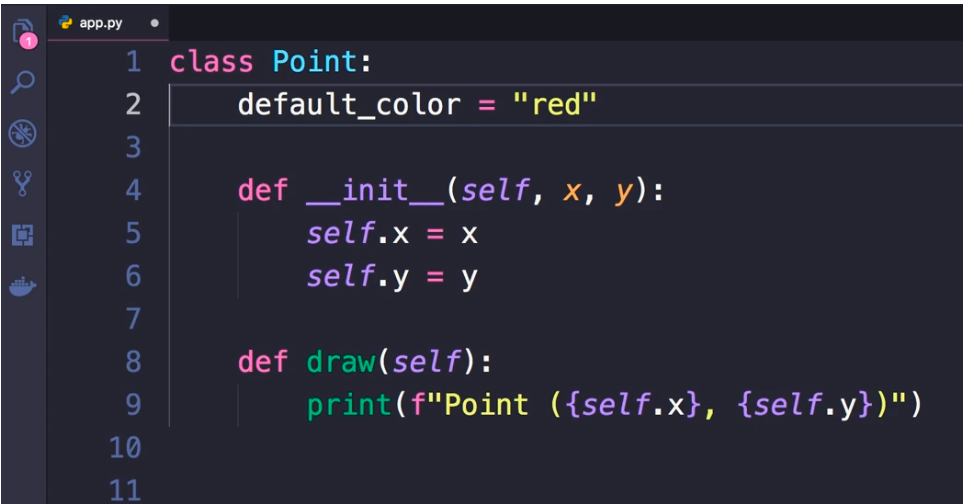
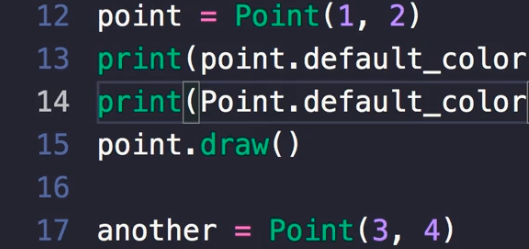
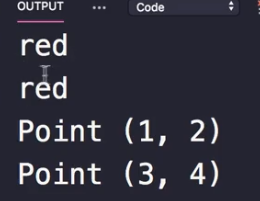


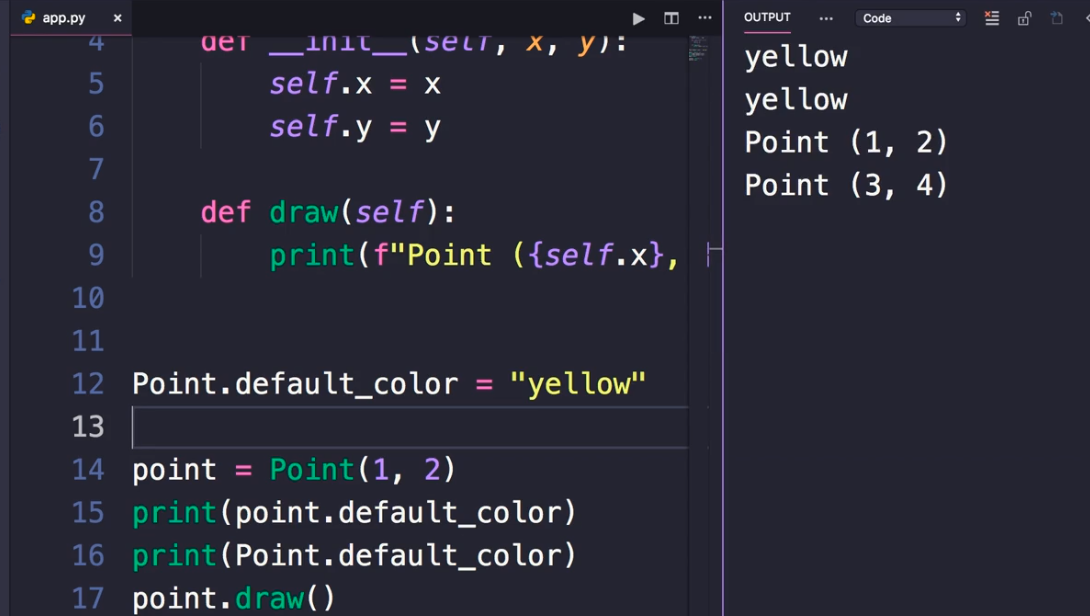


Let us see a class label attribute. **default\_color = “red”**

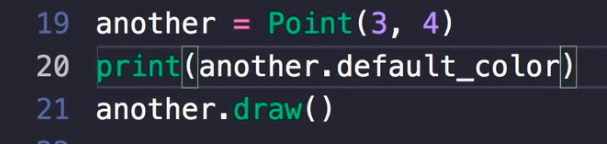
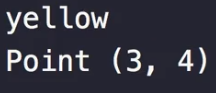


