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)
```

```
Variables:

1. x₁: Amount of raw material 1 purchased (in kg)
2. x₂: Amount of raw material 2 purchased (in kg)
3. y₁: Amount of drug 1 produced (in units)
4. y₂: Amount of drug 2 produced (in units)
5. p₁: Profit per unit of drug 1 (in $)
6. p₂: Profit per unit of drug 2 (in $)
7. c₁: Cost per kg of raw material 1 (in $)
8. c₂: Cost per kg of raw material 2 (in $)
9. o₁: Operational cost per unit of drug 1 (in $)
10. o₂: Operational cost per unit of drug 2 (in $)
11. B: Total budget for purchasing raw materials and operating the production process (in $)
12. M: Available manpower (in hours)
13. E: Available equipment hours
14. S₁: Storage capacity for raw material 1 (in kg)
15. S₂: Storage capacity for raw material 2 (in kg)
16. a₁₁: Amount of active ingredient per kg of raw material 1 (in mg)
17. a₁₂: Amount of active ingredient per kg of raw material 2 (in mg)
18. A₁: Required amount of active ingredient per unit of drug 1 (in mg)
19. A₂: Required amount of active ingredient per unit of drug 2 (in mg)
20. v₁: Maximum variation in the amount of active ingredient extracted from raw material 1 (in %)
21. v₂: Maximum variation in the amount of active ingredient extracted from raw material 2 (in %)
```

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 optimization problem is to maximize the total profit obtained from producing the drugs.

maximize Z = p₁ * y₁ + p₂ * y₂ - (c₁ * x₁ + c₂ * x₂ + o₁ * y₁ + o₂ * y₂)

where:
- p₁ * y₁ + p₂ * y₂ represents the total revenue from selling the produced drugs
- c₁ * x₁ + c₂ * x₂ represents the total cost of purchasing the raw materials
- o₁ * y₁ + o₂ * y₂ represents the total operational cost of producing the drugs
```

```
The objective is to maximize Z, which is the difference between the total revenue and the total costs (purchasing and op
```

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)
```

```
Here are the constraints for the optimization problem:

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

2. Manpower constraint:
   m₁ * y₁ + m₂ * y₂ ≤ M
   (where m₁ and m₂ are the required manpower hours per unit of drug 1 and drug 2, respectively)

3. Equipment constraint:
   e₁ * y₁ + e₂ * y₂ ≤ E
   (where e₁ and e₂ are the required equipment hours per unit of drug 1 and drug 2, respectively)

4. Storage constraints:
   x₁ ≤ S₁
   x₂ ≤ S₂

5. Active ingredient constraints (considering the maximum variation):
   (1 - v₁) * a₁₁ * x₁ + (1 - v₂) * a₁₂ * x₂ ≥ A₁ * y₁ + A₂ * y₂

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

Here's a brief explanation of each constraint:

1. The total cost of purchasing raw materials and operating the production process should not exceed the given budget B.
2. The total manpower hours required for producing the drugs should not exceed the available manpower M.
3. The total equipment hours required for producing the drugs should not exceed the available equipment hours E.
4. The amount of raw materials purchased should not exceed their respective storage capacities S₁ and S₂.
5. The amount of active ingredient extracted from the purchased raw materials, considering the maximum variation, should
6. The decision variables (x₁, x₂, y₁, y₂) should be non-negative.

These constraints ensure that the company operates within its budget, capacity, and storage limitations while meeting th
```

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)
```

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

# Create a Pyomo model
model = ConcreteModel()

# Define decision variables
model.x1 = Var(domain=NonNegativeReals)
model.x2 = Var(domain=NonNegativeReals)
model.y1 = Var(domain=NonNegativeReals)
model.y2 = Var(domain=NonNegativeReals)

# Define objective function
model.profit = Objective(expr=p1 * model.y1 + p2 * model.y2 - (c1 * model.x1 + c2 * model.x2 + o1 * model.y1 + o2 * mo

