```
'''
    classes.py -- a couple classes, plus a unit test

    Jeff Ondich, 24 October 2006
    Tianna Avery, November 2018, updated for Python 3
'''

class BSTNode:
    def __init__(self, key):
        self.key = key
        self.left = None
        self.right = None

class BST:
    ''' A binary search tree class. '''
    def __init__(self):
        self.root = None

    def traverse_subtree_in_order(self, subtree_root, visit_function):
        ''' visit_function must be callable, and must take one
parameter:
            the node to be visited. '''
        if subtree_root:
            self.traverse_subtree_in_order(subtree_root.left,
visit_function)
            visit_function(subtree_root)
            self.traverse_subtree_in_order(subtree_root.right,
visit_function)

    def traverse(self, visit_function):
        self.traverse_subtree_in_order(self.root, visit_function)

    def add_to_subtree(self, subtree_root, node):
        if node.key < subtree_root.key:
            if not subtree_root.left:
                subtree_root.left = node
            else:
                self.add_to_subtree(subtree_root.left, node)

        elif node.key > subtree_root.key:
            if not subtree_root.right:
                subtree_root.right = node
            else:
                self.add_to_subtree(subtree_root.right, node)

    def add(self, key):
        new_node = BSTNode(key)
        if not self.root:
            self.root = new_node
        else:
```

```
            self.add_to_subtree(self.root, new_node)


if __name__ == '__main__':
    def print_visit(node):
        print(node.key)

    # Build the binary search tree
    bst = BST()
    keys = ('moose', 'ant', 'tiger', 'bat', 'zebu', 'kudu', 'emu',
'auk', 'llama', 'elk')
    for key in keys:
        bst.add(key)

    # Do an in-order traversal of the tree, just printing each node's
key
    # when the node is visited.
    bst.traverse(print_visit)
```