Prac06

car.py modifications

*"""CP1404/CP5632 Practical - Car class example."""*class Car:  
 *"""Represent a Car object."""* def \_\_init\_\_(self, fuel=0, name = **'no name'**):  
 *"""Initialise a Car instance.  
 fuel: float, one unit of fuel drives one kilometre  
 """* self.fuel = fuel  
 self.odometer = 0  
 self.name = name  
  
 def \_\_str\_\_(self):  
 return **'{}, fuel={}, odometer={}'**.format(self.name, self.fuel, self.odometer)  
  
 def add\_fuel(self, amount):  
 *"""Add amount to the car's fuel."""* self.fuel += amount  
  
 def drive(self, distance):  
 *"""Drive the car a given distance.  
 Drive given distance if car has enough fuel  
 or drive until fuel runs out return the distance actually driven.  
 """* if distance > self.fuel:  
 distance = self.fuel  
 self.fuel = 0  
 else:  
 self.fuel -= distance  
 self.odometer += distance  
 return distance

used\_cars.py modifications (cars have names)

*"""CP1404/CP5632 Practical - Client code to use the Car class."""  
# Note that the import has a folder (module) in it.*from car import Car  
  
  
def main():  
 *"""Demo test code to show how to use car class."""* name = input(**'Enter cars name: '**)  
 my\_car = Car(180, name)  
 my\_car.drive(30)  
 print(**"fuel ="**, my\_car.fuel)  
 print(**"odo ="**, my\_car.odometer)  
 print(my\_car)  
  
 print(**"{} {}, {}"**.format(name, my\_car.fuel, my\_car.odometer))  
 print(**"Car {self.fuel}, {self.odometer}"**.format(self=my\_car))  
  
*# 1. Create a new Car object called "limo" that is initialised with 100 units of fuel.* limo=Car(100)  
*# 2. Add 20 more units of fuel to this new car object using the add method.* limo.add\_fuel(20)  
*# 3. Print the amount of fuel in the car.* print(**'fuel ='**, limo.fuel)  
*# 4. Attempt to drive the car 115km using the drive method.* limo.drive(115)  
*# 5. Print the car's odometer reading.* print(**'odometer ='**, limo.odometer)  
  
main()

programming\_language.py (the class)

class ProgrammingLanguage:  
  
 def \_\_init\_\_(self, field, typing, reflection, year):  
 self.field=field  
 self.typing=typing  
 self.reflection=reflection  
 self.year=year  
  
 def is\_dynamic(self):  
 if self.typing==**'Dynamic'**:  
 return True  
 else:  
 return False  
  
 def \_\_str\_\_(self):  
 return **'{}, {} Typing, Reflection={}, First appeared in {}'**.format(self.field, self.typing, self.reflection, self.year)

languages.py (the client program)

from programming\_language import ProgrammingLanguage  
  
def main():  
 ruby = ProgrammingLanguage(**"Ruby"**, **"Dynamic"**, True, 1995)  
 python = ProgrammingLanguage(**"Python"**, **"Dynamic"**, True, 1991)  
 visual\_basic = ProgrammingLanguage(**"Visual Basic"**, **"Static"**, False, 1991)  
 print(ruby)  
 print(python)  
 print(visual\_basic,**'**\n**'**)  
  
 list=[ruby, python, visual\_basic]  
 print(**'The dynamically typed languages are:'**)  
 for language in list:  
 if language.is\_dynamic():  
 print(language.field)  
  
main()

guitar.py (the class)

class Guitar:  
  
 def \_\_init\_\_(self, name=**''**, year=0, cost=0):  
 self.name=name  
 self.year=year  
 self.cost=cost  
  
 def \_\_str\_\_(self):  
 return **'{} ({}) : ${:.2f}'**.format(self.name, self.year, self.cost)  
  
 def get\_age(self):  
 age=2020-self.year  
 return age  
  
 def is\_vintage(self):  
 if self.get\_age()>=50:  
 return True  
 else:  
 return False

guitar\_test.py (the tests)

from guitar import Guitar  
  
def test():  
 guitar1=Guitar(**'Gibson L-5 CES'**,1922,16035.40)  
 guitar2=Guitar(**'Another Guitar'**,2013,10000.00)  
 print(**'{} get\_age() - Expected {}. Got {}'**.format(guitar1.name, 98, guitar1.get\_age()))  
 print(**'{} get\_age() - Expected {}. Got {}'**.format(guitar2.name, 7, guitar1.get\_age()))  
 print(**'{} is\_vintage() - Expected {}. Got {}'**.format(guitar1.name, True, guitar1.is\_vintage()))  
 print(**'{} is\_vintage() - Expected {}. Got {}'**.format(guitar2.name, False, guitar2.is\_vintage()))  
  
test()

guitars.py (the client program)

from guitar import Guitar  
  
def main():  
 guitars=[]  
 print(**'My guitars!'**)  
 name=str(input(**'Name: '**))  
 while name != **''**:  
 year=int(input(**'Year: '**))  
 cost=float(input(**'Cost: $'**))  
 print(**'{} ({}) : ${} added.'**.format(name, year, cost))  
 guitar=Guitar(name, year, cost)  
 guitars.append(guitar)  
 name = str(input(**'Name: '**))  
  
 guitars.append(Guitar(**"Gibson L-5 CES"**, 1922, 16035.40))  
 guitars.append(Guitar(**"Line 6 JTV-59"**, 2010, 1512.9))  
  
 print(**'These are my guitars:'**)  
 i=0  
 for guitar in guitars:  
 if guitar.is\_vintage():  
 vintage\_string=**' (vintage)'** else:  
 vintage\_string =**''** print(**'Guitar {}: {:>20} ({:4}), worth ${:10,.2f}{}'**.format(i + 1, guitar.name, guitar.year, guitar.cost, vintage\_string))  
 i+=1  
  
main()