**Python advanced backend 6**

This OOP exercise covers questions on the following ****topics****:

* Class and Object creation
* Instance variables and Methods, and Class level attributes
* Model systems with class inheritance i.e., inherit From Other Classes
* Parent Classes and Child Classes
* Extend the functionality of Parent Classes using Child class
* Object checking

[Home](https://pynative.com/) » [Python Exercises](https://pynative.com/python-exercises-with-solutions/) » Python Object-Oriented Programming (OOP) Exercise: Classes and Objects Exercises

# **Python Object-Oriented Programming (OOP) Exercise: Classes and Objects Exercises**

Updated on: December 8, 2021 | [38 Comments](https://pynative.com/python-object-oriented-programming-oop-exercise/" \l "llc_comments)

This Object-Oriented Programming (OOP) exercise aims to help you to learn and practice OOP concepts. All questions are tested on Python 3.

[Python Object-oriented programming (OOP)](https://pynative.com/python/object-oriented-programming/) is based on the concept of “objects,” which can contain data and code: data in the form of instance variables (often known as attributes or properties), and code, in the form method. I.e., Using OOP, we encapsulate related properties and behaviors into individual objects.

****What is included in this Python OOP exercise?****

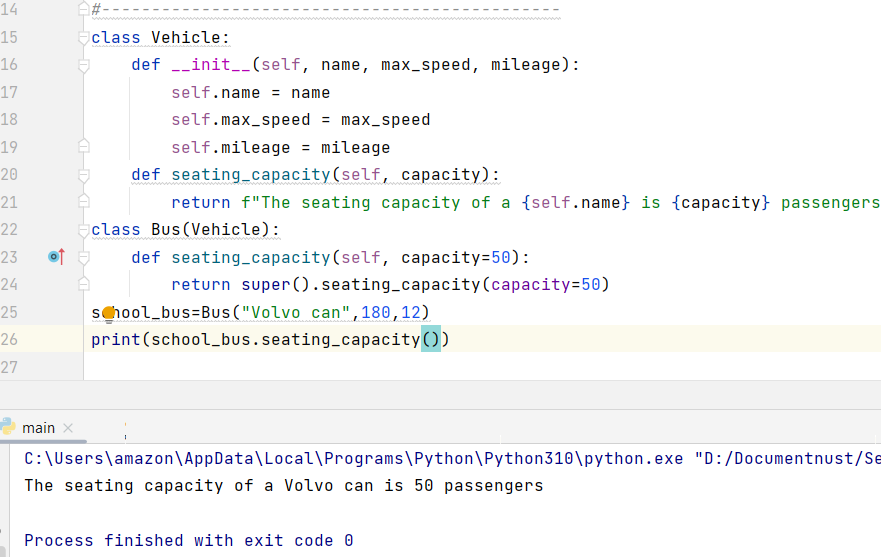
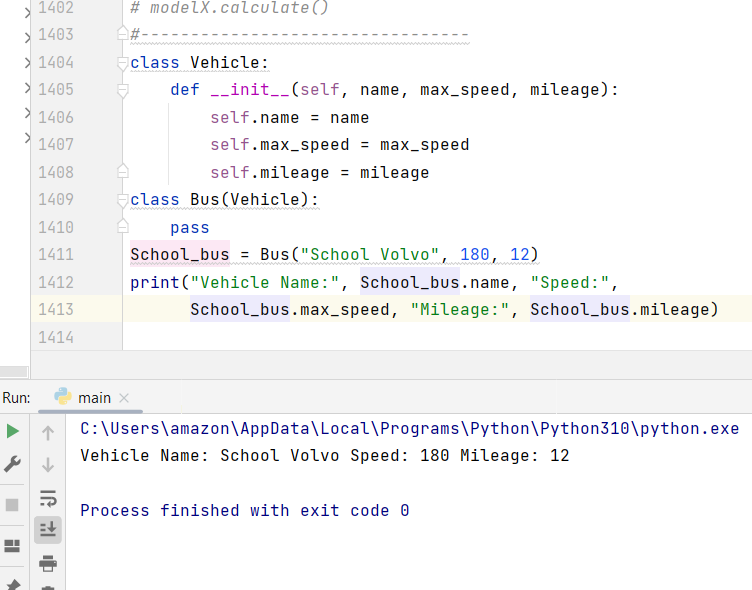
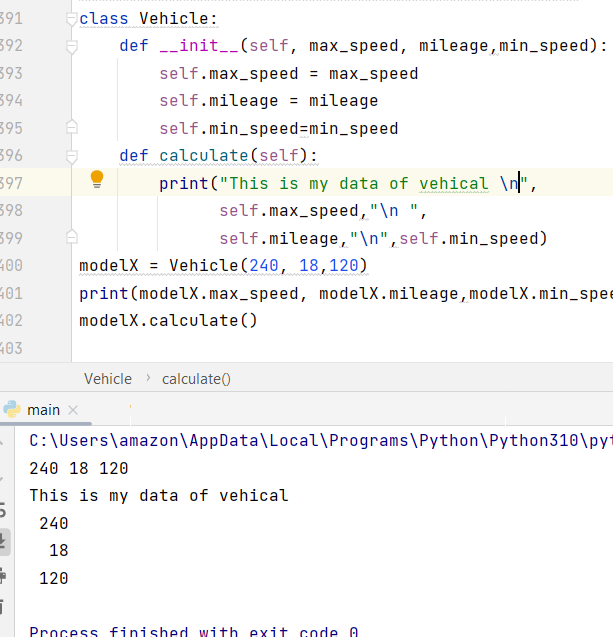
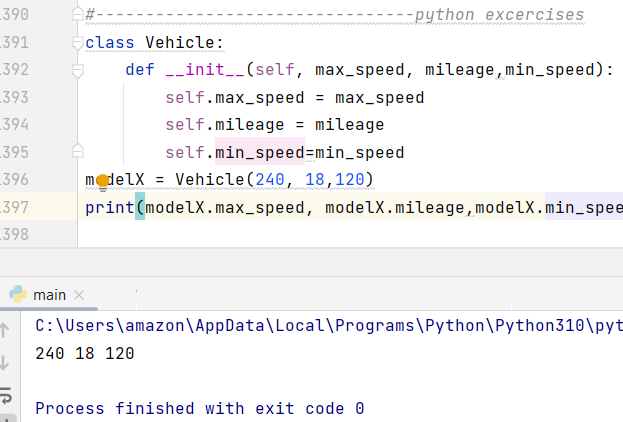
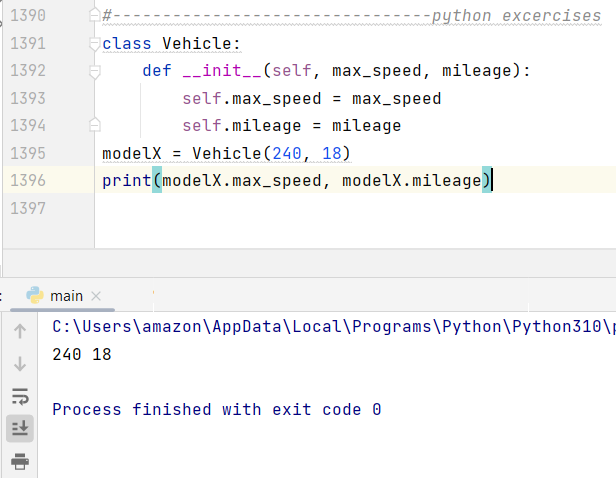
This OOP classes and objects exercise includes 8 different programs, questions, and challenges. All solutions are tested on Python 3.

