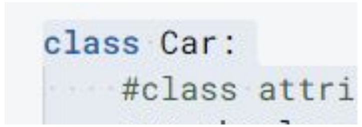
Starting to Secure Our Objects

Private Attributes

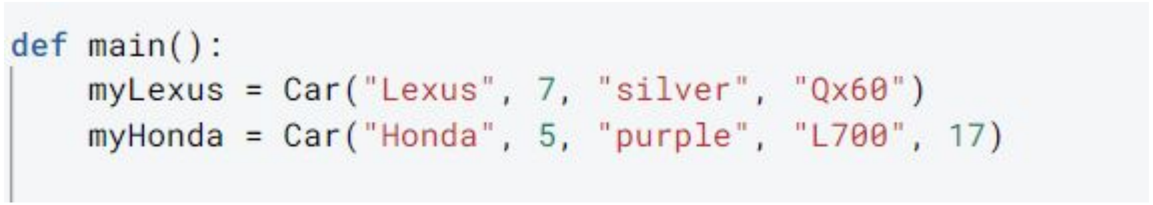
Using Objects as Parameters (point from last lesson)

- Given a class Car:



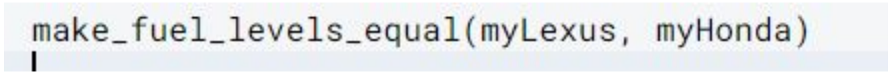
```
class Car:
    #class attri
```

- And 2 instances of a Car:



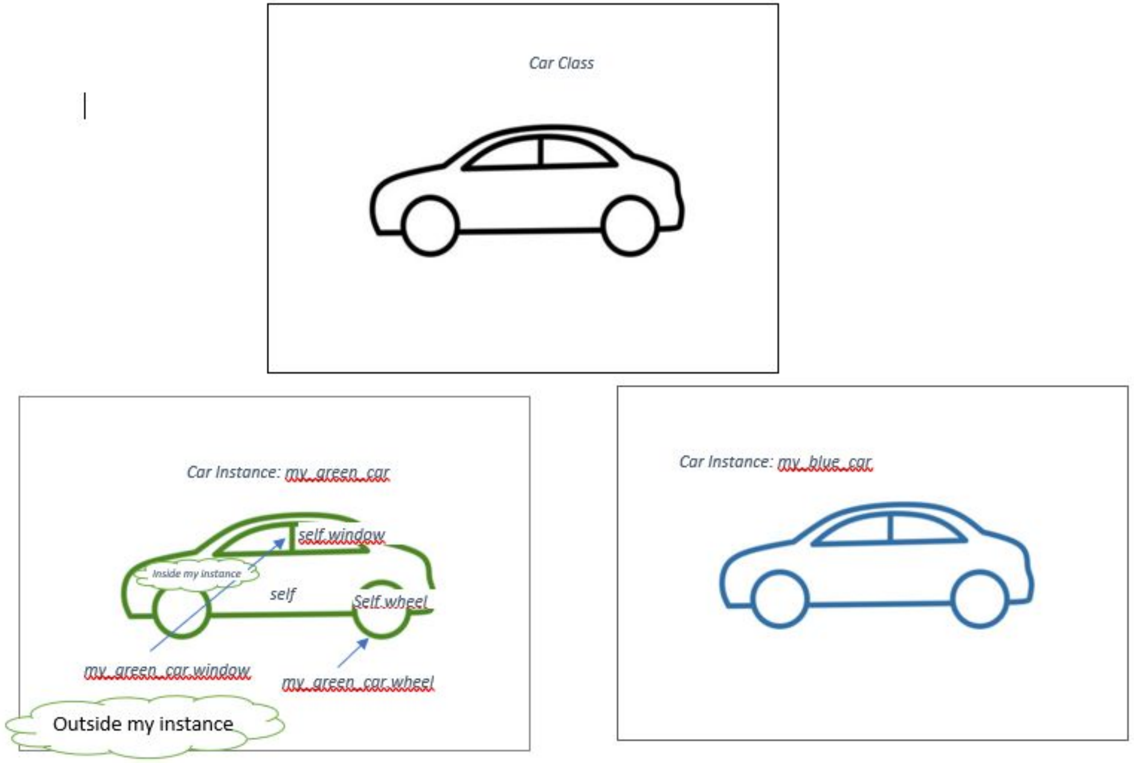
```
def main():
    myLexus = Car("Lexus", 7, "silver", "Qx60")
    myHonda = Car("Honda", 5, "purple", "L700", 17)
```

- I can pass the instances to another method as a parameter:

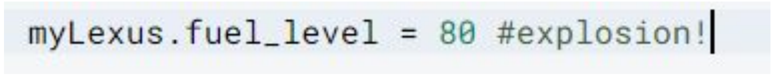


```
make_fuel_levels_equal(myLexus, myHonda)
```

Illustration of the concept of “self” (review)



Illustrating The Need For Security



```
myLexus.fuel_level = 80 #explosion!
```

- Right now, we coded a dangerous car.
- We gave access to the attribute fuel_level to anyone OUTSIDE of the Car class. That means they can modify fuel_level without checking it against the gas_tank_size!!
- How do we make the fuel_level inaccessible to anyone outside of the Car class?

Private Attributes

- Welcome to private attributes. If we prefix our attributes with a double underscore, they will not be accessible outside of the Car class.
- (Reminder: an attribute is a variable that belongs to the class. You will find an attribute either inside the __init__() method (instance attribute) or immediately after the class definition (class attribute). Any additional variables defined in methods of the class are not attributes. They are anyway inaccessible outside the class and there is no need to prefix them with underscores.)

Illustrate in the Car class

- Code together

Assignment

- Update your credit card statement class that all the attributes will be private. (Make sure you update *all references* to the attributes. Ie. anywhere that it is used)
- Ensure that you are not using those attributes outside of your class. If you are, you will see an error when you run it since they are now private to the class.
  - (Side point: If you decide that you still need to use them outside of your class, write a method in the Car class that will enable you to use the variable outside of the car class)
- Improve your code:
  - The __str__() method should not cause anything to happen. In other words, the __str__() method should not be reading the file. The __str()__ method should simply return information about your object.
  - Bonus: make sure the __str__() method will work regardless of when it is called. Meaning, the __str__() method should return a sensible response whether the file was read or was not read yet.
- Bonus: Include some aspect of validation in the usage of the CreditCardStatement class. For example, if you wrote a method that relies on another method happening first, and you did not call one from inside the other, do some sort of check that will make sure the code doesn't continue without all the information that it needs. (For a hint, on this, see next slide. But try without peeking first.)

- Bonus expounded:For example, iIf you wrote a method conclude_statement() that relies on read_file(), and you did not call read_file() inside conclude_statement, add validation to conclude_statement() that will make sure you have all the data that you need to work with. If it’s missing data, make sure it doesn’t crash and returns a response that makes sense.