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)
```

```
To formulate the variables for this mathematical optimization problem, we need to consider the following:

1. Decision variables:
   - Amount of raw material 1 to purchase (in appropriate units, e.g., kg or tons)
   - Amount of raw material 2 to purchase (in appropriate units)
   - Amount of drug 1 to produce (in appropriate units, e.g., bottles or packs)
   - Amount of drug 2 to produce (in appropriate units)

2. Parameters:
   - Cost per unit of raw material 1
   - Cost per unit of raw material 2
   - Selling price per unit of drug 1
   - Selling price per unit of drug 2
   - Operational cost per unit of drug 1
   - Operational cost per unit of drug 2
   - Budget for purchasing raw materials and operating the production process
   - Maximum capacity for producing drug 1 (considering manpower and equipment hours)
   - Maximum capacity for producing drug 2 (considering manpower and equipment hours)
   - Maximum storage capacity for raw material 1
   - Maximum storage capacity for raw material 2
   - Required amount of active ingredient per unit of drug 1
   - Required amount of active ingredient per unit of drug 2
   - Minimum and maximum percentage of active ingredient extracted from raw material 1 (considering the 0.5% variation)
   - Minimum and maximum percentage of active ingredient extracted from raw material 2 (considering the 2% variation)

By defining these variables and parameters, we can formulate the objective function and constraints for the optimization
```

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:

Let's define the decision variables and parameters as follows:
```

```
Decision variables:
- x₁: Amount of raw material 1 to purchase (in appropriate units)
- x₂: Amount of raw material 2 to purchase (in appropriate units)
- y₁: Amount of drug 1 to produce (in appropriate units)
- y₂: Amount of drug 2 to produce (in appropriate units)

Parameters:
- c₁: Cost per unit of raw material 1
- c₂: Cost per unit of raw material 2
- p₁: Selling price per unit of drug 1
- p₂: Selling price per unit of drug 2
- o₁: Operational cost per unit of drug 1
- o₂: Operational cost per unit of drug 2

The objective function for maximizing the total profit can be formulated as follows:

Maximize: (p₁ × y₁ + p₂ × y₂) - (c₁ × x₁ + c₂ × x₂) - (o₁ × y₁ + o₂ × y₂)

Explanation:
- (p₁ × y₁ + p₂ × y₂) represents the total revenue generated from selling drug 1 and drug 2.
- (c₁ × x₁ + c₂ × x₂) represents the total cost of purchasing raw material 1 and raw material 2.
- (o₁ × y₁ + o₂ × y₂) represents the total operational cost for producing drug 1 and drug 2.

By subtracting the purchasing costs and operational costs from the revenue, we obtain the total profit. The objective is

Note that this is only the objective function, and the complete optimization problem will also include the necessary con
```

