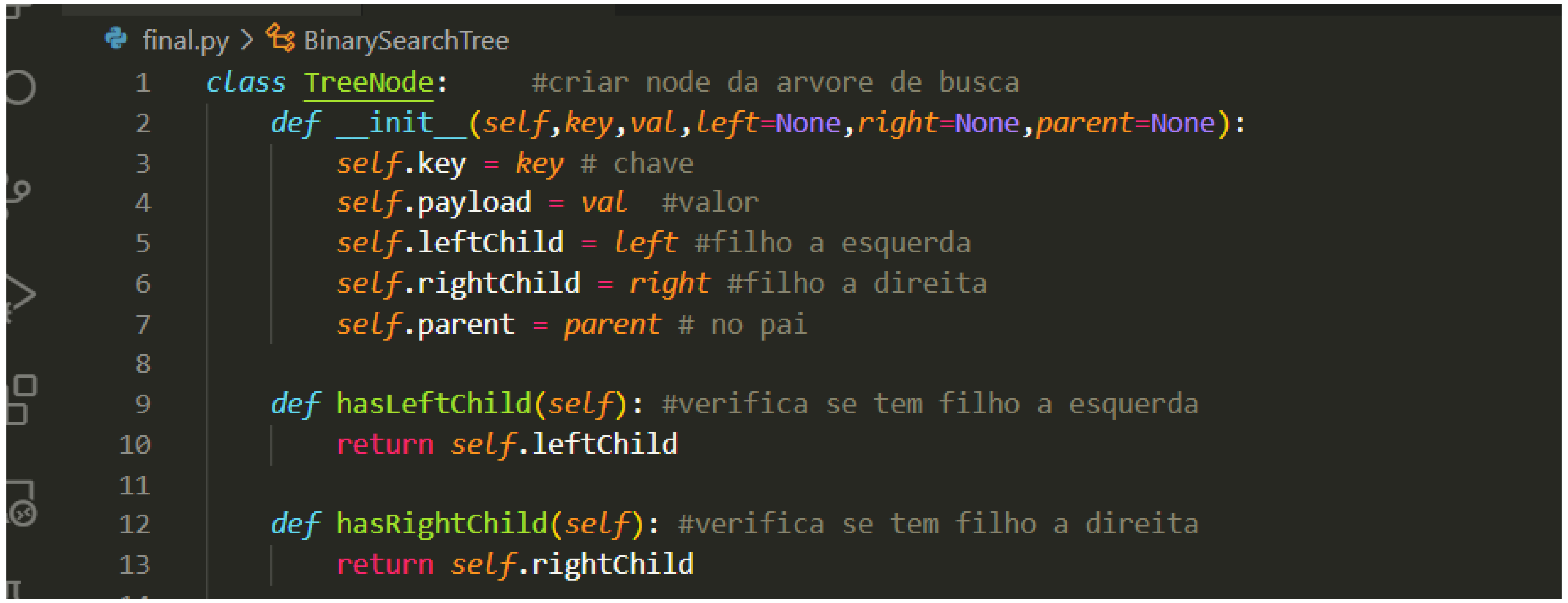
LAB 9

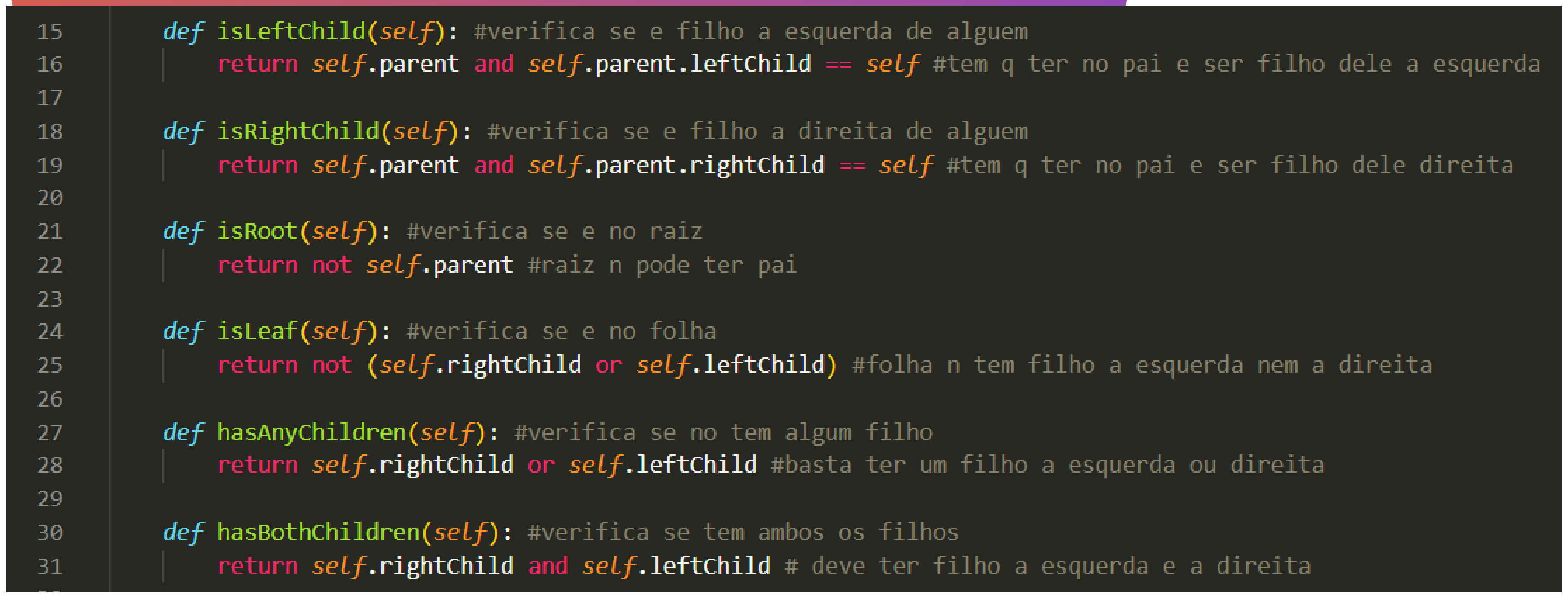
# Data structures and Algorithms

**Luzia Xavier Manuel**

**Roll number : 2021332**



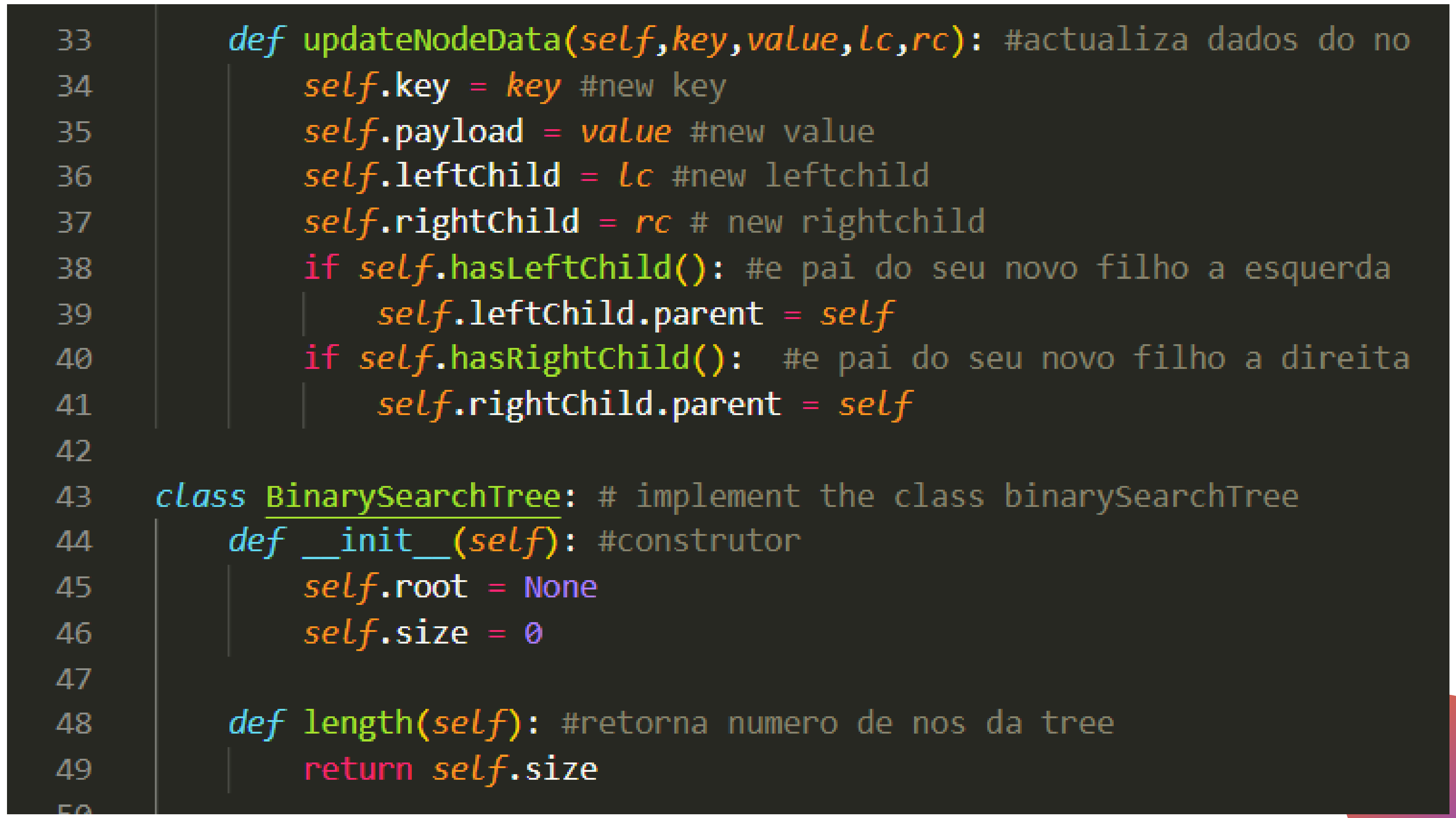
```python
class TreeNode:     #criar node da arvore de busca
    def __init__(self,key,val,left=None,right=None,parent=None):
        self.key = key # chave
        self.payload = val  #valor
        self.leftChild = left #filho a esquerda
        self.rightChild = right #filho a direita
        self.parent = parent # no pai

    def hasLeftChild(self): #verifica se tem filho a esquerda
        return self.leftChild

    def hasRightChild(self): #verifica se tem filho a direita
        return self.rightChild
```



```python
    def isLeftChild(self): #verifica se e filho a esquerda de alguem
        return self.parent and self.parent.leftChild == self #tem q ter no pai e ser filho dele a esquerda

    def isRightChild(self): #verifica se e filho a direita de alguem
        return self.parent and self.parent.rightChild == self #tem q ter no pai e ser filho dele direita

