Learning Objectives
Students will be able to:

Content:
- Define a tree
- Identify the root and leaf of a tree
- Explain the meaning of a tree’s ___slots___
- Explain how a tree data structure represents a 20 Questions game

Process:
- Write code that adds nodes to a tree
- Write code that iterates through the tree’s values.

Prior Knowledge
- Python concepts from Activities 1-19, Linked Lists, Recursion

Critical Thinking Questions:

FYI: Trees are data structures that simulate a hierarchical tree structure, represented as a set of linked nodes.

1. Examine the diagram below.

   Twenty Questions
   
   Is it alive?
   
   Does it have 8 legs?  Is it food?
   
   Is it an octopus?  Is it sweet?  Does it have 4 legs?
   
   Is it a pretzel?  Is it a table?

   a. How many possible answers does each question have? ________________

   b. On what side of the questions do “yes” responses appear? ________________

   c. If this diagram represents 3 games of 20 Questions, what is the first question asked?

   ________________

   d. What is an example final guess for one of these games of 20 Questions?

   ________________

FYI: A leaf is a node of the tree that has no children. A root is the top node of the tree that points to other nodes (children), but none of these child nodes point to the root.
2. The following code creates the topmost sub-tree of the Twenty Questions tree:

```python
    t2 = Tree('Does it have 8 legs?')
    t3 = Tree('Is it food?')
    mytree = Tree('Is it alive?', t2, t3)
```

a. What does the first parameter of a new `Tree` instance represent?

b. What does the second parameter of a new `Tree` instance represent?

c. Write a line of code to add “octopus” to the correct location in the Tree, where in the sample code would you need to place it?

3. On the left is sample code, on the right is its output when executed:

```
print(mytree.value)  'Is it alive?'
print(mytree.left.value)  'Does it have 8 legs?'
print(mytree.left.left.value)  'Is it an octopus?'
```

a. What would happen if we replaced the first line with:

```python
print(mytree.right.value)'
```

b. What would the following line output (according to the diagram)?

```python
print(mytree.right.right.left.value)
```

c. What would the following line do (according to the diagram)?:

```python
mytree.right.right.right
```

d. How does `Tree.left` differ from what’s stored in `Tree.right` for our 20 Questions game?

e. Write a function, `isLeaf`, that takes in a `Tree` as a parameter and determines if it is a **leaf**. (Question 1d. on the previous page provides some insight into the meaning of a leaf.)
4. Examine the following example code:

def mystery(self):
    if not self.right:
        return self

    return self.right.mystery()

b. What does the following line do?: `if not self.right`

c. Why doesn’t this line need to be included in an if statement?:
   `return self.right.mystery()`

d. For this recursive method, what is the base case / stopping condition?

e. For this recursive method, how is the longer journey broken down/shortened?

f. For 20 Questions, what will this `mystery` method return?

Application Questions: Use the Python Interpreter to check your work

1. Write a recursive method of `Tree` that returns the left most leaf of a `Tree`. In 20 Questions, that would be the “Is it an octopus?” node.

def leftmost(self):

   ```
2. Write the `__str__(self)` method for our Tree class so that it prints the values of all the child nodes of our Tree, not only our Tree’s values:

```python
def __str__(self):
    ______________________________________________________________________________
    ______________________________________________________________________________
    ______________________________________________________________________________
    ______________________________________________________________________________
    ______________________________________________________________________________
    ______________________________________________________________________________
    ______________________________________________________________________________
    ______________________________________________________________________________
    ______________________________________________________________________________
    ______________________________________________________________________________
    ______________________________________________________________________________
    ______________________________________________________________________________
    ______________________________________________________________________________
    ______________________________________________________________________________
    ______________________________________________________________________________
    ______________________________________________________________________________
    ______________________________________________________________________________
    ______________________________________________________________________________
    ______________________________________________________________________________
    ______________________________________________________________________________
    ______________________________________________________________________________
    ______________________________________________________________________________
```

3. Write a recursive method of `Tree` that returns `True` if the given value, `v`, exists as a value within an unsorted `Tree`, `False` if not contained in the `Tree`.

```python
def contains(self, v):
    ______________________________________________________________________________
    ______________________________________________________________________________
    ______________________________________________________________________________
    ______________________________________________________________________________
    ______________________________________________________________________________
    ______________________________________________________________________________
    ______________________________________________________________________________
    ______________________________________________________________________________
    ______________________________________________________________________________
    ______________________________________________________________________________
    ______________________________________________________________________________
    ______________________________________________________________________________
    ______________________________________________________________________________
    ______________________________________________________________________________
    ______________________________________________________________________________
    ______________________________________________________________________________
    ______________________________________________________________________________
    ______________________________________________________________________________
    ______________________________________________________________________________
    ______________________________________________________________________________
    ______________________________________________________________________________
    ______________________________________________________________________________
```
4. See questions 3 & 4 of Homework 9! (Copied/Pasted below, do individually not with partners)

3. Write a recursive method of Tree that computes the number of data values stored in a Tree.

```python
def __len__(self):  # hint: it's at least one!
    """Returns the number of values stored in the tree."""
```

4. Write a recursive method of Tree that computes the sum of the values in the tree:

```python
def sum(self):
    """Returns the sum of all the values found in this tree."""
```