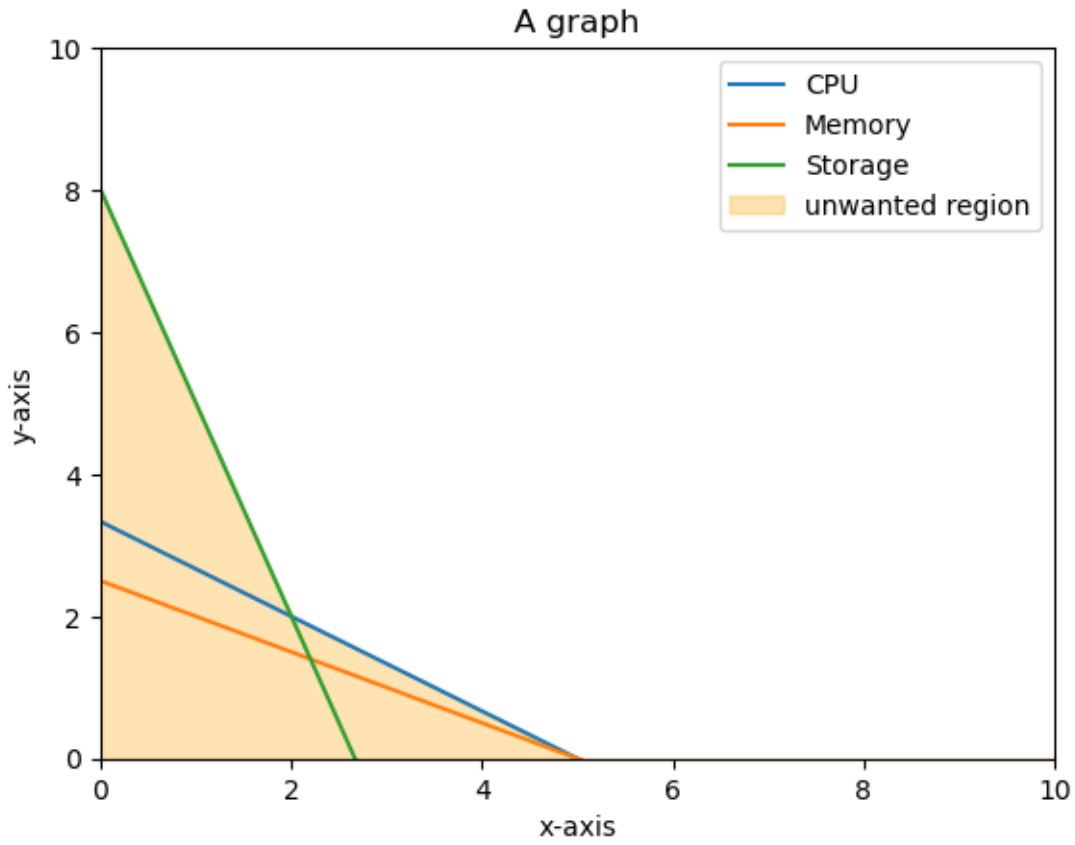
MOUNTAINS OF THE MOON UNIVERSITY
FUCULTY OF INNOVATION, SCIENCE AND TECHNOLOGY
BARCHELOR OF SCIENCE IN COMPUTER SCIENCE
NAME: KAGORO ENOCK
REG. NO.: 2023/U/MMU/BCS/00521
LECTURER: MR. OCEN SAMUEL

]

```
  # no1
# import libraries
from pulp import LpProblem,LpVariable,LpMinimize
import numpy as np
import matplotlib.pyplot as plt

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

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

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

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

# solve
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
```

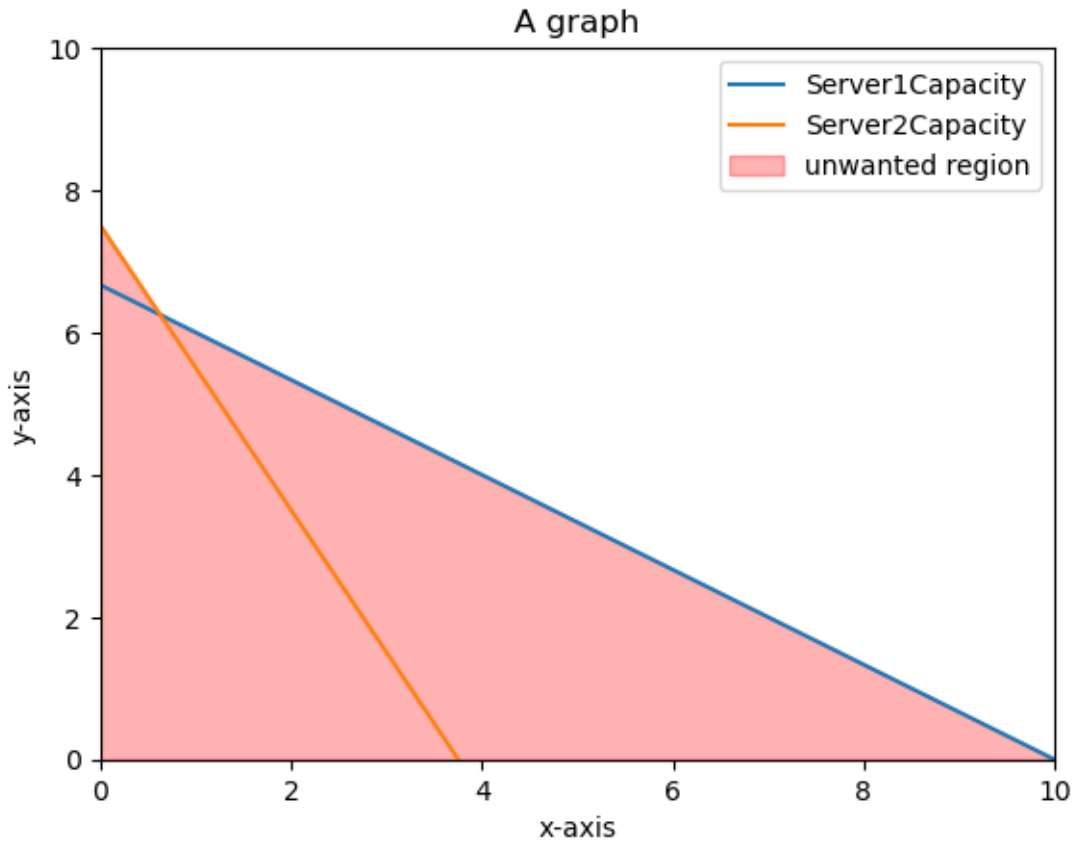
```
plt.legend()
plt.show()


Result
X: 2.0
Y: 2.0
Optimum solution: 18.0
```




```
    #Number 2
# import libraries
from pulp import LpProblem,LpVariable,LpMaximize
import numpy as np
import matplotlib.pyplot as plt

# define the lp
```

```
lp = LpProblem(name="maximize_workload" , sense=LpMaximize)

# 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 the contraints
lp += 2*x + 3*y <=10, "Server1Capacity"
lp += 4*x + 2*y <= 15, "Server2Capacity"

# solve
lp.solve()

