NAME:KATO PAUL
YEAR OF STUDY :1
SEMESTER:2
REG NO :2023/U/MMU/BCS/00104

Basic resource allocation

```
from pulp import*
import matplotlib.pyplot as plt
import numpy as np
#create lp problem
model = LpProblem(name ="cost minimization",sense= LpMinimize)
#Lp variables
x = LpVariable("x", 0)
y = LpVariable("y",0)
#objective function
model+= 4*x + 5*y
#constraints
model += 2*x +3*y >= 10, "CPU_constraint"
model += x +2*y >= 5, "MEMORY_constraint"
model += 3*x + y >= 8, "STORAGE_constraint"
#Solving
model.solve()
#display results
print("optimal solution: ")
print(f"quantity of X (x):{x.varValue}")
print(f"quantity of Y (y):{y.varValue}")
print(f"Minimum cost (Z):{model.objective.value()}")


#graph
x = np.linspace(0,16,20000)

# convert to iqualities
y1 = (10-2*x)/3
y2=(5-x)/2
y3=(8-3*x)

# get feasible region
y4 = np.maximum.reduce([y1, y2, y3])
plt.fill_between(x , y4,11,color="gray", alpha=0.3,label="wanted region")

# plotting
plt.plot(x, y1, label="2*x +3*y >= 10")
plt.plot(x, y2,label="x +2*y >= 5")
plt.plot(x, y3,label="3*x + y >= 8")
```

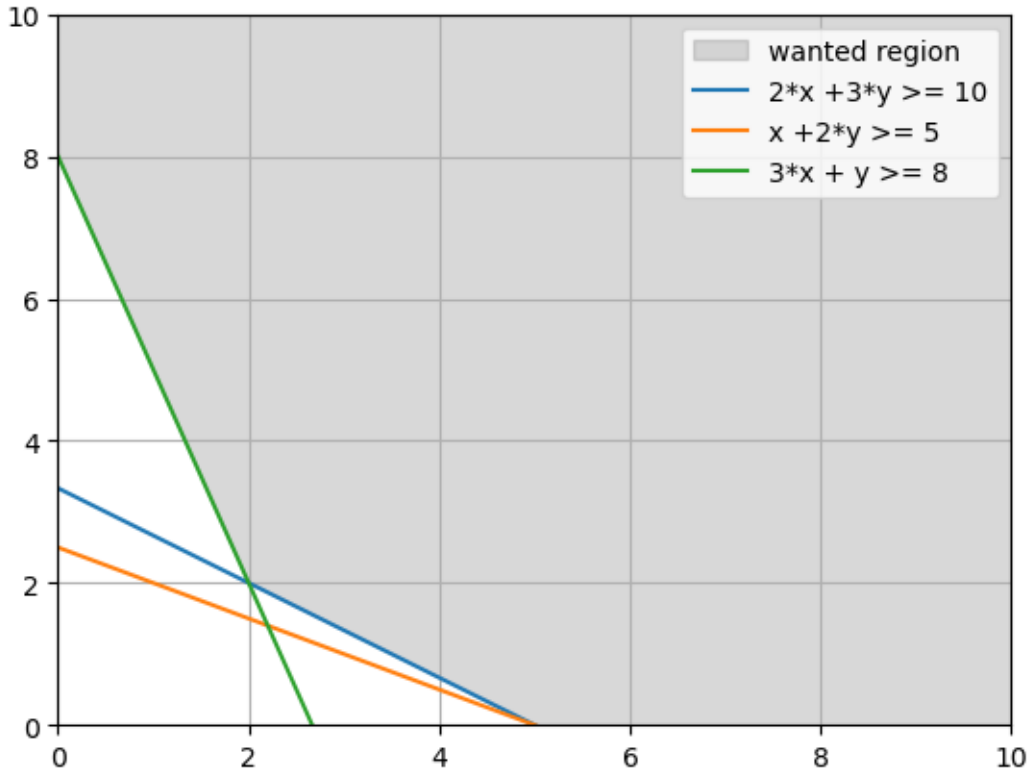

Figure 1: graph showing the wanted region

```
# limit and show
plt.xlim(0,10)
plt.ylim(0,10)
plt.legend()
plt.grid(True)
plt.show()
optimal solution:
quantity of X (x):2.0
quantity of Y (y):2.0
minimum cost (Z):18.0
```