This OOP exercise covers questions on the following ****topics****:

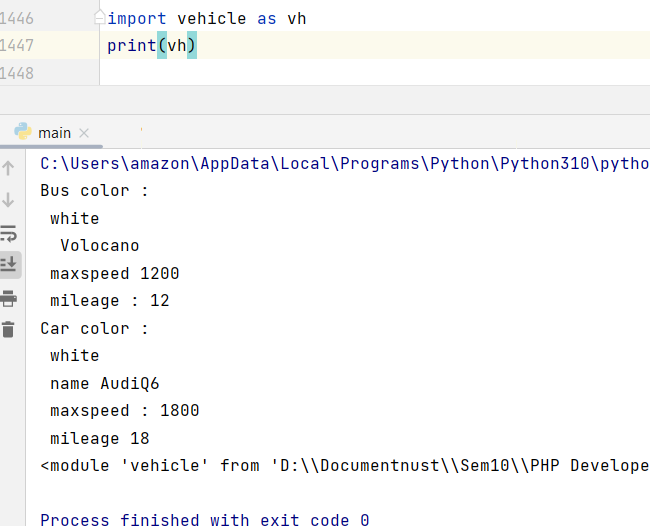
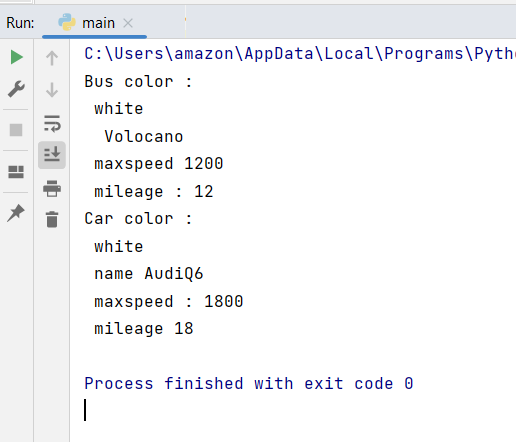
* Class and Object creation
* Instance variables and Methods, and Class level attributes
* Model systems with class inheritance i.e., inherit From Other Classes
* Parent Classes and Child Classes
* Extend the functionality of Parent Classes using Child class
* Object checking