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

# the graph

# x array
x = np.linspace(0,10,100)

# covert the constraints to ineq
y1 = (20 - 2*x)/3
y2 = (15 - 4*x)/2

plt.plot(x,y1,label="Server1Capacity")
plt.plot(x,y2, label="Server2Capacity")

# feasible region
y4  = np.maximum.reduce([y1,y2])
plt.fill_between(x,y4, color="red", alpha=0.3, label="unwanted region")

# limits
plt.xlim(0,10)
plt.ylim(0,10)
plt.title("A graph")
plt.xlabel("x-axis")
plt.ylabel("y-axis")
plt.legend()
plt.show()
```

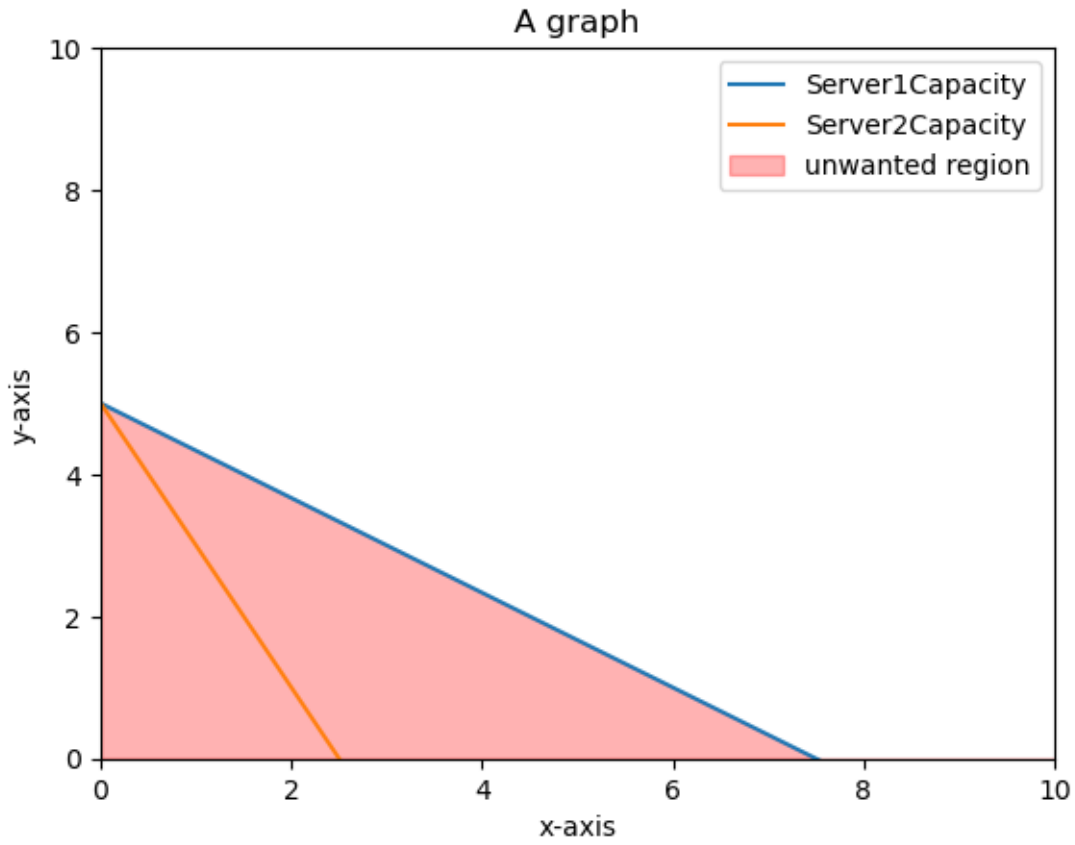
```
Result
X: 3.125
Y: 1.25
Minimum solution: 20.625
```




```
    #Number 3

# import libraries
from pulp import LpProblem,LpVariable,LpMinimize
import numpy as np
import matplotlib.pyplot as plt

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

# define decision variables
```

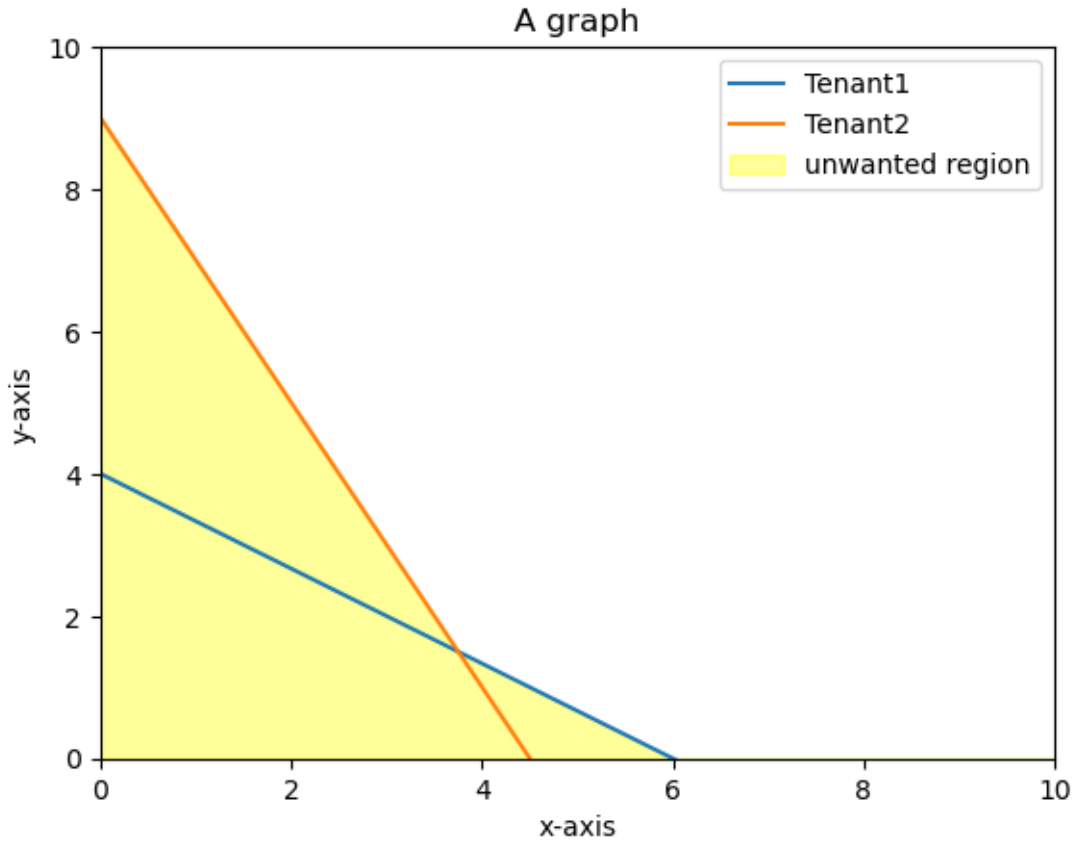
```
x = LpVariable(name="X",lowBound=0)
y = LpVariable(name="Y",lowBound=0)

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

# define the contraints
lp += 2*x + 3*y >=15, "CPUAllocation"
lp += 4*x + 2*y >= 10, "MemoryAllocation"

# solve
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