1 load balancing

```
from pulp import*
```

```
import matplotlib.pyplot as plt

import numpy as np
# create alinear programming problem
model=LpProblem(name="cost_minimization",sense=LpMinimize)
# defining variables
x=LpVariable("x",0)
y=LpVariable("y",0)
#defining the objective function
model+=5*x+4*y
#defining the constraints
model+=2*x+3*y<=20, "server1capacity"
model+=4*x+2*y<=15, "server2capacity"
#solving
model.solve()
#plotting
optimal_x= x.varValue
optimal_y =y.varValue
optimal_value = model.objective.value()
print("optimum solution")
print("x: ",optimal_x)
print("y:",optimal_y)
print("z:", optimal_value)
#graph work
x=np.linspace(0,15,300)
#defing the constraints for purpose of ploting
y1=(20-2*x)/3
y2=(15-4*x)/2
#allocating the feasible region
y3=np.minimum.reduce([y1,y2])
plt.fill_between(x,y3,color="gray",alpha=0.4, label="wanted region")
#labeling
plt.plot(x,y1,label="2*x+3*y<=20")
plt.plot(x,y2,label="4*x+2*y<=15")
plt.scatter(optimal_x,optimal_y, color="red", label="optimum_solution")
#setting the limits of the plots
plt.xlim(0,10)
plt.ylim(0,10)
plt.legend()
plt.grid(True)
plt.xlabel('x')
plt.ylabel('y')
plt.title('feasible region')
plt.show()
optimum solution:
x (X) 0.0
```