1. Prompt 1.3: Create Constraints for Mathematical Model

```python
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:

Let's define additional parameters:
- B: Budget for purchasing raw materials and operating the production process
- C₁: Maximum capacity for producing drug 1
- C₂: Maximum capacity for producing drug 2
- S₁: Maximum storage capacity for raw material 1
- S₂: Maximum storage capacity for raw material 2
- a₁: Required amount of active ingredient per unit of drug 1
- a₂: Required amount of active ingredient per unit of drug 2
- l₁: Minimum percentage of active ingredient extracted from raw material 1 (considering the 0.5% variation)
- u₁: Maximum percentage of active ingredient extracted from raw material 1 (considering the 0.5% variation)
- l₂: Minimum percentage of active ingredient extracted from raw material 2 (considering the 2% variation)
- u₂: Maximum percentage of active ingredient extracted from raw material 2 (considering the 2% variation)

The constraints for the optimization problem can be formulated as follows:

1. Budget constraint:
   c₁ × x₁ + c₂ × x₂ + o₁ × y₁ + o₂ × y₂ ≤ B

2. Production capacity constraints:
   y₁ ≤ C₁
   y₂ ≤ C₂

3. Storage capacity constraints:
   x₁ ≤ S₁
   x₂ ≤ S₂

4. Active ingredient requirement constraints (considering the variations):
   l₁ × x₁ + l₂ × x₂ ≥ a₁ × y₁ + a₂ × y₂
   u₁ × x₁ + u₂ × x₂ ≥ a₁ × y₁ + a₂ × y₂

5. Non-negativity constraints:
   x₁, x₂, y₁, y₂ ≥ 0

Explanation:
1. The budget constraint ensures that the total cost of purchasing raw materials and operating the production process do

2. The production capacity constraints ensure that the amount of drug 1 and drug 2 produced does not exceed their respec

3. The storage capacity constraints ensure that the amount of raw material 1 and raw material 2 purchased does not excee
```

```
4. The active ingredient requirement constraints ensure that the amount of active ingredient extracted from the purchase

5. The non-negativity constraints ensure that the decision variables (x₁, x₂, y₁, y₂) are non-negative, as negative amou

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)
```

```
    model.y2 = Var(within=NonNegativeReals)  # Amount of drug 2 to produce

    # Define the parameters
    model.c1 = Param(initialize=10)  # Cost per unit of raw material 1 (sample data)
    model.c2 = Param(initialize=15)  # Cost per unit of raw material 2 (sample data)
    model.p1 = Param(initialize=50)  # Selling price per unit of drug 1 (sample data)
    model.p2 = Param(initialize=60)  # Selling price per unit of drug 2 (sample data)
    model.o1 = Param(initialize=5)   # Operational cost per unit of drug 1 (sample data)
    model.o2 = Param(initialize=6)   # Operational cost per unit of drug 2 (sample data)
    model.B = Param(initialize=1000)  # Budget for purchasing raw materials and operating the production process (sample d
    model.C1 = Param(initialize=100)  # Maximum capacity for producing drug 1 (sample data)
    model.C2 = Param(initialize=120)  # Maximum capacity for producing drug 2 (sample data)
    model.S1 = Param(initialize=500)  # Maximum storage capacity for raw material 1 (sample data)
    model.S2 = Param(initialize=600)  # Maximum storage capacity for raw material 2 (sample data)
    model.a1 = Param(initialize=0.8)  # Required amount of active ingredient per unit of drug 1 (sample data)
    model.a2 = Param(initialize=0.9)  # Required amount of active ingredient per unit of drug 2 (sample data)
    model.l1 = Param(initialize=0.495)  # Minimum percentage of active ingredient extracted from raw material 1 (sample da
    model.u1 = Param(initialize=0.505)  # Maximum percentage of active ingredient extracted from raw material 1 (sample da
    model.l2 = Param(initialize=0.98)   # Minimum percentage of active ingredient extracted from raw material 2 (sample da
    model.u2 = Param(initialize=1.02)   # Maximum percentage of active ingredient extracted from raw material 2 (sample da

    # 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.o1 * model.y1 + model.o2 *
    model.capacity_constraint1 = Constraint(expr=model.y1 <= model.C1)
    model.capacity_constraint2 = Constraint(expr=model.y2 <= model.C2)
    model.storage_constraint1 = Constraint(expr=model.x1 <= model.S1)
    model.storage_constraint2 = Constraint(expr=model.x2 <= model.S2)
    model.ingredient_constraint1 = Constraint(expr=model.l1 * model.x1 + model.l2 * model.x2 >= model.a1 * model.y1 + mode
    model.ingredient_constraint2 = Constraint(expr=model.u1 * model.x1 + model.u2 * model.x2 >= model.a1 * model.y1 + mode

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

The objective function `profit` is defined to maximize the total profit, and the constraints are defined according to

The optimization problem is solved using the `glpk` solver, and the results are printed, including the optimal values

Please note that you need to have Pyomo and the GLPK solver installed to run this code. You can install them using the

