

Work with your neighbor. (This will be graded for participation only.)

1. Aliasing. Given the code:

```
x = [10, 20, 30]
```

```
y = [x, x]
```

Draw the resulting diagram:

How many aliases (references to the same data object) are there in this diagram?

2. User-defined types. Just as we can draw diagrams for Python built-in types, we can draw diagrams for objects that are instances of user-defined classes.

- a) Draw the diagrams for these `Point` objects defined below. Each `Point` object will have two boxes, one for each attribute (i.e, `self._x` and `self._y`).

```
a = Point(3, 8)
```

```
b = Point(5, 14)
```

- b) Given the assignments for `a` and `b` above, what is the diagram for `z`?

```
z = [a,b]
```

3. Compare the diagrams of a built-in Python list vs a linked list.

a) Draw the diagram for this list:

```
alist = [10, 20, 30]
```

b) Now draw it as a linked list

4. The `LinkedList` and `Node` classes (first pass...)

```
class Node:
    def __init__(self, value):
        self._value = value
        self._next = None
```

```
class LinkedList:
    def __init__(self):
        self._head = None
```

Draw a diagram that shows the `LinkedList` object `alist` after the assignment is executed:

```
alist = LinkedList()
```

Draw a diagram that shows the `Node` object `n` after the assignment is executed:

```
n = Node(10)
```

Draw a diagram that shows the `Node` object `m` after the assignment is executed:

```
m = Node(20)
```

5. We can draw these now, but how do we get them to be connected?
- How do we get the reference in `alist._head` to refer to `n`?
 - How do we get the reference in `n._next` to refer to `m`?

Discuss this with your neighbors!