# Define constraints
model.budget_constraint = Constraint(expr=c1 * model.x1 + c2 * model.x2 + o1 * model.y1 + o2 * model.y2 <= B)
model.manpower_constraint = Constraint(expr=m1 * model.y1 + m2 * model.y2 <= M)
model.equipment_constraint = Constraint(expr=e1 * model.y1 + e2 * model.y2 <= E)
model.storage_constraint1 = Constraint(expr=model.x1 <= S1)
model.storage_constraint2 = Constraint(expr=model.x2 <= S2)
model.active_ingredient_constraint = Constraint(expr=(1 - v1) * a11 * model.x1 + (1 - v2) * a12 * model.x2 >= A1 * mod

# Assign sample data (replace with actual data)
p1, p2 = 10, 12  # Profit per unit of drug 1 and drug 2
c1, c2 = 5, 6  # Cost per kg of raw material 1 and raw material 2
o1, o2 = 2, 3  # Operational cost per unit of drug 1 and drug 2
B = 1000  # Total budget
M = 200  # Available manpower (in hours)
E = 150  # Available equipment hours
S1, S2 = 100, 120  # Storage capacity for raw material 1 and raw material 2
a11, a12 = 0.8, 0.9  # Amount of active ingredient per kg of raw material 1 and raw material 2
A1, A2 = 5, 6  # Required amount of active ingredient per unit of drug 1 and drug 2
v1, v2 = 0.05, 0.1  # Maximum variation in the amount of active ingredient extracted from raw materials
m1, m2 = 1, 1.5  # Required manpower hours per unit of drug 1 and drug 2
e1, e2 = 0.8, 1.2  # Required equipment hours per unit of drug 1 and drug 2

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

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

In this code, we define the decision variables (`x1`, `x2`, `y1`, `y2`) and the objective function (`profit`) using Py

Sample data is assigned to the parameters (`p1`, `p2`, `c1`, `c2`, etc.) for demonstration purposes. Replace these val

Finally, we solve the optimization problem using the `glpk` solver (you may need to install it separately) and print t

Note: Make sure you have Pyomo and the `glpk` solver installed before running this code.
~~~

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 *

# Assign sample data (replace with actual data)
p1, p2 = 6200, 6900   # Profit per unit of drug 1 and drug 2
c1, c2 = 100, 199.90  # Cost per kg of raw material 1 and raw material 2
o1, o2 = 700, 800  # Operational cost per unit of drug 1 and drug 2
B = 100000  # Total budget
M = 2000  # Available manpower (in hours)
E = 800  # Available equipment hours
S1, S2 = 1000, 1000  # Storage capacity for raw material 1 and raw material 2
a11, a12 = 0.01, 0.02  # Amount of active ingredient per kg of raw material 1 and raw material 2
```

```
A1, A2 = 0.5, 0.6  # Required amount of active ingredient per unit of drug 1 and drug 2
v1, v2 = 0.005, 0.02  # Maximum variation in the amount of active ingredient extracted from raw materials
m1, m2 = 90, 1000  # Required manpower hours per unit of drug 1 and drug 2
e1, e2 = 40, 50  # Required equipment hours per unit of drug 1 and drug 2

# Create a Pyomo model
model = ConcreteModel()

# Define decision variables
model.x1 = Var(domain=NonNegativeReals)
model.x2 = Var(domain=NonNegativeReals)
model.y1 = Var(domain=NonNegativeReals)
model.y2 = Var(domain=NonNegativeReals)

# Define objective function
model.profit = Objective(expr=p1 * model.y1 + p2 * model.y2 - (c1 * model.x1 + c2 * model.x2 + o1 * model.y1 + o2 * model.y2)

# Define constraints
model.budget_constraint = Constraint(expr=c1 * model.x1 + c2 * model.x2 + o1 * model.y1 + o2 * model.y2 <= B)
model.manpower_constraint = Constraint(expr=m1 * model.y1 + m2 * model.y2 <= M)
model.equipment_constraint = Constraint(expr=e1 * model.y1 + e2 * model.y2 <= E)
model.storage_constraint1 = Constraint(expr=model.x1 <= S1)
model.storage_constraint2 = Constraint(expr=model.x2 <= S2)
model.active_ingredient_constraint = Constraint(expr=(1 - v1) * a11 * model.x1 + (1 - v2) * a12 * model.x2 >= A1 * model.y1 +

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

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

```
Optimal solution:
x1 = 877.731940665321
x2 = 0.0
y1 = 17.4668656192399
y2 = 0.0
Maximum profit = 8294.566839287349
```

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