Let's recall two fundamental and related concepts: **polynomial-time reducibility** and **NP-completeness**.

- A problem L is **polynomial-time reducible** to a problem R when we can solve any instance of L in polynomial time by querying an oracle of R that correctly solves any instance of R.

  More formally, there exists a Turing Machine M that works in polynomial time and outputs $f(w)$ for any input w, such that $w \in L$ if and only if $f(w) \in R$.

- A problem L is ***NP-complete*** when:
  - It is in NP. *What does this mean again?*
  - Every other problem P in NP can be *reduced to* L in polynomial time.

Today's exercises will highlight **self-reducibility**: the notion that finding a solution can be reduced to deciding if a solution exists.

---

1. ***Warm-up 1.*** Consider the **VertexCover** problem: Given an undirected graph G, a ***vertex cover*** is a set of vertices where every edge touches some vertex in the set. In terms of languages and Turing Machines, phrase the following problems:
   (a) Deciding if a graph has a vertex cover of size k. This is an example of a ***decision problem***.
   (b) Finding a vertex cover of size k in a graph G (if such a cover exists). This is an example of a ***search problem***.
2. **Warm-up 2.** What do you think is more difficult: reducing a decision problem to a search problem or the opposite (reducing a search problem to a decision problem)?
3. **Are you satisfied now?** Given a Boolean formula, the **Sat** problem asks whether there is an assignment to the variables such that the formula is satisfied. The Cook-Levin Theorem tells us that Sat is NP-complete.

   Reduce the problem of finding a valid assignment to the problem of deciding whether a valid assignment exists.
4. **Cover-up.** Show that the problem of finding a vertex cover of size k in a graph G is poly-time reducible to the problem of deciding whether such a vertex cover exists in G.

---

5. By Cook-Levin theorem, the Sat problem is NP-complete. The 3-Sat problem requires that each clause consists of at most three literals. Show that Sat reduces to the 3-Sat problem in polynomial time. *This proves that* 3-Sat *is NP-hard.*
6. ***To think about later:*** Reduce 3-Sat to the VertexCover problem in polynomial time. *This proves that* VertexCover *is also NP-hard.*