class TrieNode(object):

"""

Our trie node implementation. Very basic. but does the job

"""

def \_\_init\_\_(self, char: str):

self.char = char

self.children = []

# Is it the last character of the word.`

self.word\_finished = False

# How many times this character appeared in the addition process

self.counter = 1

def add(root, word):

"""

Adding a word in the trie structure

"""

node = root

for char in word:

found\_in\_child = False

# Search for the character in the children of the present `node`

for child in node.children:

if child.char == char:

# We found it, increase the counter by 1 to keep track that another

# word has it as well

child.counter += 1

# And point the node to the child that contains this char

node = child

found\_in\_child = True

# We did not find it so add a new chlid

if not found\_in\_child:

new\_node = TrieNode(char)

node.children.append(new\_node)

# And then point node to the new child

node = new\_node

# Everything finished. Mark it as the end of a word.

node.word\_finished = True

def find\_prefix(root, prefix):

"""

Check and return

1. If the prefix exsists in any of the words we added so far

2. If yes then how may words actually have the prefix

"""

node = root

# If the root node has no children, then return False.

# Because it means we are trying to search in an empty trie

if not root.children:

return False, 0

for char in prefix:

char\_not\_found = True

# Search through all the children of the present `node`

for child in node.children:

if child.char == char:

# We found the char existing in the child.

char\_not\_found = False

# Assign node as the child containing the char and break

node = child

# Return False anyway when we did not find a char.

if char\_not\_found:

return False, 0

# Well, we are here means we have found the prefix. Return true to indicate that

# And also the counter of the last node. This indicates how many words have this

# prefix

return True, node.counter

if \_\_name\_\_ == "\_\_main\_\_":

root = TrieNode('\*')

add(root, "hackathon")

add(root, 'hack')

add(root, 'harry')

print(find\_prefix(root, 'ha'))

print(find\_prefix(root, 'har'))