Learning Quiz 25: Python Classes

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

Instructions

Prior to completing this quiz, be sure to read:

- Sections 8.1-8.2: Defining a New Python Class and Examples of User-Defined Classes (p. 240-251)

Please also go over Practice Problems 8.1-8.4 in the textbook (solutions 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	13 minutes	4.13 out of 5

Score for this attempt: **4.13** out of 5
Submitted Nov 11 at 10:07am
This attempt took 13 minutes.

Question 1

1 / 1 pts

Classes in Python are used to keep related things together. They also serve as a blueprint for creating objects. Suppose we have an "idea" of a vehicle. Vehicles typically have a color, number of wheels, make, and model. In the code below, we create a class Vehicle, which has a color, number of wheels, make, and model. Vehicle also comes with functions setcolor(), getcolor(), getmake(), and getmodel().

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

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

    def getcolor(self):
        'returns the color of the vehicle'
        return self.color

    def setcolor(self, color):
        'sets the vehicle color'
        self.color = color

    def getmake(self):
        'returns the make of the vehicle'
        return self.make

    def getmodel(self):
        'returns the model of the vehicle'
        return self.model
```

The first argument (in this case, "self") typically refers to the object invoking the method. Although the first argument can be named anything, Python developers commonly use the name self for the object that the method is invoked on.

Following our Vehicle class creation, we can create new Vehicle objects:

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

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

    def getcolor(self):
        'returns the color of the vehicle'
        return self.color

    def setcolor(self, color):
        'sets the vehicle color'
        self.color = color     ## we need to refer to self.color to actually set the color

    def getmake(self):
        'returns the make of the vehicle'
        return self.make

    def getmodel(self):
        'returns the model of the vehicle'
        return self.model

firstCar = Vehicle('silver',4,'Toyota','Corolla')
newCar = Vehicle('black',4,'Honda','Accord')

print(firstCar.getcolor())
firstCar.setcolor('gray')
print(firstCar.getcolor())
```

When we actually create the Vehicle and use the Vehicle functions, we do not need to give the function a "self" argument. Using the same firstCar and newCar that we have created, which of the following lines of code will set the color of our newCar to the same color as our firstCar?

- ○ newCar.setcolor() = firstCar.getcolor()
- Correct! [✓ newCar.setcolor(firstCar.getcolor())]
- ○ newCar.getcolor() = firstCar.getcolor()
- ○ newCar.getcolor(firstCar.setcolor())
- ○ newCar.setcolor() = firstCar.setcolor()
- ○ firstCar.getcolor(newCar, firstCar.getcolor())

Question 2

0.33 / 1 pts

Suppose you want to make some arguments optional, and that you have a default setting if the user does not enter the number of arguments. You can create a default value in the arguments of the __init__() function:

```
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

    def getcolor(self):
        'returns the color of the vehicle'
        return self.color

    def setcolor(self, color):
        'sets the vehicle color'
        self.color = color

    def getmake(self):
        'returns the make of the vehicle'
        return self.make

    def getmodel(self):
        'returns the model of the vehicle'
        return self.model

firstCar = Vehicle('silver','Toyota','Corolla')
newCar = Vehicle('black','Honda','Accord', year = 2019)
newBike = Vehicle('red','Ducati','Panigale', 2, 2019)
```

In the __init__() function, we moved the optional arguments (wheels and year) to the end. The required arguments color, make, and model MUST come first. Thereafter, we set a default value of 4 for wheels and 2000 for year.

Our object firstCar uses these default values.

Our second object newCar uses the default value 4 for wheels, but we added the argument 'year = 2019' to change the year. We need to state 'year =' or else Python might think we are referring to the number of wheels.

Our third object newBike adjusts the default number of wheels to 2 and the year to 2019. Note, we do not need to specify 'wheels = 2, year = 2019' if we are using both optional arguments *in order*.

Suppose another programmer expects our Vehicle class to have an optional argument color and will be using the following line of code to create his own Vehicle object called myCar:

```
myCar = Vehicle('Nissan', 'Sentra', year = 2020, color = 'gold')
```

Which of the following lines of code can be used for the def __init__() line?

