Quiz #7: Binary Search Trees

Read over the declaration of class `BinarySearchTree`, and the header and docstring for method `num_in_range`:

```python
class BinarySearchTree:
    """A Binary Search Tree.
    """

    # --- Private Attributes ---
    # @type _root: object
    # The item stored at the root of the tree, or None if the tree is empty.
    # @type _left: BinarySearchTree | None
    # The left subtree, or None if the tree is empty
    # @type _right: BinarySearchTree | None
    # The right subtree, or None if the tree is empty

    # --- Representation Invariants ---
    # - If _root is None, then so are _left and _right. This represents an empty BST.
    # - If _root is not None, then _left and _right are BSTs.
    # - All items in _left are <= _root, and all items in _right are => _root.

    def num_in_range(self, start, end):
        """Return the number of items between <start> and <end>, inclusive.
        """
        Use the BST property to ensure you don’t make unnecessary recursive calls.

        @type self: BinarySearchTree
        @type start: int
        @type end: int
        @rtype: int
        """
        pass
```

Below is a picture of a **BST**, with several levels. We’ll refer to this tree as the variable `bst`.

```
10
  /  \
 3   32
 /    /  \
2    7 27 81
     /    /  \
   49   99
```
1. What is the output of `bst.num_in_range(5, 50)`?

2. What are the outputs of `bst._left.num_in_range(5, 50)` and `bst._right.num_in_range(5, 50)`?

3. Give an example of numbers \(x\) and \(y\) such that the call `bst.num_in_range(x, y)` would only require one subtree to be checked.

4. Implement `num_in_range` using only the attributes `_root`, `_left`, `_right`, and the method `is_empty`.

```python
def num_in_range(self, start, end):
```