```
%%capture
!pip install -q pyomo
!apt-get install -y -qq glpk-utils
```

Exercise 1

```
from pyomo.environ import *

# Create a model
model = ConcreteModel()

# Define variables
# Added explicit upper bounds based on problem constraints
model.x1 = Var(domain=NonNegativeReals)
model.x2 = Var(domain=NonNegativeReals)

# Objective function
model.Profit = Objective(expr=2*model.x1 + model.x2, sense=maximize)


# Constraints
model.c1 = Constraint(expr=model.x2 <= 10)
model.c2 = Constraint(expr=2*model.x1 + 5*model.x2 <= 60)
model.c3 = Constraint(expr=model.x1 + model.x2 <= 18)
model.c4 = Constraint(expr=3*model.x1 + model.x2 <= 44)

# Solve the model (requires GLPK installed)
# Explicitly specify the executable path for glpsol to ensure Pyomo finds it
results = SolverFactory('glpk', executable='/usr/bin/glpsol').solve(model)
results.write()
model.pprint()

#display results
print("x1 =", model.x1.value)
print("x2 =", model.x2.value)
print("Maximum Z =", model.Profit())
```

```
Problem:
- Name: unknown
  Lower bound: 31.0
  Upper bound: 31.0
  Number of objectives: 1
  Number of constraints: 4
  Number of variables: 2
  Number of nonzeros: 7
  Sense: maximize
# ----------------------------------------------------------
#   Solver Information
# ----------------------------------------------------------
Solver:
- Status: ok
  Termination condition: optimal
  Statistics:
    Branch and bound:
      Number of bounded subproblems: 0
      Number of created subproblems: 0
  Error rc: 0
  Time: 0.004904031753540039
# ----------------------------------------------------------
#   Solution Information
# ----------------------------------------------------------
Solution:
- number of solutions: 0
  number of solutions displayed: 0
2 Var Declarations
    x1 : Size=1, Index=None
        Key  : Lower : Value : Upper : Fixed : Stale : Domain
        None :     0 :  13.0 :  None : False : False : NonNegativeReals
    x2 : Size=1, Index=None





        None :   True : maximize : 2*x1 + x2
4 Constraint Declarations
    c1 : Size=1, Index=None, Active=True
        Key  : Lower : Body : Upper : Active
        None :  -Inf :   x2 :  10.0 :   True
    c2 : Size=1, Index=None, Active=True
        Key  : Lower : Body        : Upper : Active
        None :  -Inf : 2*x1 + 5*x2 :  60.0 :   True
    c3 : Size=1, Index=None, Active=True
        Key  : Lower : Body    : Upper : Active
        None :  -Inf : x1 + x2 :  18.0 :   True
    c4 : Size=1, Index=None, Active=True
        Key  : Lower : Body      : Upper : Active
        None :  -Inf : 3*x1 + x2 :  44.0 :   True

7 Declarations: x1 x2 Profit c1 c2 c3 c4
x1 = 13.0
x2 = 5.0
Maximum Z = 31.0
```

Exercise 2

```
from pyomo.environ import *

# Create model
model = ConcreteModel()

# Define variables: proportions of each alloy (5 variables)
model.x1 = Var(domain=NonNegativeReals)
model.x2 = Var(domain=NonNegativeReals)
model.x3 = Var(domain=NonNegativeReals)
model.x4 = Var(domain=NonNegativeReals)
model.x5 = Var(domain=NonNegativeReals)

# Cost per pound for each alloy
costs = [22, 20, 25, 24, 27]

# Objective: minimize total cost
model.Cost = Objective(expr=
    costs[0]*model.x1 + costs[1]*model.x2 + costs[2]*model.x3 + costs[3]*model.x4 + costs[4]*model.x5,
    sense=minimize)

# Constraints for tin percentage (target 40%)
model.Tin = Constraint(
    expr= 60*model.x1 + 25*model.x2 + 45*model.x3 + 20*model.x4 + 50*model.x5 == 40*(model.x1 + model.x2 + model.x3 + model.x4
)

# Constraints for zinc percentage (target 35%)
model.Zinc = Constraint(
    expr= 10*model.x1 + 15*model.x2 + 45*model.x3 + 50*model.x4 + 40*model.x5 == 35*(model.x1 + model.x2 + model.x3 + model.x4
)

# Constraints for lead percentage (target 25%)
model.Lead = Constraint(
    expr= 30*model.x1 + 60*model.x2 + 10*model.x3 + 30*model.x4 + 10*model.x5 == 25*(model.x1 + model.x2 + model.x3 + model.x4
)

# Sum of proportions must be 1 (to fully define the alloy blend)
model.Total = Constraint(expr=model.x1 + model.x2 + model.x3 + model.x4 + model.x5 == 1)

# Solve with GLPK in Colab
results = SolverFactory('glpk', executable='/usr/bin/glpsol').solve(model)
results.write()
model.pprint()

# Display results
print("Proportions of alloys:")
print(f"x1 = {model.x1.value:.4f}")
print(f"x2 = {model.x2.value:.4f}")
print(f"x3 = {model.x3.value:.4f}")