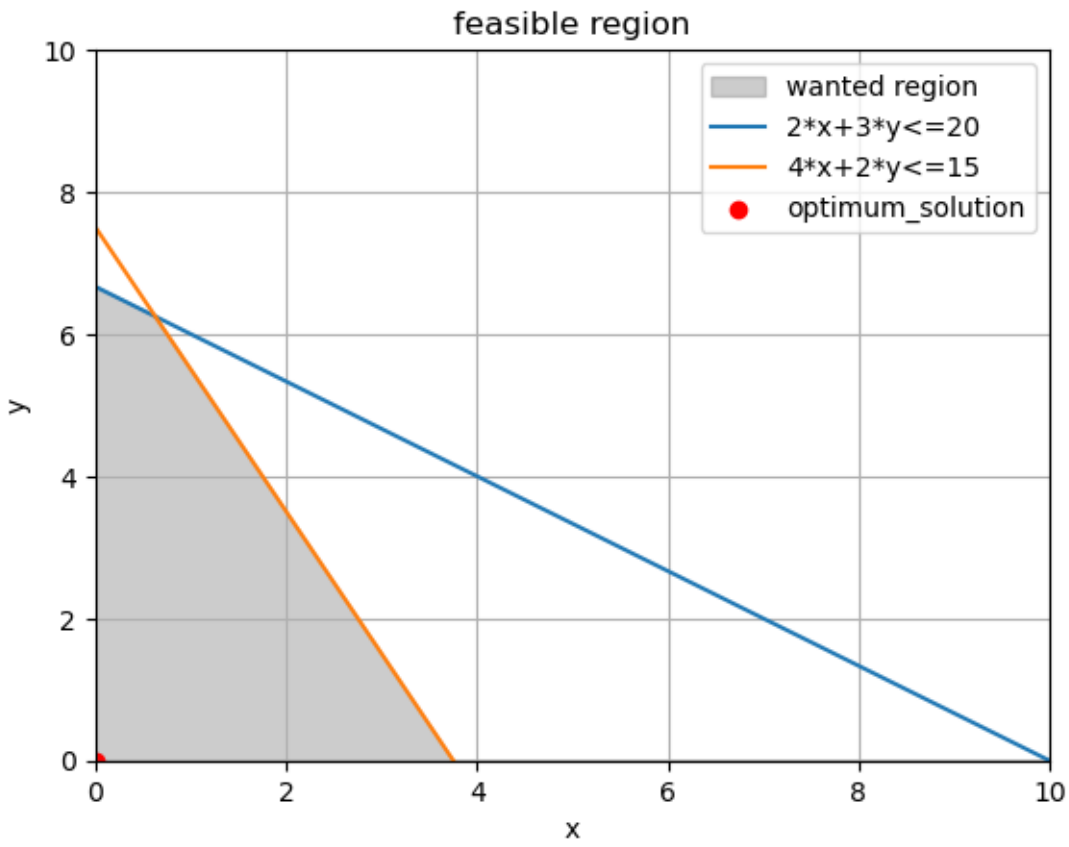

Figure 2: Caption

```
y (X) 0.0
z (X) 0.0
```

2 energy efficient and resources allocation

```
    from pulp import*
import matplotlib.pyplot as plt
import numpy as np
#defining the lp problem
model= LpProblem("energy_minimization" ,sense=LpMinimize)
#defining the variables
x=LpVariable('x',0)
y=LpVariable('y',0)
```

```
#defining the objective function
model+=3*x+2*y ,"objective"
#defining the constraints
model+=2*x+3*y>=15,"CPU Allocation"
model+=4*x+2*y>=10,"Memory Allocation"
#Solving the lp problems
model.solve()
print("optimal_solution")
print(f"x:{x.varValue}")
print(f"y:{y.varValue}")
print(f"optimum_solution: {model.objective.value()}")
#graph work
x=np.linspace(0,10,400)
#defining the constraints
y1=(15-2*x)/3
y2=(10-4*x)/2
#determining the feasible region
y3=np.maximum.reduce([y1,y2])
plt.fill_between(x,y3,color="gray",alpha=0.3,label="unwanted region")
#labeling
plt.plot(x,y1,label="2*x+3*y>=15")
plt.plot(x,y2,label="4*x+2*y>=10")
plt.legend()
plt.title("Araph")
plt.grid(True)
plt.xlim(0,10)
plt.ylim(0,10)
plt.xlabel("x")
plt.ylabel("y")
plt.show()
optimal solution
x:0.0
y:5.0
optimum solution:10.0
```

3 multi tenant resource sharing

```
    from pulp import*
import matplotlib.pyplot as plt
import numpy as np
#defining the linear programming problem
model=LpProblem(name="resource miminization",sense=LpMinimize)
#defining the variables
x=LpVariable('x',0)
```

