Please turn in answers to the following questions next Monday, in class.

1. Rewrite the following expressions in a simpler or more elegant way.

   (a) \(i = i \div 2\)

   (b) \(\text{if True:} \quad \text{print("Hello, world!")}\)

   (c) \(a = b == \text{True}\)

   (d) \(\text{if result:} \quad \text{return False} \quad \text{else:} \quad \text{return True}\)

   (e) \(i = 0 \quad \text{while True:} \quad \text{if i} \geq \text{n:} \quad \text{break} \quad \text{print(i)} \quad i = i + 1\)

2. Let’s think about the relationship between variables \(o_1\) and \(o_2\).

   (a) Explain the difference between \((o_1 \text{ is } o_2)\) and \((o_1 == o_2)\).

   (b) Possible or not: \((o_1 \text{ is } o_2) \text{ and not } (o_1 == o_2)\)? Explain.

   (c) Write Python that sets two non-empty lists \(l_1\) and \(l_2\) in such a way that \((l_1 == l_2) \text{ and not } (l_1 \text{ is } l_2)\).
3. Consider these concepts related to *abstraction*.

   (a) We use functions or *procedures* to support *procedural abstraction*.
       What part of a function is its public interface? What part is its private implementation?

   (b) We depend on *int* types to represent (somehow!) arbitrarily large integer values.
       Explain how the *int* type can be thought of as an abstraction.

   (c) This week we're learning about *modules*. A module is a file that contains a collection of
       functions and values that support a single purpose (*e.g.* the *math* module). Modules typically
       have an *__all__* variable that controls how symbols are imported.
       Explain how a module is an abstraction.