Binary Search Trees

CSC148, INTRODUCTION TO COMPUTER SCIENCE
DAVID LIU
A Binary Search Tree is a “sorted” tree

Every item is >= all items in its left subtree, and <= all items in its right subtree.
def __contains__(self, item) -> bool:
    if self.is_empty():
        return False
    elif item == self._root:
        return True
    elif item < self._root:
        return self._left.__contains__(item)
    else:
        return self._right.__contains__(item)
But not always!

def items(self) -> List:
    if self.is_empty():
        return []
    else:
        return (  
            self._left.items() +
            [self.root] +
            self._right.items()
        )
Representation invariants are key!

If `self._root` is not `None`, then `self._left` and `self._right` are `BinarySearchTree`

If you know that the BST is not empty, you **never** need to check if `self._left` or `self._right` are `None`. 
Deleting from a BST

HINT: PULL OUT THE TREE DELETION WORKSHEET FROM LAST WEEK
def delete(self, item: Any) -> None:
    """Remove *one* occurrence of <item> from this BST.

    Do nothing if <item> is not in the BST.
    """
First, need to find the item

delete this
Introducing Assignment 2!

YAY!
Autocompletion

- hello
- hello google
- hello world
- hello google how are you
- hello fresh
- hello magazine
- hello 123
- hello canada
- hello neighbor
- hello kitty
- hello lyrics
Autocompletion for melodies