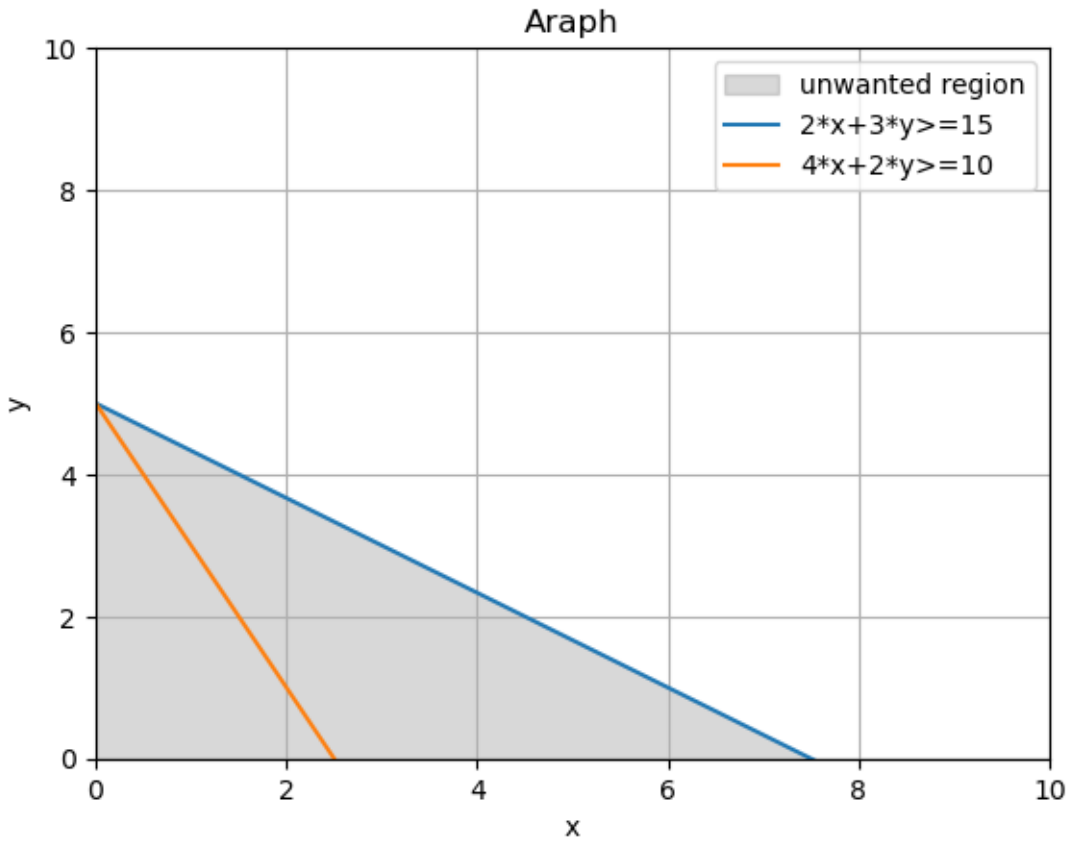

Figure 3: Caption

```
y=LpVariable('y',0)
#defining the objective function
model+=5*x+4*y
# defining the variables
model+=2*x+3*y>=12,"tenant 1"
model+=4*x+2*y>=18,"tenant 2"
#solving
model.solve()
print("optimal solution")
print(f"x:{x.varValue}")
print(f"y:{y.varValue}")
print(f"optimum solution:{model.objective.value()}")
#plotting
x=np.linspace(0,10,300)
#defining the constraits
y1=(12-2*x)/3
y2=(18-4*x)/2
#determing the feasible region
y3=np.maximum.reduce ([y1,y2])
plt.fill_between(x,y3,color="gray",alpha=0.4,label="unwanted region")
#labeling
plt.plot(x,y1,label="2*x+3*y>=12")
plt.plot(x,y2,label="4*x+2*y>=18")
#plots

plt.xlabel("x")
plt.ylabel("y")
plt.title('Agraph')
plt.legend()
plt.xlim(0,10)
plt.ylim(0,10)
plt.grid(True)
plt.show()
optimal solution
x:3.75
y:1.5
optimal solution:24.75
```

4 diet optimization

```
    from pulp import*
import matplotlib.pyplot as plt
import numpy as np
#defining the linear programmimg problem
```