Class attributes are shared across all instance of a class. As all human instances have 2 ears and 2 eyes

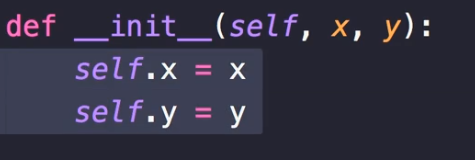


Please note that **Point.default\_color** = “yellow” changes the original label attribute value from red to “yellow”

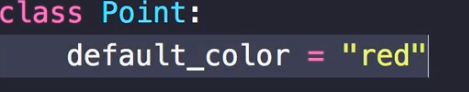
 

**So, class label attributes are shared across all instances of a class. If you change their value, the change is visible across all instances of the class**

In practical terms, you will be using instance attributes most of the time



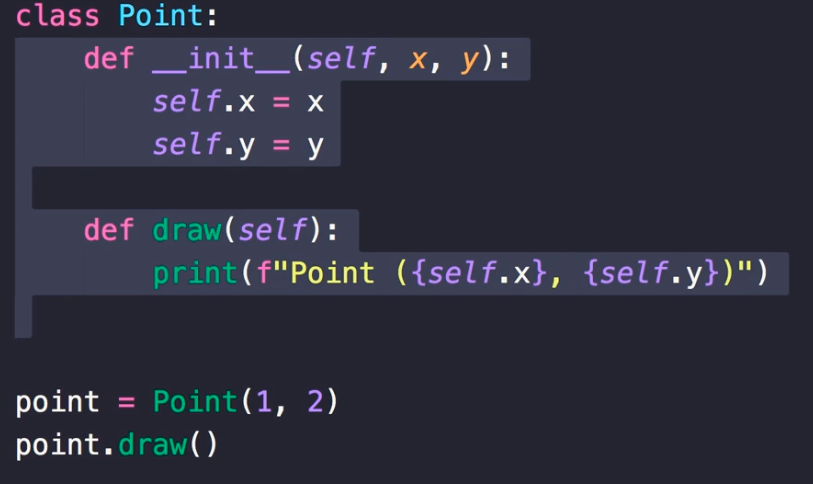
But there are times we want to define a class label attribute like default\_color below:



Default\_color is shared across all objects of **Point** class or a given type



\_\_init\_\_(self, x, y) and draw(self) are instance methods



So we can call them using an instance (point) of the **Point** class

Generally speaking use these instance methods where we need an object reference.

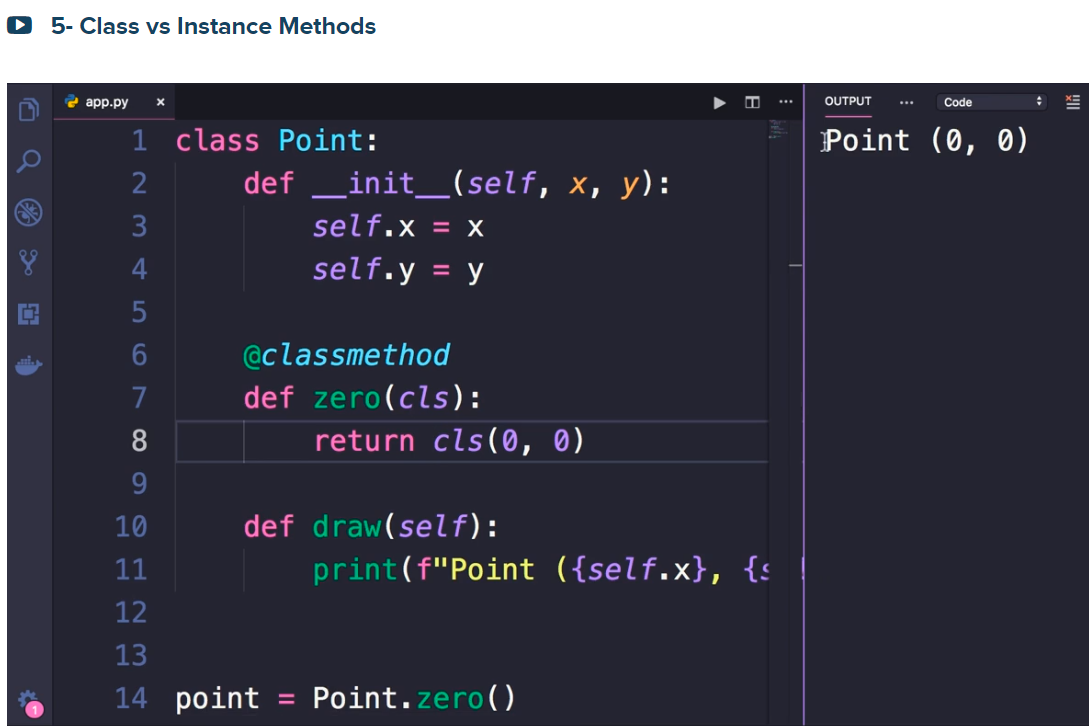
However, there are times where you do not need an existing object, and that is when you use a class method