# x array
x = np.linspace(0,10,100)

# covert the constraints to ineq
y1 = (15 - 2*x)/3
y2 = (10 - 4*x)/2

plt.plot(x,y1,label="Server1Capacity")
plt.plot(x,y2, label="Server2Capacity")

# feasible region
y4  = np.maximum.reduce([y1,y2])
plt.fill_between(x,y4, color="red", alpha=0.3, label="unwanted region")

# limits
plt.xlim(0,10)
plt.ylim(0,10)
plt.title("A graph")
plt.xlabel("x-axis")
plt.ylabel("y-axis")
plt.legend()
plt.show()

Result
X: 0.0
```

```
Y: 5.0
Optimum solution: 10.0
```



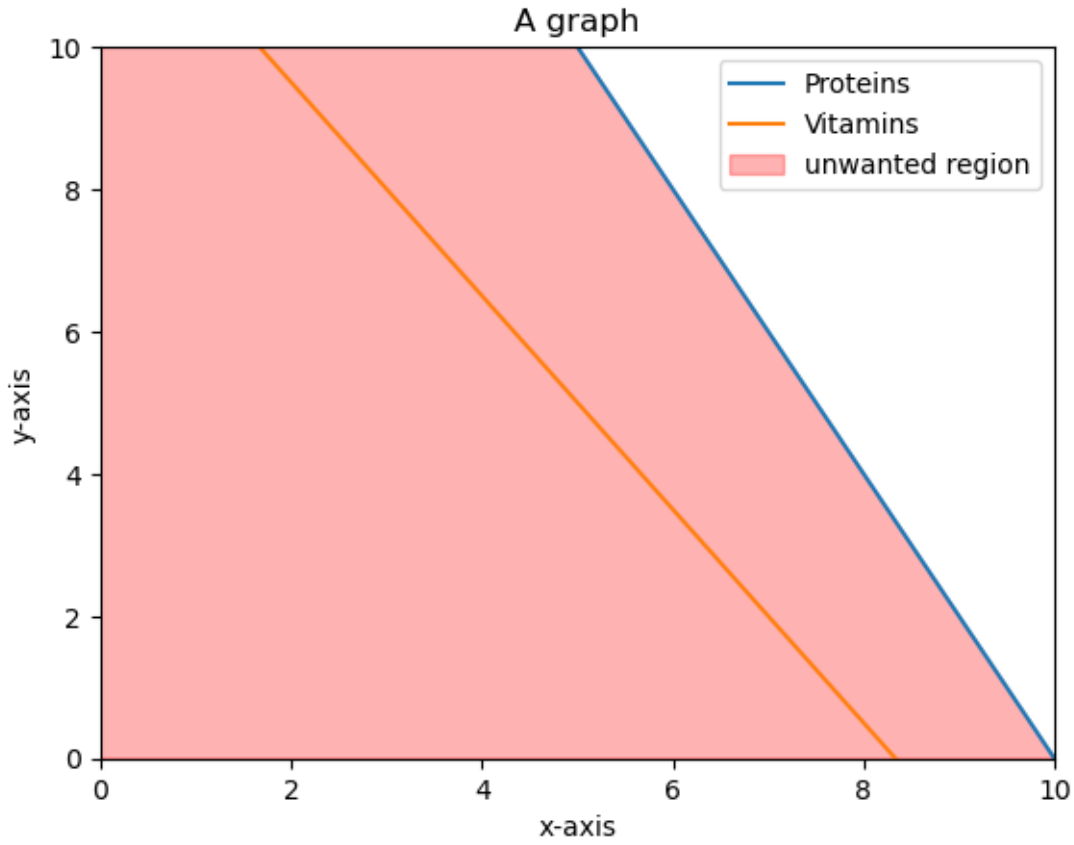

```
    #Number 4
# import libraries
from pulp import LpProblem,LpVariable,LpMinimize
import numpy as np
import matplotlib.pyplot as plt

# define the lp
lp = LpProblem(name="minimize_tenant_Resource" , 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 the contraints
lp += 2*x + 3*y >=12, "Tenant1"
lp += 4*x + 2*y <= 18, "Tenant2"

# solve
lp.solve()

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

# the graph

# x array
x = np.linspace(0,10,100)

# covert the constraints to ineq
y1 = (12 - 2*x)/3
y2 = (18 - 4*x)/2

plt.plot(x,y1,label="Tenant1")
plt.plot(x,y2, label="Tenant2")

# feasible region
y4  = np.maximum.reduce([y1,y2])