    def isRoot(self): #verifica se e no raiz
        return not self.parent #raiz n pode ter pai

    def isLeaf(self): #verifica se e no folha
        return not (self.rightChild or self.leftChild) #folha n tem filho a esquerda nem a direita

    def hasAnyChildren(self): #verifica se no tem algum filho
        return self.rightChild or self.leftChild #basta ter um filho a esquerda ou direita

    def hasBothChildren(self): #verifica se tem ambos os filhos
        return self.rightChild and self.leftChild # deve ter filho a esquerda e a direita
```

CLASS TO STORE A TREE NODE
I CREATED THE CONSTRUCTOR
AFTER CREATED :
KEY
VALUE STORED IN KEY
SON ON THE LEFT
SON ON THE RIGHT
NODE parent

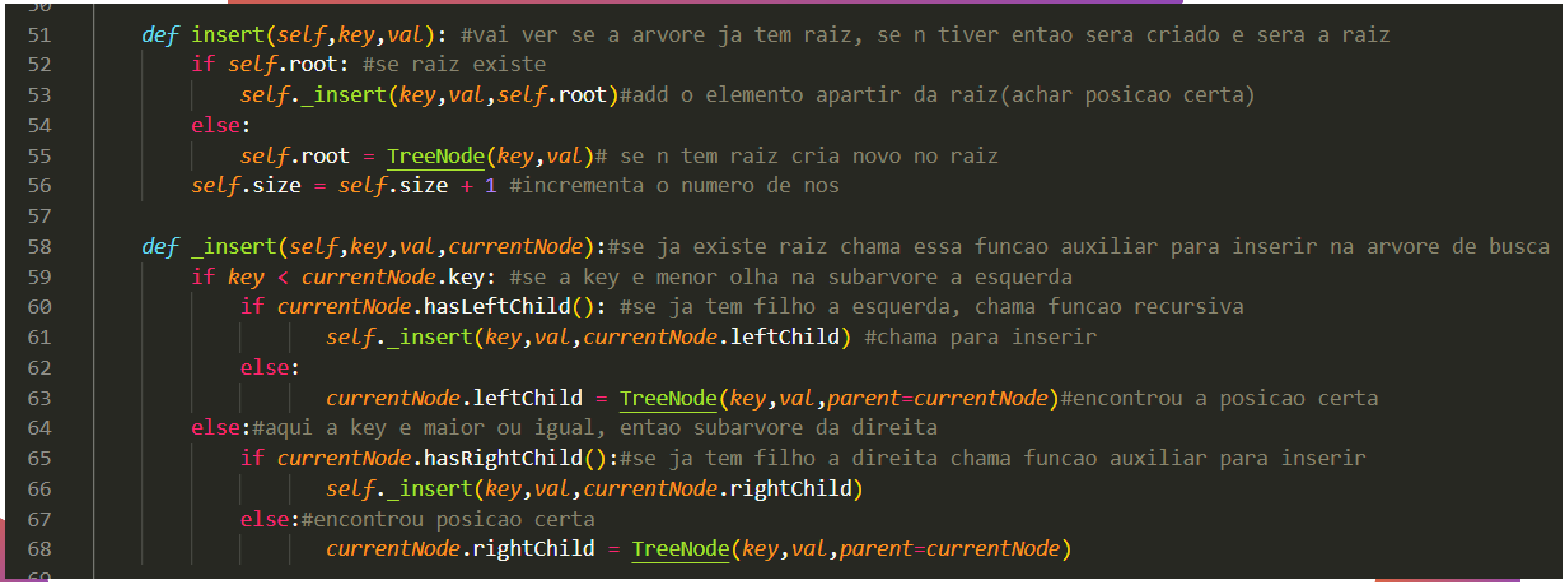
CHECKS IF IT HAS CHILDREN ON THE LEFT AND RIGHT
IF IT RETURNS NONE, THERE IS NO CHILD ON THE LEFT OR RIGHT
CHECKS IF THE NODE IS A CHILD TO SOMEONE'S LEFT AND SOMEONE'S RIGHT
IT HAS TO HAVE ONE IN THE PARENT AND BE A CHILD ON THE LEFT OF THIS PARENT NODE AND ON THE RIGHT
CHECK IF NODE IS ROOT