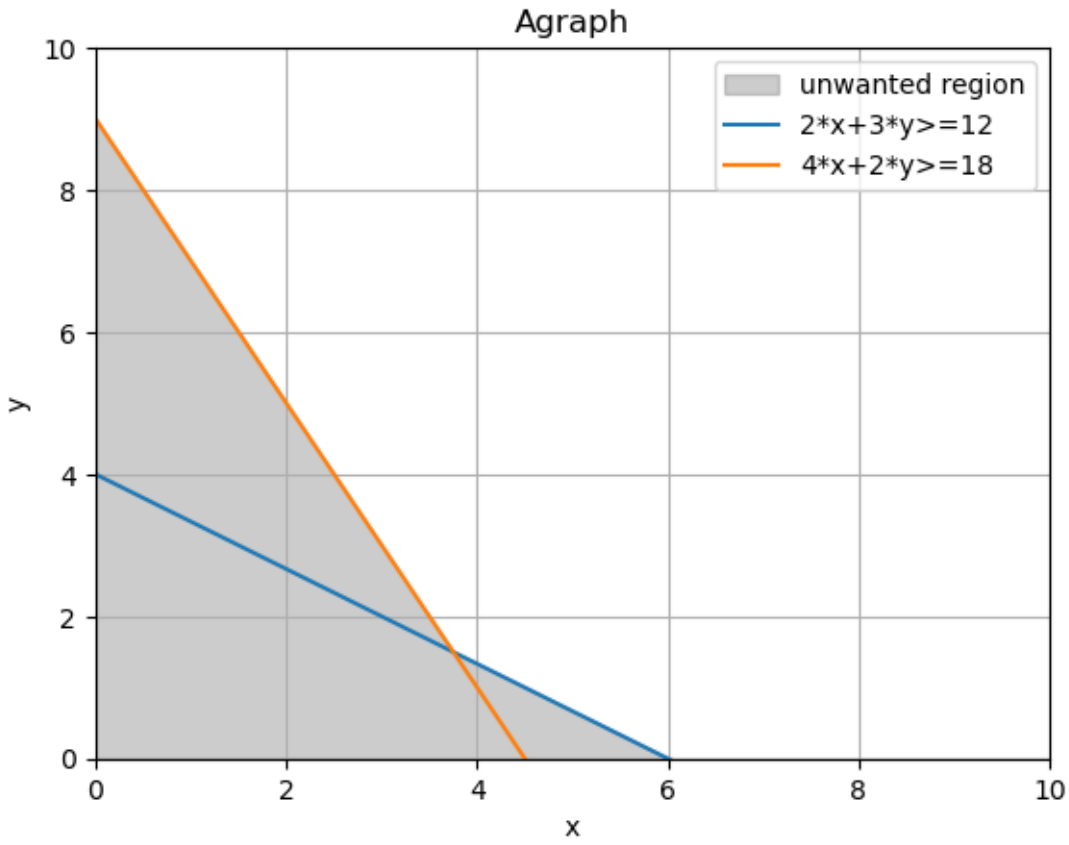

Figure 4: Caption

```
model=LpProblem(name="cost_minimization",sense=LpMinimize)
#defining the variable
x1=LpVariable("x1",0)
x2=LpVariable("x2",0)
#defining the objectivefunction
model+=3*x1+2*x2 ,"objective"
#defining the constraitnts
model+=2*x1+x2>=20, "proteins"
model+=3*x1+2*x2>=25 ,"vitamins"
#solving
model.solve()
optimal_x1=x1.varValue
optimal_x2=x2.varValue
optimal_Value=model.objective.value()
print("optimal solution")
print(f"x1:",optimal_x1)
print(f"x2:",optimal_x2)
print(f"z:",optimal_Value)
#ploting
x1=np.linspace(0,20,300)
#defining the objective
y1=(20-2*x1)
y2=(25-3*x1)/2
#labeling the constraitnts
plt.plot(x,y1,label="2*x1+x2>=20")
plt.plot(x,y2,label="3*x1+2*x2>=25")
#finding tyhe feasible region
y3=np.maximum.reduce([y1,y2])
plt.fill_between(x1,y3,color="red",alpha=0.4,label="feasible_region")
plt.legend()
plt.xlabel("x")
plt.ylabel("y")
plt.xlim(0,30)
plt.ylim(0,30)
plt.title("graph")
plt.grid(True)
plt.legend()
plt.show()
optimal solution
x1:10.0
x2:0.0
z:30.0
```

