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/IP1.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:
 We are delighted to welcome you, our newest intern on the Analytics team of Massachusetts General Hospital! You have be

You are responsible for determining the best treatment plan for 17 patients who require radiotherapy. Your goal is to op

To determine the best course of action for each patient, you will use a scoring system called the Biological Equivalent

We have n=17 patients who need radiotherapy. Each patient i needs 15 fractions, which can be photon fractions, proton fr

The data file "ProblemData.csv" contains a 2D matrix of BED scores. It does not have an index. It was made in Excel and

Suppose that the total maximal capacity C is 100 proton fractions. To maximize the total BED scores for all the patients
```

```
!pip install anthropic
```

```
Collecting anthropic
  Downloading anthropic-0.26.0-py3-none-any.whl (877 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 877.7/877.7 kB 14.6 MB/s eta 0:00:00
Requirement already satisfied: anyio<5,>=3.5.0 in /usr/local/lib/python3.10/dist-packages (from anthropic) (3.7.1)
Requirement already satisfied: distro<2,>=1.7.0 in /usr/lib/python3/dist-packages (from anthropic) (1.7.0)
Collecting httpx<1,>=0.23.0 (from anthropic)
  Downloading httpx-0.27.0-py3-none-any.whl (75 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 75.6/75.6 kB 10.0 MB/s eta 0:00:00
Requirement already satisfied: pydantic<3,>=1.9.0 in /usr/local/lib/python3.10/dist-packages (from anthropic) (2.7.1)
Requirement already satisfied: sniffio in /usr/local/lib/python3.10/dist-packages (from anthropic) (1.3.1)
Requirement already satisfied: tokenizers>=0.13.0 in /usr/local/lib/python3.10/dist-packages (from anthropic) (0.19.1)
Requirement already satisfied: typing-extensions<5,>=4.7 in /usr/local/lib/python3.10/dist-packages (from anthropic) (4.
Requirement already satisfied: idna>=2.8 in /usr/local/lib/python3.10/dist-packages (from anyio<5,>=3.5.0->anthropic) (3
Requirement already satisfied: exceptiongroup in /usr/local/lib/python3.10/dist-packages (from anyio<5,>=3.5.0->anthropi
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages (from httpx<1,>=0.23.0->anthropic) (20
Collecting httpcore==1.* (from httpx<1,>=0.23.0->anthropic)
  Downloading httpcore-1.0.5-py3-none-any.whl (77 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 77.9/77.9 kB 6.6 MB/s eta 0:00:00
Collecting h11<0.15,>=0.13 (from httpcore==1.*->httpx<1,>=0.23.0->anthropic)
```

```
  Downloading h11-0.14.0-py3-none-any.whl (58 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 58.3/58.3 kB 7.3 MB/s eta 0:00:00
Requirement already satisfied: annotated-types>=0.4.0 in /usr/local/lib/python3.10/dist-packages (from pydantic<3,>=1.9.
Requirement already satisfied: pydantic-core==2.18.2 in /usr/local/lib/python3.10/dist-packages (from pydantic<3,>=1.9.0
Requirement already satisfied: huggingface-hub<1.0,>=0.16.4 in /usr/local/lib/python3.10/dist-packages (from tokenizers>
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.16.4->t
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.16.4->t
Requirement already satisfied: tqdm>=4.42.1 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.16.
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.16.4
Requirement already satisfied: packaging>=20.9 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->huggi
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->huggingface
Installing collected packages: h11, httpcore, httpx, anthropic
Successfully installed anthropic-0.26.0 h11-0.14.0 httpcore-1.0.5 httpx-0.27.0
```

```
# 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)
```

```
Thank you for the warm welcome and the exciting challenge! I'm eager to contribute to this important project and help op

To solve this problem, I will formulate an integer linear optimization model. The objective is to maximize the total BED

Parameters:
- n: The number of patients (n = 17)
- C: The total maximal capacity of proton fractions (C = 100)
- BEDi(j,15-j): The BED score for patient i when receiving j proton fractions and 15-j photon fractions

Decision Variables:
- xij: Binary variable; xij = 1 if patient i receives j proton fractions, and 0 otherwise

Objective Function:
Maximize ∑i=1 to n ∑j=0 to 15 BEDi(j,15-j) * xij

Constraints:
1. Each patient must receive exactly one treatment plan:
   ∑j=0 to 15 xij = 1, for all i = 1 to n

2. The total number of proton fractions used must not exceed the maximal capacity:
   ∑i=1 to n ∑j=0 to 15 j * xij ≤ C

3. Binary constraints:
   xij ∈ {0,1}, for all i = 1 to n and j = 0 to 15

In this model:
- The objective function maximizes the total BED scores for all patients by summing the products of BEDi(j,15-j) and xij
- Constraint 1 ensures that each patient receives exactly one treatment plan, i.e., a combination of proton and photon f
- Constraint 2 guarantees that the total number of proton fractions used does not exceed the maximal capacity C.
- Constraint 3 specifies that the decision variables xij are binary, meaning they can only take values of 0 or 1.

