C1M5L3_Code_Reuse_V2

March 29, 2020

1 Code Reuse

Let's put what we learned about code reuse all together. First, let's look back at **inheritance**. Run the following cell that defines a generic `Animal` class.

```
In [7]: class Animal:
            name = ""
            category = ""

            def __init__(self, name):
                self.name = name

            def set_category(self, category):
                self.category = category
```

What we have is not enough to do much – yet. That's where you come in. In the next cell, define a `Turtle` class that inherits from the `Animal` class. Then go ahead and set its category. For instance, a turtle is generally considered a reptile. Although modern cladistics call this categorization into question, for purposes of this exercise we will say turtles are reptiles!

```
In [10]: class Turtle(Animal):
             category = "Reptile"
```

Run the following cell to check whether you correctly defined your `Turtle` class and set its category to reptile.

```
In [11]: print(Turtle.category)
```

```
Reptile
```

Was the output of the above cell reptile? If not, go back and edit your `Turtle` class making sure that it inherits from the `Animal` class and its category is properly set to reptile. Be sure to re-run that cell once you've finished your edits. Did you get it? If so, great!

Next, let's practice **composition** a little bit. This one will require a second type of `Animal` that is in the same category as the first. For example, since you already created a `Turtle` class, go ahead and create a `Snake` class. Don't forget that it also inherits from the `Animal` class and that its category should be set to reptile.

```
In [12]: class Snake(Animal):
             category ="Reptile"
```

Now, let's say we have a large variety of `Animals` (such as turtles and snakes) in a Zoo. Below we have the `Zoo` class. We're going to use it to organize our various `Animals`. Remember, inheritance says a Turtle is an `Animal`, but a `Zoo` is not an `Animal` and an `Animal` is not a `Zoo` – though they are related to one another.

Fill in the blanks of the `Zoo` class below so that you can use **zoo.add_animal( )** to add instances of the `Animal` subclasses you created above. Once you've added them all, you should be able to use **zoo.total_of_category( )** to tell you exactly how many individual `Animal` types the `Zoo` has for each category! Be sure to run the cell once you've finished your edits.

```
In [36]: class Zoo:
             def __init__(self):
                 self.current_animals = {}

             def add_animal(self, animal):
                 self.current_animals[animal.name] = animal.category

             def total_of_category(self, category):
                 result = 0
                 for animal in self.current_animals.values():
                     if animal == category:
                         result += 1
                 return result

         zoo = Zoo()
```

Run the following cell to check whether you properly filled in the blanks of your `Zoo` class.

```
In [37]: turtle = Turtle("Turtle") #create an instance of the Turtle class
         snake = Snake("Snake") #create an instance of the Snake class

         zoo.add_animal(turtle)
         zoo.add_animal(snake)

         print(zoo.total_of_category("Reptile")) #how many zoo animal types in the reptile cat

2
```

Was the output of the above cell 2? If not, go back and edit the `Zoo` class making sure to fill in the blanks with the appropriate attributes. Be sure to re-run that cell once you've finished your edits.

Did you get it? If so, perfect! You have successfully defined your `Turtle` and `Snake` subclasses as well as your `Zoo` class. You are all done with this notebook. Great work!