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/IP/IP4.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:
 Imagine you own a small shop, and you're trying to manage your employees' working hours and sales targets efficiently.
You have three kinds of goals overall: sales goals, employee hours goals and overtime goals.

Sales Goals:
You have a target of selling 5500 units.
But sometimes, you might sell fewer (negative deviation) or more (positive deviation) than this target.
On average, your full-time employees will sell 5 records per hour and your part-time employees will sell 2 records per h

Employee Hours:
You have both full-time and part-time employees.
Your full-time employees are expected to work a total of 800 hours, while part-time employees are expected to work 320 h
Similarly, like sales, there could be deviations in these working hours.
```

```
Overtime:
Additionally, your employees might end up working overtime.
You have a target of 100 hours of overtime for full-time employees.
You are not concerned about limiting the overtime of part-time employees.
Again, you might deviate from this goal.

Each of these deviations has a different priority for you.
P1 to P4 are priority factors for each deviation.
Your most important goal is to achieve your sales goal.
Second to that is the goal to not exceed 100 overtime hours for full-time employees.
Then, you next priority is to fully utilize your full-time and part time employees.
Your last goal with the lowest priority, you want to keep general overtime hours for both kinds of employees as low as p
In addition to the general priorities of you goals, you are twice as concerned about fully utilizing the full-time emplo
Finally, overtime hours of part-time employees have a relative cost of three times that of full-time employees.

Now, the goal here is to minimize the overall impact of these deviations, considering their priorities.
This means finding the best balance between sales targets, employee hours, and overtime, while keeping in mind which dev
```

```
!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)
```

```
Great! Let's formulate the variables for this goal programming problem. We'll define variables for the decision variable

Decision Variables:
- Let "F" be the number of hours worked by full-time employees
- Let "P" be the number of hours worked by part-time employees

Deviation Variables:
- Let "d1+" be the positive deviation from the sales goal (selling more than 5500 units)
- Let "d1-" be the negative deviation from the sales goal (selling less than 5500 units)
- Let "d2+" be the positive deviation from the full-time employee hours goal (working more than 800 hours)
- Let "d2-" be the negative deviation from the full-time employee hours goal (working less than 800 hours)
- Let "d3+" be the positive deviation from the part-time employee hours goal (working more than 320 hours)
- Let "d3-" be the negative deviation from the part-time employee hours goal (working less than 320 hours)
- Let "d4+" be the positive deviation from the full-time employee overtime goal (exceeding 100 hours of overtime)
- Let "d4-" be the negative deviation from the full-time employee overtime goal (having less than 100 hours of overtime)
- Let "d5+" be the positive deviation from the general overtime hours goal (having more overtime hours than desired)

Priority Factors:
- Let "P1" be the priority factor for deviations from the sales goal
- Let "P2" be the priority factor for deviations from the full-time employee overtime goal
- Let "P3" be the priority factor for deviations from the full-time employee hours goal
- Let "P4" be the priority factor for deviations from the part-time employee hours goal and the general overtime hours g

These variables will be used in the objective function and constraints of the goal programming model to minimize the ove
```

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)
```

```
Here's the objective function for the goal programming problem:

    Minimize:
    P1 * (d1+ + d1-) + P2 * d4+ + P3 * 2 * (d2+ + d2-) + P4 * (d3+ + d3-) + P4 * (d5+ + 3*d5-)

    Explanation:
    - The objective function aims to minimize the weighted sum of deviations from the goals.
    - P1, P2, P3, and P4 are the priority factors for each goal, representing their relative importance.
    - (d1+ + d1-) represents the absolute deviation from the sales goal.
    - d4+ represents the positive deviation from the full-time employee overtime goal. We only include the positive deviatio
    - (d2+ + d2-) represents the absolute deviation from the full-time employee hours goal. This term is multiplied by 2 to
    - (d3+ + d3-) represents the absolute deviation from the part-time employee hours goal.
    - (d5+ + 3*d5-) represents the deviation from the general overtime hours goal. The negative deviation (d5-) is multiplie

    The objective function combines these deviations, weighted by their respective priority factors, to minimize the overall
```

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 goal programming problem:

    1. Sales Goal Constraint:
       5F + 2P + d1- - d1+ = 5500

       Explanation: This constraint ensures that the total sales (5 units per hour for full-time employees and 2 units per h

    2. Full-time Employee Hours Constraint:
       F + d2- - d2+ = 800

       Explanation: This constraint ensures that the total hours worked by full-time employees plus the deviation variables

    3. Part-time Employee Hours Constraint:
       P + d3- - d3+ = 320

       Explanation: This constraint ensures that the total hours worked by part-time employees plus the deviation variables

    4. Full-time Employee Overtime Constraint:
       F - 800 + d4- - d4+ = 100

       Explanation: This constraint ensures that the overtime hours for full-time employees (hours worked minus the expected

    5. General Overtime Hours Constraint:
       F + P - 1120 - d5+ = 0

       Explanation: This constraint calculates the total overtime hours for both full-time and part-time employees (total ho

    6. Non-negativity Constraints:
       F, P, d1+, d1-, d2+, d2-, d3+, d3-, d4+, d4-, d5+ ≥ 0

       Explanation: These constraints ensure that all decision variables and deviation variables are non-negative.

