CS 120: Intro to Computer Programming II

In-Class Activity - 04 Classes and Objects - Day 2

In this ICA, we're going to write a class called `Counter`, which was discussed in the slides and the video. We're going to do this in stages.

Activity 1 - Turn in this one

Write a class called `Counter`, which represents a click-counter like I mentioned in the video.

For the first version of your class, just include a constructor (that is, a method named `__init__()`). You should design it so that someone can create an object of your class with no parameters, like this:

```
my_counter = Counter()
```

How many parameters does `__init__()` need, and what are their name(s)? Review the slides (and discuss with your group) how this is supposed to work.

Your class must initialize, in your constructor, any variable(s) that your object will need. Discuss with your group what variable(s) you will need, and what their types will be.

Solution:

```
def Counter:
    def __init__(self):
        self.count = 0

    def click(self):
        self.count += 1

    def get_count(self):
        return self.count
```

Activity 2 - Turn in this one

Write some test code for your class. Create a snippet of code that creates three different `Counter` objects. Increment the counter in each a few times (do it a different number of times in each counter). Then print out the current counts in each one.

If your classes are working properly - if you are using fields inside the object, instead of global variables or something like that - then it should be possible for all three objects to have different counts.

Solution:

```
a = Counter()
b = Counter()
c = Counter()

a.click()
a.click()
a.click()
```

```
b.click()
b.click()
b.click()
b.click()
b.click()

c.click()

print(a.get_count())
print(b.get_count())
print(c.get_count())
```

(activity continues on the next page)

Activity 3 - Optional

OPTIONAL. Complete this if you have time, and turn it in. If you don't have time, you may report to your TA that you ran out of time.

Now, it's time to start using some methods, instead of accessing the fields of the objects directly. (Next video, I'll tell you all about encapsulation - but you don't need to know all that quite yet.)

For now, add methods to perform each of the following tasks:

- A method which will increment the count by 1.
- A method which returns the current count.
- A method which will reset the count to zero.

Modify your test code (from the previous activity) to use **only** the methods - don't access the fields directly anymore. Add code to test the reset method as well. Confirm that you class works as you might expect, in a variety of combinations.

Challenge Activity - Do not turn in this one

Update your class with the following features:

- Keep track of how many times the counter has been reset. (Include a getter to read this information.)
- Add code to your click function, which will "wrap around" (that is, go back to zero) if the counter hits ten thousand.
- Add a "total click" counter, which ignores both the resets and the wrap-around feature; include a getter for it.

Solution:

```
def Counter:
    def __init__(self):
        self.count       = 0
        self.reset_count = 0
        self.total_clicks = 0

    def click(self):
        self.count += 1
        self.total_clicks += 1

        if self.count == 10_000:
            self.count = 0

    def reset(self):
        self.count = 0
        self.reset_count += 1

    def get_count(self):
        return self.count
```

```
        def get_reset_count(self):
            return self.reset_count

        def get_total_clicks(self):
            return self.total_clicks
```