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
```

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)
```

```
Thank you for the warm welcome and the challenging problem! I will formulate an integer linear optimization model to det

Given:
- n = 17 patients requiring radiotherapy
- Each patient needs 15 fractions in total (proton, photon, or a mix)
- BEDi(p, 15-p) is the BED score for patient i when p proton fractions and 15-p photon fractions are used
- C = 100 is the total maximal capacity for proton fractions

Decision Variables:
- xij: Binary variable indicating whether patient i receives j proton fractions (1 if true, 0 otherwise)

Objective Function:
Maximize the total BED scores for all patients:
max ∑(i=1 to n) ∑(j=0 to 15) BEDi(j, 15-j) * xij

Constraints:
1. Each patient must receive exactly 15 fractions in total:
   ∑(j=0 to 15) xij = 1, for all i = 1, 2, ..., n

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

3. Binary constraint for the decision variables:
   xij ∈ {0, 1}, for all i = 1, 2, ..., n and j = 0, 1, ..., 15

The complete integer linear optimization model is as follows:

max ∑(i=1 to n) ∑(j=0 to 15) BEDi(j, 15-j) * xij

subject to:
∑(j=0 to 15) xij = 1, for all i = 1, 2, ..., n
∑(i=1 to n) ∑(j=0 to 15) j * xij ≤ C
xij ∈ {0, 1}, for all i = 1, 2, ..., n and j = 0, 1, ..., 15

Solving this optimization model will determine the optimal number of proton fractions for each patient, maximizing the t
```

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)
```

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

# Sample BED scores (replace with actual data)
BED_scores = np.random.randint(50, 100, size=(n, 16))

# Create the model
model = pyo.ConcreteModel()
```

```
    model.fraction_constraint = pyo.ConstraintList()
    for i in range(n):
        model.fraction_constraint.add(sum(model.x[i, j] for j in range(16)) == 1)

    model.capacity_constraint = pyo.Constraint(expr=sum(sum(j * model.x[i, j] for j in range(16)) for i in range(n)) <= C)

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

    # Print the results
    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")
    ```

    In this code:

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

    2. We define the sample data:
       - `n = 17` represents the number of patients.
       - `C = 100` represents the total maximal capacity for proton fractions.
       - `BED_scores` is a sample 2D array of size (n, 16) containing random BED scores for each patient and each possible

    3. We create the pyomo model using `pyo.ConcreteModel()`.

    4. We define the decision variables `x` as binary variables using `pyo.Var()`. The indices represent the patient (i) a

    5. We define the objective function using `pyo.Objective()`. It maximizes the sum of BED scores for each patient based

    6. We define the constraints:
       - The `fraction_constraint` ensures that each patient receives exactly 15 fractions in total (proton + photon).
       - The `capacity_constraint` ensures that the total number of proton fractions used does not exceed the maximal capa

    7. We solve the model using the 'glpk' solver (you can change this to any other appropriate solver).

    8. Finally, we print the objective value and the treatment plan for each patient, specifying the number of proton and
```

## 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 for proton fractions

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

# Create the model
model = pyo.ConcreteModel()

# Decision variables
model.x = pyo.Var(range(n), range(16), within=pyo.Binary)

# Objective function
model.obj = pyo.Objective(expr=sum(sum(BED_scores[i, j] * model.x[i, j] for j in range(16)) for i in range(n)), sense=pyo.max

# Constraints
model.fraction_constraint = pyo.ConstraintList()
for i in range(n):
    model.fraction_constraint.add(sum(model.x[i, j] for j in range(16)) == 1)

model.capacity_constraint = pyo.Constraint(expr=sum(sum(j * model.x[i, j] for j in range(16)) for i in range(n)) <= C)

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

```
# Print the results
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")
```

```
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)