```
pip install pyomo
```

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 the model
model = ConcreteModel()

# Define the decision variables
model.x1 = Var(within=NonNegativeReals)  # Amount of raw material 1 to purchase
model.x2 = Var(within=NonNegativeReals)  # Amount of raw material 2 to purchase
model.y1 = Var(within=NonNegativeReals)  # Amount of drug 1 to produce
model.y2 = Var(within=NonNegativeReals)  # Amount of drug 2 to produce

# Define the parameters
model.c1 = Param(initialize=100)  # Cost per unit of raw material 1 (sample data)
model.c2 = Param(initialize=199.90)  # Cost per unit of raw material 2 (sample data)
model.p1 = Param(initialize=6200)  # Selling price per unit of drug 1 (sample data)
model.p2 = Param(initialize=6900)  # Selling price per unit of drug 2 (sample data)
model.o1 = Param(initialize=700)   # Operational cost per unit of drug 1 (sample data)
model.o2 = Param(initialize=800)   # Operational cost per unit of drug 2 (sample data)
model.B = Param(initialize=100000)  # Budget for purchasing raw materials and operating the production process (sample data)
model.C1 = Param(initialize=2800)  # Maximum capacity for producing drug 1 (sample data) USED COMBINATION OF MPH AND EH
model.C2 = Param(initialize=2800)  # Maximum capacity for producing drug 2 (sample data) USED COMBINATION OF MPH AND EH
model.S1 = Param(initialize=1000)  # Maximum storage capacity for raw material 1 (sample data)
model.S2 = Param(initialize=1000)  # Maximum storage capacity for raw material 2 (sample data)
model.a1 = Param(initialize=0.5)  # Required amount of active ingredient per unit of drug 1 (sample data)
model.a2 = Param(initialize=0.6)  # Required amount of active ingredient per unit of drug 2 (sample data)
model.l1 = Param(initialize=0.00995)  # Minimum percentage of active ingredient extracted from raw material 1 (sample data)
model.u1 = Param(initialize=0.01005)  # Maximum percentage of active ingredient extracted from raw material 1 (sample data)
model.l2 = Param(initialize=0.0196)   # Minimum percentage of active ingredient extracted from raw material 2 (sample data)
model.u2 = Param(initialize=0.0204)   # Maximum percentage of active ingredient extracted from raw material 2 (sample data)

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

# Define the constraints
model.budget_constraint = Constraint(expr=model.c1 * model.x1 + model.c2 * model.x2 + model.o1 * model.y1 + model.o2 * model.
model.capacity_constraint1 = Constraint(expr=model.y1 <= model.C1)
model.capacity_constraint2 = Constraint(expr=model.y2 <= model.C2)
model.storage_constraint1 = Constraint(expr=model.x1 <= model.S1)
model.storage_constraint2 = Constraint(expr=model.x2 <= model.S2)
model.ingredient_constraint1 = Constraint(expr=model.l1 * model.x1 + model.l2 * model.x2 >= model.a1 * model.y1 + model.a2 *
model.ingredient_constraint2 = Constraint(expr=model.u1 * model.x1 + model.u2 * model.x2 >= model.a1 * model.y1 + model.a2 *

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

# Print the results
print("Optimization Results:")
print("Amount of raw material 1 to purchase (x1):", model.x1.value)
print("Amount of raw material 2 to purchase (x2):", model.x2.value)
print("Amount of drug 1 to produce (y1):", model.y1.value)
print("Amount of drug 2 to produce (y2):", model.y2.value)
print("Total profit:", model.profit.expr())
```

```
Optimization Results:
Amount of raw material 1 to purchase (x1): 877.731940665321
Amount of raw material 2 to purchase (x2): 0.0
Amount of drug 1 to produce (y1): 17.4668656192399
Amount of drug 2 to produce (y2): 0.0
```

```
Total profit: 8294.566839287345
```

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