Clicker Questions
for
L11: Lists
Q1: Changing an Element in a List

If `val = [ 27, False, "hockey", 3.14159 ]`, what command(s) would change the element containing "hockey" to be 42?

A. `val[3] = 42`
B. `val[2] = 42`
C. `val["hockey"] = 42`
D. `for i in range(len(val)):`
   
   ```python
   if(val[i] == "hockey"): val[i] = 42
   ```
E. both B and D
Q2: List Methods

- If \( \text{vals} = [\text{'a'}, \text{'b'}, \text{'c'}, \text{'d'}] \), what would the value of \( \text{vals} \) be after performing the following operations?
  
  \[
  \begin{align*}
  \text{vals}.\text{insert}(2, \text{'b'}) \\
  \text{vals}.\text{append}(\text{'f'}) \\
  \text{vals}[2] = \text{'f'}
  \end{align*}
  \]

A. ['a', 'b', 'c', 'd']
B. ['a', 'b', 'c', 'd', 'f']
C. ['a', 'b', 'f', 'c', 'd', 'f']
D. ['a', 'f', 'b', 'c', 'd', 'f']
E. none of the above
Q3: Trace the call

Consider the following code:

```python
def below_limit(list1, limit):
    alist = []
    for item in list1:
        if item < limit:
            alist.append(item)
    return alist

alist = [1, 5, 3]
a = 4
blist = below_limit(alist, a)
```

- What is the state of memory just after the `below_limit` function call begins executing? (Exclude functions from frame)

<table>
<thead>
<tr>
<th></th>
<th>Global frame</th>
<th>Local (below_limit)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>alist ➔ [0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>a ➔ 4</td>
<td>a ➔ 4</td>
</tr>
<tr>
<td>B</td>
<td>alist ➔ [0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>a ➔ 4</td>
<td>limit ➔ 4</td>
</tr>
<tr>
<td></td>
<td>blist ➔ below_limit</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>alist ➔ [0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>a ➔ 4</td>
<td>limit ➔ 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>alist ➔ [1</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>a ➔ 4</td>
<td>limit ➔ 4</td>
</tr>
<tr>
<td>E</td>
<td>None of the above</td>
<td></td>
</tr>
</tbody>
</table>


Q4: Trace the call

Consider the following code:

```python
def below_limit(list1, limit):
    alist = []
    for item in list1:
        if item < limit:
            alist.append(item)
    return alist
```

alist = [1, 5, 3]
a = 4
blist = below_limit(alist, a)

What is the state of memory just **before** the function call to `below_limit` finishes? (Exclude functions from frame)

- **A**: `alist → [0|1|2] 1 5 3`  
- **B**: `alist → [0|1] 1 3`  
- **C**: `alist → [0|1|2] 1 5 3`
Q5: Trace the call

Consider the following code:

```python
def below_limit(list1, limit):
    alist = []
    for item in list1:
        if item < limit:
            alist.append(item)
    return alist
```

```plaintext
alist = [1,5,3]
a = 4
blist = below_limit(alist, a)
```

- What is the state of memory after the last line executes? (Exclude functions from frame)

<table>
<thead>
<tr>
<th></th>
<th>Global frame</th>
<th>Local (below_limit)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td><code>alist</code> ➞ [0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>↓↓↓↓</td>
<td>↓↓↓↓</td>
</tr>
<tr>
<td></td>
<td>1 5 3</td>
<td>1 5 3</td>
</tr>
<tr>
<td></td>
<td><code>a</code> ➞ 4</td>
<td><code>limit</code> ➞ 4</td>
</tr>
<tr>
<td></td>
<td><code>blist</code> ➞ [0</td>
<td>1]</td>
</tr>
<tr>
<td></td>
<td>↓↓</td>
<td><code>alist</code> ➞ [0</td>
</tr>
<tr>
<td></td>
<td>1 3</td>
<td>↓↓</td>
</tr>
<tr>
<td></td>
<td>Frame doesn’t exist</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td><code>alist</code> ➞ [0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>↓↓↓↓</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 5 3</td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>a</code> ➞ 4</td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>blist</code> ➞ [0</td>
<td>1]</td>
</tr>
<tr>
<td></td>
<td>↓↓</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 3</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td><code>alist</code> ➞ [0</td>
<td>1]</td>
</tr>
<tr>
<td></td>
<td>↓↓</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 3</td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>a</code> ➞ 4</td>
<td></td>
</tr>
<tr>
<td></td>
<td><code>blist</code> ➞ [0</td>
<td>1]</td>
</tr>
<tr>
<td></td>
<td>↓↓</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 3</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>Neither frame exists</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>None of the above</td>
<td></td>
</tr>
</tbody>
</table>
Q6: Extracting a Sublist

If `vals = ['a', 'b', 'c', 'd', 'e']`, how do we extract the third, fourth and fifth elements (where 1st is ‘a’)?

A. `vals[2,3,4]`
B. `vals[3,4,5]`
C. `vals[2:4]`
D. `vals[2:5]`
E. `vals[3:5]`
Q7: Extracting a Sublist

- If `vals = ['a', 'b', 'c', 'd', 'e']`, how do we extract every element except the first and last?

   A. `vals[:]
   B. `vals[0:-1]
   C. `vals[1:0]
   D. `vals[1:-1]
   E. `vals[1:-2]`
Q8: Extracting a Sublist

If `vals = ['a', 'b', 'c', 'd', 'e']`, how could we extract the first and last elements if we didn’t know how long the list was?

A. `vals[0:1:-1:]`

B. `vals[0:1] + vals[-1:]`

C. `vals[0:1] + vals[-1:-2]`

D. `vals[0:1] + vals[-2:-1]`

E. `vals[0:1] + vals[4:]`