Learning Objectives
Students will be able to:

Content:
- Define a wrapper class
- Explain the shortcomings of a solitary Element class

Process:
- Write code that adds and modifies elements of a LinkedList
- Write code to access elements of a LinkedList
- Write code to make an object iterable.

Prior Knowledge
- Python concepts from Activities 1-19, recursion, special methods, Linked Lists - Elements

Folks, this is a brand new activity. If you encounter any issues/typos, please let Iris know!

Critical Thinking Questions:

FYI: Continuing on from Linked Lists – Elements, we’re building a LinkedList data structure which is a series of Elements linked together, one pointing to the next. Review Activity 42 first!

1. Below are the special methods, __init__(..) and __len__(..) for an Element object.

<table>
<thead>
<tr>
<th>In the Element Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 def <strong>init</strong>(self, value, next=None):</td>
</tr>
<tr>
<td>1 self._next = next</td>
</tr>
<tr>
<td>2 self._value = value</td>
</tr>
<tr>
<td>3 def <strong>len</strong>(self):</td>
</tr>
<tr>
<td>4 if self.next is None:</td>
</tr>
<tr>
<td>5 return 1</td>
</tr>
<tr>
<td>6 else:</td>
</tr>
<tr>
<td>7 return 1 + len(self.next)</td>
</tr>
</tbody>
</table>

a. How many parameters does initializing a new Element require? ________________

b. At line 1, what is stored in ee.next? ________________

c. At line 2, what is stored in ee.value? ________________

d. If we added the following code at line 9, what would be returned?: len(ee) ________________

e. Is it possible to write a line of code to create an empty Element list (such that the length would be 0)? How?: ________________
2. The following code uses the LinkedList class in interactive python (indicated by >>>):

```
0 >>> ll = LinkedList()
1 >>> ll._head
2 None
3 >>> ll.append(3)
4 >>> ll._head.value
5 3
6 >>> type(ll._head)
7 <class 'Element'>
```

a. If we replaced line 1 with len(ll), what should be returned? ______________
b. If we replaced line 4 with len(ll), what should be returned? ______________
c. What is stored in ll._head at line 0? Where does that change?
   Line 0: ______________________  ll._head changes: ______________________
d. What type of object is LinkedList._head? ________________________________
e. What might LinkedList._head represent?
_____________________________________________________________________

3. The following code is the __len__(..) method from LinkedList:

```
In the LinkedList Class
0 def __len__(self):
1     if self._head is None:
2         return 0
3     else:
4         return len(self._head)
```

a. Write a line of code that implicitly calls __len__(..) on our LinkedList, ll:

b. If we constructed a new LinkedList, ll = LinkedList(), what would __len__(..) return?

```
c. len(..) is being called on self._head on line 4. What class defines that len(..) method?

d. If we appended a value to ll, as in the previous example, what would this method return?
```

FYI: A wrapper class is any class which wraps/encapsulates the functionality of another class or component. In this case, LinkedList is a wrapper class which encapsulates the container class, Element.
4. Examine the following example method from the LinkedList class:

```python
0 def mystery(self, a):
1     if self._head is None:
2         self._head = Element(a)
3     else:
4         self._head.mystery(a)
```

a. What does the following line 1 do?: `if self._head is None:`

b. What happens `if self._head is None` (line 2)?

c. What type of object is `self._head`? (*Hint: see line 2*)

d. When `self._head.mystery(a)` is invoked, what class's `mystery` method is being called?

e. If we called this new mystery method on an empty LinkedList with `ll.mystery(55)`, what would happen?

f. If we replaced line 4 with `self._head.append(a)` and called this method on a LinkedList which already contains the values 1, 2, 3, 4, with `ll.mystery(55)`, what would happen?

g. What should the `mystery` method be renamed to?

---

FYI: `__setitem__(self, i, v)` is a special method in python that is called when assigning a value to an indexed item on the left-hand side of an assignment operator.

5. In examining this code, the method on the top is called when line 4 is evaluated:

```python
0 def __setitem__(self, i, v):
1     self.head[i] = v

>>> ll = LinkedList()
>>> ll.append(3)
>>> ll[0] = 55
>>> print(ll)
[55]
```

a. What is stored in `ll` after line 2?

b. What might happen if we executed `None.__setitem__(0, 55)`?

c. What might happen if we were to replace line 3 with `ll[0] = 77`?
d. What is stored in \texttt{ll} after the original line 3 above? ______________________________

e. Write some code to add a second element to our list, \texttt{ll}, to have the value 99, and then change it to 33 using the \texttt{setitem} special method, implicitly:

Application Questions: Use the Python Interpreter to check your work

1. Write the \texttt{__str__} method for our LinkedList class so that it prints the values of our list, just like when we print a python list (remember the square brackets and commas):

   ```python
def \texttt{__str__}(\texttt{self}):
   
   
   
   
   
   
   
   ```

2. Write the \texttt{extend(\texttt{self}, \texttt{v})} method for our LinkedList class so that it adds all the objects stored in \texttt{v} to the end of our list. Make use of the \texttt{Element.append(\texttt{self}, \texttt{v})} method to avoid code redundancy:

   ```python
def \texttt{extend}(\texttt{self}, \texttt{v}):
   
   
   
   
   
   
   ```

3. Write a method of LinkedList that returns \texttt{True} if the given value, \texttt{v}, exists as a value within the list, \texttt{False} if not contained in the LinkedList. Refer to the LinkedList.\texttt{__len__}(\texttt{self}) code in this activity for some hints on the approach and structure.

   ```python
def \texttt{__contains__}(\texttt{self}, \texttt{v}):
   
   
   
   
   
   
   ```
4. Write a method, \texttt{\textunderscore\textunderscore\texttt{iter\textunderscore\textunderscore}(self)}, for the LinkedList class that \textit{yields} one item from our list at a time. This method would be called with a line like “\texttt{for item in ll:}”. Refer to the activity on generators for more insight.

```python
def __iter__(self):
    # implementation
```

\textbf{FYI:} \texttt{\textunderscore\textunderscore\texttt{iter\textunderscore\textunderscore}(self)} is a special method in python that is called when using a for loop over that sequence object. Any class that has this method defined is \textit{iterable}.

5. Refer to Homework and Lab assignments for more application questions!