⌄ 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/IP/IP4.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:
     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: 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)
```

⇄ Let's formulate this problem as a goal programming model. We'll define the parameters, decision variables, objective fun

```
Parameters:
- ST: Sales Target (5500 units)
- FT: Full-time employee hours target (800 hours)
- PT: Part-time employee hours target (320 hours)
- OT: Overtime target for full-time employees (100 hours)
- FS: Full-time employee sales rate (5 units per hour)
- PS: Part-time employee sales rate (2 units per hour)
- P1 to P4: Priority factors for each deviation

Decision Variables:
- F: Full-time employee hours
- P: Part-time employee hours
- OF: Overtime hours for full-time employees
- OP: Overtime hours for part-time employees
- S+: Positive deviation from sales target
- S-: Negative deviation from sales target
- F+: Positive deviation from full-time employee hours target
- F-: Negative deviation from full-time employee hours target
- P+: Positive deviation from part-time employee hours target
- P-: Negative deviation from part-time employee hours target
- O+: Positive deviation from overtime target for full-time employees
- O-: Negative deviation from overtime target for full-time employees

Objective Function:
Minimize Z = P1 * (S+ + S-) + P2 * O+ + P3 * (2F+ + 2F- + P+ + P-) + P4 * (OF + 3OP)

Constraints:
1. Sales Constraint: FS * F + PS * P + S- - S+ = ST
2. Full-time Employee Hours Constraint: F + F- - F+ = FT
3. Part-time Employee Hours Constraint: P + P- - P+ = PT
4. Overtime Constraint for Full-time Employees: OF + O- - O+ = OT
5. Non-negativity Constraints: All decision variables ≥ 0

The objective function minimizes the weighted sum of deviations based on their priorities. The sales constraint ensures

By solving this goal programming model, you can find the optimal balance between sales targets, employee hours, and over
```

⌄ 2. Prompt 2: Write Code based on Mathematical Model

```python
message2 = client.messages.create(
    model="claude-3-opus-20240229",
    system=prompt2,
    max_tokens=4069,
    messages=[
        {"role": "user", "content": response_p1}
    ]
)

response_p2 = message2.content[0].text


# Print response
print(response_p2)
```

⮞ Here's the Python code using Pyomo for the optimization problem:

```python
import pyomo.environ as pyo

# Create a Pyomo model
model = pyo.ConcreteModel()

# Define parameters
model.ST = pyo.Param(initialize=5500)  # Sales Target
model.FT = pyo.Param(initialize=800)   # Full-time employee hours target
model.PT = pyo.Param(initialize=320)   # Part-time employee hours target
model.OT = pyo.Param(initialize=100)   # Overtime target for full-time employees
model.FS = pyo.Param(initialize=5)     # Full-time employee sales rate
model.PS = pyo.Param(initialize=2)     # Part-time employee sales rate
model.P1 = pyo.Param(initialize=1)     # Priority factor for sales deviation
model.P2 = pyo.Param(initialize=2)     # Priority factor for overtime deviation
model.P3 = pyo.Param(initialize=3)     # Priority factor for employee hours deviation
model.P4 = pyo.Param(initialize=4)     # Priority factor for overtime cost

# Define decision variables
model.F = pyo.Var(domain=pyo.NonNegativeReals)   # Full-time employee hours
model.P = pyo.Var(domain=pyo.NonNegativeReals)   # Part-time employee hours
model.OF = pyo.Var(domain=pyo.NonNegativeReals)  # Overtime hours for full-time employees
model.OP = pyo.Var(domain=pyo.NonNegativeReals)  # Overtime hours for part-time employees
model.S_plus = pyo.Var(domain=pyo.NonNegativeReals)   # Positive deviation from sales target
model.S_minus = pyo.Var(domain=pyo.NonNegativeReals)  # Negative deviation from sales target
model.F_plus = pyo.Var(domain=pyo.NonNegativeReals)   # Positive deviation from full-time employee hours target
model.F_minus = pyo.Var(domain=pyo.NonNegativeReals)  # Negative deviation from full-time employee hours target
model.P_plus = pyo.Var(domain=pyo.NonNegativeReals)   # Positive deviation from part-time employee hours target
model.P_minus = pyo.Var(domain=pyo.NonNegativeReals)  # Negative deviation from part-time employee hours target
model.O_plus = pyo.Var(domain=pyo.NonNegativeReals)   # Positive deviation from overtime target for full-time employee
model.O_minus = pyo.Var(domain=pyo.NonNegativeReals)  # Negative deviation from overtime target for full-time employee

# Define objective function
model.objective = pyo.Objective(expr=model.P1 * (model.S_plus + model.S_minus) +
                                model.P2 * model.O_plus +
                                model.P3 * (2 * model.F_plus + 2 * model.F_minus + model.P_plus + model.P_minus)
                                model.P4 * (model.OF + 3 * model.OP),
                        sense=pyo.minimize)