    These constraints, along with the objective function, form the complete goal programming model for the employee scheduli
```

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)
```

```
Here's the Python code using Pyomo to solve the goal programming problem:

    ```python
    from pyomo.environ import *

    # Create a Pyomo model
    model = ConcreteModel()

    # Define decision variables
    model.F = Var(within=NonNegativeReals)  # Hours worked by full-time employees
    model.P = Var(within=NonNegativeReals)  # Hours worked by part-time employees

    # Define deviation variables
    model.d1_plus = Var(within=NonNegativeReals)
    model.d1_minus = Var(within=NonNegativeReals)
    model.d2_plus = Var(within=NonNegativeReals)
    model.d2_minus = Var(within=NonNegativeReals)
    model.d3_plus = Var(within=NonNegativeReals)
    model.d3_minus = Var(within=NonNegativeReals)
    model.d4_plus = Var(within=NonNegativeReals)
    model.d4_minus = Var(within=NonNegativeReals)
    model.d5_plus = Var(within=NonNegativeReals)

    # Define priority factors (sample data)
    P1 = 1
    P2 = 2
    P3 = 3
    P4 = 4

    # Define the objective function
    model.objective = Objective(expr=P1 * (model.d1_plus + model.d1_minus) +
                                     P2 * model.d4_plus +
                                     P3 * 2 * (model.d2_plus + model.d2_minus) +
                                     P4 * (model.d3_plus + model.d3_minus) +
                                     P4 * (model.d5_plus + 3 * model.d5_plus),
                                sense=minimize)

    # Define constraints
    model.sales_constraint = Constraint(expr=5 * model.F + 2 * model.P + model.d1_minus - model.d1_plus == 5500)
    model.fulltime_hours_constraint = Constraint(expr=model.F + model.d2_minus - model.d2_plus == 800)
    model.parttime_hours_constraint = Constraint(expr=model.P + model.d3_minus - model.d3_plus == 320)
    model.fulltime_overtime_constraint = Constraint(expr=model.F - 800 + model.d4_minus - model.d4_plus == 100)
    model.general_overtime_constraint = Constraint(expr=model.F + model.P - 1120 - model.d5_plus == 0)

    # Solve the model
    solver = SolverFactory('glpk')  # Specify the solver (e.g., 'glpk', 'gurobi', 'cplex')
    results = solver.solve(model)

    # Print the results
    print("Objective value:", value(model.objective))
    print("Full-time employee hours (F):", value(model.F))
    print("Part-time employee hours (P):", value(model.P))
    ```

    In this code:
    - We create a Pyomo model using `ConcreteModel()`.
    - We define the decision variables `F` and `P` for the hours worked by full-time and part-time employees, respectively
```

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 decision variables
model.F = Var(within=NonNegativeReals)  # Hours worked by full-time employees
model.P = Var(within=NonNegativeReals)  # Hours worked by part-time employees

# Define deviation variables
model.d1_plus = Var(within=NonNegativeReals)
model.d1_minus = Var(within=NonNegativeReals)
model.d2_plus = Var(within=NonNegativeReals)
model.d2_minus = Var(within=NonNegativeReals)
model.d3_plus = Var(within=NonNegativeReals)
model.d3_minus = Var(within=NonNegativeReals)
model.d4_plus = Var(within=NonNegativeReals)
model.d4_minus = Var(within=NonNegativeReals)
model.d5_plus = Var(within=NonNegativeReals)

# Define priority factors (sample data)
P1 = 4
P2 = 3
P3 = 2
P4 = 1

# Define the objective function
model.objective = Objective(expr=P1 * (model.d1_plus + model.d1_minus) +
                                 P2 * model.d4_plus +
                                 P3 * 2 * (model.d2_plus + model.d2_minus) +
                                 P4 * (model.d3_plus + model.d3_minus) +
                                 P4 * (model.d5_plus + 3 * model.d5_plus),
                            sense=minimize)