print(f"x4 = {model.x4.value:.4f}")
print(f"x5 = {model.x5.value:.4f}")
print(f"Minimum cost per pound = ${model.Cost():.2f}")
```

```
# ==========================================================
# = Solver Results                                         =
# ==========================================================
# ----------------------------------------------------------
#   Problem Information
# ----------------------------------------------------------
Problem:
- Name: unknown
  Lower bound: 23.4565217391304
  Upper bound: 23.4565217391304
  Number of objectives: 1
  Number of constraints: 4
  Number of variables: 5
  Number of nonzeros: 20
  Sense: minimize
# ----------------------------------------------------------
#   Solver Information
# ----------------------------------------------------------
Solver:
- Status: ok
  Termination condition: optimal
  Statistics:
    Branch and bound:
      Number of bounded subproblems: 0
      Number of created subproblems: 0
  Error rc: 0
  Time: 0.005146980285644531
# ----------------------------------------------------------
#   Solution Information
# ----------------------------------------------------------
Solution:
- number of solutions: 0
  number of solutions displayed: 0
5 Var Declarations
    x1 : Size=1, Index=None
        Key  : Lower : Value               : Upper : Fixed : Stale : Domain
        None :     0 : 0.0434782608695652 :  None : False : False : NonNegativeReals
    x2 : Size=1, Index=None
        Key  : Lower : Value              : Upper : Fixed : Stale : Domain
        None :     0 : 0.282608695652174 :  None : False : False : NonNegativeReals
    x3 : Size=1, Index=None
        Key  : Lower : Value              : Upper : Fixed : Stale : Domain
        None :     0 : 0.673913043478261 :  None : False : False : NonNegativeReals
    x4 : Size=1, Index=None
        Key  : Lower : Value : Upper : Fixed : Stale : Domain
        None :     0 :   0.0 :  None : False : False : NonNegativeReals
    x5 : Size=1, Index=None
        Key  : Lower : Value : Upper : Fixed : Stale : Domain
        None :     0 :   0.0 :  None : False : False : NonNegativeReals

1 Objective Declarations
    Cost : Size=1, Index=None, Active=True
        Key  : Active : Sense    : Expression
        None :   True : minimize : 22*x1 + 20*x2 + 25*x3 + 24*x4 + 27*x5

4 Constraint Declarations
    Lead : Size=1, Index=None, Active=True
```

Exercise 3

```
from pyomo.environ import *

# Create the model
model = ConcreteModel()

# Decision variables
model.x1 = Var(domain=NonNegativeReals)
model.x2 = Var(domain=NonNegativeReals)

# Objective function: maximize profit
model.Profit = Objective(expr=32*model.x1 + 24*model.x2, sense=maximize)

# Constraints
model.Material = Constraint(expr=3*model.x1 + 2*model.x2 <= 5000)
model.CollegiateSales = Constraint(expr=model.x1 <= 1000)
model.MiniSales = Constraint(expr=model.x2 <= 1200)
model.Labor = Constraint(expr=45*model.x1 + 40*model.x2 <= 35*40*60)

# Solve the model using GLPK solver on Colab
results = SolverFactory('glpk', executable='/usr/bin/glpsol').solve(model)
results.write()
model.pprint()
```

```
# Print solution
print(f"Optimal production quantities:")
print(f"Collegiate backpacks (x1): {model.x1.value:.2f}")
print(f"Mini backpacks (x2): {model.x2.value:.2f}")
print(f"Maximum total profit: ${model.Profit():.2f}")
```

```
# ==========================================================
# = Solver Results                                         =
# ==========================================================
# ----------------------------------------------------------
#   Problem Information
# ----------------------------------------------------------
Problem:
- Name: unknown
  Lower bound: 55400.0
  Upper bound: 55400.0
  Number of objectives: 1
  Number of constraints: 4
  Number of variables: 2
  Number of nonzeros: 6
  Sense: maximize
# ----------------------------------------------------------
#   Solver Information
# ----------------------------------------------------------
Solver:
- Status: ok
  Termination condition: optimal
  Statistics:
    Branch and bound:
      Number of bounded subproblems: 0
      Number of created subproblems: 0
  Error rc: 0
  Time: 0.004709959030151367
# ----------------------------------------------------------
#   Solution Information
# ----------------------------------------------------------
Solution:
