Basics of COPTPY Solver

AIE1901 - AI Exploration - LLM for Optimization

Jie Wang, 2025/11/04

Key Ingredients of an Optimization Problem

1. **Decision Variables**: The choices we can make
2. **Objective**: The thing we want to minimize or maximize
3. **Constraints**: The rules or limitations we have to follow

Meet the Solver: COPT

- **What is COPT?** Cardinal Optimizer (COPT) is a high-performance mathematical solver developed by Cardinal Operations
- **What does it do?** It takes your problem and uses advanced algorithms to find the provably best solution
- **COPTPY**: A python package that enables us to talk with COPT solver using Python.

Our Coding Environment: Google Colab

- **Action:** Go to colab.research.google.com
- Step 1: Create a new notebook
- Step 2: Install COPTPY

```
[1]
8s
!pip install coptpy

Collecting coptpy
  Downloading coptpy-8.0.1-cp312-cp312-manylinux2014_x86_64.whl.metadata (292 bytes)
Downloading coptpy-8.0.1-cp312-cp312-manylinux2014_x86_64.whl (22.2 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 22.2/22.2 MB 39.5 MB/s eta 0:00:00
Installing collected packages: coptpy
Successfully installed coptpy-8.0.1
```

The "Hello World" of Optimization

Problem Statement:

$$
\begin{aligned}
\text{Maximize}_{x,y} \quad & x + y \\
\text{Subject to} \quad & x + 2y \leq 10 \\
& 2x + y \leq 10 \\
& x \geq 0,\ y \geq 0
\end{aligned}
$$

Step 1: Import and Create Environment

Problem Statement:

$$
\begin{aligned}
\text{Maximize}_{x,y} \quad & x + y \\
\text{Subject to} \quad & x + 2y \leq 10 \\
& 2x + y \leq 10 \\
& x \geq 0,\ y \geq 0
\end{aligned}
$$

```python
# import the coptpy package
import coptpy as cp
from coptpy import COPT
# Create COPT environment
env = cp.Envr()

# Create COPT model
model = env.createModel("Hello_World")

```

```
Cardinal Optimizer v8.0.1. Build date Oct 22 2025
Copyright Cardinal Operations 2025. All Rights Reserved
```

- Envr() is our connection to the COPT Solver
- createModel() creates a new “container” for our problem

Step 2: Add Variables

Problem Statement:

$$
\begin{aligned}
\text{Maximize}_{x,y} \quad & x + y \\
\text{Subject to} \quad & x + 2y \leq 10 \\
& 2x + y \leq 10 \\
& x \geq 0,\ y \geq 0
\end{aligned}
$$

```
# Create variables. They have a name and a lower bound (lb)
# "x" and "y" are our decision variables
x = model.addVar(lb=0, name="x")
y = model.addVar(lb=0, name="y")
```

- Create variables. They have a name and a lower bound (lb)
- These variables can be any number from 0 to infinity

Step 3: Set the Objective

Problem Statement:

$$
\begin{aligned}
\text{Maximize}_{x,y} \quad & x + y \\
\text{Subject to} \quad & x + 2y \leq 10 \\
& 2x + y \leq 10 \\
& x \geq 0,\ y \geq 0
\end{aligned}
$$

```python
# set the objective: Maximize (x+y)
model.setObjective(x+y, sense=COPT.MAXIMIZE)
```

- Our goal is to maximize the objective $x + y$
- COPT.MAXIMIZE tells the solver we want the largest possible value

Step 4: Add Constraints

Problem Statement:

$$
\begin{aligned}
\text{Maximize}_{x,y} \quad & x + y \\
\text{Subject to} \quad & x + 2y \leq 10 \\
& 2x + y \leq 10 \\
& x \geq 0,\ y \geq 0
\end{aligned}
$$

[6] 0s

```python
# Add the constraints
model.addConstr(x + 2*y <= 10, name = "c1") # constraint 1
model.addConstr(2*x + y <= 10, name = "c2") # constriant 2
```

```
<coptpy.Constraint: c2>
```

