Example:

To illustrate the knapsack problem with a simple example, let's consider a concrete situation:

Context

Imagine you are packing a bag for an expedition and you have several items with different weights and values. Your backpack cannot hold more than 10 kg. Here are the items available:

Item	Weight (kg)	Value
Lamp	3	15
Tent	5	40
Water	2	10
Food	5	30

Objective

Maximize the total value of the items in the backpack without exceeding the maximum capacity of 10 kg.

Modeling

Define binary variables for each item, where xi=1 if item i is chosen, and xi=0 otherwise.

- $x1$: Lamp
- $x2$: Tent
- $x3$: Water
- $x4$: Food

Objective Function

Maximize the total value:

$$V = 15x_1 + 40x_2 + 10x_3 + 30x_4$$

Weight Constraint

The sum of the weights of the selected items must not exceed 10 kg:

$$3x_1 + 5x_2 + 2x_3 + 5x_4 \leq 10$$

Potential Solution

To solve this problem, one can use different methods such as dynamic programming, greedy methods, or even more sophisticated approaches like genetic algorithms. A simple solver might use dynamic programming to find the optimal combination.

Example of Resolution (Simple Approach)

By evaluating combinations that maximize value without exceeding the weight limit, we find:

- Taking the tent ($x2$=1) and water ($x3$=1) gives a total value of 40+10=5040+10=50 and a total weight of 5+2=75+2=7 kg, which is below the 10 kg limit.
- Adding other combinations (like the lamp or food) would exceed the weight limit, so they are excluded.

In this simple example, the best solution is to take the tent and water to maximize the value without exceeding the permitted weight.

This example can be implemented in a Python script to automate finding the optimal solution, using an approach like dynamic programming, for instance.