### **OOP Exercise 1: Create a Class with instance attributes**

Write a Python program to create a Vehicle class with max\_speed and mileage instance attributes



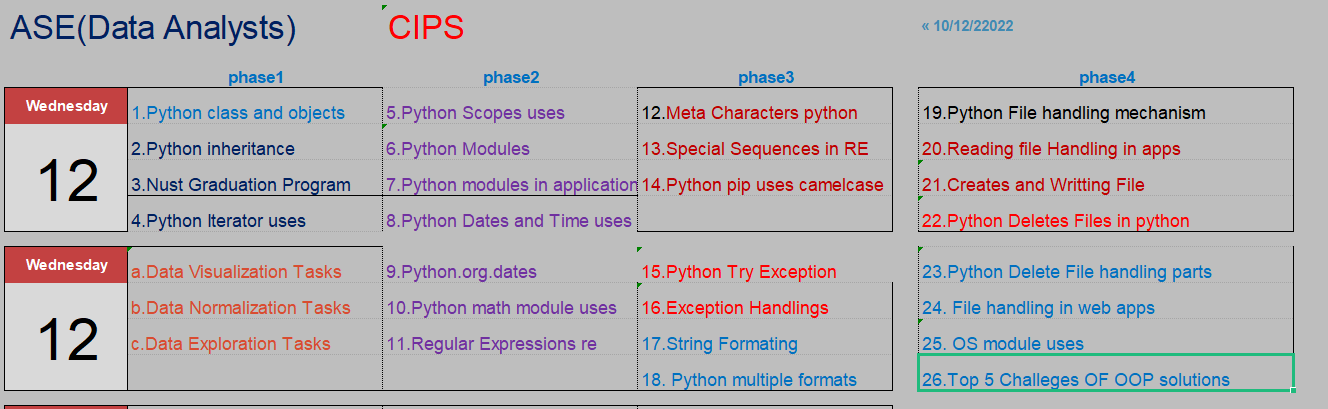
class Vehicle:  
 def \_\_init\_\_(self, name, max\_speed, mileage):  
 self.name = name  
 self.max\_speed = max\_speed  
 self.mileage = mileage  
 def seating\_capacity(self, capacity):  
 return f"The seating capacity of a {self.name} is {capacity} passengers"  
class Bus(Vehicle):  
 def seating\_capacity(self, capacity=50):  
 return super().seating\_capacity(capacity=50)  
school\_bus=Bus("Volvo can",180,12)  
print(school\_bus.seating\_capacity())



**Important links:**

**Code challenge:**

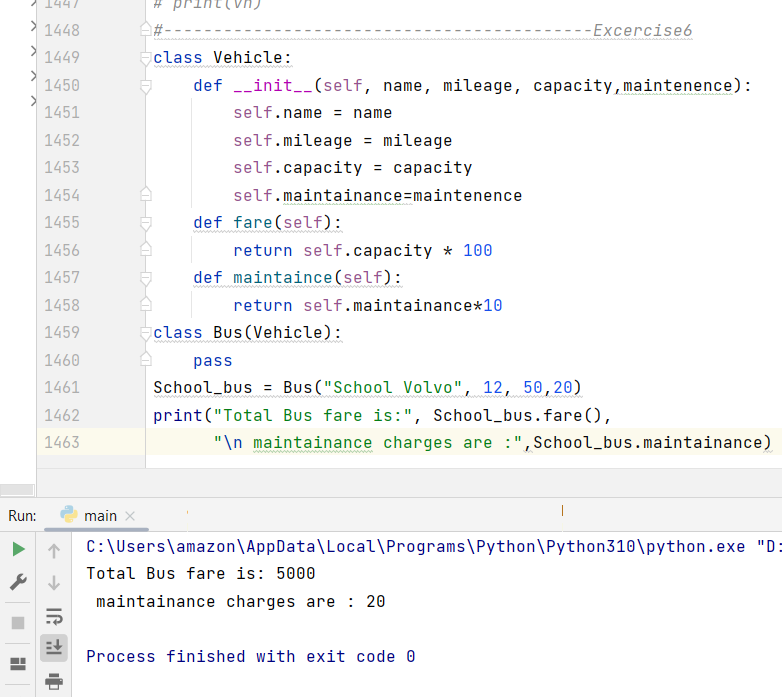
<https://pynative.com/python-object-oriented-programming-oop-exercise/#h-oop-exercise-1-create-a-class-with-instance-attributes>