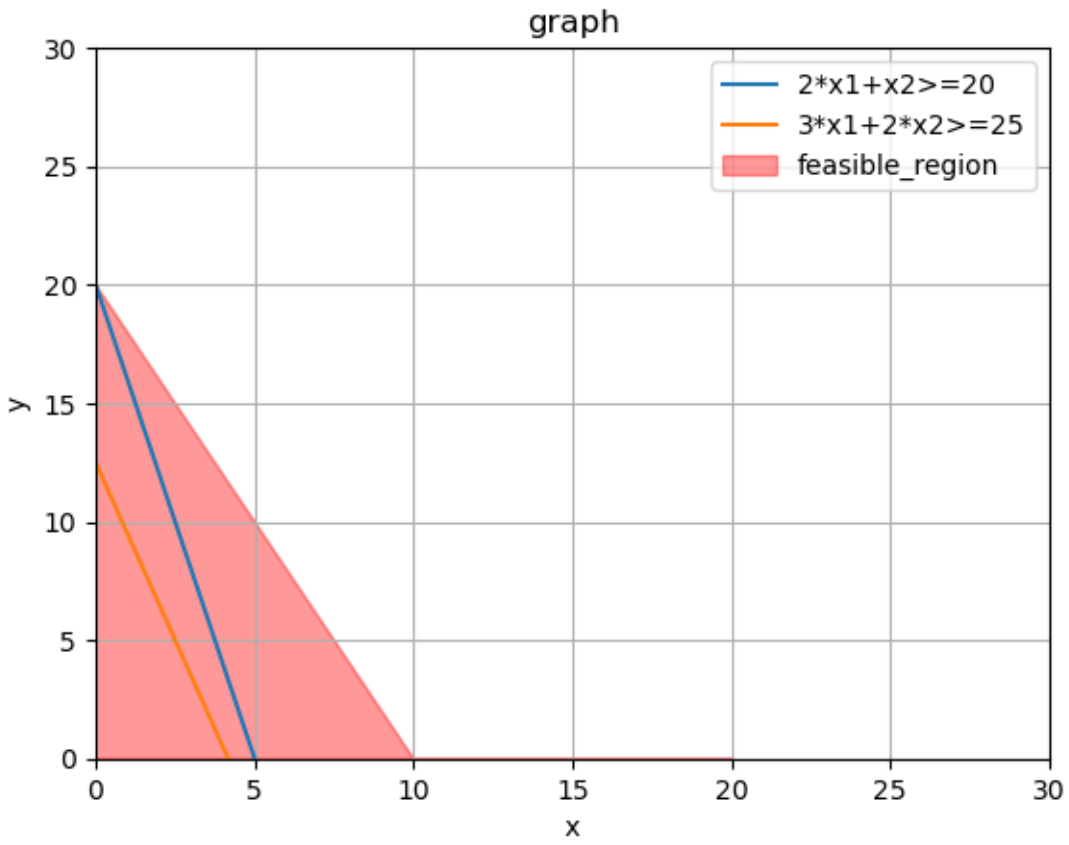

Figure 5: Caption

5 production planing

```
    from pulp import *
import matplotlib.pyplot as plt
import numpy as np

# Defining the linear programming problem
model = LpProblem(name="cost_minimization", sense=LpMinimize)

# Defining the variables
x1 = LpVariable("x1", lowBound=0)  # Adding lowBound to enforce non-negative constraint
x2 = LpVariable("x2", lowBound=0)  # Adding lowBound to enforce non-negative constraint

# Defining the objective function
model += 5 * x1 + 3 * x2, "objective"

# Defining the constraints
model += 2 * x1 + 3 * x2 <= 60, "labor"
model += 4 * x1 + 2 * x2 <= 80, "raw materials"

# Solving
model.solve()
optimal_x1 = x1.varValue
optimal_x2 = x2.varValue
optimal_value = value(model.objective)

print("Optimal solution:")
print(f"x1: {optimal_x1}")
print(f"x2: {optimal_x2}")
print(f"z: {optimal_value}")

# Plotting
x = np.linspace(0, 20, 300)

# Defining the constraints
y1 = (60 - 2 * x) / 3  # Fixing typo: x1 -> x
y2 = (80 - 4 * x) / 2  # Fixing typo: x1 -> x

# Labeling the constraints
plt.plot(x, y1, label="2*x1+3*x2<=60")
plt.plot(x, y2, label="4*x1+2*x2<=80")

# Finding the feasible region
y3 = np.minimum(y1, y2)  # Fixing maximum -> minimum
plt.fill_between(x, 0, y3, where=(y3 >= 0), color="red", alpha=0.4, label="unwanted region")
```

