MOUNTAINS OF THE MOON UNIVERSITY

FACULTY OF SCIENCE TECHNOLOGY

AND INNOVATION

DEPARTMENT OF COMPUTER SCIENCE

Individual Coursework

NAME: AYESIGA ROONEY

REGISTRATION NUMBER: 2023/U/MMU/BCS/00517

COURSE CODE: BCS 1201

LECTURER'S NAME: Mr. Samuel Ocen

1 Resource Allocation in Cloud Computing

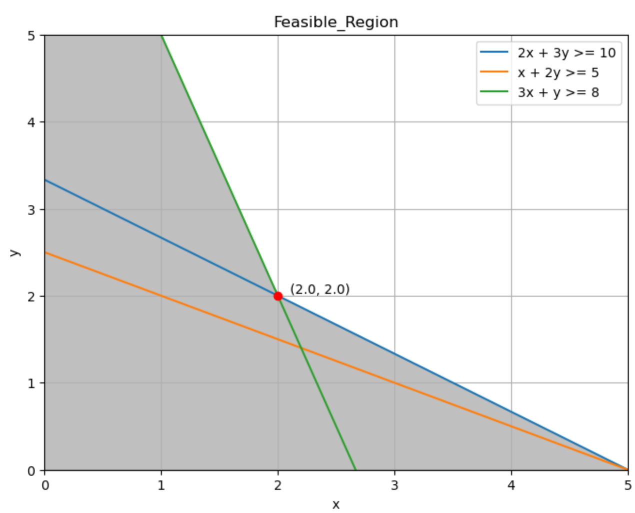
1.1 Number One: Basic Resource Allocation

```
# NUMBER 1


# import necessary libraries
from pulp import LpVariable, LpProblem,LpMinimize

# define the linear problem
lp = LpProblem(name="minimizing_cost", sense=LpMinimize)

# define the decision variables
x = LpVariable(name="X")
y = LpVariable(name="Y")

#define the objectives
lp += 4*x + 5*y ,"objective"

# define constraints
lp += 2*x + 3*y>=10, "CPU"
lp += x + 2*y>=5, "Memory"
lp += 3*x + y>=8, "Storage"

# solve the results
lp.solve()

# print results
print("OPTIMUM SOLUTION")
print(f"X = {x.varValue}")
print(f"Y = {y.varValue}")
print(f"optimum solution: {lp.objective.value()}")

# the graph
#import libraries
import numpy as np
from scipy.optimize import linprog
import matplotlib.pyplot as plt
#x array
x=np.linspace(0,10,100)
#convert constraints to inequalities
y1 = (10-2*x)/3
y2 = (5-x)/2
y3 = (8-3*x)
#plot constraits
```

```
plt.plot(x,y1, label="2x + 3y >=10 (cpu)")
plt.plot(x,y2, label="x + 2y >=5 (memory)")
plt.plot(x,y3, label="3x + y >=8 (storage)")
#feasible region
plt.fill_between(x,0,np.minimum.reduce([y1,y2,y3]),color="blue",alpha=0.5,label="feasible re
plt.xlabel("x")
plt.ylabel("y")
plt.title("feasible region for Basic Resource Allocation")
plt.ylim(0,5)
plt.xlim(0,5)
plt.legend()
plt.show()
```