Define factory method as shown below:

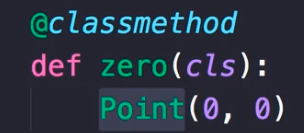
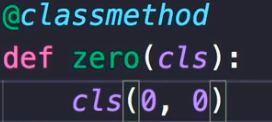


cls is a reference to the class itself

@classmehod is a decorater. It is method to extend the behavior of a method



Please note

if you use cls(0, 0) at run time it will be changed to Point(0, 0)



These are methods like

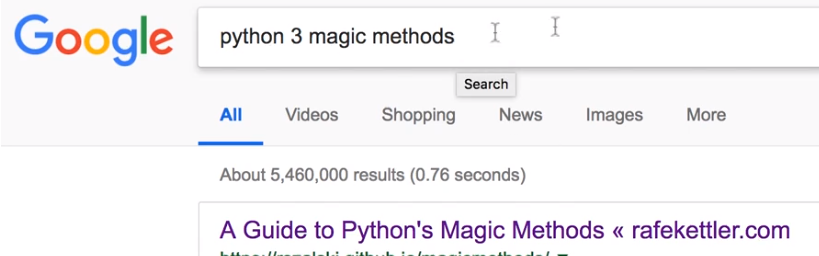


They are called automatically by python interpreter internally depending on how we use our objects and classes

 is called where create



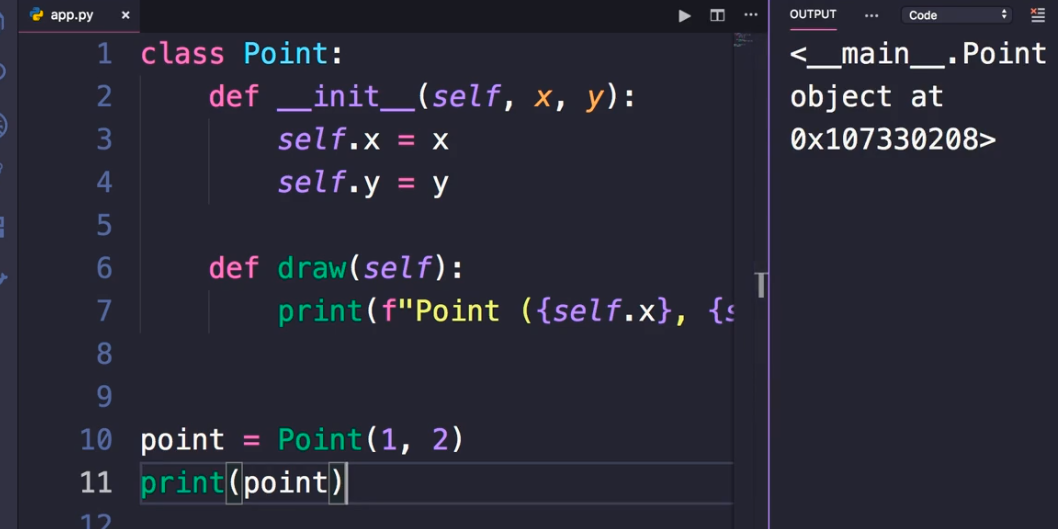
To see the lists:



