1. What would Python display? If an error occurs, write ”Error”. If a function is displayed, write ”Function”. If nothing is returned, write ”Nothing”.

```python
>>> a = [1, 2]
>>> a.append([3, 4])
>>> a

Solution:
[1, 2, [3, 4]]

>>> b = list(a)
>>> a[0] = 5
>>> a[2][0] = 6
>>> b

Solution:
[1, 2, [6, 4]]

>>> a.extend([7])
>>> a += [8]
>>> a += 9

Solution:
TypeError: 'int' object is not iterable
```

>>> a
**Solution:**

```
[5, 2, [6, 4], 7, 8]
```

**Challenge:**

```python
>>> b[2][1] = a[2:]
>>> a[2][1][0][0]
```

**Solution:**

```
6
```
2. Draw the environment diagram that results from running the following code.

```python
a = [1, 2, [3]]
def mystery(s, t):
    s.pop(1)
    return t.append(s)
b = a
a += [b[0]]
a = mystery(b, a[1:])
```

Solution: https://goo.gl/s2XKiG
3. Given some list `lst`, possibly a deep list, mutate `lst` to have the accumulated sum of all elements so far in the list. If there is a nested list, mutate it to similarly reflect the accumulated sum of all elements so far in the nested list. Return the total sum of `lst`.

**Hint:** The `isinstance` function returns True for `isinstance(l, list)` if `l` is a list and False otherwise.

```python
def accumulate(lst):
    """
    >>> l = [1, 5, 13, 4]
    >>> accumulate(l)
    23
    >>> l
    [1, 6, 19, 23]
    >>> deep_l = [3, 7, [2, 5, 6], 9]
    >>> accumulate(deep_l)
    32
    >>> deep_l
    [3, 10, [2, 7, 13], 32]
    """

    for _______________:  
        ________________
        if isinstance(______________, list):
            inside = ________________
        else:
            ________________
    __________________________

    Solution:
    sum_so_far = 0
    for i in range(len(lst)):
        item = lst[i]
        if isinstance(item, list):
            inside = accumulate(item)
            sum_so_far += inside
        else:
            sum_so_far += item
        lst[i] = sum_so_far
    return sum_so_far
```
1. **Nonlocal Kale**  
Draw the environment diagram for the following code.
```python
eggplant = 8
def vegetable(kale):
    def eggplant(spinach):
        nonlocal eggplant
        nonlocal kale
        kale = 9
        eggplant = spinach
        print(eggplant, kale)
        eggplant(kale)
    return eggplant

spinach = vegetable('kale')
```

**Solution:** [https://goo.gl/F43X2Q](https://goo.gl/F43X2Q)
2. Pingpong again...

Implement a function `make_pingpong_tracker` that returns the next value in the pingpong sequence each time it is called. You may use assignment statements.

```python
def has_seven(k):  # Use this function for your answer below
    if k % 10 == 7:
        return True
    elif k < 10:
        return False
    else:
        return has_seven(k // 10)
```

Solution:
```python
def make_pingpong_tracker():
    index, current, add = 1, 0, True
def pingpong_tracker():
    nonlocal index, current, add
    if add:
        current = current + 1
    else:
        current = current - 1
    if has_seven(index) or index % 7 == 0:
        add = not add
    index += 1
    return current
return pingpong_tracker
```