# Define constraints
model.sales_constraint = pyo.Constraint(expr=model.FS * model.F + model.PS * model.P + model.S_minus - model.S_plus ==
model.fulltime_constraint = pyo.Constraint(expr=model.F + model.F_minus - model.F_plus == model.FT)
model.parttime_constraint = pyo.Constraint(expr=model.P + model.P_minus - model.P_plus == model.PT)
model.overtime_constraint = pyo.Constraint(expr=model.OF + model.O_minus - model.O_plus == model.OT)

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

# Print the results
print("Optimal Solution:")
print("Full-time employee hours:", pyo.value(model.F))
print("Part-time employee hours:", pyo.value(model.P))
print("Overtime hours for full-time employees:", pyo.value(model.OF))
print("Overtime hours for part-time employees:", pyo.value(model.OP))
print("Positive deviation from sales target:", pyo.value(model.S_plus))
```

˅ 4. Input Problem Data and Test Model Code

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

⮞ Show hidden output

```python
import pyomo.environ as pyo
```

```python
# Create a Pyomo model
model = pyo.ConcreteModel()

# Define parameters
model.ST = pyo.Param(initialize=5500)  # Sales Target
model.FT = pyo.Param(initialize=800)   # Full-time employee hours target
model.PT = pyo.Param(initialize=320)   # Part-time employee hours target
model.OT = pyo.Param(initialize=100)   # Overtime target for full-time employees
model.FS = pyo.Param(initialize=5)     # Full-time employee sales rate
model.PS = pyo.Param(initialize=2)     # Part-time employee sales rate
model.P1 = pyo.Param(initialize=4)     # Priority factor for sales deviation
model.P2 = pyo.Param(initialize=3)     # Priority factor for overtime deviation
model.P3 = pyo.Param(initialize=2)     # Priority factor for employee hours deviation
model.P4 = pyo.Param(initialize=1)     # Priority factor for overtime cost

# Define decision variables
model.F = pyo.Var(domain=pyo.NonNegativeReals)   # Full-time employee hours
model.P = pyo.Var(domain=pyo.NonNegativeReals)   # Part-time employee hours
model.OF = pyo.Var(domain=pyo.NonNegativeReals)  # Overtime hours for full-time employees
model.OP = pyo.Var(domain=pyo.NonNegativeReals)  # Overtime hours for part-time employees
model.S_plus = pyo.Var(domain=pyo.NonNegativeReals)   # Positive deviation from sales target
model.S_minus = pyo.Var(domain=pyo.NonNegativeReals)  # Negative deviation from sales target
model.F_plus = pyo.Var(domain=pyo.NonNegativeReals)   # Positive deviation from full-time employee hours target
model.F_minus = pyo.Var(domain=pyo.NonNegativeReals)  # Negative deviation from full-time employee hours target
model.P_plus = pyo.Var(domain=pyo.NonNegativeReals)   # Positive deviation from part-time employee hours target
model.P_minus = pyo.Var(domain=pyo.NonNegativeReals)  # Negative deviation from part-time employee hours target
model.O_plus = pyo.Var(domain=pyo.NonNegativeReals)   # Positive deviation from overtime target for full-time employees
model.O_minus = pyo.Var(domain=pyo.NonNegativeReals)  # Negative deviation from overtime target for full-time employees

# Define objective function
model.objective = pyo.Objective(expr=model.P1 * (model.S_plus + model.S_minus) +
                                     model.P2 * model.O_plus +
                                     model.P3 * (2 * model.F_plus + 2 * model.F_minus + model.P_plus + model.P_minus) +
                                     model.P4 * (model.OF + 3 * model.OP),
                                sense=pyo.minimize)

# Define constraints
model.sales_constraint = pyo.Constraint(expr=model.FS * model.F + model.PS * model.P + model.S_minus - model.S_plus == model.
model.fulltime_constraint = pyo.Constraint(expr=model.F + model.F_minus - model.F_plus == model.FT)
model.parttime_constraint = pyo.Constraint(expr=model.P + model.P_minus - model.P_plus == model.PT)
model.overtime_constraint = pyo.Constraint(expr=model.OF + model.O_minus - model.O_plus == model.OT)

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

# Print the results
print("Optimal Solution:")
print("Full-time employee hours:", pyo.value(model.F))
print("Part-time employee hours:", pyo.value(model.P))
print("Overtime hours for full-time employees:", pyo.value(model.OF))
print("Overtime hours for part-time employees:", pyo.value(model.OP))
print("Positive deviation from sales target:", pyo.value(model.S_plus))
print("Negative deviation from sales target:", pyo.value(model.S_minus))
print("Objective function value:", pyo.value(model.objective))
```

```
Optimal Solution:
Full-time employee hours: 972.0
Part-time employee hours: 320.0
Overtime hours for full-time employees: 0.0
Overtime hours for part-time employees: 0.0
Positive deviation from sales target: 0.0
Negative deviation from sales target: 0.0
Objective function value: 688.0
```

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