class Tree:
    """A recursive tree data structure."
    ""
    # === Private Attributes ===
    # The item stored at this tree's root, or None if the tree is empty.
    _root: Optional[object]
    # The list of all subtrees of this tree.
    _subtrees: List['Tree']

    # === Representation Invariants ===
    # - If self._root is None then self._subtrees is an empty list. This represents an empty Tree.
    # - self._subtrees may be empty when self._root is not None. This represents a tree consisting of one node.

    # === Methods ===
    def __init__(self, root: object, subtrees: List['Tree']) -> None:
        """Initialize a new Tree with the given root value and subtrees.

        If <root> is None, the tree is empty.
        Precondition: if <root> is None, then <subtrees> is empty.
        """

    def is_empty(self) -> bool:
        """Return True if this tree is empty."
        """

1. Write a Python expression to create an empty tree.

2. Write a Python expression to create a tree containing just the single item 'hello'. Don't touch the private attributes. (You don’t need to.)

3. Write a sequence of Python expressions which create a tree with a root value of 1, whose two children are 10 and 20. You may use temporary variables in your answer, but don’t touch the private attributes!
4. We will define the **length** of a tree to be the number of values in the tree.

What is the **length** of the following tree? ____

5. Draw each subtree of this tree. Underneath each one, write its length.

6. **In English**, explain the relationship between the length of a tree and the length of its subtrees.

7. Implement the method `__len__` for the Tree class, which returns the length of the tree.

   ```python
def __len__(self):
        """Return the length of this tree."""
```