Solution

```
Optimal_Solution:
x = 2.0
y = 2.0
Minimum_cost (Z) = 18.0
```



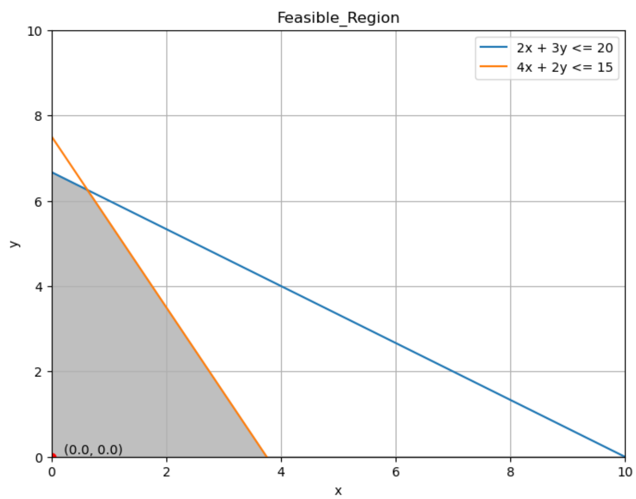
1.2 Number Two: Load Balancing

```
# NUMBER 2


# import libraries
from pulp import *

# define lp
lp = LpProblem(name="minimizing_the_overall_response_time", sense=LpMinimize)

# define decision variables
x = LpVariable(name="x", lowBound=0)
y = LpVariable(name="y", lowBound=0)

# define the objective
lp += 5*x + 4*y, "objective"

# define constraints
lp += 2*x + 3*y <=20, "server1"
lp += 4*x + 2*y <=15, "server2"

# solve
lp.solve()

print("RESULTS")
print(f"x = {x.varValue}")
print(f"y = {y.varValue}")
print(f"optimum solution = {lp.objective.value()}")


# the graph
#import libraries
import numpy as np
from scipy.optimize import linprog
import matplotlib.pyplot as plt
#x array
x=np.linspace(0,10,100)
#convert constraints to inequalities
y1 = (20-2*x)/3
y2 = (15-4*x)/2
#plot constraits
plt.plot(x,y1, label="2x + 3y <=20 (server1)")
plt.plot(x,y2, label="4x + 2y <=15 (server2)")
#feasible region
plt.fill_between(x,0,np.minimum(y1,y2),color="grey",alpha=0.5,label="feasible region")
```

```
plt.xlabel("x")
plt.ylabel("y")
plt.title("feasible region for Load balancing")
plt.ylim(0,10)
plt.xlim(0,10)
plt.legend()
plt.show()
```

Solution

```
Optimal_Solution:
x = 0.0
y = 0.0
Minimum_response_time (Z) = 0.0
```



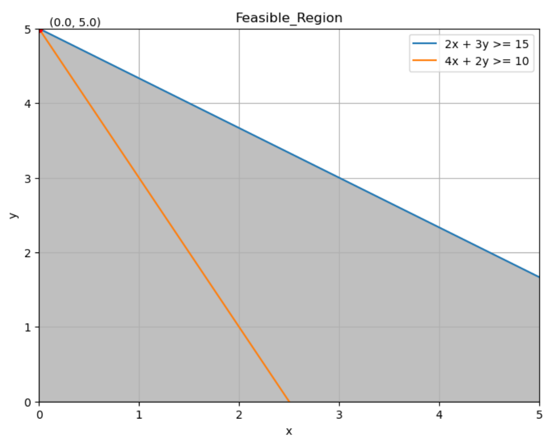
1.3 Number Three: Energy Efficient Resource Allocation

```
# NUMBER 3


# import necessary libraries
from pulp import *

# define the linear problem
Lp = LpProblem(name="cloud data", sense=LpMinimize)

#  define the decision values
x = LpVariable(name="x", lowBound=0)
y = LpVariable(name="y", lowBound=0)

# define the objectives
Lp += 3*x + 2*y, "objective"

# define the constraints
Lp += 2*x + 3*y >=15, "cpu_allocation"
Lp += 4*x + 2*y >=10, "memory_allocation"

# solve the results
Lp.solve()

# print results
print("Result")
print(f"X = {x.varValue}")
print(f"Y = {y.varValue}")
print(f"optimum solution: {Lp.objective.value()}")

# the graph
#import libraries
import numpy as np
from scipy.optimize import linprog
import matplotlib.pyplot as plt
#x array
x=np.linspace(0,20,100)
#convert constraints to inequalities
y1 = (15-2*x)/3
y2 = (10-4*x)/2
#plot constraits
plt.plot(x,y1, label="2x + 3y >=15 (cpu allocation)")
plt.plot(x,y2, label="4x + 2y >=10 (memory allocation")
#feasible region
plt.fill_between(x,100,np.maximum.reduce([y1,y2]),color="orange",alpha=0.5,label="feasible r
```

```
plt.xlabel("x")
plt.ylabel("y")
plt.title("feasible region for energy Efficient resource allocation")
plt.ylim(0,15)
plt.xlim(0,10)
plt.legend()
plt.show()
```

Solution

```
Optimal_Solution:
x = 0.0
y = 5.0
Minimum_total_energy_consumption (Z) = 10.0
```