☐ def __init__(self, make, color = 'black', model, wheels = 4, year = 2000):

Correct!

☑ def __init__(self, make, model, wheels = 4, year = 2000, color = 'black'):

Correct Answer

☐ def __init__(self, make, model, wheels = 4, color = 'black', year = 2000):

☐ def __init__(self, make, model, wheels, color = 'black', year = 2000):

Correct Answer

☐ def __init__(self, make, model, color = 'black', wheels = 4, year = 2000):

☐ def __init__(self, color = 'black', make, model, wheels = 4, year = 2000):

☐ def __init__(self, make, model, color = 'black', wheels, year = 2000):

Question 3

1 / 1 pts

In the code below, we create a new variable called 'fuel' and a new function called 'refuel()':

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

    def __init__(self, color, make, model = 'Accord', 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 getcolor(self):
        'returns the color of the vehicle'
        return self.color

    def setcolor(self, color):
        'sets the vehicle color'
        self.color = color

    def getmake(self):
        'returns the make of the vehicle'
        return self.make

    def getmodel(self):
        'returns the model of the vehicle'
        return self.model

    def refuel(self, gallons):
        'returns the model of the vehicle'
        self.fuel += gallons

firstCar = Vehicle('silver','Toyota','Corolla')
newCar = Vehicle('black','Honda','Accord', year = 2019)
newBike = Vehicle('red','Ducati','Panigale', year = 2019, wheels = 2)
firstCar.refuel(5)
```

Notice that fuel does not need to be part of the argument. Depending on the properties of your object, this may be something you'd like access later, after object creation.

Which of the following functions should we add to our class so that we can return the number of gallons available in our vehicle?

☐
```
def getFuel():
    'returns the available fuel in the vehicle'
    return fuel
```

Correct!

✓
```
def getFuel(self):
    'returns the available fuel in the vehicle'
    return self.fuel
```

☐
```
def getFuel(self):
    'returns the available fuel in the vehicle'
    return fuel
```

☐
```
def getFuel():
    'returns the available fuel in the vehicle'
    return self.fuel
```

Question 4

0.8 / 1 pts

Suppose we now have the following Vehicle class and an object called firstCar:

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

    def __init__(self, color, make, model = 'Accord', 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 getcolor(self):
        'returns the color of the vehicle'
        return self.color

    def setcolor(self, color):
        'sets the vehicle color'
        self.color = color

    def getmake(self):
        'returns the make of the vehicle'
        return self.make

    def getmodel(self):
        'returns the model of the vehicle'
        return self.model

    def refuel(self, gallons):
        'returns the model of the vehicle'
        self.fuel += gallons

firstCar = Vehicle('silver','Toyota','Corolla')
```

Which of the following functions are available to us?

Correct! ☑ firstCar.getcolor()

☐ firstCar.setyear(1999)

☐ firstCar.setfuel(2)

Correct! ☑ firstCar.getmodel()

Correct! ☑ firstCar.setcolor('black')

☐ firstCar.getfuel()

Correct Answer ☐ firstCar.getmake()

☐ firstCar.getyear()

☐ firstCar.getwheels()

Correct!

☑ firstCar.refuel(2)

Question 5

1 / 1 pts

It's important to make it a habit to add a comment in the line immediately after defining a class or function. That way, many lines later, if someone need to look-up properties of your class, they can use the help function to read your notes.

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

    def __init__(self, color, make, model = 'Accord', 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 getcolor(self):
        'returns the color of the vehicle'
        return self.color

    def setcolor(self, color):
        'sets the vehicle color'
        self.color = color

    def getmake(self):
        'returns the make of the vehicle'
        return self.make

    def getmodel(self):
        'returns the model of the vehicle'
        return self.model

    def refuel(self, gallons):
        'returns the model of the vehicle'
        self.fuel += gallons

help(Vehicle)
```

Where would be an appropriate location to add comments about the new getwheels() function below?

```
    def getwheels(self):
        return self.wheels
```

○ After return self.wheels

○ Before def getwheels(self)

Correct!

● Between def getwheels(self): and return self.wheels

Quiz Score: **4.13** out of 5