ROOT CANNOT HAVE PARENT
CHECK IF IT IS LEAF
LEAF HAS NO CHILDREN ON THE LEFT OR RIGHT
CHECK IF YOU HAVE ANY CHILDREN IN THE TREE
JUST HAVE A CHILD ON THE LEFT OR RIGHT
CHECK IF THEY BOTH HAVE CHILDREN
MUST HAVE CHILDREN ON THE LEFT AND THE RIGHT



```python
    def updateNodeData(self,key,value,lc,rc): #actualiza dados do no
        self.key = key #new key
        self.payload = value #new value
        self.leftChild = lc #new leftchild
        self.rightChild = rc # new rightchild
        if self.hasLeftChild(): #e pai do seu novo filho a esquerda
            self.leftChild.parent = self
        if self.hasRightChild():  #e pai do seu novo filho a direita
            self.rightChild.parent = self

class BinarySearchTree: # implement the class binarySearchTree
    def __init__(self): #construtor
        self.root = None
        self.size = 0

    def length(self): #retorna numero de nos da tree
        return self.size
```

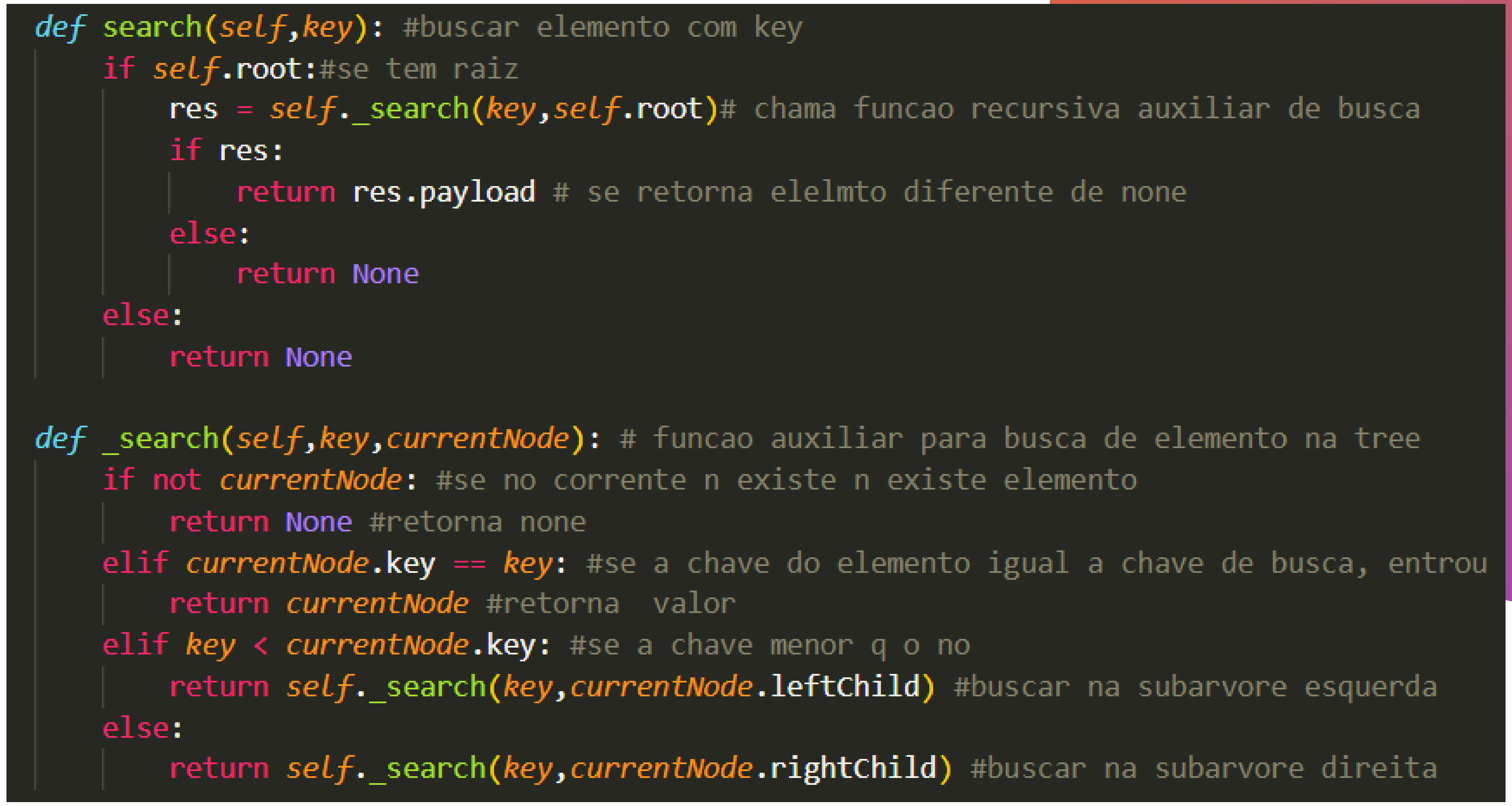
UPDATING THE DATA OF THE NODE
NEW KEY
NEW VALUE
NEW SON ON THE LEFT
NEW SON ON THE RIGHT

