a1 * model.y1 + mo
model.ingredient_constraint2 = Constraint(expr=model.r12 * model.x1 + model.r22 * model.x2 >= model.a1 * model.y1 + mo

# Solve the optimization problem
solver = SolverFactory('glpk')  # You can use any other solver (e.g., 'cplex', 'gurobi')
results = solver.solve(model)

# Print the optimal solution
print("Optimal Solution:")
print("x1 =", model.x1.value)
print("x2 =", model.x2.value)
print("y1 =", model.y1.value)
print("y2 =", model.y2.value)
print("Total Profit =", model.profit.expr())
```

In this code, we first create a Pyomo model using `ConcreteModel()`. Then, we define the decision variables (`x1`, `x2

Next, we define the parameters using `Param(initialize=value)`. The values provided here are sample data, and you can

We define the objective function using `Objective(expr=expression, sense=maximize)`, where `expression` represents the

Then, we define the constraints using `Constraint(expr=expression)`, where `expression` represents the constraint equa

Finally, we solve the optimization problem using a solver (in this case, `'glpk'`). You can choose any other solver su

After solving the problem, we print the optimal values of the decision variables and the total profit.
```
```

4. Input Problem Data and Test Model Code

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

Show hidden output

```
from pyomo.environ import *

# Create a Pyomo model
model = ConcreteModel()

# Define the decision variables
model.x1 = Var(domain=NonNegativeReals)  # Amount of raw material 1 purchased (in kg)
model.x2 = Var(domain=NonNegativeReals)  # Amount of raw material 2 purchased (in kg)
model.y1 = Var(domain=NonNegativeReals)  # Amount of drug 1 produced (in units)
model.y2 = Var(domain=NonNegativeReals)  # Amount of drug 2 produced (in units)

# Define the parameters (sample data)
model.c1 = Param(initialize=100)  # Cost per kg of raw material 1
model.c2 = Param(initialize=199.90)  # Cost per kg of raw material 2
model.p1 = Param(initialize=6200)  # Price per unit of drug 1
model.p2 = Param(initialize=6500)  # Price per unit of drug 2
model.B = Param(initialize=100000)  # Total budget for purchasing raw materials and operating the production process
model.M = Param(initialize=2000)  # Available manpower hours
model.E = Param(initialize=800)  # Available equipment hours
model.S1 = Param(initialize=1000)  # Storage capacity for raw material 1 (in kg)
model.S2 = Param(initialize=1000)  # Storage capacity for raw material 2 (in kg)
model.a1 = Param(initialize=0.5)  # Amount of active ingredient required per unit of drug 1
model.a2 = Param(initialize=0.6)  # Amount of active ingredient required per unit of drug 2
model.r11 = Param(initialize=0.00995)  # Amount of active ingredient extracted from 1 kg of raw material 1 (lower bound)
model.r12 = Param(initialize=0.01005)  # Amount of active ingredient extracted from 1 kg of raw material 1 (upper bound)
model.r21 = Param(initialize=0.0196)  # Amount of active ingredient extracted from 1 kg of raw material 2 (lower bound)
model.r22 = Param(initialize=0.0204)  # Amount of active ingredient extracted from 1 kg of raw material 2 (upper bound)
model.m1 = Param(initialize=90)  # Manpower hours required to produce one unit of drug 1
model.m2 = Param(initialize=100)  # Manpower hours required to produce one unit of drug 2
model.e1 = Param(initialize=40)  # Equipment hours required to produce one unit of drug 1
model.e2 = Param(initialize=50)  # Equipment hours required to produce one unit of drug 2

# Define the objective function
model.profit = Objective(expr=model.p1 * model.y1 + model.p2 * model.y2 - (model.c1 * model.x1 + model.c2 * model.x2), sense=

# Define the constraints
model.budget_constraint = Constraint(expr=model.c1 * model.x1 + model.c2 * model.x2 <= model.B)
model.manpower_constraint = Constraint(expr=model.m1 * model.y1 + model.m2 * model.y2 <= model.M)
model.equipment_constraint = Constraint(expr=model.e1 * model.y1 + model.e2 * model.y2 <= model.E)
model.storage_constraint1 = Constraint(expr=model.x1 <= model.S1)
model.storage_constraint2 = Constraint(expr=model.x2 <= model.S2)
model.ingredient_constraint1 = Constraint(expr=model.r11 * model.x1 + model.r21 * model.x2 >= model.a1 * model.y1 + model.a2
model.ingredient_constraint2 = Constraint(expr=model.r12 * model.x1 + model.r22 * model.x2 >= model.a1 * model.y1 + model.a2

# Solve the optimization problem
solver = SolverFactory('glpk')  # You can use any other solver (e.g., 'cplex', 'gurobi')
results = solver.solve(model)

# Print the optimal solution
print("Optimal Solution:")
print("x1 =", model.x1.value)
print("x2 =", model.x2.value)
print("y1 =", model.y1.value)
print("y2 =", model.y2.value)
print("Total Profit =", model.profit.expr())
```

```
Optimal Solution:
x1 = 1000.0
x2 = 5.9505912441227e-14
y1 = 19.9
y2 = 0.0
Total Profit = 23379.99999999997
```

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