class Car:

def \_\_init\_\_(self, make, model, year, price):

self.\_\_make = make

self.\_\_model = model

self.\_\_year = year

self.\_\_price = price

def get\_make(self):

return self.\_\_make

def get\_model(self):

return self.\_\_model

def get\_year(self):

return self.\_\_year

def get\_price(self):

return self.\_\_price

class Sedan(Car):

def \_\_init\_\_(self, make, model, year, price, num\_doors):

super().\_\_init\_\_(make, model, year, price)

self.\_\_num\_doors = num\_doors

def get\_num\_doors(self):

return self.\_\_num\_doors

class SUV(Car):

def \_\_init\_\_(self, make, model, year, price, seating\_capacity):

super().\_\_init\_\_(make, model, year, price)

self.\_\_seating\_capacity = seating\_capacity

def get\_seating\_capacity(self):

return self.\_\_seating\_capacity

class SportsCar(Car):

def \_\_init\_\_(self, make, model, year, price, top\_speed):

super().\_\_init\_\_(make, model, year, price)

self.\_\_top\_speed = top\_speed

def get\_top\_speed(self):

return self.\_\_top\_speed

Encapsulation

class Car:

def \_\_init\_\_(self, make, model, year, price):

self.\_\_make = make

self.\_\_model = model

self.\_\_year = year

self.\_\_price = price

def get\_make(self):

return self.\_\_make

def get\_model(self):

return self.\_\_model

def get\_year(self):

return self.\_\_year

def get\_price(self):

return self.\_\_price

def set\_make(self, make):

self.\_\_make = make

def set\_model(self, model):

self.\_\_model = model

def set\_year(self, year):

self.\_\_year = year

def set\_price(self, price):

self.\_\_price = price

polymorphism

class Car:

def \_\_init\_\_(self, make, model, year, price):

self.make = make

self.model = model

self.year = year

self.price = price

def \_\_str\_\_(self):

return f"{self.make} {self.model} ({self.year}) - ${self.price}"

def print\_info(self, cars):

if isinstance(cars, Car):

print(str(cars))

elif isinstance(cars, list):

for car in cars:

print(str(car))