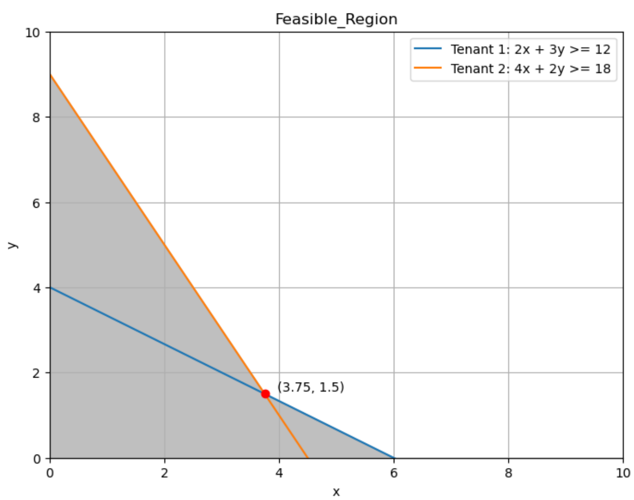
1.4 Number Four: Multi-Tenant Resource Sharing

```
# NUMBER 4


# import libraries
from pulp import *

# define lp
lp = LpProblem(name="allocating_resources", sense=LpMinimize)

# define decision variables
x = LpVariable(name="x", lowBound=0)
y = LpVariable(name="y", lowBound=0)

# define the objective
lp += 5*x + 4*y, "objective"

# define constraints
lp += 2*x + 3*y >=12, "tenant 1"
lp += 4*x + 2*y >=18, "tenant 2"

# solve
lp.solve()

print("RESULTS")
print(f"x = {x.varValue}")
print(f"y = {y.varValue}")
print(f"optimum solution = {lp.objective.value()}")

# the graph
#import libraries
import numpy as np
from scipy.optimize import linprog
import matplotlib.pyplot as plt
#x array
x=np.linspace(0,25,100)
#convert constraints to inequalities
y1 = (12-2*x)/3
y2 = (18-4*x)/2
#plot constraits
plt.plot(x,y1, label="2x + 3y >=12 (tenant1)")
plt.plot(x,y2, label="4x + 2y >=18 (tenant2)")
#feasible region
plt.fill_between(x,200,np.maximum.reduce([y1,y2]),color="red",alpha=0.5,label="feasible regi
plt.xlabel("x")
```

```
plt.ylabel("y")
plt.title("feasible region for Multi_Tenant Resource Sharing")
plt.ylim(0,20)
plt.xlim(0,20)
plt.legend()
plt.show()
```

Solution

```
Optimal_Solution:
x = 3.75
y = 1.5
Minimum_total_cost (Z) = 24.75
```



2 Transportation Logistics Optimization

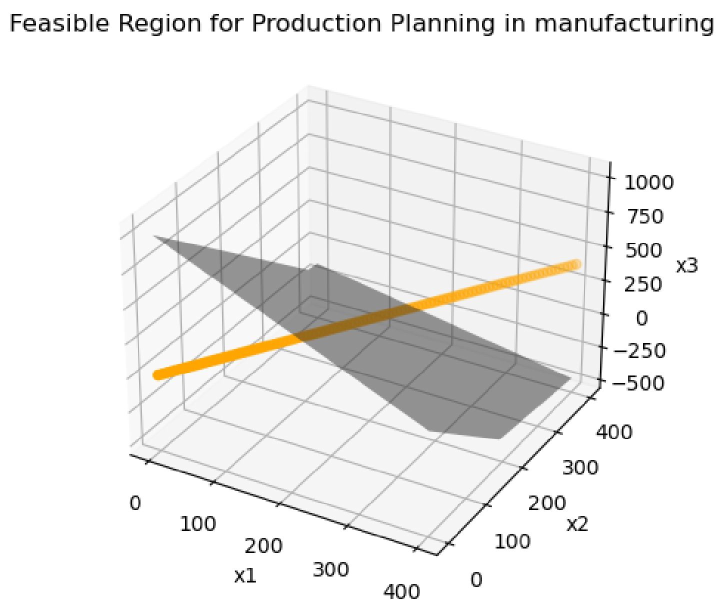
2.1 Number Five: Production Planning in Manufacturing

```
# NUMBER 5

# import libraries
from pulp import *

# define lp
lp = LpProblem(name="production_costs", sense=LpMinimize)

# define decision variables
x1 = LpVariable(name="x1", lowBound=0)
x2= LpVariable(name="x2", lowBound=0)
x3= LpVariable(name="x3", lowBound=0)

# define the objective
lp += 5*x1 + 3*x2 + 4*x3, "objective"

# define constraints
lp += 2*x1 + 3*x2 + 1*x3 <=1000, "raw material"
lp += 4*x1 + 2*x2 + 5*x3 <=120, "labor hours"
lp += x1 >=200
lp += x2 >=300
lp += x3 >=150

# solve
lp.solve()

print("RESULTS")
print(f"x1 = {x1.varValue}")
print(f"x2 = {x2.varValue}")
print(f"x3 = {x3.varValue}")
print(f"optimum solution x1,x2,x3 = {lp.objective.value()}")


# Plotting the feasible region (in 3D space)
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

x = np.linspace(0, 400, 100)
y = np.linspace(0, 400, 100)
x, y = np.meshgrid(x, y)
z1 = (1000 - 2*x - 3*y) / 1
z2 = (120 - 4*x - 2*y) / 5
```

```
ax.plot_surface(x, y, np.maximum(z1, z2), alpha=0.5, rstride=100, cstride=100, color='gray'
ax.set_xlabel('x1')
ax.set_ylabel('x2')
ax.set_zlabel('x3')
ax.set_title('Feasible Region for Production Planning in manufacturing')
ax.scatter(x[0], x[1], x[2], color='orange', label='Optimal Solution')

plt.show()
```