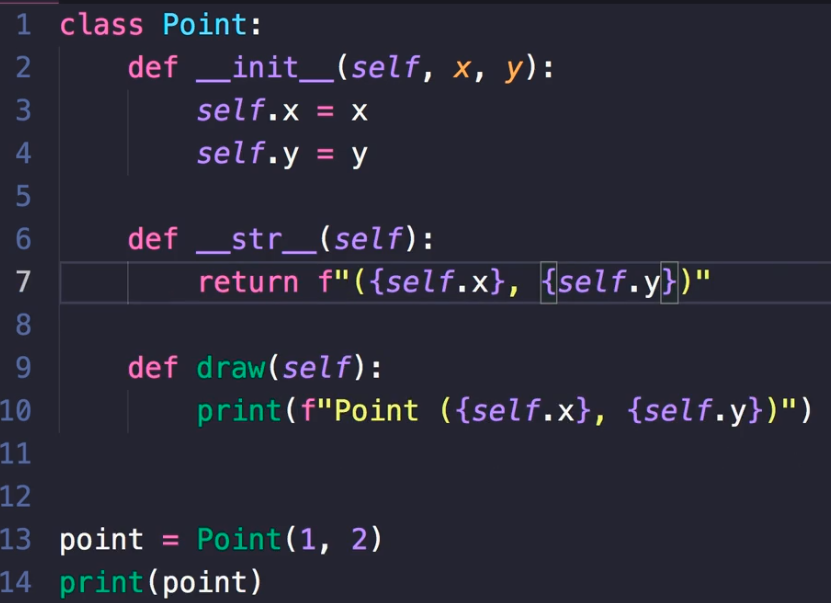
Or click on:

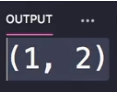
<https://rszalski.github.io/magicmethods/>

look what you get when you print an object like:

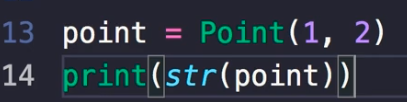
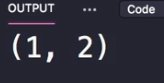


Now let us define \_\_str\_\_(self) as:

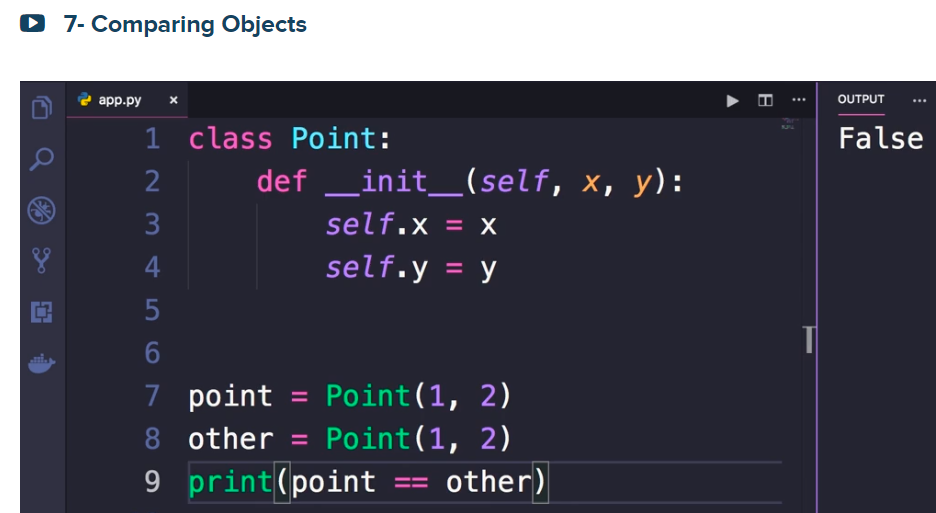


 when you print(point) you get the result (1,2) as shown in the back.

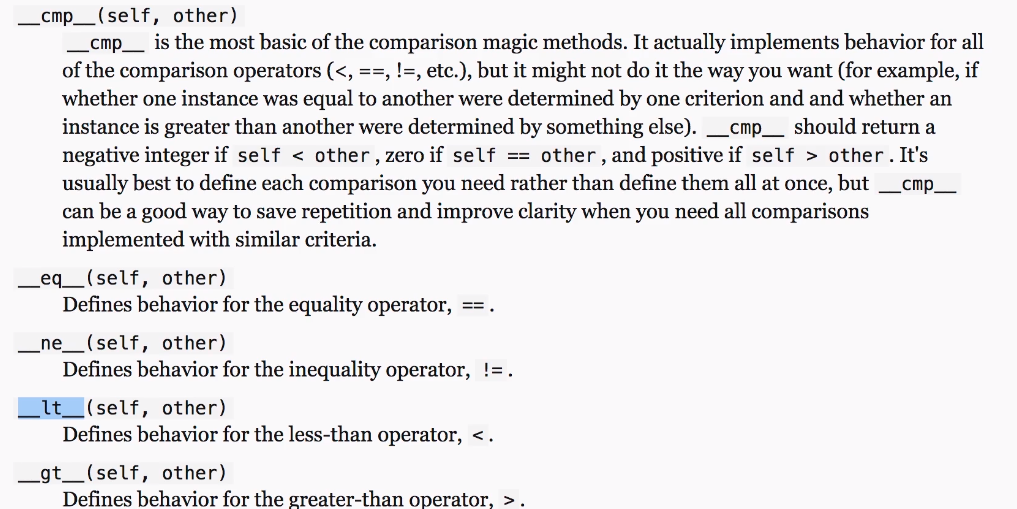
Also, you can use str(point)

