* See BT\_Assgn 10 for ref

1. When the \_\_iter\_\_ was not added then we can’t use iterable functions like any, as the class and the objects created from it are not itreable yet.
2. To fix this, you need to make the Tree\_node class iterable. You can do this by defining the \_\_iter\_\_ method, which will allow you to traverse the tree and yield each node's data.

* The code added:-

def \_\_iter\_\_(self):

if self.left:

yield from self.left

yield self.data

if self.right:

yield from self.right

* Imp pts abt abv code

 **def \_\_iter\_\_(self):**

* This defines the \_\_iter\_\_ method, which is a special method in Python used to make an object iterable. This means you can use it in loops or with functions that require an iterable.

 **if self.left:**

* This checks if the current node has a left child. If it does, it proceeds to the next line.

 **yield from self.left**

* This is a generator expression that recursively calls the \_\_iter\_\_ method on the left child node. The yield from statement is used to yield all values from the left subtree.

 **yield self.data**

* This yields the data of the current node. This means that when iterating over the tree, you will get the data stored in this node.

 **if self.right:**

* This checks if the current node has a right child. If it does, it proceeds to the next line.

 **yield from self.right**

* This is a generator expression that recursively calls the \_\_iter\_\_ method on the right child node. The yield from statement is used to yield all values from the right subtree.