IMPLEMENTING THE BINARY SEARCH TREE CLASS
CREATING THE CONSTRUCTOR
CREATE EMPTY ROOT AND RETURN THE NUMBER OF NODES
CHECK IF YOU HAVE ANY CHILDREN IN THE TREE

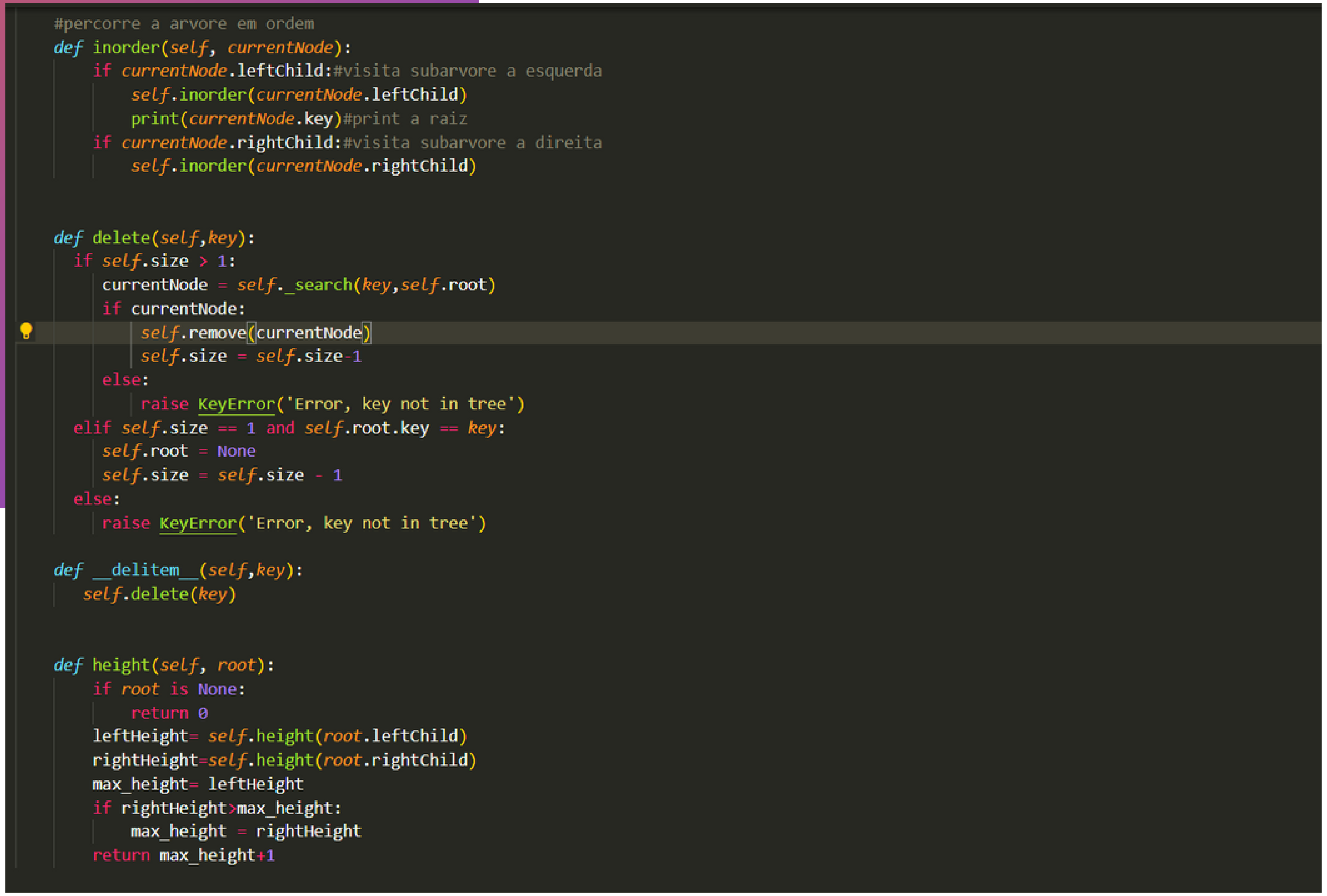


```python
    def insert(self,key,val): #vai ver se a arvore ja tem raiz, se n tiver entao sera criado e sera a raiz
        if self.root: #se raiz existe
            self._insert(key,val,self.root)#add o elemento apartir da raiz(achar posicao certa)
        else:
            self.root = TreeNode(key,val)# se n tem raiz cria novo no raiz
        self.size = self.size + 1 #incrementa o numero de nos

    def _insert(self,key,val,currentNode):#se ja existe raiz chama essa funcao auxiliar para inserir na arvore de busca
        if key < currentNode.key: #se a key e menor olha na subarvore a esquerda
            if currentNode.hasLeftChild(): #se ja tem filho a esquerda, chama funcao recursiva
                    self._insert(key,val,currentNode.leftChild) #chama para inserir
            else:
                    currentNode.leftChild = TreeNode(key,val,parent=currentNode)#encontrou a posicao certa
