More OOP and Inheritance

Computer Science Mentors CS 88

March 5th to April 10th

1 Inheritance

1. Fill in the classes `Emotion`, `Joy`, and `Sadness` below so that you get the following output from the Python interpreter.

```
>>> Emotion.num
0

>>> joy = Joy()
>>> sadness = Sadness()
>>> emotion = Emotion()
>>> Emotion.num # number of Emotion instances created
3

>>> joy.power
5

>>> joy.catchphrase() # Print Joy's catchphrase
Think positive thoughts

>>> sadness.catchphrase() #Print Sad's catchphrase
I'm positive you will get lost

>>> sadness.power
5

>>> emotion.catchphrase()
I'm just an emotion.

>>> joy.feeling(sadness) # print "Together" if same power
Together

>>> sadness.feeling(joy)
Together

>>> joy.power = 7
>>> joy.feeling(sadness) # Print the catchphrase of the more
   powerful feeling before the less powerful feeling
Think positive thoughts
I'm positive you will get lost

>>> sadness.feeling(joy)
Think positive thoughts
I'm positive you will get lost
```

```
class Emotion

    def __init__(self):



    def feeling(self, other):









    def catchphrase(self):


class Joy
    def catchphrase(self):
```

```
class Sadness
    def catchphrase(self):
```

2. **(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
```

```
>>> neil = Baller('Neil', True)
>>> michelle = BallHog('Michelle')
>>> len(Baller.all_players)

>>> Baller.name

>>> len(michelle.all_players)
```

```
>>> neil.pass_ball()

>>> neil.pass_ball(michelle)

>>> neil.pass_ball(michelle)

>>> BallHog.pass_ball(michelle, neil)

>>> michelle.pass_ball(neil)

>>> michelle.pass_ball(michelle, neil)
```

3. **(H)OOP**

Here is the Baller code again

```
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
```

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

```
class TeamBaller(________________):
    """
    >>> alex = BallHog('Alex')
    >>> cheerballer = TeamBaller('Richard', has_ball=True)
    >>> cheerballer.pass_ball(alex)
    Yay!
    True
    >>> cheerballer.pass_ball(alex)
    I don't have the ball
    False
    """
    def pass_ball(________________, ________________):
```

4. **FrOOPt**

Given the following code, what will Python output for the following prompts?

```
class Fruit:
    ripe = False
    def __init__(self, taste, size):
        self.taste = taste
        self.size = size
        self.ripe = True

    def eat(self, eater):
        print(eater, 'eats the', 'self.name)
        if not self.ripe:
            print('But it isn't ripe!')
        else:
            print('What a', self.taste, self.size, 'fruit!')

class Tomato(Fruit):
    name = 'tomato'
    def eat(self, eater):
        print('Adding some sugar first')
        self.taste = 'sweet'
        Fruit.eat(self, eater)
```

```
>>> mystery = Friut('tart', 'small')
>>> tommy = Tomato('plain', 'normal')
>>> mystery.taste

>>> mystery.name

>>> tommy.eat('Brian')


>>> Tomato.ripe

>>> tommy.name = 'sweet tomato'
>>> Fruit.eat = lambda self, own : print(self.name, 'is too
   sweet!')
>>> tommy.eat('Marvin')
```

5. **Flying the cOOP** What would Python display?
   Write the result of executing the code and the prompts below.

   If a function is returned, write "Function". If nothing is returned, write "Nothing". If an error occurs, write "Error".

```
class Bird:
    def __init__(self, call):
        self.call = call
        self.can_fly = True
    def fly(self):
        if self.can_fly:
            return "Don't stop
               me now!"
        else:
            return "Ground
               control to Major
               Tom..."
    def speak(self):
        print(self.call)

class Chicken(Bird):
    def speak(self, other):
        Bird.speak(self)
        other.speak()

class Penguin(Bird):
    can_fly = False
    def speak(self):
        call = "Ice to meet you
           "
        print(call)

andre = Chicken("cluck")
gunter = Penguin("noot")
```

```
>>> andre.speak(Bird("coo"))

>>> andre.speak()

>>> gunter.fly()

>>> andre.speak(gunter)

>>> Bird.speak(gunter)
```