LINKED LISTS

COMPUTER SCIENCE MENTORS 61A

February 22 to February 26, 2016

1. **(H)OOP** Given the following code, what will Python output for the following prompts?

```
class Baller:
        all_players = []
        def __init__(self, name, has_ball = False):
                self.name = name
                self.has_ball = has_ball
                Baller.all_players.append(self)

        def pass_ball(self, other_player):
                if self.has_ball:
                self.has_ball = False
                other_player.has_ball = True
                return True
        else:
                return False

class BallHog(Baller):
        def pass_ball(self, other_player):
                return False
>>> tiffany = Baller('Tiffany', True)
>>> garrett = BallHog('Garrett')
>>> len(Baller.all_players)

>>> Baller.name

>>> len(garrett.all_players)
```

```
>>> tiffany.pass_ball()

>>> tiffany.pass_ball(garrett)

>>> tiffany.pass_ball(garrett)

>>> BallHog.pass_ball(garrett, tiffany)

>>> garrett.pass_ball(tiffany)

>>> garrett.pass_ball(garrett, tiffany)
```

2. **TeamBaller** Write `TeamBaller`, a subclass of `Baller`. An instance of `TeamBaller` cheers on the team every time it passes a ball.
   Hint: What can we use to avoid writing duplicate code?
   "Super" Hint: There are two ways to implement `pass_ball`

```
>>> cheerballer = TeamBaller('Susanna', has_ball=True)
>>> cheerballer.pass_ball(garrett)
Yay!!!!
True
>>> cheerballer.pass_ball(garrett)
I dont have the ball :(
False
```

```
class TeamBaller(__________):
        def pass_ball(__________, ___________):
```

3. **Nonlocal Kale** Draw the environment diagram for the following code.

```
eggplant = 8
carrot = 0
def vegetable(kale):
    carrot = 10
    def eggplant(spinach):
        nonlocal eggplant
        nonlocal kale
        kale = 9
        carrot = 20
        eggplant = spinach
    eggplant(kale)
    return eggplant
spinach = vegetable(lambda kale: carrot*kale)(eggplant)
```

4. **Pinpong again...** Recap of ping-pong: The ping-pong sequence counts up starting from 1 and is always either counting up or counting down. At element k, the direction switches if k is a multiple of 7 or contains the digit 7. The first 30 elements of the ping-pong sequence are listed below, with direction swaps marked using brackets at the 7th, 14th, 17th, 21st, 27th, and 28th elements:

```
1 2 3 4 5 6 [7] 6 5 4 3 2 1 [0] 1 2 [3] 2 1 0 [-1] 0 1 2 3 4
   [5] [4] 5 6
```

Implement a function `pingpong` that returns the nth element of the ping-pong sequence.

```
1 2 3 4 5 6 [7] 6 5 4 3 2 1 [0] 1 2 [3] 2 1 0 [-1] 0 1 2 3 4
   [5] [4] 5 6
```

5. **(Optional)** Instead of using nonlocal for pingpong, let's use OOP!

```
>>> tracker1 = PingPongTracker()
>>> tracker2 = PingPongTracker()
>>> tracker1.next()
1
>>> tracker1.next()
2
>>> tracker2.next()
1

Bonus points if you can get the following syntax.
>>> tracker1()
1
>>> tracker1()
2

class PingPongTracker:
    def __init__(self):
        self.current = 0
        self.index = 0
        self.add = True
    def next(self):
        *** Enter solution below ***
```