- Add two linear constraints
- Give each constraint a name for clarity

Step 5: Solve and Get Results

Problem Statement:

$$
\begin{aligned}
\text{Maximize}_{x,y} \quad & x + y \\
\text{Subject to} \quad & x + 2y \leq 10 \\
& 2x + y \leq 10 \\
& x \geq 0,\ y \geq 0
\end{aligned}
$$

```python
# solve the model
model.solve()

# check if the solution is optimal
if model.status == COPT.OPTIMAL:
    print("Optimal solution found")
    print("Objective value: {}".format(model.ObjVal))

    print("Optimal solution: ")
    print("x: {}".format(x.x)) # .x gets the value of the variable
    print("y: {}".format(y.x))
else:
    print("No solution found")
```

- Run the entire code block in Colab.
- Show the output:
    - Optimal Solution:
    - Optimal Value:

\- Visualize the solution

Rethinking the Example

Problem Statement:

$$\text{Maximize}_{x,y} \quad x + y$$
$$\text{Subject to} \quad x + 2y \leq 10$$
$$2x + y \leq 10$$
$$x \geq 0,\ y \geq 0$$

- Objective coefficient:

$$c = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

- Constraint coefficient:

$$A = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} 10 \\ 10 \end{bmatrix}$$

- Decision variable:

$$\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$$

What is a Matrix?

Let $A = (a_{ij})$ be an $m \times n$ matrix.

A= np. array([[1, 2], [3,4]])

- The jth column of A is denoted by a column vector $\mathbf{a}_j$, i.e.,

A= $\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$

$$\mathbf{a}_j = \begin{bmatrix} a_{1j} \\ a_{2j} \\ \vdots \\ a_{mj} \end{bmatrix}$$

- The ith row of A is denoted by a row vector $\vec{\mathbf{a}}_i$, i.e.,

$$\vec{\mathbf{a}}_i = (a_{i1}, a_{i2}, \ldots, a_{in})$$

- Matrix A can be represented in terms of either its columns and rows:

$$A = [\mathbf{a}_1, \cdots, \mathbf{a}_n] = \begin{bmatrix} \vec{\mathbf{a}}_1 \\ \vec{\mathbf{a}}_2 \\ \vdots \\ \vec{\mathbf{a}}_m \end{bmatrix}$$

Matrix-Vector Multiplication

For an $m \times n$ matrix A with the ith column $\mathbf{a}_i$, and a vector $\mathbf{u} = (u_1, u_2, \ldots, u_n)^\top$, the multiplication of A and $\mathbf{u}$ is defined as

$$A\mathbf{u} = u_1\mathbf{a}_1 + u_2\mathbf{a}_2 + \cdots + u_n\mathbf{a}_n$$

Example

$$\underset{2\times 4}{\begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 5 \end{bmatrix}} \underset{4\times 1}{\begin{bmatrix} 6 \\ -7 \\ 8 \\ -9 \end{bmatrix}} = 6\begin{bmatrix} 1 \\ 2 \end{bmatrix} - 7\begin{bmatrix} 2 \\ 3 \end{bmatrix} + 8\begin{bmatrix} 3 \\ 4 \end{bmatrix} - 9\begin{bmatrix} 4 \\ 5 \end{bmatrix}$$

Inner Product

- Given a vector $\mathbf{a} = (a_1, \ldots, a_n)^\top$ and a vector $\mathbf{b} = (b_1, \ldots, b_n)^\top$, following the rule of matrix-vector product, we have

$$\mathbf{a}^\top \mathbf{b} = a_1 b_1 + a_2 b_2 + \cdots a_n b_n$$

- We call this special vector-vector multiplication the **inner product** (scalar product) of $\mathbf{a}$ and $\mathbf{b}$ (denoted by $\mathbf{a}^\top \mathbf{b}$ or $\langle \mathbf{a}, \mathbf{b} \rangle$)
- Properties: Commutative, bilinear
- Application: Cosine similarity, $\cos\theta = \frac{\mathbf{a}^\top \mathbf{b}}{\|\mathbf{a}\| \|\mathbf{b}\|}$

Row Perspective of Multiplication

The matrix-vector multiplication $A\mathbf{u}$ has a row formula as

$$A\mathbf{u} = \begin{bmatrix} \vec{\mathbf{a}}_1 \mathbf{u} \\ \vec{\mathbf{a}}_2 \mathbf{u} \\ \vdots \\ \vec{\mathbf{a}}_m \mathbf{u} \end{bmatrix}$$

