1. The problem
   1. Knapsack can carry only a certain weight
   2. Clocks have a certain weight and value
   3. Maximize the value of the stolen clocks in the knapsack
2. The solution
   1. Knapsack needs to take items with highest (value/weight) ratio and weight < (weight that knapsack can carry) – ALSO called “Greedy” approach. This approach is not “optimal”
   2. The correct way is to use dynamic programming to “make choices” using memorization
      1. If we don’t use memorization, then we are choosing all subsets and return the most valuable subset, but the time complexity would be exponential.
   3. The choices
      1. Consider the weight constraint
      2. We’ll find the optimum solution if a clock is chosen
      3. For a list of clocks from A-P
         1. Choose A, and then calculate the maximum value achieved from the rest
         2. Don’t Choose A, and then calculate the maximum value achieved from the rest
         3. Find the maximum between these two values and that will be the optimum