Python Operator Overloading

Updated on Jan 07, 2020

You have already seen you can use `+` operator for adding numbers and at the same time to concatenate strings. It is possible because `+` operator is overloaded by both `int` class and `str` class. The operators are actually methods defined in respective classes. Defining methods for operators is known as operator overloading. For e.g: To use `+` operator with custom objects you need to define a method called `__add__` .

Let's take an example to understand better

```
1  import math
2
3  class Circle:
4
5      def __init__(self, radius):
6          self.__radius = radius
7
8      def setRadius(self, radius):
9          self.__radius = radius
10
11     def getRadius(self):
12         return self.__radius
13
14     def area(self):
15         return math.pi * self.__radius ** 2
16
17     def __add__(self, another_circle):
18         return Circle( self.__radius + another_circle.__radius )
```

```
        return Circle( self.__radius + another_circle.__radius )

c1 = Circle(4)
print(c1.getRadius())

c2 = Circle(5)
print(c2.getRadius())

c3 = c1 + c2 # This became possible because we have overloaded + operator by adding
print(c3.getRadius())
```

Expected Output:

```
4
5
9
```

```
import math

class Circle:

    def __init__(self, radius):
        self.__radius = radius

    def setRadius(self, radius):
        self.__radius = radius

    def getRadius(self):
        return self.__radius

```

Submit

Output Input

In the above example we have added `__add__()` method which allows use to use the `+` operator to add two circle objects. Inside the `__add__()` method we are creating a new object and returning it to the caller.

Python has many other special methods like `__add__()`, see the list below.

Operator	Function	Method Description
`+`	`__add__(self, other)`	Addition
`*`	`__mul__(self, other)`	Multiplication
`-`	`__sub__(self, other)`	Subtraction
`%`	`__mod__(self, other)`	Remainder
`/`	`__truediv__(self, other)`	Division
`<`	`__lt__(self, other)`	Less than
`<=`	`__le__(self, other)`, Less than or equal to	
`==`	`__eq__(self, other)`, Equal to	
`!=`	`__ne__(self, other)`, Not equal to	
`>`	`__gt__(self, other)`, Greater than	

`>=`, `__ge__(self, other)`, Greater than or equal to `[index]`, `__getitem__(self, index)`, Index operator `in`, `__contains__(self, value)`, Check membership `len`, `__len__(self)`, The number of elements `str`, `__str__(self)`, The string representation

Program below is using some of the above mentioned functions to overload operators.

```
import math

class Circle:

    def __init__(self, radius):
        self.__radius = radius

    def setRadius(self, radius):
        self.__radius = radius

    def getRadius(self):
        return self.__radius

    def area(self):
        return math.pi * self.__radius ** 2

    def __add__(self, another_circle):
        return Circle( self.__radius + another_circle.__radius )

    def __gt__(self, another_circle):
        return self.__radius > another_circle.__radius

    def __lt__(self, another_circle):
        return self.__radius < another_circle.__radius

    def __str__(self):
        return "Circle with radius " + str(self.__radius)

c1 = Circle(4)
print(c1.getRadius())

c2 = Circle(5)
print(c2.getRadius())

c3 = c1 + c2
print(c3.getRadius())

print( c3 > c2) # Became possible because we have added __gt__ method

print( c1 < c2) # Became possible because we have added __lt__ method

print(c3) # Became possible because we have added __str__ method
```

Expected Output:

```
4
5
9
True
True
Circle with radius 9
```

```
import math

class Circle:

    def __init__(self, radius):
        self.__radius = radius

    def setRadius(self, radius):
        self.__radius = radius

    def getRadius(self):
        return self.__radius

    def area(self):
```

Submit

Output Input

Next lesson is inheritance and polymorphism (/python-inheritance-and-polymorphism/).

Other Tutorials (Sponsors)

This site generously supported by DataCamp (https://www.datacamp.com/?utm_source=thepythonguru&utm_campaign=thepythonguru_tutorials). DataCamp offers online interactive Python Tutorials (https://www.datacamp.com/courses/?

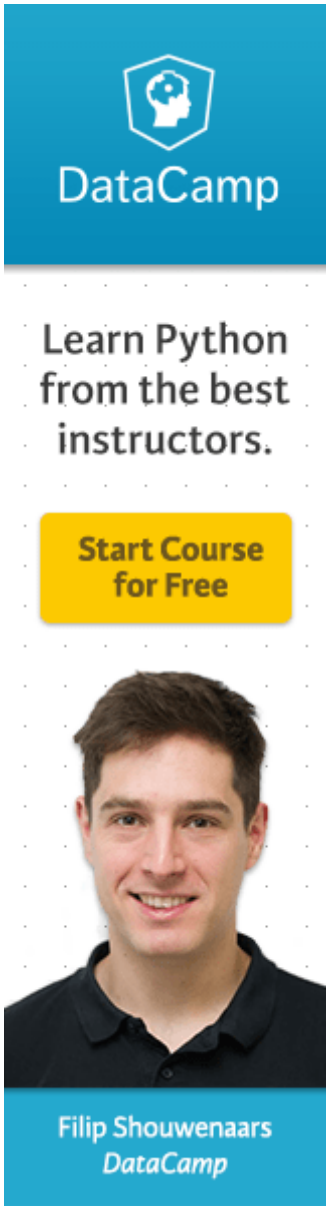
utm_source=thepythonguru&utm_campaign=thepythonguru_tutorials) for Data Science. Join over a million other learners and get started learning Python for data science today!

← Python Object and Classes (/python-object-and-classes/)

Python inheritance and polymorphism → (/python-inheritance-and-polymorphism/)

View Comments

Search here



(https://www.datacamp.com/)

Recent Posts

- # Comparing Python and Node.Js: Which Is Best for Your Project? (/comparing-python-and-nodejs-which-is-best-for-your-project/)
- Windows 10 for a Python User: Tips for Optimizing Performance (/windows-10-for-a-python-user-tips-for-optimizing-performance/)
- What Skills Do You Need to Succeed as a Python Dev in 2020? (/what-skills-do-you-need-to-succeed-as-a-python-dev-in-2020/)
- 9 Reasons Why You Should Use CICD (/9-reasons-why-you-should-use-cicd/)
- How To Make Money If You Have Python Skills (/how-to-make-money-if-you-have-python-skills/)