        else:#aqui a key e maior ou igual, entao subarvore da direita
            if currentNode.hasRightChild():#se ja tem filho a direita chama funcao auxiliar para inserir
                    self._insert(key,val,currentNode.rightChild)
            else:#encontrou posicao certa
                    currentNode.rightChild = TreeNode(key,val,parent=currentNode)
```

THIS METHOD WILL CHECK IF THE TREE ALREADY HAS A ROOT
IF NOT, A NEW NODE WILL BE CREATED AND IT WILL BE THE ROOT OF THE TREE
IF THE ROOT ALREADY EXISTS THEN THE METHOD CALLS THE INSERT FUNCTION TO
LOOK FOR THE RIGHT LOCATION OF THE ELEMENT IN THE TREE, RECURSIVELY



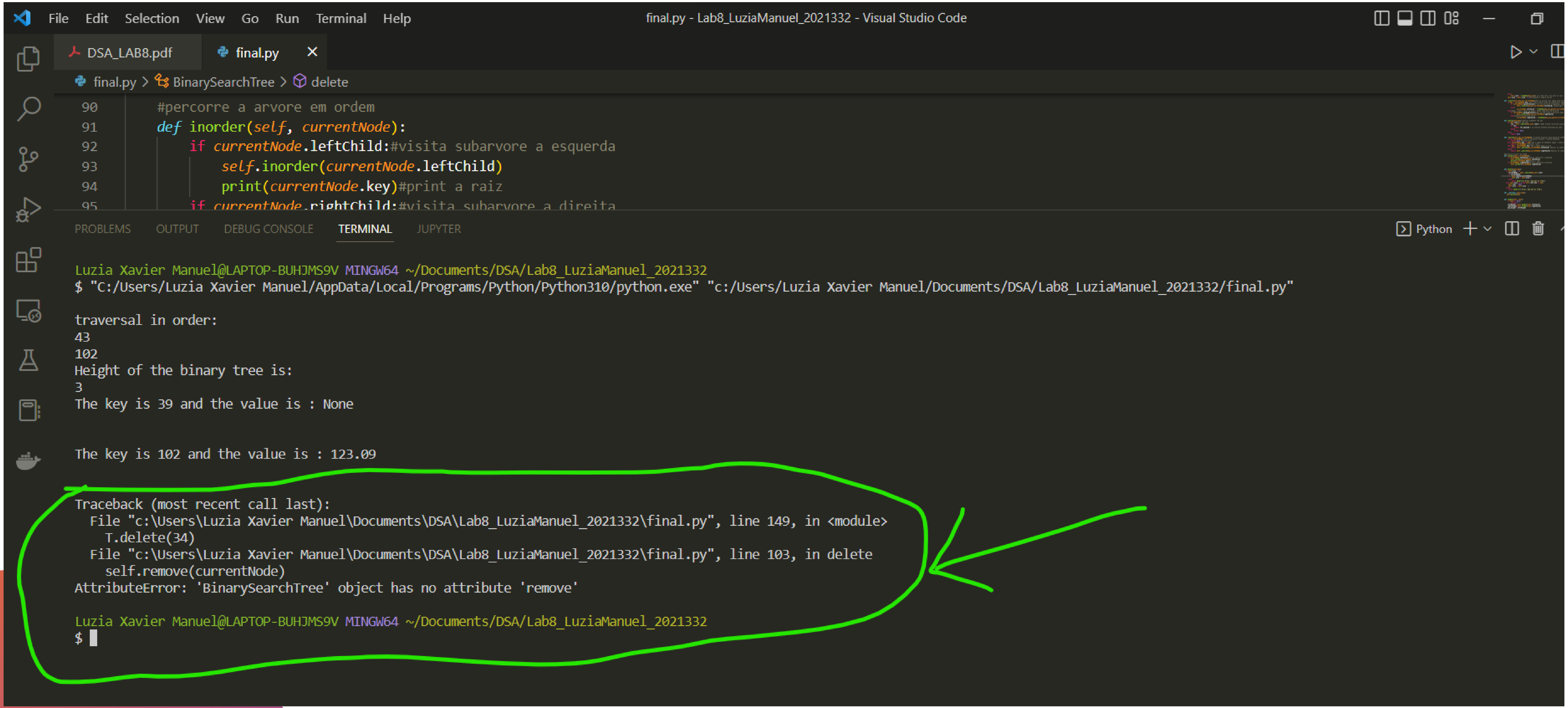
```
def search(self,key): #buscar elemento com key
    if self.root:#se tem raiz
        res = self._search(key,self.root)# chama funcao recursiva auxiliar de busca
        if res:
            return res.payload # se retorna elelmto diferente de none
        else:
            return None
    else:
        return None

