Khoi Duong

Prof. Yang

CE450

12/12/2022

HW#6

GitHub link: <https://github.com/MynameisKoi/CE450/tree/main/HW%236>



Source code: <https://github.com/MynameisKoi/CE450/blob/main/HW%236/1.py>

class Tree:

def \_\_init\_\_(*self*, *label*, *branches*=[]):

*self*.label = *label*

for branch in *branches*:

assert isinstance(branch, Tree)

*self*.branches = list(*branches*)

def \_\_repr\_\_(*self*):

if *self*.branches:

return 'Tree({0}, {1})'.format(*self*.label, repr(*self*.branches))

else:

return 'Tree({0})'.format(repr(*self*.label))

def \_\_str\_\_(*self*):

return '\n'.join(*self*.indented())

def indented(*self*):

lines = []

for b in *self*.branches:

for line in b.indented():

lines.append(' ' + line)

return [str(*self*.label)] + lines

def is\_leaf(*self*):

return not *self*.branches

def has\_itm(*t*, *e*):

# Return True if a Tree t contains the value e

# t: Tree

# e: element

# return: True or False

if *t*.is\_leaf():

return False

# if e is in the Tree t, return True

elif *t*.label == *e*:

return True

# if e is not in the Tree t, return False

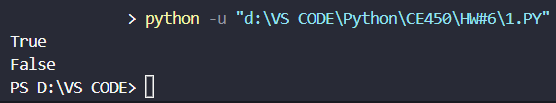
else:

return has\_itm(*t*.branches[0], *e*) or has\_itm(*t*.branches[1], *e*)

print(has\_itm (Tree(11, [Tree(12), Tree(13, [Tree(14),Tree(15)])] ), 11))

print(has\_itm (Tree(11, [Tree(12), Tree(13, [Tree(14),Tree(15)])] ), 16))

Run program & result:





Source code: <https://github.com/MynameisKoi/CE450/blob/main/HW%236/2.py>

class Tree:

def \_\_init\_\_(*self*, *label*, *branches*=[]):

*self*.label = *label*

for branch in *branches*:

assert isinstance(branch, Tree)

*self*.branches = list(*branches*)

def \_\_repr\_\_(*self*):

if *self*.branches:

return 'Tree({0}, {1})'.format(*self*.label, repr(*self*.branches))

else:

return 'Tree({0})'.format(repr(*self*.label))

def \_\_str\_\_(*self*):

return '\n'.join(*self*.indented())

def indented(*self*):

lines = []

for b in *self*.branches:

for line in b.indented():

lines.append(' ' + line)

return [str(*self*.label)] + lines

def is\_leaf(*self*):

return not *self*.branches

def lst(*t*):

# Return a list of the labels in Tree t

if *t*.is\_leaf():

return [*t*.label]

# if t is not empty, return the list of the labels in Tree t

else:

return [*t*.label] + lst(*t*.branches[0]) + lst(*t*.branches[1])

def ave(*t*):

# Return the average of the labels in Tree t

# t: Tree

# return: average

# pass empty (null) values

if *t*.is\_leaf():

return *t*.label

# if t is not empty, return the average of the labels in Tree t

else:

return sum(lst(*t*))/len(lst(*t*))

t = Tree(11, [Tree(12), Tree(13, [Tree(14),Tree(15)])] )

# print(lst(t)) => the list of the tree will be [11, 12, 13, 14, 15]

print(ave(t))

Run program & result:

