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

```python
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
```

---

1. **Object Oriented Programming**

---

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

```python
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
```
>>> paul = Baller('Paul', True)
>>> jason = BallHog('Jason')

>>> len(Baller.all_players)

>>> Baller.name

>>> len(jason.all_players)

>>> paul.pass_ball()

>>> paul.pass_ball(jason)

>>> paul.pass_ball(jason)

>>> BallHog.pass_ball(jason, paul)

>>> jason.pass_ball(paul)

>>> jason.pass_ball(jason, paul)

2. Write TeamBaller, a subclass of Baller. An instance of TeamBaller cheers on the team every time it passes a ball.

```python
class TeamBaller(_______________):
    """
    >>> cheerballer = TeamBaller('Thomas', has_ball=True)
    >>> cheerballer.pass_ball(jason)
    Yay!
    True
    >>> cheerballer.pass_ball(jason)
    I don't have the ball
    False
    """
    def pass_ball(_______________, ________________):
```
3. **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')
   ```
4. **Pingpong again...**

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 `make_pingpong_tracker` that returns the next value in the pingpong sequence each time it is called. In the body of `make_pingpong_tracker`, you can 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)

def make_pingpong_tracker():
    """ Returns a function that returns the next value in the pingpong sequence each time it is called."
    >>> output = []
    >>> x = make_pingpong_tracker()
    >>> for _ in range(9):
    ...     output += [x()]
    >>> output
    [1, 2, 3, 4, 5, 6, 7, 6, 5]
    """

    index, current, add = 1, 0, True
    def pingpong_tracker():
        ____________________________
        if add:
            _____________________________
        else:
            ____________________________
            if ________________________:
                add = not add
            _____________________________
        return pingpong_tracker
```

---

Computer Science Mentors CS61A Fall 2017: Paul Bitutsky and Jason Goodman, with Colin Schoen, Kevin Lin, Thomas Zhang, Danielle Nachum, Jennie Chen, Nipun Ramakrishnan, Christopher Allsman, Nikunj Jain, Shreya Sahoo, Joshua Zeitsoff
5. (Optional) Instead of using nonlocal for pingpong, let’s use OOP!

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

class PingPongTracker:
    def __init__(self):
        self.current = 0
        self.index = 1
        self.add = True
    def next(self):
        """*** Enter solution below ***"
```