# Define constraints
model.sales_constraint = Constraint(expr=5 * model.F + 2 * model.P + model.d1_minus - model.d1_plus == 5500)
model.fulltime_hours_constraint = Constraint(expr=model.F + model.d2_minus - model.d2_plus == 800)
model.parttime_hours_constraint = Constraint(expr=model.P + model.d3_minus - model.d3_plus == 320)
model.fulltime_overtime_constraint = Constraint(expr=model.F - 800 + model.d4_minus - model.d4_plus == 100)
model.general_overtime_constraint = Constraint(expr=model.F + model.P - 1120 - model.d5_plus == 0)

# Solve the model
solver = SolverFactory('glpk')  # Specify the solver (e.g., 'glpk', 'gurobi', 'cplex')
results = solver.solve(model)

# Print the results
print("Objective value:", value(model.objective))
print("Full-time employee hours (F):", value(model.F))
print("Part-time employee hours (P):", value(model.P))
```

```
Objective value: 1592.0
Full-time employee hours (F): 972.0
Part-time employee hours (P): 320.0
```

```
print("d1^-:", model.d1_minus())
print("d1^+:", model.d1_plus())
print("d2^-:", model.d2_minus())
print("d2^+:", model.d2_plus())
print("d3^-:", model.d3_minus())
print("d3^+:", model.d3_plus())
print("d4^-:", model.d4_minus())
print("d4^+:", model.d4_plus())
print("d5^+:", model.d5_plus())
```

```
d1^-: 0.0
d1^+: 0.0
d2^-: 0.0
```

```
d2^+: 172.0
d3^-: 0.0
d3^+: 0.0
d4^-: 0.0
d4^+: 72.0
d5^+: 172.0
```

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

```
from pyomo.environ import *

# Create a Pyomo model
model = ConcreteModel()

# Define decision variables
model.F = Var(within=NonNegativeReals)  # Hours worked by full-time employees
model.P = Var(within=NonNegativeReals)  # Hours worked by part-time employees

# Define deviation variables
model.d1_plus = Var(within=NonNegativeReals)
model.d1_minus = Var(within=NonNegativeReals)
model.d2_plus = Var(within=NonNegativeReals)
model.d2_minus = Var(within=NonNegativeReals)
model.d3_plus = Var(within=NonNegativeReals)
model.d3_minus = Var(within=NonNegativeReals)
model.d4_plus = Var(within=NonNegativeReals)
model.d4_minus = Var(within=NonNegativeReals)
model.d5_plus = Var(within=NonNegativeReals)
model.d5_minus = Var(within=NonNegativeReals)

# Define priority factors (sample data)
P1 = 4
P2 = 3
P3 = 2
P4 = 1

# Define the objective function
model.objective = Objective(expr=P1 * (model.d1_plus + model.d1_minus) +
                                 P2 * model.d4_plus +
                                 P3 * 2 * (model.d2_plus + model.d2_minus) +
                                 P4 * (model.d3_plus + model.d3_minus) +
                                 P4 * (model.d5_plus + 3 * model.d5_minus),
                            sense=minimize)

# Define constraints
model.sales_constraint = Constraint(expr=5 * model.F + 2 * model.P + model.d1_minus - model.d1_plus == 5500)
model.fulltime_hours_constraint = Constraint(expr=model.F + model.d2_minus - model.d2_plus == 800)
model.parttime_hours_constraint = Constraint(expr=model.P + model.d3_minus - model.d3_plus == 320)
model.fulltime_overtime_constraint = Constraint(expr=model.F - 800 + model.d4_minus - model.d4_plus == 100)
model.general_overtime_constraint = Constraint(expr=model.F + model.P - 1120 - model.d5_plus == 0)

# Solve the model
solver = SolverFactory('glpk')  # Specify the solver (e.g., 'glpk', 'gurobi', 'cplex')
results = solver.solve(model)

# Print the results
print("Objective value:", value(model.objective))
print("Full-time employee hours (F):", value(model.F))
print("Part-time employee hours (P):", value(model.P))
print("d1^-:", model.d1_minus())
print("d1^+:", model.d1_plus())
print("d2^-:", model.d2_minus())
print("d2^+:", model.d2_plus())
print("d3^-:", model.d3_minus())
print("d3^+:", model.d3_plus())
print("d4^-:", model.d4_minus())
print("d4^+:", model.d4_plus())
print("d5^+:", model.d5_plus())
```

```
Objective value: 860.0
Full-time employee hours (F): 900.0
Part-time employee hours (P): 500.0
d1^-: 0.0
d1^+: 0.0
d2^-: 0.0
d2^+: 100.0
d3^-: 0.0
d3^+: 180.0
d4^-: 0.0
d4^+: 0.0
d5^+: 280.0
```