plt.fill_between(x,y4, color="yellow", alpha=0.4, label="unwanted region")

# limits
plt.xlim(0,10)
plt.ylim(0,10)
plt.title("A graph")
plt.xlabel("x-axis")
plt.ylabel("y-axis")
plt.legend()
plt.show()

Result
X: 0.0
Y: 4.0
Minimum solution: 16.0
```




```
    #Number 7
# import libraries
from pulp import LpProblem,LpVariable,LpMinimize
import numpy as np
import matplotlib.pyplot as plt

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

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

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

# define the contraints
lp += 2*x + y >=20, "Proteins"
```

```
lp += 3*x + 2*y >= 25, "Vitamins"

# solve
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

# x array
x = np.linspace(0,10,100)

# covert the constraints to ineq
y1 = (20 - 2*x)
y2 = (25 - 3*x)/2

plt.plot(x,y1,label="Proteins")
plt.plot(x,y2, label="Vitamins")

# feasible region
y4  = np.maximum.reduce([y1,y2])
plt.fill_between(x,y4, color="red", alpha=0.3, label="unwanted region")

# limits
plt.xlim(0,10)
plt.ylim(0,10)
plt.title("A graph")
plt.xlabel("x-axis")
plt.ylabel("y-axis")
plt.legend()
plt.show()

Result
X: 10.0
Y: 0.0