Solution

```
RESULTS
x1 = 200.0
x2 = 300.0
x3 = 0.0
optimum solution x1,x2,x3 = 1900.0
```

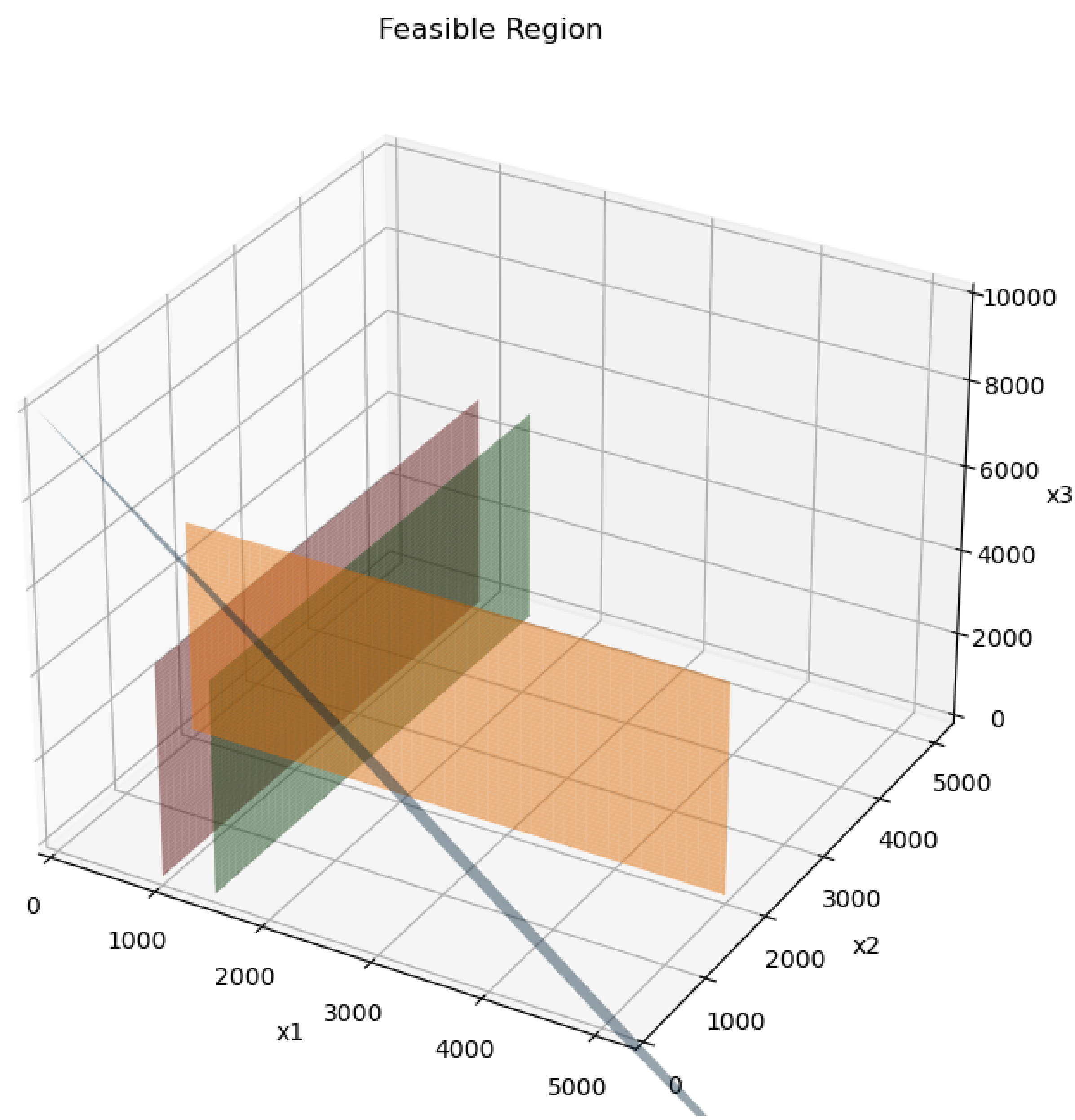


2.2 Number Six: Financial Portfolio Optimization

```
#libraries
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from pulp import LpVariable, LpMaximize, LpProblem

#the problem
problem = LpProblem(name="Financial_portfolio_optimization", sense=LpMaximize)

#decision variables
x1 = LpVariable(name="x", lowBound=0)
x2 = LpVariable(name="y", lowBound=0)
x3 = LpVariable(name="z", lowBound=0)

# Define the objective function coefficients
problem += 0.08*x1 + 0.1*x2 + 0.12*x3, "objective"

# Coefficients of the inequality constraints
problem += 2*x1 + 3*x2 + x3 <= 10000, "budget"
problem += x1 >= 2000
problem += x2 >= 1500
problem += x3 >= 1000


# Solve linear programming problem
problem.solve()

# Display the results
print("OPTIMAL SOLUTION:")
print(f"X: {x1.varValue}")
print(f"Y: {x2.varValue}")
print(f"Z: {x3.varValue}")
print(f"Minimum cost: {problem.objective.value()}")


# Create a meshgrid for A, B, and C
A_vals = np.linspace(0, 2500, 50)
B_vals = np.linspace(0, 2000, 50)