By solving this integer linear optimization model, we can determine the optimal number of proton fractions for each pati
```

2. Prompt 2: Write Code based on Mathematical Model

```
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)
```

```
BED = np.random.randint(50, 100, size=(n, 16))

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

# Define decision variables
model.x = pyo.Var(range(n), range(16), domain=pyo.Binary)

# Define objective function
def obj_func(model):
    return sum(BED[i, j] * model.x[i, j] for i in range(n) for j in range(16))
model.obj = pyo.Objective(rule=obj_func, sense=pyo.maximize)

# Define constraints
def one_treatment_plan_rule(model, i):
    return sum(model.x[i, j] for j in range(16)) == 1
model.one_treatment_plan = pyo.Constraint(range(n), rule=one_treatment_plan_rule)

def capacity_rule(model):
    return sum(j * model.x[i, j] for i in range(n) for j in range(16)) <= C
model.capacity = pyo.Constraint(rule=capacity_rule)

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

# Print the results
print("Optimization Status:", results.solver.termination_condition)
print("Objective Value:", pyo.value(model.obj))

for i in range(n):
    for j in range(16):
        if pyo.value(model.x[i, j]) == 1:
            print(f"Patient {i+1}: {j} proton fractions, {15-j} photon fractions")
```

This code does the following:

1. We import the necessary libraries: `pyomo.environ` for modeling and `numpy` for generating sample data.

2. We define the sample data:
   - `n`: The number of patients (set to 17).
   - `C`: The total maximal capacity of proton fractions (set to 100).
   - `BED`: A randomly generated 2D array of BED scores for each patient and combination of proton and photon fraction

3. We create a Pyomo model using `pyo.ConcreteModel()`.

4. We define the decision variables `x` as binary variables using `pyo.Var()` with the `domain=pyo.Binary` argument.

5. We define the objective function using `pyo.Objective()` and the `obj_func` rule, which maximizes the total BED sco

6. We define the constraints:
   - Constraint 1: Each patient must receive exactly one treatment plan, using `pyo.Constraint()` and the `one_treatme
   - Constraint 2: The total number of proton fractions used must not exceed the maximal capacity, using `pyo.Constrai

7. We solve the optimization problem using the 'glpk' solver.

8. We print the optimization status, objective value, and the optimal treatment plan for each patient.

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

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

```
import pandas as pd

# Sample data
n = 17  # Number of patients
C = 100  # Total maximal capacity of proton fractions

# Generate sample BED scores (replace with actual data)
data = pd.read_csv("/content/drive/MyDrive/Thesis/ProblemData/IP/IP1/ProblemData.csv", header=None)
BED = pd.DataFrame(data).to_numpy()

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

# Define decision variables
model.x = pyo.Var(range(n), range(16), domain=pyo.Binary)

# Define objective function
def obj_func(model):
    return sum(BED[i, j] * model.x[i, j] for i in range(n) for j in range(16))
model.obj = pyo.Objective(rule=obj_func, sense=pyo.maximize)

# Define constraints
def one_treatment_plan_rule(model, i):
    return sum(model.x[i, j] for j in range(16)) == 1
model.one_treatment_plan = pyo.Constraint(range(n), rule=one_treatment_plan_rule)

def capacity_rule(model):
    return sum(j * model.x[i, j] for i in range(n) for j in range(16)) <= C
model.capacity = pyo.Constraint(rule=capacity_rule)

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

# Print the results
print("Optimization Status:", results.solver.termination_condition)
print("Objective Value:", pyo.value(model.obj))

for i in range(n):
    for j in range(16):
        if pyo.value(model.x[i, j]) == 1:
            print(f"Patient {i+1}: {j} proton fractions, {15-j} photon fractions")
```

```
Optimization Status: optimal
Objective Value: 8.239999999999998
Patient 1: 8 proton fractions, 7 photon fractions
Patient 2: 8 proton fractions, 7 photon fractions
Patient 3: 3 proton fractions, 12 photon fractions
Patient 4: 0 proton fractions, 15 photon fractions
Patient 5: 5 proton fractions, 10 photon fractions
Patient 6: 0 proton fractions, 15 photon fractions
Patient 7: 4 proton fractions, 11 photon fractions
Patient 8: 15 proton fractions, 0 photon fractions
Patient 9: 4 proton fractions, 11 photon fractions
Patient 10: 5 proton fractions, 10 photon fractions
Patient 11: 6 proton fractions, 9 photon fractions
Patient 12: 0 proton fractions, 15 photon fractions
Patient 13: 10 proton fractions, 5 photon fractions
Patient 14: 0 proton fractions, 15 photon fractions
Patient 15: 10 proton fractions, 5 photon fractions
Patient 16: 10 proton fractions, 5 photon fractions
Patient 17: 12 proton fractions, 3 photon fractions
```

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