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/IP1.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:
 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-0.28.0-py3-none-any.whl (862 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 862.7/862.7 kB 6.3 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 7.9 MB/s eta 0:00:00
Collecting jiter<1,>=0.4.0 (from anthropic)
  Downloading jiter-0.4.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (328 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 328.3/328.3 kB 10.3 MB/s eta 0:00:00
Requirement already satisfied: pydantic<3,>=1.9.0 in /usr/local/lib/python3.10/dist-packages (from anthropic) (2.7.2)
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 8.1 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 6.3 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==2.18.3 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: packaging>=20.9 in /usr/local/lib/python3.10/dist-packages (from huggingface-hub<1.0,>=0.
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: 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: 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: jiter, h11, httpcore, httpx, anthropic
Successfully installed anthropic-0.28.0 h11-0.14.0 httpcore-1.0.5 httpx-0.27.0 jiter-0.4.1
```

```
# 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 integer linear optimization problem.

Decision Variables:
Let x[i,j] be a binary variable that indicates whether patient i receives j proton fractions.
- x[i,j] = 1 if patient i receives j proton fractions (and 15-j photon fractions)
- x[i,j] = 0 otherwise

where:
- i = 1, 2, ..., n (n = 17, the number of patients)
- j = 0, 1, ..., 15 (the possible number of proton fractions)

These variables will help us determine the optimal treatment plan for each patient, considering the limited proton thera
```

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)
```

```
Objective Function:

Maximize the total BED scores for all patients:

Max Z = Σ(i=1 to n) Σ(j=0 to 15) BED[i,j] * x[i,j]

where:
- BED[i,j] represents the BED score for patient i receiving j proton fractions (and 15-j photon fractions). These values
- x[i,j] are the binary decision variables, as defined earlier, that indicate whether patient i receives j proton fracti

The objective function aims to maximize the sum of BED scores across all patients by selecting the optimal number of pro
```

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)
```

```
Great! Now let's formulate the constraints for this integer linear optimization problem.

Constraints:

1. Each patient must receive a total of 15 fractions (either proton or photon):
   Σ(j=0 to 15) x[i,j] = 1, for all i = 1, 2, ..., n

   This constraint ensures that each patient is assigned exactly one treatment plan, which consists of a specific number

2. The total number of proton fractions used across all patients must not exceed the maximum capacity C:
   Σ(i=1 to n) Σ(j=0 to 15) j * x[i,j] ≤ C

   This constraint limits the total number of proton fractions used across all patients to be within the maximum capacit

3. Binary variable constraint:
   x[i,j] ∈ {0, 1}, for all i = 1, 2, ..., n, and j = 0, 1, ..., 15

   This constraint ensures that the decision variables x[i,j] are binary, meaning they can only take the values 0 or 1.

These constraints, along with the objective function and decision variables defined earlier, complete the formulation of
```

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_p11 + response_p12 + response_p13)}
    ]
)

response_p2 = message2.content[0].text


# Print response
print(response_p2)
```

```
# Sample data (replace with actual data from "ProblemData.csv")
n = 17  # Number of patients
C = 100  # Maximum proton therapy capacity
BED = {
    (1, 0): 72.0, (1, 1): 72.8, (1, 2): 73.6, ..., (1, 15): 86.0,
    (2, 0): 69.0, (2, 1): 69.8, (2, 2): 70.6, ..., (2, 15): 83.0,
    ...
    (17, 0): 70.5, (17, 1): 71.3, (17, 2): 72.1, ..., (17, 15): 84.5
}

# Create the Pyomo model
model = ConcreteModel()

# Define the decision variables
model.x = Var(range(1, n+1), range(0, 16), domain=Binary)

# Define the objective function
def obj_rule(model):
    return sum(BED[i, j] * model.x[i, j] for i in range(1, n+1) for j in range(0, 16))
model.OBJ = Objective(rule=obj_rule, sense=maximize)

# Define the constraints
def fraction_constraint_rule(model, i):
    return sum(model.x[i, j] for j in range(0, 16)) == 1
model.FractionConstraint = Constraint(range(1, n+1), rule=fraction_constraint_rule)

def capacity_constraint_rule(model):
    return sum(j * model.x[i, j] for i in range(1, n+1) for j in range(0, 16)) <= C
model.CapacityConstraint = Constraint(rule=capacity_constraint_rule)

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

# Print the results
print("Optimization Results:")
print("Objective Value:", model.OBJ())

print("Treatment Plan:")
for i in range(1, n+1):
    for j in range(0, 16):
        if model.x[i, j]() == 1:
            print(f"Patient {i}: {j} proton fractions, {15-j} photon fractions")
```

Note: The sample data for the BED scores (`BED` dictionary) is just a placeholder. Replace it with the actual data f

In this code:
- The decision variables `x[i,j]` are defined as binary variables using `Var(range(1, n+1), range(0, 16), domain=Bin
- The objective function is defined using `Objective(rule=obj_rule, sense=maximize)`, which maximizes the total BED s
- The constraint that each patient must receive a total of 15 fractions is defined using `Constraint(range(1, n+1),
- The constraint that the total number of proton fractions used across all patients must not exceed the maximum capa
- The `glpk` solver is used to solve the optimization problem.
- The optimal objective value and the treatment plan for each patient are printed as the results.

Remember to install the necessary dependencies (Pyomo and GLPK) before running the 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 *
```

```
import pandas as pd

# Sample data (replace with actual data from "ProblemData.csv")
n = 17  # Number of patients
C = 100  # Maximum proton therapy capacity
data = pd.read_csv("/content/drive/MyDrive/Thesis/ProblemData/IP/IP1/ProblemData.csv", header=None)
data = pd.DataFrame(data)

BED = {(i + 1, j): value for i, row in data.iterrows() for j, value in enumerate(row)} # ADJUSTED TO CORRECTLY LOAD THE DATA

# Create the Pyomo model
model = ConcreteModel()

# Define the decision variables
model.x = Var(range(1, n+1), range(0, 16), domain=Binary)

# Define the objective function
def obj_rule(model):
    return sum(BED[i, j] * model.x[i, j] for i in range(1, n+1) for j in range(0, 16))
model.OBJ = Objective(rule=obj_rule, sense=maximize)

# Define the constraints
def fraction_constraint_rule(model, i):
    return sum(model.x[i, j] for j in range(0, 16)) == 1
model.FractionConstraint = Constraint(range(1, n+1), rule=fraction_constraint_rule)

def capacity_constraint_rule(model):
    return sum(j * model.x[i, j] for i in range(1, n+1) for j in range(0, 16)) <= C
model.CapacityConstraint = Constraint(rule=capacity_constraint_rule)

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

# Print the results
print("Optimization Results:")
print("Objective Value:", model.OBJ())

print("Treatment Plan:")
for i in range(1, n+1):
    for j in range(0, 16):
        if model.x[i, j]() == 1:
            print(f"Patient {i}: {j} proton fractions, {15-j} photon fractions")
```

```
Optimization Results:
Objective Value: 8.239999999999998
Treatment Plan:
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)