Create a ****Bus**** child class that inherits from the Vehicle class. The default fare charge of any vehicle is ****seating capacity \* 100****. If Vehicle is ****Bus**** instance, we need to add an extra 10% on full fare as a maintenance charge. So total fare for bus instance will become the ****final amount = total fare + 10% of the total fare.****

 The bus seating capacity is ****50****. so the final fare amount should be****5500.****You need to override the fare() method of a Vehicle class in Bus class.

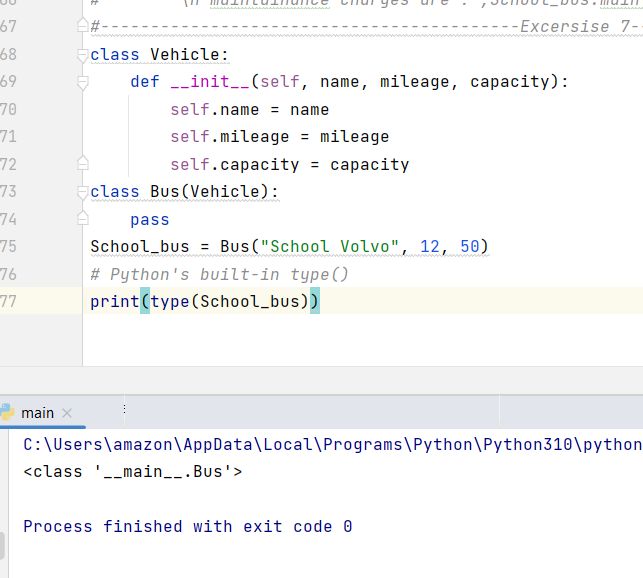
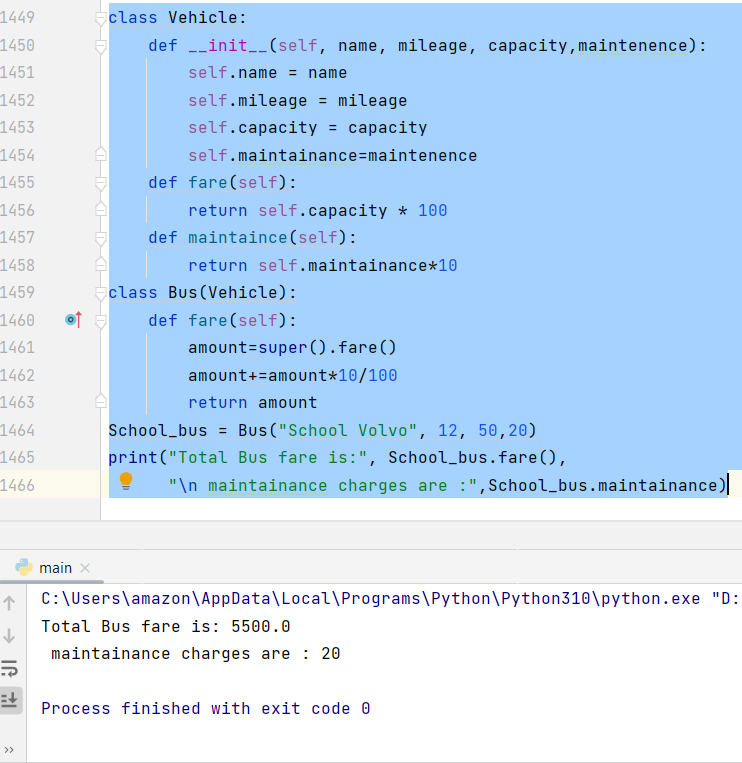
Use the following code for your parent Vehicle class. We need to access the parent class from inside a method of a child class.

