01 Create_a_binary_tree

May 3, 2020

1 Create a binary tree

1.1 Task 01: build a node

- on a piece of paper, draw a tree.
- Define a node, what are the three things you'd expect in a node?
- Define class called `Node`, and define a constructor that takes no arguments, and sets the three instance variables to `None`.
- Note: coding from a blank cell (or blank piece of paper) is good practice for interviews!

```
In [1]: ## Define a node
```

```
In [2]: node0 = Node()
        print(f"""
        value: {node0.value}
        left: {node0.left}
        right: {node0.right}
        """)
```

```
value: None
left: None
right: None
```

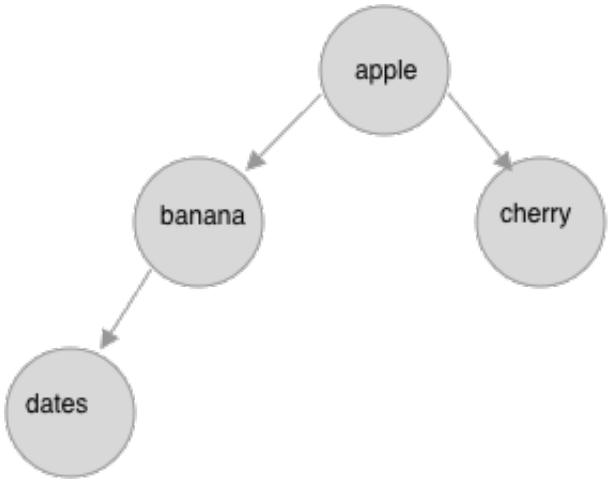
1.2 Task 02: add a constructor that takes the value as a parameter

Copy what you just made, and modify the constructor so that it takes in an optional value, which it assigns as the node's value. Otherwise, it sets the node's value to `None`.

```
In [3]: ## Your code here
        ## Define a node
```

```
In [13]:
```

```
In [4]: ## Check
```



tree image

```
node0 = Node()
print(f"""
value: {node0.value}
left: {node0.left}
right: {node0.right}
""")

node0 = Node("apple")
print(f"""
value: {node0.value}
left: {node0.left}
right: {node0.right}
""")
```

```
value: None
left: None
right: None


value: apple
left: None
right: None
```

1.3 Task 03: add functions to set and get the value of the node

Add functions `get_value` and `set_value`

```
In [ ]: # add set_value and get_value functions

In [ ]:
```

1.4 Task 04: add functions that assign a left child, or right child

Define a function `set_left_child` and a function `set_right_child`. Each function takes in a node that it assigns as the left or right child, respectively. Note that we can assume that this will replace any existing node if it's already assigned as a left or right child.

Also, define `get_left_child` and `get_right_child` functions.

```
In [5]: ## your code here
```

```
In [14]:
```

```
In [6]: ## check

        node0 = Node("apple")
        node1 = Node("banana")
        node2 = Node("orange")
        node0.set_left_child(node1)
        node0.set_right_child(node2)

        print(f"""
        node 0: {node0.value}
        node 0 left child: {node0.left.value}
        node 0 right child: {node0.right.value}
        """)
```

```
node 0: apple
node 0 left child: banana
node 0 right child: orange
```

1.5 Task 05: check if left or right child exists

Define functions `has_left_child`, `has_right_child`, so that they return true if the node has left child, or right child respectively.

```
In [7]: ## your code here
```

```
In [43]:
```

```
In [8]: ## check

        node0 = Node("apple")
        node1 = Node("banana")
        node2 = Node("orange")

        print(f"has left child? {node0.has_left_child()}")
```

```
        print(f"has right child? {node0.has_right_child()}")

        print("adding left and right children")
        node0.set_left_child(node1)
        node0.set_right_child(node2)

        print(f"has left child? {node0.has_left_child()}")
        print(f"has right child? {node0.has_right_child()}")
```

```
has left child? False
has right child? False
adding left and right children
has left child? True
has right child? True
```

1.6 Task 06: Create a binary tree

Create a class called `Tree` that has a "root" instance variable of type `Node`.
Also define a get_root method that returns the root node.

```
In [ ]: # define a Tree class here
```

```
In [1]:
```

1.7 Task 07: setting root node in constructor

Let's modify the `Tree` constructor so that it takes an input that initializes the root node. Choose between one of two options: 1) the constructor takes a `Node` object
2) the constructor takes a value, then creates a new `Node` object using that value.
Which do you think is better?

```
In [ ]: # choose option 1 or 2 (you can try both), and explain why you made this choice
```

```
In [9]:
```

```
In [16]:
```

Discussion Write your thoughts here:

1.8 Next:

Before we learn how to insert values into a tree, we'll first want to learn how to traverse a tree. We'll practice tree traversal next!