- Consider $A = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 5 \end{bmatrix}$ and $\mathbf{u} = \begin{bmatrix} 6 & -7 & 8 & -9 \end{bmatrix}^\top$.
- We calculate

$$\vec{\mathbf{a}}_1 \mathbf{u} = 6 \cdot 1 - 7 \cdot 2 + 8 \cdot 3 - 9 \cdot 4 = -20$$

$$\vec{\mathbf{a}}_2 \mathbf{u} = 6 \cdot 2 - 7 \cdot 3 + 8 \cdot 4 - 9 \cdot 5 = -22$$

- We see that $A\mathbf{u} = \begin{bmatrix} -20 & -22 \end{bmatrix}^\top$

Linear Systems as Matrix Equations

Write the following linear systems into compact matrix form:

$$\begin{cases} 2x_1 + x_2 + x_3 = 5 \\ 4x_1 - 6x_2 = -2 \\ -2x_1 + 7x_2 + 2x_3 = 9 \end{cases} \Rightarrow A\mathbf{x} = \mathbf{b}$$

where

$$A = \begin{bmatrix} 2 & 1 & 1 \\ 4 & -6 & 0 \\ -2 & 7 & 2 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} 5 \\ -2 \\ 9 \end{bmatrix}$$

Rethinking the Example

Problem Statement:

$$\begin{aligned} \text{Maximize}_{x,y} \quad & x + y \\ \text{Subject to} \quad & x + 2y \leq 10 \\ & 2x + y \leq 10 \\ & x \geq 0,\ y \geq 0 \end{aligned}$$

Compact Form:

$$\begin{aligned} \text{Maximize}_{\mathbf{x}} \quad & c^\top \mathbf{x} \\ \text{Subject to} \quad & A\mathbf{x} \leq b \end{aligned}$$

- Objective coefficient:

$$c = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

- Constraint coefficient:

$$A = \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} 10 \\ 10 \end{bmatrix}$$

- Decision variable:

