Glorious Strings

Part (a): Working with a partner, figure out what is displayed when the Python code is executed Consult the “Strings Cheat Sheet,” as needed. The diagrams are provided to help you keep track of indices. (You could use a strategy like this on an exam to reduce the chance of errors due to miscounting.)

Do not use Spyder to check your answers.

<table>
<thead>
<tr>
<th></th>
<th>t</th>
<th>e</th>
<th>l</th>
<th>e</th>
<th>p</th>
<th>h</th>
<th>o</th>
<th>n</th>
<th>e</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-1</td>
</tr>
<tr>
<td>-9</td>
<td>-8</td>
<td>-7</td>
<td>-6</td>
<td>-5</td>
<td>-4</td>
<td>-3</td>
<td>-2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>s</th>
<th>p</th>
<th>o</th>
<th>t</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-4</td>
<td>-3</td>
<td>-2</td>
<td>-1</td>
<td></td>
</tr>
</tbody>
</table>

Python Code

```python
word = 'telephone'
print( word[1] )
print( word[1:4] )
print( word.find('ph') )
print( word.replace('e', 'oi') )
print( word[-2:-2] )
word = 'spot'
print( word[1:] )
print( word + 'ty' )
print( word[:len(word)] )
print( word[:len(word)-1][:-1] )
i = word.index('p')
print( word[:i] + word[i+1:] )
print( word[:word.find('e')] )
print( (2*word)[:2] )
print( word[:len(word)-2].upper() )
r = ''
for c in word:
    r = c + r
print( word, r )
```

To check your answers, step through Visualization 0 using the link in the Artifacts section of the website. Ask about any answers that you don’t understand.
Part (b): Tracing execution of an algorithm on a specific example can help you figure out what the algorithm does in general. To trace an execution, you list the line numbers of the statements that are executed, in order, and show the values of any variables that are changed at each statement.

DO NOT USE SPYDER TO CHECK YOUR ANSWERS.

Finish the following trace of mystery1.py assuming the user enters 'Cu L8r!' at the prompt (without the quotes).

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Variable(s) changed</th>
<th>Line no.</th>
<th>Variable(s) changed</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>w: 'Cu L8r!'</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>r: ''</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>c: 'C'</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

When done, step through Visualization 1 to check your trace.

What will be printed if the user enters 'I do not like them, Sam-I-Am.' at the prompt?

What will be printed if the user just hits the enter/return-key at the prompt?

How is the output of the program related to the input?
**Part (c):** Finish the following two traces of `mystery2.py`. For the first trace, assume the user enters "Endive" at the prompt (without the quotes); for the second, assume the user enters "Madam". DO NOT USE SPYDER TO CHECK YOUR ANSWERS.

<table>
<thead>
<tr>
<th>Line no.</th>
<th>Variable(s) changed</th>
<th>Line no.</th>
<th>Variable(s) changed</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td><code>w: &quot;Endive&quot;</code></td>
<td>4</td>
<td><code>w: &quot;Madam&quot;</code></td>
</tr>
<tr>
<td>6</td>
<td><code>e: 5</code></td>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>

When done, step through **Visualization 2** to check your trace. (Once you give a user input in a visualization, you cannot provide different inputs by just restarting. To input "Madam" after already inputting "Endive", you need to press the *Edit Code* link and then, in the editor, press the *Visualize Execution* button. This starts up an entirely new execution.)

- What will be printed if the user enters "A man, a plan, a canal: Panama" at the prompt?
- What will be printed if the user just presses the enter key at the prompt?
- For what inputs does the program print "Bingo!"?
Part (d) [Extra for experts]: Download `mystery2.py` into a folder for this week’s class; bring it up in Spyder. Modify the program so that it ignores spaces and punctuation. For example, it should print "Bingo!" for the input "A man, a plan, a canal: Panama", in which the letters form a palindrome when spaces and punctuation are removed.

Implementation note: The string module can be a big help. If you import the string module, then `string.punctuation` and `string.whitespace` return strings containing, respectively, all the punctuation characters and all the whitespace characters.