def _search(self,key,currentNode): # funcao auxiliar para busca de elemento na tree
    if not currentNode: #se no corrente n existe n existe elemento
        return None #retorna none
    elif currentNode.key == key: #se a chave do elemento igual a chave de busca, entrou
        return currentNode #retorna  valor
    elif key < currentNode.key: #se a chave menor q o no
        return self._search(key,currentNode.leftChild) #buscar na subarvore esquerda
    else:
        return self._search(key,currentNode.rightChild) #buscar na subarvore direita
```



```
#percorre a arvore em ordem
def inorder(self, currentNode):
    if currentNode.leftChild:#visita subarvore a esquerda
        self.inorder(currentNode.leftChild)
        print(currentNode.key)#print a raiz
    if currentNode.rightChild:#visita subarvore a direita
        self.inorder(currentNode.rightChild)


def delete(self,key):
  if self.size > 1:
    currentNode = self._search(key,self.root)
    if currentNode:
        self.remove(currentNode)
        self.size = self.size-1
    else:
        raise KeyError('Error, key not in tree')
  elif self.size == 1 and self.root.key == key:
    self.root = None
    self.size = self.size - 1
  else:
    raise KeyError('Error, key not in tree')

def __delitem__(self,key):
   self.delete(key)


def height(self, root):
    if root is None:
        return 0
    leftHeight= self.height(root.leftChild)
    rightHeight=self.height(root.rightChild)
    max_height= leftHeight
    if rightHeight>max_height:
        max_height = rightHeight
    return max_height+1
```

SEARCH FOR ELEMENT WITH KEY
CHECK IF THE TREE HAS A ROOT
CALL RECURSIVE SEARCH HELPER FUNCTION
IF IT RETURNS AN ELEMENT OTHER THAN NONE,
BECAUSE IT FOUND ITS ELEMENT, OTHERWISE IT RETURNS THE VALUE
IF CURRENT NODE DOES NOT EXIST, THERE IS NO ELEMENT
IF ELEMENT KEY EQUALS SEARCH KEY, FOUND
IF THE KEY IS SMALLER THAN THE NODE, SEARCH IN THE LEFT TREE,
OTHERWISE SEARCH IN THE RIGHT TREE

DELETE function is not working,...



```python
    #percorre a arvore em ordem
    def inorder(self, currentNode):
        if currentNode.leftChild:#visita subarvore a esquerda
