CMPSC-122: Intermediate Programming
Spring 2018

Lab #7

Due Date: 03/02/2018, 11:59PM

Instructions:

- The work in this lab must be completed alone.
- If you need guidance, attend to your recitation class.
- Read the "Submitting assignments to Vocareum" file for instructions on how to submit this lab
- Do not change the function names or given code on your script
- The file name must be LAB7.py (incorrect name files will get a 0 score)
- You are responsible for testing your code. Use `python -i LAB7.py` in your terminal (or command prompt) to provide input to your functions. Test with as many data as you feel comfortable
- Each function must return the output (Do not use print in your final submission)
- Do not include test code outside any function in the upload. Remove all your testing code before uploading your file. If you are using input() to insert values in your functions and print to see the values, remove them.

Exercise 1 [4 pts]. Without using the Python interpreter, what is printed after the following statements are executed? Insert your answers in the function *answers()* provided in the starter code.

a)

```
class FirstOp:

    def __init__(self,num):
        self.num = num

    def modify(self):
        self.num *= 2
        return self.num


class SecondOp(FirstOp):

    def __init__(self, num):
        FirstOp.__init__(self, num)

    def change(self):
        self.num *= 3
        return self.num

x = SecondOp(6)
print(x.num)

x.modify()
print(x.num)

x.change()
print(x.num)
```

b)

```
      class Person:
          def __init__(self, name):
              self.name = name

          def getName(self):
              return self.name

          def isEmployee(self):
              return False

      class Employee(Person):
          def __init__(self, name, eid):
              Person.__init__(self, name)
              self.empID = eid

          def isEmployee(self):
              return True

          def getID(self):
              return self.empID


      alex= Employee("Alexander Hudson", "123456")
      print(alex.isEmployee())
```

EXAMPLE:

```
          question_1_a = "7,-5,12"

          question_1_b = "False"
```

Exercise 2 [6 pts]. In class, we learned how inheritance allows us to create a general class first then later extend it to more specialized classes.

Imagine you run a car dealership. You sell several types of vehicles, including cars and trucks. To set apart from the competition, you determine the price of a vehicle in you lot as $4,000×number of wheels a vehicle has. You also buy vehicles. You offer a flat rate - 10% of the miles driven on the vehicle. The flat rates are: for cars $7,500, for trucks: $9,000

Your current sales system has a *Car* and a *Truck* class. Identify common attributes and methods among those classes, and using the OOP principle of inheritance, create the parent class *Vehicle*. The classes *Car* and *Truck* will inherit everything from *Vehicle*. (code available in your starter code)

The functionality of the program should not change. Do not change any method name. Remember, a parent class contains all attributes and methods common to the child classes.

```
class Car:
    def __init__(self, wheels, miles, make, model, year, gear, color):
        self.wheels = wheels
        self.miles = miles
        self.make = make
        self.model = model
        self.year = year
        self.gear = gear
        self.color = color
        self.sold_on = False
        self.flat_rate = 7500

    def sell(self):
        if self.sold_on == True:
            print("This item has been sold")
        else:
            self.sold_on = True

    def sale_price(self):
        if self.sold_on:
            return 0.0  # Vehicle already sold
        return 4000 * self.wheels

    def purchase_price(self):
        return self.flat_rate - (.10 * self.miles)

    def getDescription(self):
        sale_price=self.sale_price()
        return "{} {} {} - {}, {} miles >>> ${}".format(self.make, self.model,
self.year, self.color, self.miles, sale_price)

class Truck:
    def __init__(self, seats, wheels, miles, make, model, year):
        self.seats=seats
        self.wheels = wheels
        self.miles = miles
        self.make = make
        self.model = model
        self.year = year
        self.sold_on = False
        self.flat_rate = 9000


    def sell(self):
        if self.sold_on == True:
            print("This item has been sold")
        else:
            self.sold_on = True

    def sale_price(self):
        if self.sold_on:
            return 0.0  # Vehicle already sold
        return 4000 * self.wheels

    def purchase_price(self):
        return self.flat_rate - (.10 * self.miles)

    def getDescription(self):
        sale_price=self.sale_price()
        return "{} {} {}, {} miles - {} seats >>> ${}".format(self.make,
self.model, self.year,self.miles, self.seats, sale_price)
```

EXAMPLE:

```
>>> x=Car(4,12000,'Mazda','CX5',2017,'Automatic','Red')
>>> y=Truck(7,8,15000,'Ford','Engine',1987)
>>> x.sale_price()
16000
>>> x.purchase_price()
6300.0
>>> x.getDescription()
'Mazda CX5 2017 - Red, 12000 miles >>> $16000'
>>> x.sell()
>>> x.sale_price()
0.0
>>> x.sell()
This item has been sold
>>> y.sale_price()
32000
>>> y.purchase_price()
7500.0
>>> y.getDescription()
'Ford Engine 1987, 15000 miles - 7 seats >>> $32000'
>>> y.sell()
>>> y.sell()
This item has been sold
>>> y.sale_price()
0.0
```