CS 61A
Summer 2017

Mutability & OOP

Mentoring 9: July 19, 2017

1 List Mutation

1.1 What would Python display?

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

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

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

```
>>> a
```

Challenge:

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

1.2 Draw the box-and-pointer diagram.

```
>>> corgi = [3, 15, 18, 7, 9]
>>> husky = [8, 21, 19, 11, 25]
>>> poodle = corgi.pop()
>>> corgi += husky[-3:]
```

1.3 Draw the box-and-pointer diagram.

```
>>> pom = [16, 15, 13]
>>> pompom = pom * 2
>>> pompom.append(pom[:])
>>> pom.extend(pompom)
```

1.4 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: You may find it useful to use the **`isinstance`** function, which returns true for **`isinstance`**`(l, `**`list`**`)` if `l` is a list and false otherwise.

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

2 OOP

2.1 Given the following code, what would Python display?

```
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):
        if self.has_ball:
            self.has_ball = False
            other.has_ball = True
            return True
        else:
            return False

class BallHog(Baller):
    def pass_ball(self, other):
        return False
```

(a)
```
anwar = Baller('Anwar', True)
jerry = BallHog('Jerry')
len(Baller.all_players)
```

(b)
```
Baller.name
```

(c)
```
len(jerry.all_players)
```

(d)
```
anwar.pass_ball()
```

(e)
```
anwar.pass_ball(jerry)
```

(f)
```
anwar.pass_ball(jerry)
```

(g)
```
BallHog.pass_ball(jerry, anwar)
```

(h)
```
jerry.pass_ball(anwar)
```

(i)
```
jerry.pass_ball(jerry, anwar)
```

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

```
class TeamBaller(_________________):
    """
    >>> cheerballer = TeamBaller('Thomas', has_ball=True)
    >>> cheerballer.pass_ball(jerry)
    Yay!
    True
    >>> cheerballer.pass_ball(jerry)
    I don't have the ball
    False
    """
    def pass_ball(_________________, _________________):
```