            self.inorder(currentNode.leftChild)
            print(currentNode.key)#print a raiz
        if currentNode.rightChild:#visita subarvore a direita
```

```
Luzia Xavier Manuel@LAPTOP-BUHJMS9V MINGW64 ~/Documents/DSA/Lab8_LuziaManuel_2021332
$ "C:/Users/Luzia Xavier Manuel/AppData/Local/Programs/Python/Python310/python.exe" "c:/Users/Luzia Xavier Manuel/Documents/DSA/Lab8_LuziaManuel_2021332/final.py"

traversal in order:
43
102
Height of the binary tree is:
3
The key is 39 and the value is : None


The key is 102 and the value is : 123.09


Traceback (most recent call last):
  File "c:\Users\Luzia Xavier Manuel\Documents\DSA\Lab8_LuziaManuel_2021332\final.py", line 149, in <module>
    T.delete(34)
  File "c:\Users\Luzia Xavier Manuel\Documents\DSA\Lab8_LuziaManuel_2021332\final.py", line 103, in delete
    self.remove(currentNode)
AttributeError: 'BinarySearchTree' object has no attribute 'remove'

Luzia Xavier Manuel@LAPTOP-BUHJMS9V MINGW64 ~/Documents/DSA/Lab8_LuziaManuel_2021332
$ 
```

this message is bacause, delete function is not working