$$\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix}$$

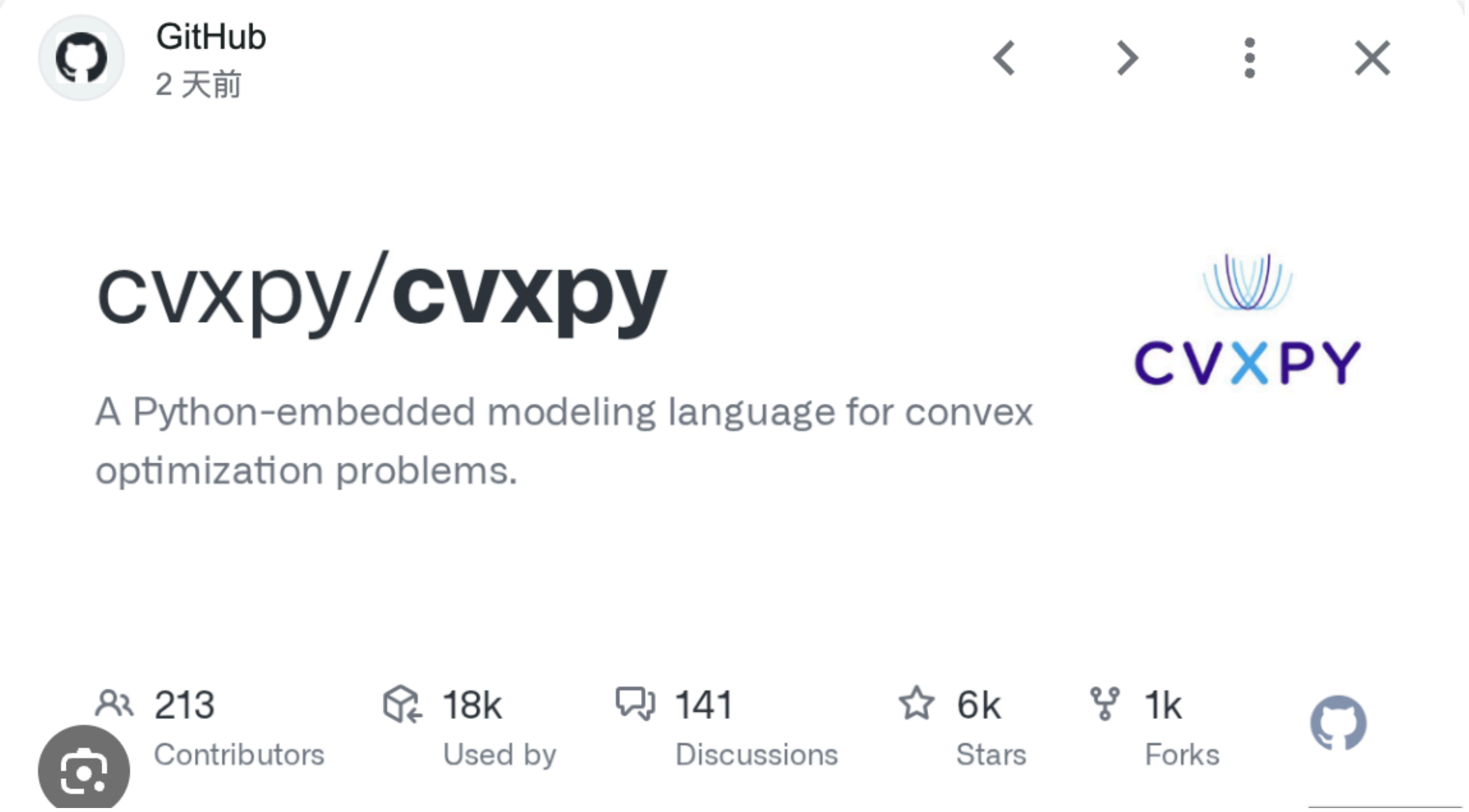
Full Coding Demo

Problem Statement:

$$
\begin{aligned}
\text{Maximize}_{x,y} \quad & x + y \\
\text{Subject to} \quad & x + 2y \leq 10 \\
& 2x + y \leq 10 \\
& x \geq 0,\ y \geq 0
\end{aligned}
$$

