## 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:

Using __str__
Using __repr__
```

## 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)

\[ \begin{align*} \begin{align*}
```

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\_\_ #

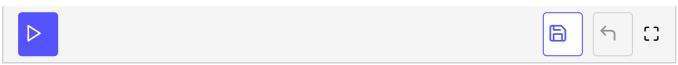
**Lines 11 and 12** in the code below show how this is done.

```
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.