1. Consider this code:

```python
phrase = 'Laughing Out Loud'
```

Assuming the code above has been executed, complete the indices in the expression below that will produce the string 'LOL'. Use at least one negative index in your answer.

```python
phrase[ ] + phrase[ ] + phrase[ ]
```

2. Consider this code:

```python
phrase = 'big orange cat'
slice1 = phrase[:3]
slice2 = phrase[-4:]
slice3 = phrase[3:8]
```

Assuming the code above has been executed, complete the table with the values that each variable refers to.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>phrase</td>
<td></td>
</tr>
<tr>
<td>slice1</td>
<td></td>
</tr>
<tr>
<td>slice2</td>
<td></td>
</tr>
<tr>
<td>slice3</td>
<td></td>
</tr>
</tbody>
</table>

3. Consider this code:

```python
lyrics = 'abc easy as 123'
```

Assuming the code above has been executed, circle the expression(s) that produce `False`.

(a) 'easy' in lyrics   (b) str(len('mj')) in lyrics
(c) 'cab' in lyrics   (d) '' in lyrics

4. Consider this code:

```python
s = 'Jacqueline'
```

You know that the slicing operation `s[1:4]` will produce the string 'acq'. The slicing operation has an optional third parameter that determines the `stride` (or distance between characters) in the slice. For example, the slicing operation `s[::2]` will produce the string 'Jculn', which has every other character in 'Jacqueline', starting from the first character in the string, and up to the end of the string. Use a negative stride to work backwards through a string.

(a) Write an expression that uses slicing on `s` to produce the string 'aqeie'.

(b) Write an expression that uses slicing on `s` to produce the string 'enileuqcaJ'.

(c) Write an expression that uses slicing on `s` to produce the string 'eieqa'.