Unit 8

Lesson 2 - Built-In Methods and Operator Overloading

Learning Targets

- I can override the repr and str method to define how the object will create a string representation of itself.
- I can override the eq method to define equivalence for this specific class.
- I can override mathematical operators to redefine the way mathematical functions are performed.

Printing Information About Objects

```
12   p1 = Point(3, 4)
13   print(p1)
14
```

```
<__main__.Point object at 0x10255e930>
[Finished in 32ms]
```

Printing Information About Objects - More dunder methods

To print information about an object we can use either the

__str__ OR __repr__

__str__(self): Defines behavior for when str() is called on an instance of your class.

Typically str is for the user and meant to be readable while repr is for the programmer/debugger. I will user str.

```
def __str__(self):
    return f"({self.x},{self.y})"
```

```
p1 = Point(3, 4)
print(p1)
```

```
(3,4)
```

Determining if Objects are Equal - More dunder methods

To determine if objects are equal we use __eq__.

It is up to you to determine what makes two objects equal

What makes two points equal?

Determining if Objects are Equal - More dunder methods

To determine if objects are equal we use __eq__.

It is up to you to determine what makes two objects equal

```
def __eq__(self, point_2):
    return self.x == point_2.x and self.y == point_2.y
```

```
18  p1 = Point(3, 4)
19  p2 = Point(3, 4)
20  print(p1)
21  print(p1 == p2)
22
```

```
(3,4)
True
```

Point class

```
class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y

    def __str__(self):
        return f"({self.x},{self.y})"

    def __eq__(self, point_2):
        return self.x == point_2.x and self.y == point_2.y

    def distance(self, point_2):
        x_dist = abs(point_2.x - self.x)
        y_dist = abs(point_2.y - self.y)
        return math.sqrt(x_dist**2 + y_dist**2)
```

Your turn

- Make a Circle class in the same module as the Point class.
- Our implementation will use 2 data attributes
  - Point object representing the center
  - int object representing the radius

Your turn

- Make a Circle class in the same module as the Point class.
- Our implementation will use 2 data attributes
  - Point object representing the center
  - int object representing the radius
- Add code to the init method to check that the type of center is a Coordinate obj and the type of radius is an int. If either are not these types, raise a ValueError.

Your turn

- Make a Circle class in the same module as the Point class.
- Our implementation will use 2 data attributes
  - Point object representing the center
  - int object representing the radius
- Add code to the init method to check that the type of center is a Coordinate obj and the type of radius is an int. If either are not these types, raise a ValueError.
- Add a __str__ to print the center and radius of the circle, nicely formatted

Your turn

- Make a Circle class in the same module as the Point class.
- Our implementation will use 2 data attributes
  - Point object representing the center
  - int object representing the radius
- Add code to the init method to check that the type of center is a Coordinate obj and the type of radius is an int. If either are not these types, raise a ValueError.
- Add a __str__ to print the center and radius of the circle, nicely formatted
- Add a method to return the area of circle as a float
- Add a method to return the circumference of a circle as a float

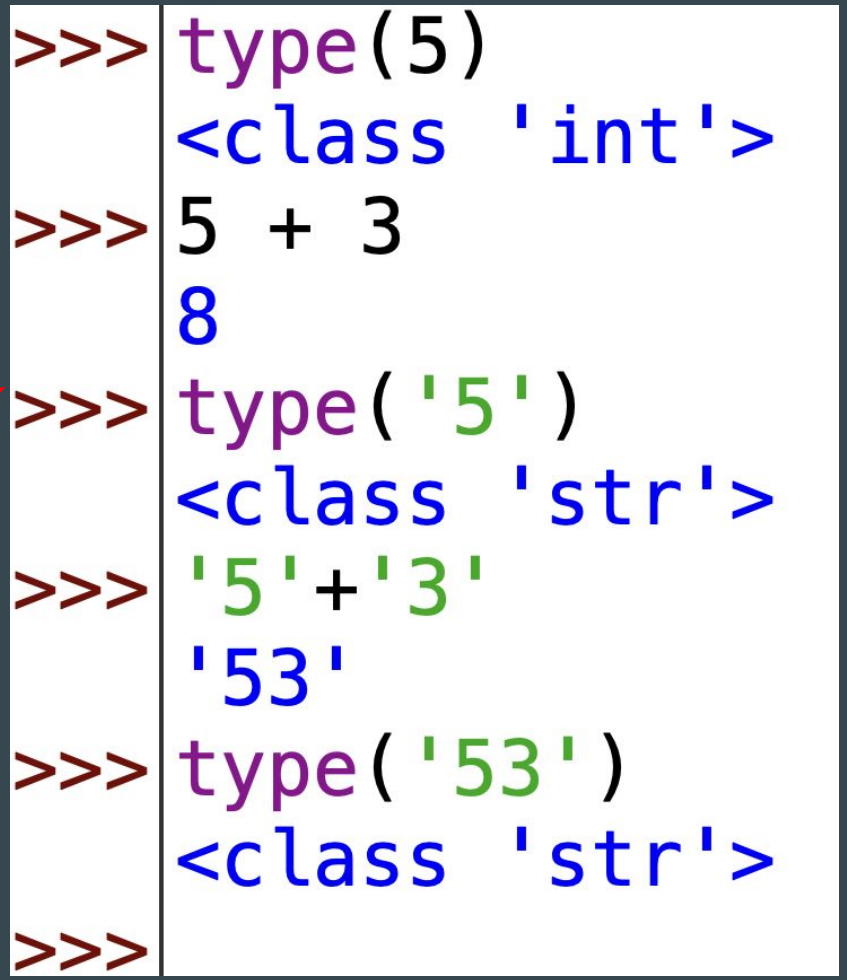
Your turn

- Make a Circle class in the same module as the Point class.
- Our implementation will use 2 data attributes
  - Point object representing the center
  - int object representing the radius
- Add code to the init method to check that the type of center is a Coordinate obj and the type of radius is an int. If either are not these types, raise a ValueError.
- Add a __str__ to print the center and radius of the circle, nicely formatted
- Add a method to return the area of circle as a float
- Add a method to return the circumference of a circle as a float
- Add a __eq__ to determine if two circles are equal

Operator Overriding

You have actually seen this before (not just __eq__ and __str__)

The + operator performed differently depending upon the class that the objects belonged to



Operator Overriding

Built-In Method Name	Operator
__add__()	+ (add)
__sub__()	- (minus)
__mul__()	* (multiply)
__div__()	/ (divide)
__mod__()	% (remainder/modulo)
__pow__()	** (power)
__lt__()	< (less than)
__gt__()	> (greater than)
__le__()	<= (less than or equal to)
__ge__()	>= (greater than or equal to)
__eq__()	== (equality check)
__ne__()	!= (inequality check)

```
18  p1 = Point(3, 4)
19  p2 = Point(1, 5)
20  print(p1 + p2)  × File "/Users/brandon/Documents/RandomPython/s
21
Traceback (most recent call last):
  File "/Users/brandon/Documents/RandomPython/sandbox.py", line 20
    print(p1 + p2)
          ~~~^~~~
TypeError: unsupported operand type(s) for +: 'Point' and 'Point'
```

Operator Overriding - adding two points

This is tricky. The result of adding two points is a third point. How do we do that?

Operator Overriding - adding two points

This is tricky. The result of adding two points is a third point. How do we do that?

```
def __add__(self, point_2):
    return Point(self.x + point_2.x, self.y + point_2.y)
```

```
21  p1 = Point(3, 4)
22  p2 = Point(1, 5)
23  print(p1 + p2)
```

```
(4,9)
```