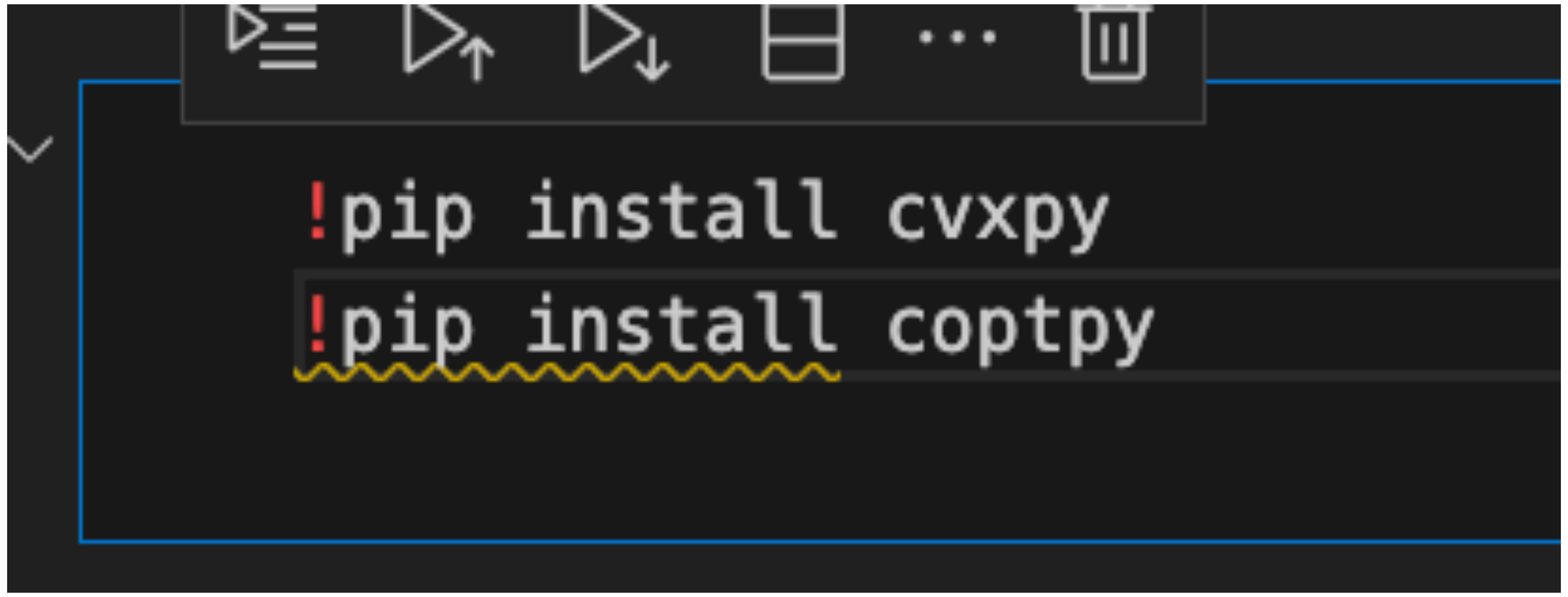
```python
# import the coptpy package
import coptpy as cp
from coptpy import COPT
# Create COPT environment
env = cp.Envr()

# Create COPT model
model = env.createModel("Hello_World")

# Create variables. They have a name and a lower bound (lb)
# "x" and "y" are our decision variables
x = model.addVar(lb=0, name="x")
y = model.addVar(lb=0, name="y")

# Add the constraints
model.addConstr(x + 2*y <= 10, name = "c1") # constraint 1
model.addConstr(2*x + y <= 10, name = "c2") # constriant 2

model.setObjective(x+y, sense=COPT.MAXIMIZE)
# solve the model
model.solve()

# check if the solution is optimal
if model.status == COPT.OPTIMAL:
    print("Optimal solution found")
    print("Objective value: {}".format(model.ObjVal))

    print("Optimal solution: ")
    print("x: {}".format(x.x)) # .x gets the value of the variable
    print("y: {}".format(y.x))
else:
    print("No solution found")
```

Alternative Modeling: CVXPY



- **What is CVXPY?** a Python-embedded modeling language for convex optimization that:
- Provides intuitive mathematical expression syntax
- Supports multiple backend solvers (including COPT, ECOS, SCS, etc.)
- Easy to learn and use, ideal for rapid prototyping and education

Installing CVXPY in Google Colab



Note: CVXPY comes with free solvers by default, but can also interface with commercial solvers like COPT.

CVXPY Modeling Approach - Comparison with COPTPY

COPT-PY	CVXPY
`model.addVar()`	`cp.Variable()`
`model.addConstr()`	Constraints with `<=`, `==`, `>=`
`model.setObjective()`	`cp.Maximize()` or `cp.Minimize()`
`model.solve()`	`problem.solve()`

Step 1: Import and Create Variables

Problem Statement:

$$\begin{aligned} \text{Maximize}_{x,y} \quad & x + y \\ \text{Subject to} \quad & x + 2y \leq 10 \\ & 2x + y \leq 10 \\ & x \geq 0,\ y \geq 0 \end{aligned}$$

```python
import cvxpy as cp

# Create decision variables
x = cp.Variable(nonneg=True, name="x")
y = cp.Variable(nonneg=True, name="y")
```

- nonneg=True automatically sets lower bound to zero
- Variable names are optional but helpful for debugging

Step 2: Define Constraints

Problem Statement:

$$
\begin{aligned}
\text{Maximize}_{x,y} \quad & x + y \\
\text{Subject to} \quad & x + 2y \leq 10 \\
& 2x + y \leq 10 \\
& x \geq 0,\ y \geq 0
\end{aligned}
$$

```python
# Define constraints
constraints = [
    x + 2*y <= 10,      # First constraint
    2*x + y <= 10,      # Second constraint
    x >= 0,             # Non-negativity (redundant but explicit)
    y >= 0              # Non-negativity
]
```

- Note: Constraints are defined using natural mathematical syntax with Python operators.

