Solution Review: Implement a Print Method

This lesson discusses the __str__ method in Python for the string representation of an object.

WE'LL COVER THE FOLLOWING

Solution:

In Python, and in many other languages for that matter, if we make a class and print an instance of that class the output may vary every time. It prints the address of the object in memory. Consider the following code:

```
class Rectangle:
  def __init__(self, x1, y1, x2, y2): # class constructor
    if x1 < x2 and y1 > y2:
      self.x1 = x1 # class variable
      self.y1 = y1 # class variable
      self.x2 = x2 # class variable
      self.y2 = y2 # class variable
    else:
      print("Incorrect coordinates of the rectangle!")


# test your code
r = Rectangle (2, 7, 8, 4)
print (r)
```

However, python has a built-in method `__str__` used for the string representation of an object. `__repr__` is another built-in method which is similar to `__str__`. Both of them can be overridden for any class and there are only minor differences.

`str()`:

1. makes the object readable
2. generates output for end-user

`repr()`:

1. needs code that reproduces the object
2. generates output for developer

If both methods are defined in a class, `__str__` is used.

In the previous exercise, you had to implement the `__str__` method in the `Rectangle` class; therefore, when you print one of the objects using the `print()` command, it prints the coordinates as `x1, y1, x2, y2`.

Using `__str__`

Lines 11 and 12 in the code below show how this is done.

```
class Rectangle:
  def __init__(self, x1, y1, x2, y2): # class constructor
    if x1 < x2 and y1 > y2:
      self.x1 = x1 # class variable
      self.y1 = y1 # class variable
      self.x2 = x2 # class variable
      self.y2 = y2 # class variable
    else:
      print("Incorrect coordinates of the rectangle!")

  def __str__(self):
    return(str(self.x1) + ', ' + str(self.y1) + ', ' + str(self.x2) + ', '+str(self.y2))

# test your code
r = Rectangle (2, 7, 8, 4)
print (r)
```

Using `__repr__`

Lines 11 and 12 in the code below show how this is done.

```
class Rectangle:
  def __init__(self, x1, y1, x2, y2): # class constructor
    if x1 < x2 and y1 > y2:
      self.x1 = x1 # class variable
      self.y1 = y1 # class variable
      self.x2 = x2 # class variable
      self.y2 = y2 # class variable
    else:
      print("Incorrect coordinates of the rectangle!")

  def __repr__(self):
    return(str(self.x1) + ', ' + str(self.y1) + ', ' + str(self.x2) + ', '+str(self.y2))

# test your code
r = Rectangle (2, 7, 8, 4)
print (r)
```

Now that you know the basic concepts of classes, let’s move on to another concept in object-oriented programming - inheritance.