

```

1
2 class node:
3     def __init__(self,data=None):
4         self.data=data
5         self.next=None
6
7 class linked_list:
8     def __init__(self):
9         self.head=node()
10
11     # Adds new node containing 'data' to the end of the linked list.
12     def append(self,data):
13         new_node=node(data)
14         cur=self.head
15         while cur.next!=None:
16             cur=cur.next
17         cur.next=new_node
18
19     # Returns the length (integer) of the linked list.
20     def length(self):
21         cur=self.head
22         total=0
23         while cur.next!=None:
24             total+=1
25             cur=cur.next
26         return total
27
28     # Prints out the linked list in traditional Python list format.
29     def display(self):
30         elems=[]
31         cur_node=self.head
32         while cur_node.next!=None:
33             cur_node=cur_node.next
34             elems.append(cur_node.data)
35         print elems
36
37     # Returns the value of the node at 'index'.
38     def get(self,index):
39         if index>=self.length() or index<0: # added 'index<0' post-video
40             print "ERROR: 'Get' Index out of range!"
41             return None
42
43         cur_idx=0
44         cur_node=self.head
45         while True:
46             cur_node=cur_node.next
47             if cur_idx==index: return cur_node.data
48             cur_idx+=1
49
50     # Deletes the node at index 'index'.
51     def erase(self,index):
52         if index>=self.length() or index<0: # added 'index<0' post-video
53             print "ERROR: 'Erase' Index out of range!"
54             return
55
56         cur_idx=0
57         cur_node=self.head
58         while True:
59             last_node=cur_node
60             cur_node=cur_node.next
61             if cur_idx==index:
62                 last_node.next=cur_node.next
63                 return
64             cur_idx+=1
65
66     # Allows for bracket operator syntax (i.e. a[0] to return first item).
67     def __getitem__(self,index):
68         return self.get(index)
69
70     #####
71     # Functions added after video tutorial
72
73     # Inserts a new node at index 'index' containing data 'data'.
74     # Indices begin at 0. If the provided index is greater than or
75     # equal to the length of the linked list the 'data' will be appended.
76     def insert(self,index,data):
77         if index>=self.length() or index<0:
78             return self.append(data)
79
80         cur_node=self.head
81         prior_node=self.head
82         cur_idx=0
83         while True:
84             cur_node=cur_node.next

```

```

83         if cur_idx==index:
84             new_node=node(data)
85             prior_node.next=new_node
86             new_node.next=cur_node
87             return
88
89         prior_node=cur_node
90         cur_idx+=1
91
92     # Inserts the node 'node' at index 'index'. Indices begin at 0.
93     # If the 'index' is greater than or equal to the length of the linked
94     # list the 'node' will be appended.
95     def insert_node(self,index,node):
96         if index<0:
97             print "ERROR: 'Erase' Index cannot be negative!"
98             return
99
100         if index>=self.length(): # append the node
101             cur_node=self.head
102             while cur_node.next!=None:
103                 cur_node=cur_node.next
104             cur_node.next=node
105             return
106
107         cur_node=self.head
108         prior_node=self.head
109         cur_idx=0
110         while True:
111             cur_node=cur_node.next
112             if cur_idx==index:
113                 prior_node.next=node
114                 return
115             prior_node=cur_node
116             cur_idx+=1
117
118     # Sets the data at index 'index' equal to 'data'.
119     # Indices begin at 0. If the 'index' is greater than or equal
120     # to the length of the linked list a warning will be printed
121     # to the user.
122     def set(self,index,data):
123         if index>=self.length() or index<0:
124             print "ERROR: 'Set' Index out of range!"
125             return
126
127         cur_node=self.head
128         cur_idx=0
129         while True:
130             cur_node=cur_node.next
131             if cur_idx==index:
132                 cur_node.data=data
133                 return
134             cur_idx+=1

```