Learning Quiz 26: Overloaded Operators

Due Nov 13 at 1pm **Points** 5 **Questions** 2 **Available** until Dec 4 at 11:59pm **Time Limit** None
Allowed Attempts Unlimited

Instructions

Prior to completing this quiz, be sure to read:

- Sections 8.4: Overloaded Operators (p. 256-264)

Please also go over Practice Problems 8.6 and 8.7 in the textbook (solution at the end of the chapter) before attempting this quiz.

This quiz was created for learning purposes. You may attempt this quiz as many times as you would like. The highest score *prior to the deadline* will count towards the final course grade. No late submissions will be accepted.

Take the Quiz Again

Attempt History

	Attempt	Time	Score
LATEST	Attempt 1	10 minutes	4.92 out of 5

Score for this attempt: **4.92** out of 5
Submitted Nov 11 at 11:10am
This attempt took 10 minutes.

Question 1

0.92 / 1 pts

The '+' operator can be used to add numbers and strings. While these additions make sense, they functionally do different tasks. For numbers, the '+' operator mathematically adds the values. For strings, the '+' operator concatenates the two strings. We can use these operators to perform specific tasks in our own user-defined classes.

For example, we might use == to equate whether two cars are equal to each other. Note that **self** refers to the current vehicle while **other** refers to the comparison vehicle. (The '\' below indicates that the code continues in the next line)

```
class Vehicle:
    'Represents a drive-able vehicle'

    def __init__(self, color, make, model, wheels = 4, year = 2000):
        'initializes the vehicle, the color, number of wheels, make, and model'
        self.color = color
        self.wheels = wheels
        self.make = make
        self.model = model
        self.year = year
        self.fuel = 0

    def __eq__(self,other):
        'returns true if two vehicles have the same make, model, year, wheels, and color'
        return self.color == other.color and self.wheels == other.wheels and \
               self.make == other.make and self.model == other.model and \
               self.year == other.year

hisCar = Vehicle('green', 'Honda', 'Accord')
herCar = Vehicle('green', 'Honda', 'Accord')
print(hisCar == herCar)
```

Refer to Table 8.3 in the textbook. Match the operator with the method name

Correct!	+	__add__
Correct!	-	__sub__
Correct!	*	__mul__
Correct!	/	__truediv__
Correct!	//	__floordiv__
Correct!	%	__mod__
Correct!	==	__eq__
Correct!	!=	__ne__
Correct!	>	__gt__
Correct!	>=	__ge__
You Answered	<	__le__
	Correct Answer	__lt__
Correct!	<=	__le__

Other Incorrect Match Options:

- __div__

Question 2

4 / 4 pts

Suppose we would like to be able to compare Vehicles based on their year. We would like to be able to use ==, !=, <=, >=, <, and > for this comparison. Again, **self** refers to the current vehicle while **other** refers to the comparison vehicle.

Note, after answering this question, you will not receive a score immediately as it is manually scored.

```
class Vehicle:
    'Represents a drive-able vehicle'

    def __init__(self, color, make, model, wheels = 4, year = 2000):
        'initializes the vehicle, the color, number of wheels, make, and model'
        self.color = color
        self.wheels = wheels
        self.make = make
        self.model = model
        self.year = year
        self.fuel = 0

    def __eq__(self,other):
        'returns true if two vehicles have the same year'
        return self.year == other.year
```

Complete the class for the other five operators. Be sure to use comments where needed/appropriate.

Your Answer:

```
def __ne__(self, other):
    'returns true if two vehicles don't have the same year '
    return self.year != other.year

def __le__(self, other):
    'returns true if vehicle year is less then OR equal to the the year another vehicle '
    return self.year <= other.year

def __ge__(self, other):
    'returns true if vehicle year is greater then OR equal to the year of another vehicle '
    return self.year >= other.year

def __lt__(self, other):
    'returns true if vehicle year is less then the year of another vehicle '
    return self.year < other.year

def __gt__(self, other):
    'returns true if vehicle year is greater then the year of another vehicle '
    return self.year > other.year
```

Hint, here's a skeleton:

```
class Vehicle:
    'Represents a drive-able vehicle'

    def __init__(self, color, make, model, wheels = 4, year = 2000):
        'initializes the vehicle, the color, number of wheels, make, and model'
        self.color = color
        self.wheels = wheels
        self.make = make
        self.model = model
        self.year = year
        self.fuel = 0

    def __eq__(self,other):
        'returns true if two vehicles have the same year'
        return self.year == other.year

    def __ne__(self,other):

    def __gt__(self,other):

    def __ge__(self,other):

    def __lt__(self,other):

    def __le__(self,other):
```

Quiz Score: **4.92** out of 5