CS 115 - Introduction to Programming in Python

Lab Guide 07

Lab Objectives: Inheritance

Notes:

a) Upload your solutions as **a single .zip file** to the Lab07 assignment for your section on Moodle. You must use the following naming convention: Lab07_Surname_FirstName.zip where Surname is your family name and FirstName is your first name.
b) Solutions sent through email will not be accepted.
c) You should only use functionality covered in CS115 in your solution.
d) Include a docstring for your functions.

You will write a program for a cab owner to store and display information about their taxi cabs.

1. Create a class, `Cab`, with the following data attributes and methods. Note all data attributes and class variables should be private.

Class Cab:

Data Members:

- `typeOfCab`: private attribute that stores the type of cab, hatch back or sedan.
- `kms`: private attribute that stores the number of kilometers travelled.

Methods:

- `__init__()`: initializes the data members to the values passed as parameters.
- Get methods for kms, type.
- `__lt__()`: compares Cab objects by number of kms.
- `__eq__()`: returns True if two Cabs have the same number of kms and are the same type, False if not.
- `__repr__()`: returns a string representation of a Cab object.
  See sample run for details.

2. Create a subclass, `Sedan`, by extending the superclass `Cab`, with the following data attributes and methods. Note all data attributes should be private.

   Data Members:
   - `price_per_km`: private class attribute (not instance) that stores the price per km ($2).

   Methods:
   - `__init__()`: initializes the inherited data members to the values passed as parameters.
   - `calculate_fare()`: calculates and returns the cab fare using the price per km and the number of kms.

3. Create a subclass, ***Hatchback***, by extending the superclass *Cab***,** with the following data attributes and methods. Note all data attributes should be private.

   Data Members:
   - `price_per_km`: private class attribute (not instance) that stores the price per km ($1.5).

   Methods:
   - `__init__()`: initializes the inherited data members to the values passed as parameters.
   - `calculate_fare()`: calculates and returns the cab fare using the price per km and the number of kms.

4. Write a script `CabApp` with the following functions:

- `find_equal()`: Takes a list of Cabs, and a **Cab object** as parameters. The function should find and return the number of Cabs in the list that are equal to the Cab passed as a parameter

- `read_file()`: Takes a filename as a parameter. Assume each line of the file contains the type of Cab and the number of kilometers, separated by a colon. A sample file is shown below, however the data may change. Using data in the file, return a list of Cab objects (Sedan or Hatchback).

The script should do the following:

- Creates a list containing Cabs using data in the file, *`cabs.txt`*.
- Store the kms of all Sedans in one list and the kms of all Hatchbacks in another.
- Display the total kilometers for each Cab type (Sedans and Hatchbacks).
- Display the total fare for all Cabs.
- Sort the list of Cabs by kms and display the sorted list.
- Find and display the number of Sedans with 200 kms. Use the *`find_equal`* function.

Sample Run:

```
---Kilometers driven for each cab---
Hatchback: 1700 kilometers
Sedan: 1660 kilometers

Total number of kilometers driven by all Cabs: 3360
Total Fare Earned from all cabs (in dollars): 5870.0

Sorted Cabs:
[Hatchback    10
, Sedan 10
, Hatchback   20
, Hatchback   20
, Sedan 20
, Sedan 20
, Hatchback   20
, Sedan 20
, Hatchback   30
, Sedan 50
, Hatchback   100
, Sedan 100
, Sedan 100
, Sedan 200
, Sedan 200
, Hatchback   200
, Sedan 200
, Sedan 200
, Hatchback   200
, Hatchback   200
, Hatchback   300
, Hatchback   300
, Hatchback   300
, Sedan 540
]

There are 4 Sedan cabs with 200 kms.
```