A_grid, B_grid = np.meshgrid(A_vals, B_vals)

# Calculate the corresponding z-values (ROI function)
C_vals = (10000 - 2 * A_grid - 3 * B_grid)  # Budget constraint
ROI_vals = 0.08 * A_grid + 0.1 * B_grid + 0.12 * C_vals
```

```
# Create the 3D plot
fig = plt.figure(figsize=(10, 6))
ax = fig.add_subplot(111, projection='3d')

# Plot the feasible region (constraints)
ax.plot([2000, 2000], [0, 2000], [0, 470], color='red', linestyle='--', linewidth=2, label=
ax.plot([0, 2500], [1500, 1500], [0, 470], color='green', linestyle='--', linewidth=2, label
ax.plot([0, 2500], [0, 2000], [470, 470], color='blue', linestyle='--', linewidth=2, label=

# Highlight the optimum point
optimum_A = 2000
optimum_B = 1500
optimum_ROI = 470
ax.the plotscatter(optimum_A, optimum_B, optimum_ROI, color='purple', s=100, label='Optimum

# Set labels and title
ax.set_xlabel('A')
ax.set_ylabel('B')
ax.set_zlabel('ROI')
ax.set_title('Return On Investiment Maximization')

# Add a legend
ax.legend()

# Show
plt.show()
```

Solution



Feasible Region
x1
x2
x3
0
1000
2000
3000
4000
5000
0
1000
2000
3000
4000
5000
0
2000
4000
6000
8000
10000

