Python for data scientists SP1 2021/2022

Assignment 3

There are two problems in this assignment.
You can either submit:

- One Python file for Problem 1 (solutionA3_P1.py)
- One Python file for Problem 2 (solutionA3_P2.py)
- A PDF file for the explanation part (if any).

Or a Jupyter Notebook with clearly marked solutions for which problems/which tasks and any explanation.
Note: If you add any details or make any assumptions, please clearly describe in your submission.

Problem 1

Define a class Circle() where the objects constructed from this class will be circles of various sizes. Define the following methods for this class:

- constructor method which has one parameter, called radius
- circumference() method which returns the circumference of the circle
- surface_area() method which returns the area of the circle.

Next, define a class Cylinder() which is derived from the class Circle. The constructor of this new class will include two parameters, i.e., radius and height. Then define a surface_area() method which returns the surface area of a closed cylinder (a cylinder that has top and bottom). After that, define volume() method which returns the volume of the cylinder. Both surface_area and volume methods should make use of one or more method(s) that Cylinder inherited from its parent's.

Finally, define a class Cone(), which is derived from the class Cylinder. The constructor of this class includes two parameters, radius and height. This class has its own volume() method which returns the volume of the cone. However, the volume method is built based on the volume method it inherits from its parent. This class also has its own surface_area() method which returns the total surface area of the cone (the base surface area + the lateral surface area).

Example of use:

```
if __name__ == '__main__':
    cyl = Cylinder(4, 9)
    print ('The surface of the cylinder : ', cyl.surface_area())
    print ('The volume of the cylinder : ', cyl.volume())
    co = Cone(4, 9)
    print ('The volume of the cone : ', co.volume())
    print ('Surface area of the cone: ', co.surface_area())
```

```
Output:
The surface of the cylinder :  326.7264
The volume of the cylinder :  452.3904
The volume of the cone :  150.7968
Surface area of the cone:  174.03028668049055
```

Problem 2

This task is about a very simplistic car domain (not necessarily representing reality). Your task is to model the domain specifications given here in an object oriented fashion using Python. The main purpose is to illustrate multiple inheritance and how to implement that. Checking validity of input is an optional task.

Here is the domain specification. All instances of cars have *id* and *build year* as their base attributes. We considered three categories of cars, i.e., petrol cars, electric cars, and plug-in hybrid vehicles (PHEV).

The petrol cars have the following extra attributes:

- *tank_capacity* that tells the maximum capacity of the petrol tank (in l). Let's say petrol cars can have tank capacity 40-70 l.
- *fuel_consumption* that tells the amount of fuel used per unit distance, stated in l/100km (e.g., 5.5 means that 5.5 litre of petrol for 100 km driven).
- *fuel_status* that tells how full is the tank (i.e., fuel equals to 0.5 means that the tank is only half-full). This attribute *fuel_status* is initialy set to 0 at instance initiation. The value of attribute *fuel_status* can be modified via two functions, i.e., *fill_fuel_to()* and *reduce_fuel_to()*.

The petrol cars have a function to calculate the estimated range remaining (the distance that can be driven with the remaining fuel in the tank). Here, we just calculate estimated ideal range. In reality, real range is affected by many things like weather, total weight of the vehicle and its content, and driving style, therefore the formula for calculating the estimated range could change. As a result of this, you are asked to make sure that the function for estimating range cannot be accessed directly from an outside application.

Similarly to the petrol cars, the electric cars have the following attributes in addition to the base attributes of cars:

- *battery_capacity* that tells the maximum capacity of battery (in kWh). Let's say electric cars can have battery size 17-100 kWh.
- *battery_consumption* that tells the amount of electricity used per unit distance, stated in kWh/km (e.g., 0.2 means that the car needs 0.2 kwh to drive 1 km distance).
- *battery_status* that tells the percentage of the battery left (i.e., attribute *battery_status* equals to 10 means that 10% of battery power left). The battery power is initially set to 0% at instance initiation. The value of attribute *battery_status* is modified via two functions, i.e., *charge_battery_to()*, and *reduce_battery_to()*.

The electric cars also have a function to calculate the estimated range remaining. Same request as in the petrol cars was made here (i.e., enough with simple range estimation and that the function must not be accessible directly from an outside application).

The plug-in hybrid cars (PHEV) have petrol tank and battery and can run on either of petrol or battery. These cars have inherited the base attributes of cars, as well as *tank_capacity*, *fuel_consumption*, and *fuel_status* from petrol cars, and *battery_capacity*, *battery_consumption*, and *battery_status* from electric cars.

The PHEV cars have their own function to calculate estimated ideal range; this function must not be accessible directly from an outside application. Here, the total ideal range comes from both the estimated ideal range from the petrol tank and the electric power in the battery. The *battery_capacity* in PHEV cars is much smaller compared to the electric cars. Let's say the battery capacity for PHEV is 8-9.5 kWh. Let's say the battery comsumption is about the same with the electric cars, i.e., between 0.15-0.25 kWh/km. We assume that the petrol tank size in the PHEV cars is about the same as in the petrol cars. The fuel consumption when PHEV is running on petrol is assumed to be comparable with that of petrol cars.

Example of use:

```
if __name__ =='__main__':
    c = Petrol_car('c1', 1995, 40, 5.5)
    c.fill_fuel_to(1)
    c.get_range()
    c.reduce_fuel_to(0.4)
    c.get_range()
    e = Electric_car('e1', 2015, 70, 0.2)
    e.charge_battery_to(90)
    e.get_range()
    e.reduce_battery_to(20)
    e.get_range()
    h = PHEV('h1', 2016, 40, 5.5, 9.4, 0.2)
    h.fill_fuel_to(0.5)
    h.charge_battery_to(50)
    h.get_range()
    h.reduce_battery_to(0)
    h.get_range()
```

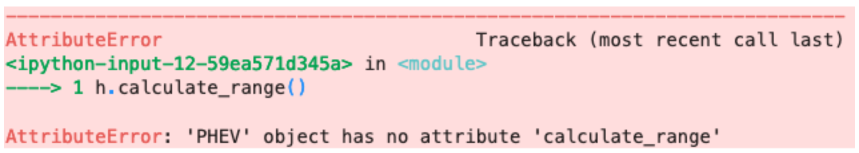
```
output:

Fuel status of c1 is 1 of full tank now!
The remaining range of c1: 727 km.
Fuel status of c1 is 0.4 of full tank now!
The remaining range of c1: 290 km.
Battery power of e1 is 90% now!
The remaining range of e1: 315 km.
Battery power of e1 is 20% now!
The remaining range of e1: 70 km.
Fuel status of h1 is 0.5 of full tank now!
Battery power of h1 is 50% now!
The remaining range of h1: 387 km.
Battery power of h1 is 0% now!
The remaining range of h1: 363 km.
```

You should get an AttributeError message if we try to access *calculate_range()* from outside (like the example below).

```
h.calculate_range()
```

```
output:
```



```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-12-59ea571d345a> in <module>
----> 1 h.calculate_range()

AttributeError: 'PHEV' object has no attribute 'calculate_range'
```