Discussion 13: Object-Oriented Programming

Creating Objects

a. Your good friend Matthew created some objects in Python, but then accidentally deleted his file! Lucky for him, he still has the output of running his code saved. Use this output to help Matthew reconstruct his original code.

```
>>> x = X(5, 6)
>>> X.foo
123
>>> x.foo
123
>>> x.show_vars()
[5, 6]
>>> x.say_hello()
ERROR

>>> y = Y(3, 4, 10)
>>> y.foo
10
>>> Y.foo
123
>>> y.show_vars()
[3, 4]
>>> y.say_hello()
hello
>>> x.incr_other(y)
>>> y.show_vars()
[4, 5]

>>> z = Z(1)
>>> z.show_vars()
ERROR
```

b. Now, based on the class you wrote above, what would Python display if you ran the lines below? If any line would cause an error, write "ERROR."

```
>>> new_y = Y(10, 20, 30)
>>> Y.foo += 1
>>> new_y.foo
```

_____

```
>>> x.foo
```

_____

```
>>> z.foo
```

_____

What Would Python Display?

Determine what Python would output while running the code below. You may assume that each part is independent of the others: the code run in one part is not run in parts after. If any line would cause an error, write "ERROR." You may not need all lines.

a.
```
class A:
      def __init__(self, x, y):
            self.first = x
            self.second = y
            print('I have been initialized!')

>>> a = A(1, 2)
```

_____

b.
```
class Insect:
      insects = 0
      def __init__(self, energy):
            self.energy = energy
            Insect.insects += 1

      def move():
            self.energy -= 1
            if self.energy == 0:
                  print('I am tired!')

      def fly(self, spider):
            self.energy -= 3
            if self.energy == 0:
                  spider.catch(self)

class Spider:
      def __init__(self, energy):
            self.energy = energy
            self.insects = 0

      def catch(self, insect):
            self.insects += Insect.insects
            self.energy += insect.energy


>>> spider = Spider(0)
>>> insect = Insect(3)
>>> insect.move()
```

_____

```
>>> insect.fly(Spider(1))
>>> insect.energy
```

_____

```
>>> spider.insects
```

```
>>> spider.catch(insect)
>>> spider.catch(Insect(2))
>>> spider.insects
```

---

c.
```
class Dessert:
      sweetness = 10
      def __init__(self):
            self.sweetness = Dessert.sweetness

      def eat(self):
            if self.sweetness < Dessert.sweetness:
                  return ‘Yuck!’
            print(‘Yum!’)
            self.sweetness -= 1

class SuperDessert(Dessert):
      sweetness = 20

      def eat(self):
            print(‘Yum!’)
            self.sweetness -= 1
            Dessert.eat(SuperDessert)

class NestedDessert(Dessert):
      def __init__(self, dessert1, dessert2):
            self.dessert1 = dessert1
            self.dessert2 = dessert2
            self.sweetness = dessert1.sweetness + dessert2.sweetness

      def nest_me(self, dessert):
            print(‘What have you done’)
            return NestedDessert(dessert, self)

>>> macaron = Dessert()
>>> macaron.eat()
```

```
>>> Dessert.sweetness = 9
>>> cheesecake= SuperDessert()
>>> cheesecake.eat()
```

```
>>> macaron.eat()
```

```
>>> yum = NestedDessert(macaron, Dessert())
>>> yummy = NestedDessert(Dessert, SuperDessert)
>>> yum.sweetness
```

_____

```
>>> yummy.sweetness
```

_____

```
>>> delicious = yummy.nest_me(yum)
```

_____

```
>>> delicious.eat()
```

_____

Bonus Questions:

i. What would `Dessert.sweetness` need to be set to for `delicious.eat()` to return `'Yuck!'`?

_____

ii. Complete the expression below to return `macaron`.

`delicious.`_____