<第一次嘗試>

class MyLinkedList:

def \_\_init\_\_(self,data,nextnode):

"""

Initialize your data structure here.

"""

self.val = data

self.next = nextnode

def \_init\_(self):

self.head = None

self.count = 0

def get(self, index):

"""

Get the value of the index-th node in the linked list. If the index is invalid, return -1.

:type index: int

:rtype: int

"""

p = self.head

if self.count == 0 or index > self.count -1 :

return -1

else:

i = 1

while i <=index

p = p.next

i+=1

return p.val

def addAtHead(self, val):

"""

Add a node of value val before the first element of the linked list. After the insertion, the new node will be the first node of the linked list.

:type val: int

:rtype: None

"""

node = self.Node(val,self.head)

self.head = node

self.count += 1

def addAtTail(self, val):

"""

Append a node of value val to the last element of the linked list.

:type val: int

:rtype: None

"""

node = self.Node(val,None)

if self.count == 0:

self.head = node

self.count +=1

else:

p = self.head

while p.next != None:

p = p.next

p.next = node

self.count += 1

def addAtIndex(self, index, val):

"""

Add a node of value val before the index-th node in the linked list. If index equals to the length of linked list, the node will be appended to the end of linked list. If index is greater than the length, the node will not be inserted.

:type index: int

:type val: int

:rtype: None

"""

p = self.head

post = self.head

if index == self.count:

self.addAtTail(val)

elif index > self.count:

return

else:

i=1

while i <= index:

post = p

p = p.next

i += 1

post.next = self.Node(val,p)

self.count += 1

def deleteAtIndex(self, index):

"""

Delete the index-th node in the linked list, if the index is valid.

:type index: int

:rtype: None

"""

p = self.head

if self.count == 0 or index > (self.count - 1) or index < 0:

return

else:

i =1

while i <= index:

post = p

p = p.next

i += 1

post.next = p.next

self.count -= 1

<第二次嘗試>

class MyLinkedList:

class Node:

def \_\_init\_\_(self):

self.val = None

self.next = None

def \_init\_(self):

self.head = None

self.count = 0

def get(self, index):

p = self.head

if self.count == 0 or index > self.count -1:

return -1

else:

i = 1

while i <= index:

p = p.next

i += 1

return p.val

def addAtHead(self, val):

node = self.Node(val,self.head)

self.head = node

self.count += 1

def addAtTail(self, val):

node = self.Node(val,None)

if self.count == 0:

self.head = node

self.count +=1

else:

p = self.head

while p != None:

p = p.next

p.next = node

self.count += 1

def addAtIndex(self, index, val):

p = self.head

post = self.head

if index == self.count:

self.addAtTail(val)

elif index > self.count:

return

else:

i=1

while i <= index:

post = p

p = p.next

i += 1

post.next = self.Node(val,p)

self.count += 1

def deleteAtIndex(self, index):

p = self.head

if self.count == 0 or index > (self.count - 1) or index < 0:

return

else:

i =1

while i <= index:

post = p

p = p.next

i += 1

post.next = p.next

self.count -= 1

<第三次嘗試>

class Node:

def \_\_init\_\_(self,dataval=0):

self.dataval = 0

self.nextval = None

class MyLinkedList:

def \_init\_(self):

self.val = None

self.next = None

def get(self, index):

p = self.val

if p == None :

return -1

else:

i = 1

while i <= index:

p = p.next

i += 1

return p.val

def addAtHead(self, newdata):

NewNode = Node(newdata)

def addAtTail(self, val):

NewNode = Node(val)

if self.val == None:

self.val = NewNode

return

laste = self.val

while(laste.nextval):

laste = laste.nextval

laste.nextval=NewNode

def addAtIndex(self, middle\_node,newdata):

if not middle\_node:

print("The mentioned node is absent")

return

NewNode = Node()

NewNode.nextval = middle\_node.nextval

middle\_node.nextval = NewNode

def deleteAtIndex(self, removekey):

HeadVal = self.head

if not HeadVal:

if (HeadVal.data == Removekey):

self.head = HeadVal.next

HeadVal = None

return

while not HeadVal:

if HeadVal.data == Removekey:

break

prev = HeadVal

HeadVal = HeadVal.next

if (HeadVal == None):

return

prev.next = HeadVal.next

HeadVal = None

#要注意def 是包含在 class裡面的