```
class node:
        def __init__(self,data=None):
                self.data=data
                self.next=None

class linked_list:
        def __init__(self):
                self.head=node()

        # Adds new node containing 'data' to the end of the linked list.
        def append(self,data):
                new_node=node(data)
                cur=self.head
                while cur.next!=None:
                        cur=cur.next
                cur.next=new_node

        # Returns the length (integer) of the linked list.
        def length(self):
                cur=self.head
                total=0
                while cur.next!=None:
                        total+=1
                        cur=cur.next
                return total

        # Prints out the linked list in traditional Python list format.
        def display(self):
                elems=[]
                cur_node=self.head
                while cur_node.next!=None:
                        cur_node=cur_node.next
                        elems.append(cur_node.data)
                print elems

        # Returns the value of the node at 'index'.
        def get(self,index):
                if index>=self.length() or index<0: # added 'index<0' post-video
                        print "ERROR: 'Get' Index out of range!"
                        return None
                cur_idx=0
                cur_node=self.head
                while True:
                        cur_node=cur_node.next
                        if cur_idx==index: return cur_node.data
                        cur_idx+=1

        # Deletes the node at index 'index'.
        def erase(self,index):
                if index>=self.length() or index<0: # added 'index<0' post-video
                        print "ERROR: 'Erase' Index out of range!"
                        return
                cur_idx=0
                cur_node=self.head
                while True:
                        last_node=cur_node
                        cur_node=cur_node.next
                        if cur_idx==index:
                                last_node.next=cur_node.next
                                return
                        cur_idx+=1

        # Allows for bracket operator syntax (i.e. a[0] to return first item).
        def __getitem__(self,index):
                return self.get(index)


        ######################################################
        # Functions added after video tutorial

        # Inserts a new node at index 'index' containing data 'data'.
        # Indices begin at 0. If the provided index is greater than or
        # equal to the length of the linked list the 'data' will be appended.
        def insert(self,index,data):
                if index>=self.length() or index<0:
                        return self.append(data)
                cur_node=self.head
                prior_node=self.head
                cur_idx=0
                while True:
                        cur_node=cur_node.next
                        if cur_idx==index:
                                new_node=node(data)
                                prior_node.next=new_node
                                new_node.next=cur_node
                                return
                        prior_node=cur_node
                        cur_idx+=1

        # Inserts the node 'node' at index 'index'. Indices begin at 0.
        # If the 'index' is greater than or equal to the length of the linked
        # list the 'node' will be appended.
        def insert_node(self,index,node):
                if index<0:
                        print "ERROR: 'Erase' Index cannot be negative!"
                        return
                if index>=self.length(): # append the node
                        cur_node=self.head
                        while cur_node.next!=None:
                                cur_node=cur_node.next
                        cur_node.next=node
                        return
                cur_node=self.head
                prior_node=self.head
                cur_idx=0
                while True:
                        cur_node=cur_node.next
                        if cur_idx==index:
                                prior_node.next=node
                                return
                        prior_node=cur_node
                        cur_idx+=1

        # Sets the data at index 'index' equal to 'data'.
        # Indices begin at 0. If the 'index' is greater than or equal
        # to the length of the linked list a warning will be printed
        # to the user.
        def set(self,index,data):
                if index>=self.length() or index<0:
                        print "ERROR: 'Set' Index out of range!"
                        return
                cur_node=self.head
                 cur_idx=0
                 while True:
                         cur_node=cur_node.next
                         if cur_idx==index:
                                 cur_node.data=data
                                 return
                         cur_idx+=1
```