- number of solutions: 0
  number of solutions displayed: 0
2 Var Declarations
    x1 : Size=1, Index=None
        Key  : Lower : Value  : Upper : Fixed : Stale : Domain
        None :     0 : 1000.0 :  None : False : False : NonNegativeReals
    x2 : Size=1, Index=None
        Key  : Lower : Value : Upper : Fixed : Stale : Domain
        None :     0 : 975.0 :  None : False : False : NonNegativeReals

1 Objective Declarations
    Profit : Size=1, Index=None, Active=True
        Key  : Active : Sense    : Expression
        None :   True : maximize : 32*x1 + 24*x2

4 Constraint Declarations
    CollegiateSales : Size=1, Index=None, Active=True
        Key  : Lower : Body : Upper  : Active
        None :  -Inf :   x1 : 1000.0 :   True
    Labor : Size=1, Index=None, Active=True
        Key  : Lower : Body          : Upper   : Active
        None :  -Inf : 45*x1 + 40*x2 : 84000.0 :   True
    Material : Size=1, Index=None, Active=True
        Key  : Lower : Body        : Upper  : Active
        None :  -Inf : 3*x1 + 2*x2 : 5000.0 :   True
    MiniSales : Size=1, Index=None, Active=True
        Key  : Lower : Body : Upper  : Active
```

Excercise 4

```
from pyomo.environ import *

# Create model
model = ConcreteModel()

# Define variables
model.xA = Var(domain=NonNegativeReals)
model.xB = Var(domain=NonNegativeReals)

# Objective: minimize cost
model.Cost = Objective(expr=1.20*model.xA + 1.70*model.xB, sense=minimize)

# Constraints
model.MinFuel = Constraint(expr=model.xA + model.xB >= 3300)
model.MinFuel = Constraint(expr=model.xA + model.xB >= 3300)
model.MaxFuel = Constraint(expr=model.xA + model.xB <= 4300)
model.AvailA = Constraint(expr=model.xA <= 2200)
model.AvailB = Constraint(expr=model.xB <= 3600)
model.Octane = Constraint(expr=10*model.xA - 12*model.xB >= 0)

# Solve
results = SolverFactory('glpk', executable='/usr/bin/glpsol').solve(model)
results.write()
model.pprint()

# Print solution
print(f"Optimal fuel amounts:")
print(f"Fuel A (xA): {model.xA.value:.2f} gallons")
print(f"Fuel B (xB): {model.xB.value:.2f} gallons")
print(f"Minimum total cost: ${model.Cost():.2f}")
```

```
  Number of objectives: 1
  Number of constraints: 5
  Number of variables: 2
  Number of nonzeros: 8
  Sense: minimize
# ----------------------------------------------------------
#   Solver Information
# ----------------------------------------------------------
Solver:
- Status: ok
  Termination condition: optimal
  Statistics:
    Branch and bound:
      Number of bounded subproblems: 0
      Number of created subproblems: 0
  Error rc: 0
  Time: 0.0033943653106689453
# ----------------------------------------------------------
#   Solution Information
# ----------------------------------------------------------
Solution:
- number of solutions: 0
  number of solutions displayed: 0
2 Var Declarations
    xA : Size=1, Index=None
        Key  : Lower : Value  : Upper : Fixed : Stale : Domain
        None :     0 : 2200.0 :  None : False : False : NonNegativeReals
    xB : Size=1, Index=None
        Key  : Lower : Value  : Upper : Fixed : Stale : Domain
        None :     0 : 1100.0 :  None : False : False : NonNegativeReals

1 Objective Declarations
    Cost : Size=1, Index=None, Active=True
        Key  : Active : Sense    : Expression
        None :   True : minimize : 1.2*xA + 1.7*xB

5 Constraint Declarations
    AvailA : Size=1, Index=None, Active=True
        Key  : Lower : Body : Upper  : Active
        None :  -Inf :   xA : 2200.0 :   True
    AvailB : Size=1, Index=None, Active=True
        Key  : Lower : Body : Upper  : Active
        None :  -Inf :   xB : 3600.0 :   True
    MaxFuel : Size=1, Index=None, Active=True
        Key  : Lower : Body    : Upper  : Active
        None :  -Inf : xA + xB : 4300.0 :   True
    MinFuel : Size=1, Index=None, Active=True
        Key  : Lower  : Body    : Upper : Active
        None : 3300.0 : xA + xB :  +Inf :   True
    Octane : Size=1, Index=None, Active=True
        Key  : Lower : Body          : Upper : Active
        None :   0.0 : 10*xA - 12*xB :  +Inf :   True

8 Declarations: xA xB Cost MinFuel MaxFuel AvailA AvailB Octane
Optimal fuel amounts:
Fuel A (xA): 2200.00 gallons
Fuel B (xB): 1100.00 gallons
Minimum total cost: $4510.00
```