Step 3: Formulate the Problem

Problem Statement:

$$\begin{aligned} \text{Maximize}_{x,y} \quad & x + y \\ \text{Subject to} \quad & x + 2y \leq 10 \\ & 2x + y \leq 10 \\ & x \geq 0,\ y \geq 0 \end{aligned}$$

```python
# Define objective function
objective = cp.Maximize(x + y)

# Create optimization problem
problem = cp.Problem(objective, constraints)
```

- cp.Maximize() for maximization problems
- cp.Minimize() for minimization problems
- Problem combines objective and constraints into one object

Step 4: Solve and Get Results

Problem Statement:

$$\begin{aligned} \text{Maximize}_{x,y} \quad & x + y \\ \text{Subject to} \quad & x + 2y \leq 10 \\ & 2x + y \leq 10 \\ & x \geq 0,\ y \geq 0 \end{aligned}$$

```python
# Solve the problem
problem.solve(solver=cp.COPT)  # Specify COPT as solver

# Display results
print("Status:", problem.status)
print("Optimal value:", problem.value)
print("Optimal solution:")
print(f"x = {x.value:.2f}")
print(f"y = {y.value:.2f}")
```

Alternative solvers:

- solver=cp.ECOS (free, for convex problems)
- solver=cp.SCS (free, for larger problems)
- solver=cp.COPT (commercial, high-performance)

Full Coding Demo using CVXPY

Problem Statement:

$$\begin{aligned} \text{Maximize}_{x,y} \quad & x + y \\ \text{Subject to} \quad & x + 2y \leq 10 \\ & 2x + y \leq 10 \\ & x \geq 0,\ y \geq 0 \end{aligned}$$

```python
!pip install cvxpy
!pip install coptpy

import cvxpy as cp

# Step 1: Create variables
x = cp.Variable(nonneg=True, name="x")
y = cp.Variable(nonneg=True, name="y")

# Step 2: Define constraints
constraints = [x + 2*y <= 10, 2*x + y <= 10]

# Step 3: Formulate problem
objective = cp.Maximize(x + y)
problem = cp.Problem(objective, constraints)

# Step 4: Solve
problem.solve(solver=cp.COPT)

# Step 5: Output results
print("Status:", problem.status)
print("Optimal value: {:.2f}".format(problem.value))
print("x = {:.2f}, y = {:.2f}".format(x.value, y.value))
```

Matrix Form with CVXPY

Problem Statement:

$$
\begin{aligned}
\text{Maximize}_{x,y} \quad & x + y \\
\text{Subject to} \quad & x + 2y \leq 10 \\
& 2x + y \leq 10 \\
& x \geq 0,\ y \geq 0
\end{aligned}
$$

```python
import cvxpy as cp
import numpy as np

# Problem data in matrix form
c = np.array([1, 1])           # Objective coefficients
A = np.array([[1, 2], [2, 1]]) # Constraint matrix
b = np.array([10, 10])         # RHS values

# Variables
x = cp.Variable(2, nonneg=True)  # Vector of 2 variables

# Problem formulation
objective = cp.Maximize(c.T @ x)  # cᵀx
constraints = [A @ x <= b]        # Ax ≤ b

problem = cp.Problem(objective, constraints)
problem.solve()

print("Solution:", x.value)
```

COPTPY versus CVXPY

COPT-PY (Direct Interface)

- **Pros**: Full control, access to advanced solver features, better for large-scale problems
- **Cons**: More verbose syntax, steeper learning curve

CVXPY (Modeling Language)

- **Pros**: Clean mathematical syntax, easy prototyping, solver-agnostic
- **Cons**: Some overhead for very large problems, limited to convex problems

Recommendation:

- **CVXPY** for education, research, and rapid prototyping
- **COPT-PY** for production systems and performance-critical applications

When to use Which?

Choose CVXPY when:

- Learning optimization concepts
- Rapid prototyping and experimentation
- Working with multiple solvers
- Solving convex optimization problems

Choose COPT-PY when:

- Building production systems
- Need maximum performance
- Require advanced solver features
- Working exclusively with COPT solver

Summary

- CVXPY provides a high-level, intuitive interface for optimization modeling
- Natural mathematical syntax makes code readable and maintainable
- Same optimization problem can be solved with identical results
- Choice depends on your specific needs: education vs production

Next: Try converting more complex problems between both interfaces!