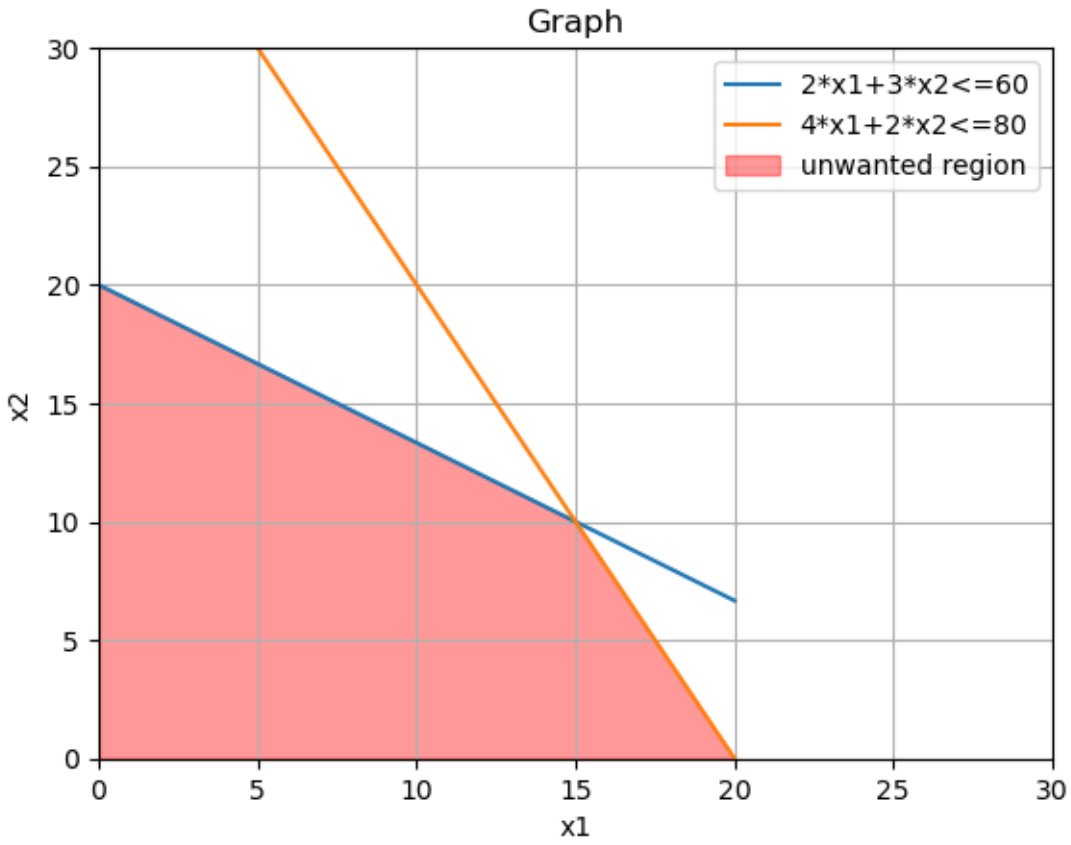

Figure 6: Caption

```
plt.legend()
plt.xlabel("x1")  # Fixing xlabel
plt.ylabel("x2")  # Fixing ylabel
plt.xlim(0, 30)
plt.ylim(0, 30)
plt.title("Graph")
plt.grid(True)
plt.show()
optimal solution
x:0.0
y:0.0
z:0.0
```

6 financial portfolio

```
    from pulp import *
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

# Defining the linear programming problem
model = LpProblem(name="constraint_maximization", sense=LpMaximize)

# Defining the variables
x1 = LpVariable("x1", lowBound=2000)  # Adding lower bounds
x2 = LpVariable("x2", lowBound=1500)  # Adding lower bounds
x3 = LpVariable("x3", lowBound=1000)  # Adding lower bounds

# Defining the objective function
model += 0.08 * x1 + 0.1 * x2 + 0.12 * x3, "objective"

# Defining the constraints
model += 2 * x1 + 3 * x2 + x3 <= 10000, "budget constraints "

# Solving
model.solve()
optimal_x1 = x1.varValue
optimal_x2 = x2.varValue
optimal_x3 = x3.varValue
optimal_value = value(model.objective)

print("Optimal solution:")
print(f"x1: {optimal_x1}")
print(f"x2: {optimal_x2}")
print(f"x3: {optimal_x3}")
print(f"z: {optimal_value}")

# Plotting 3D graph
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

# Generating x, y, z data points
x = np.linspace(2000, optimal_x1, 50)
y = np.linspace(1500, optimal_x2, 50)
x, y = np.meshgrid(x, y)
z = (10000 - 2 * x - 3 * y) / 1

# Plotting the surface
ax.plot_surface(x, y, z, cmap='viridis')
```