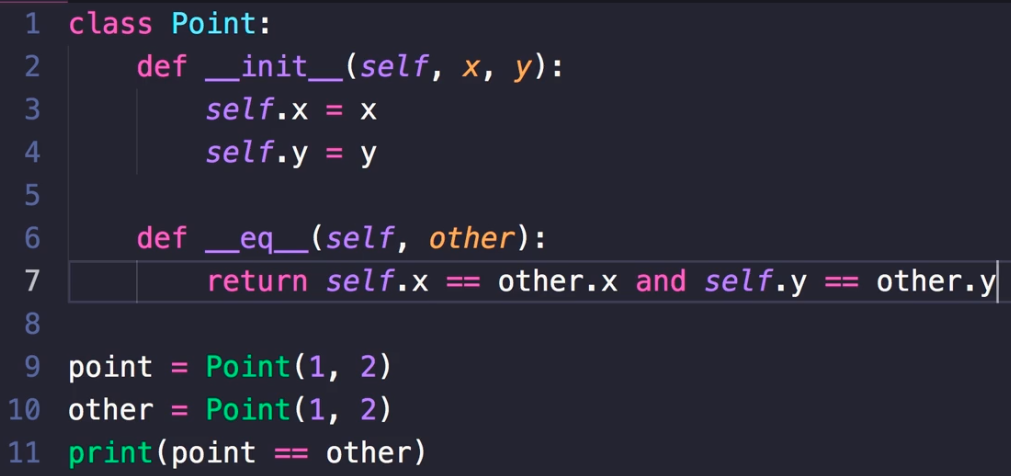
 





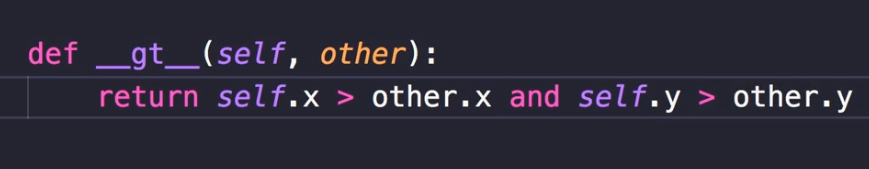
The reason we are getting False is that the equality or (==) compares the references of the two different objects: see

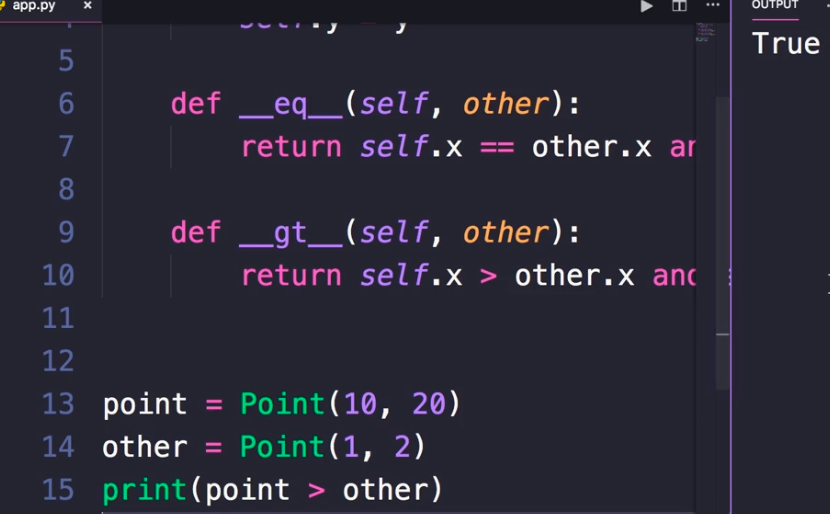






Now, implement greater than





We got False to less than as shown below:

