2105623 Optimization of Chemical Processes

Department of Chemical Engineering, Chulalongkorn University

Pyomo quickstart

Week 4 | Semester 1/2026 | Keep this beside the notebook

Weeks 1 to 3 used Excel Solver and OpenSolver. From this week the course also uses **Pyomo**, a modelling library for Python. You do not need to become a Python programmer. You need to be able to *read* a model, *run* it, and *recognise* when the answer it reports should not be trusted — which is the whole point of Week 4.

You do not have to install anything. Use the first route below.

Route 1 — Google Colab. Nothing installed, works on any machine

1. Go to `colab.research.google.com` and sign in with any Google account.
2. **File** → **Upload notebook**, and choose `w04-modeling3-blending-pooling.ipynb` from the course page.
3. Run the **first code cell**. It takes about two minutes.
4. It should print `ipopt :  True` and `HiGHS : True`. You are ready.

That first cell is already in the notebook. If you ever need it in a notebook of your own, it is this:

```
import os, shutil, subprocess, sys

if shutil.which("ipopt") is None and not os.path.exists(
        os.path.expanduser("~/.idaes/bin/ipopt")):
    subprocess.run([sys.executable, "-m", "pip", "install", "-q",
                    "pyomo", "idaes-pse", "highspy"], check=False)
    subprocess.run(["idaes", "get-extensions"], check=False)

os.environ["PATH"] = os.path.expanduser("~/.idaes/bin") + os.pathsep + os.environ["PATH"]

import pyomo.environ as pyo
print("ipopt :", pyo.SolverFactory("ipopt").available(exception_flag=False))
```

Colab gives you a fresh machine each session, so **run that cell every time you open the notebook**. Download your work before you close the tab.

Route 2 — your own machine

Only worth doing if you intend to keep using Pyomo after this course.

```
conda install -c conda-forge ipopt
pip install pyomo highspy
```

Check it, in Python:

```
from pyomo.environ import SolverFactory
print(SolverFactory('ipopt').available())
```

If that prints `False`, do not lose an afternoon to it. Use Route 1 today and e-mail the instructor the exact error message.

The five things you need to read a Pyomo model

```
import pyomo.environ as pyo

m = pyo.ConcreteModel()                              # 1. the model object
m.x = pyo.Var(domain=pyo.NonNegativeReals)           # 2. a decision variable
m.y = pyo.Var(bounds=(0, 100))                       #    with bounds
m.cap = pyo.Constraint(expr=2*m.x + m.y <= 180)      # 3. a constraint
m.profit = pyo.Objective(expr=40*m.x + 30*m.y,       # 4. the objective
                         sense=pyo.maximize)

res = pyo.SolverFactory('appsi_highs').solve(m)      # 5. solve, then read values
print(pyo.value(m.profit), pyo.value(m.x), pyo.value(m.y))
```

Indexed models add a set: `m.T = pyo.Set(initialize=[1,2,3])`, then `m.x = pyo.Var(m.T)` and a `Constraint(m.T, rule=...)` whose rule function returns one inequality per index. That is the entire vocabulary used in the Week 4 notebook and in Homework 2.

Which solver for which model. `appsi_highs` or `glpk` for a linear or mixed-integer program; `ipopt` for a nonlinear one. The notebook chooses for you through `pick_solver()`. Activity 3 is the only part of Week 4 that genuinely needs `ipopt` — because the point of Activity 3 is that a *local* nonlinear solver gives you a different answer depending on where it starts.

A warning you will meet today. `ipopt` reporting `optimal` means it found a point where it can no longer improve locally. On a nonconvex problem that is not the same as the best answer, and today you will see the same model with the same data report `optimal` twice with two different profits. Reading that correctly is the skill being assessed, not the typing.

Homework 2, Problem 6 asks you to audit a Pyomo listing. It is answered by *reading* the code, not by running it — as the instructions say, the reasoning is what is graded. You can score every one of those 20 points without a working solver.