Optimum solution: 30.0
```




```
    #Number 8
# import libraries
from pulp import LpProblem,LpVariable,LpMaximize
import numpy as np
import matplotlib.pyplot as plt

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

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

# define the objective
lp += 5*x + 3*y, "objective"

# define the contraints
lp += 2*x + 3*y <=60, "Labour"
```

```
lp += 4*x + 2*y <= 80, "Raw_Materials"

# solve
lp.solve()

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

# the graph

# x array
x = np.linspace(0,50,2000)

# covert the constraints to ineq
y1 = (60 - 2*x)/3
y2 = (80 - 4*x)/2

# Ensure the constraints are non negative
y1 = np.maximum(y1,0)
y2 = np.maximum(y2,0)

#plot constraints
plt.plot(x,y1,label="Labour")
plt.plot(x,y2, label="Raw_Materials")

# feasible region
y3  = np.minimum(y1,y2) #feasible region is where both constraints are satisified
plt.fill_between(x,y3, color="yellow", alpha=0.3, label="feasible region")

# limits
plt.xlim(0,25)
plt.ylim(0,45)
plt.title("A graph")
plt.xlabel("x-axis")
plt.ylabel("y-axis")
plt.legend()
plt.grid()

plt.show()

Result
X: 15.0
Y: 10.0
```

```
Minimum solution: 105.0
```