3 Project Resource Allocation Optimization

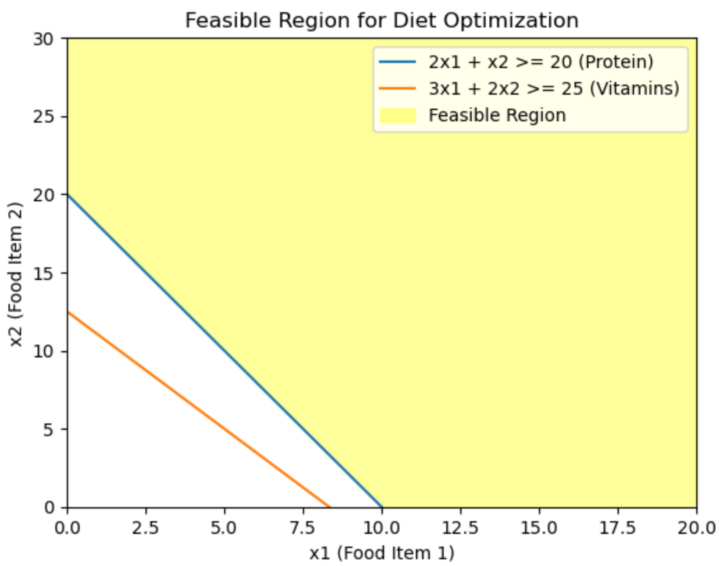
3.1 Number Seven: Diet Optimization

```
# NUMBER 7



# import libraries
from pulp import *

# define lp
lp = LpProblem(name="daily_diet", sense=LpMinimize)

# define decision variables
x1 = LpVariable(name="x1", lowBound=0)
x2 = LpVariable(name="x2", lowBound=0)

# define the objective
lp += 3*x1 + 4*x2, "objective"

# define constraints
lp += 2*x1 + 1*x2 >=20, "protein"
lp += 3*x1 + 2*x2 >=25, "vitamin"

# solve
lp.solve()

print("RESULTS")
print(f"x1 = {x1.varValue}")
print(f"x2 = {x2.varValue}")
print(f"optimum solution = {lp.objective.value()}")


#graph
#import libraries
import matplotlib.pyplot as plt
import numpy as np

# Plotting the feasible region
x1 = np.linspace(0, 20, 100)

x2_1 = 20 - 2*x1  # Protein constraint
x2_2 = (25 - 3*x1) / 2  # Vitamins constraint

plt.plot(x1, x2_1, label='2x1 + x2 >= 20 (Protein)')
```

```
plt.plot(x1, x2_2, label='3x1 + 2x2 >= 25 (Vitamins)')

plt.fill_between(x1,200, np.maximum.reduce([x2_1,x2_2 ]), color='yellow', alpha=0.4, label='
plt.xlim(0,20)
plt.ylim(0,30)
plt.xlabel('x1 (Food Item 1)')
plt.ylabel('x2 (Food Item 2)')
plt.title('Feasible Region for Diet Optimization')
plt.legend()
plt.show()
```

Solution

```
RESULTS
x1 = 10.0
x2 = 0.0
optimum solution = 30.0
```



3.2 Number Eight: Production Planning

```
# NUMBER 8



# import libraries
from pulp import *

# define lp
lp = LpProblem(name="profi_maximization", sense=LpMaximize)

# define decision variables
x1 = LpVariable(name="x1", lowBound=0)
x2 = LpVariable(name="x2", lowBound=0)

# define the objective
lp += 5*x1 + 3*x2, "objective"

# define constraints
lp += 2*x1 + 3*x2 <=60, "labor"
lp += 4*x1 + 2*x2 <=80, "raw materials"

# solve
lp.solve()

print("RESULTS")
print(f"x1 = {x1.varValue}")
print(f"x2 = {x2.varValue}")
print(f"optimum solution = {lp.objective.value()}")


#graph
#import libraries
import matplotlib.pyplot as plt
import numpy as np

# Plotting the feasible region
x1 = np.linspace(0, 100, 100)
# making one of the variables the subject in each constrait
x2_1 = (60 - 2*x1)/3  # labor
x2_2 = (80 - 4*x1) / 2  # raw materials

plt.plot(x1, x2_1, label='2x1 + 3x2 <= 60 (labor)')
plt.plot(x1, x2_2, label='4x1 + 2x2 <= 80 ( raw materials)')
```

```
plt.fill_between(x1,0, np.minimum.reduce([x2_1,x2_2] ), color='pink', alpha=0.6, label='Feas
plt.xlim(0,50)
plt.ylim(0,50)
plt.xlabel('x1 (Labor)')
plt.ylabel('x2 (raw materials)')
plt.title('Feasible Region for Production planning')
plt.legend()
plt.show()
```

Solution

```
Optimal solution:
x1 = 15.0
x2 = 10.0
Maximum profit (Z) = 105.0
```