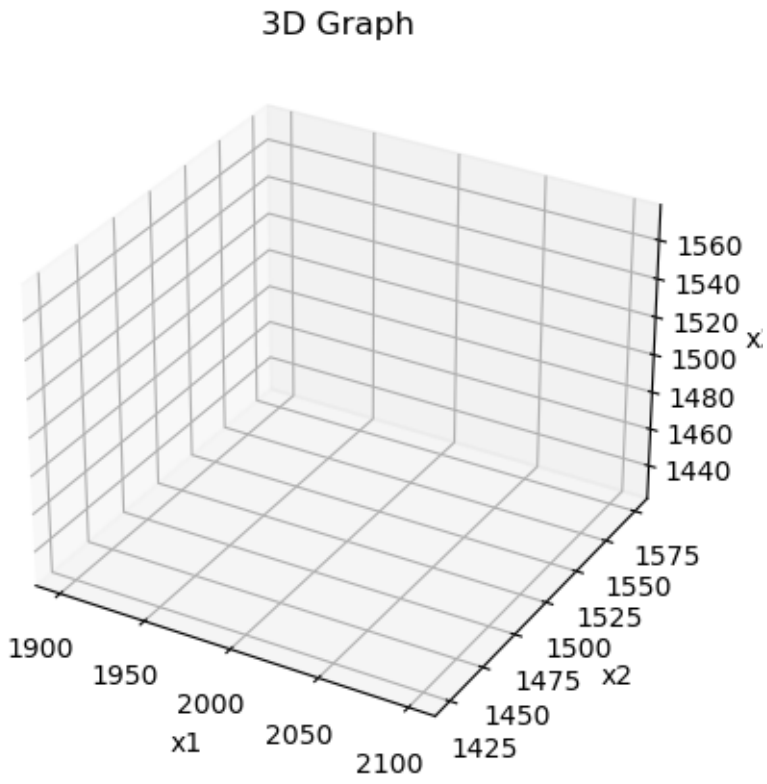

Figure 7: Caption

```
# Adding labels and title
ax.set_xlabel('x1')
ax.set_ylabel('x2')
ax.set_zlabel('x3')
ax.set_title('3D Graph')

plt.show()
optimal solution
x1:2000.0
x2:1500.0
x3:1500.0
z:490.0
```

7 production planning in manufacturing

```
    import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from scipy.optimize import linprog

c = np.array([-5, -3, -4])
A = np.array([[2, 3, 1], [4, 2, 5]])
b = np.array([1000, 120])
x_bounds = [(200, None), (300, None), (150, None)]

res = linprog(c, A_ub=-A, b_ub=-b, bounds=x_bounds, method='highs')

print("Minimum cost:", res.fun)
print("Minimum values of variables x1, x2, and x3:", res.x)

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

x = np.linspace(0, 350, 100)
y = np.linspace(0, 450, 100)
x, y = np.meshgrid(x, y)
z = (1000 - 2*x - 3*y) / 1
ax.plot_surface(x, y, z, alpha=0.5)

x = np.linspace(200, 350, 100)
z = np.linspace(0, 150, 100)
x, z = np.meshgrid(x, z)
y = (120 - 4*x - 5*z) / 2
ax.plot_surface(x, y, z, alpha=0.5)

y = np.linspace(300, 450, 100)
z = np.linspace(0, 150, 100)
y, z = np.meshgrid(y, z)
x = (1000 - 3*y - z) / 2
ax.plot_surface(x, y, z, alpha=0.5)

plt.show()
optimal solution;
x1:0.0
x2:0.0
x3:0.0
optimal solution 0.0
```

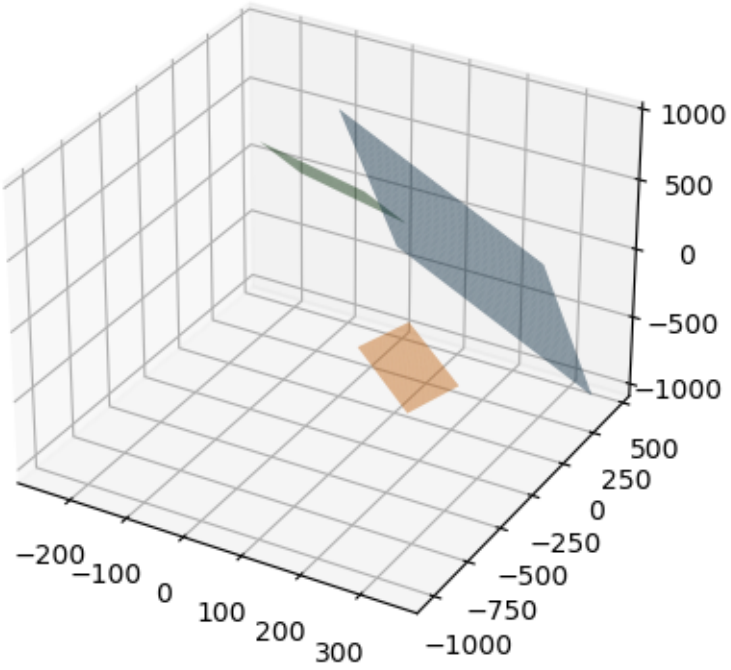

Figure 8: Caption