.

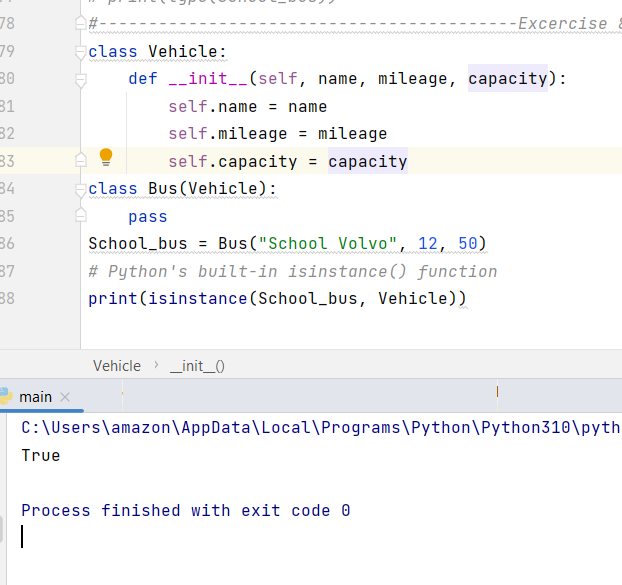
class Vehicle:  
 def \_\_init\_\_(self, name, mileage, capacity,maintenence):  
 self.name = name  
 self.mileage = mileage  
 self.capacity = capacity  
 self.maintainance=maintenence  
 def fare(self):  
 return self.capacity \* 100  
 def maintaince(self):  
 return self.maintainance\*10  
class Bus(Vehicle):  
 pass  
School\_bus = Bus("School Volvo", 12, 50,20)  
print("Total Bus fare is:", School\_bus.fare(),  
 "\n maintainance charges are :",School\_bus.maintainance)

/////////////////////////////////////////////////////////////////////////////////////////////////////

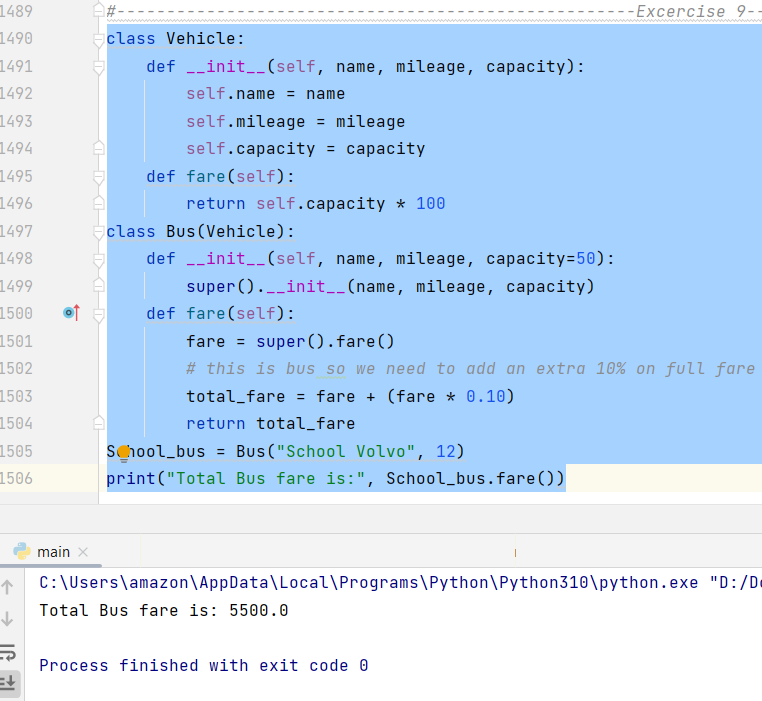
class Vehicle:  
 def \_\_init\_\_(self, name, mileage, capacity,maintenence):  
 self.name = name  
 self.mileage = mileage  
 self.capacity = capacity  
 self.maintainance=maintenence  
 def fare(self):  
 return self.capacity \* 100  
 def maintaince(self):  
 return self.maintainance\*10  
class Bus(Vehicle):  
 def fare(self):  
 amount=super().fare()  
 amount+=amount\*10/100  
 return amount  
School\_bus = Bus("School Volvo", 12, 50,20)  
print("Total Bus fare is:", School\_bus.fare(),  
 "\n maintainance charges are :",School\_bus.maintainance)



### **OOP Exercise 8: Determine if School\_bus is also an instance of the Vehicle class**



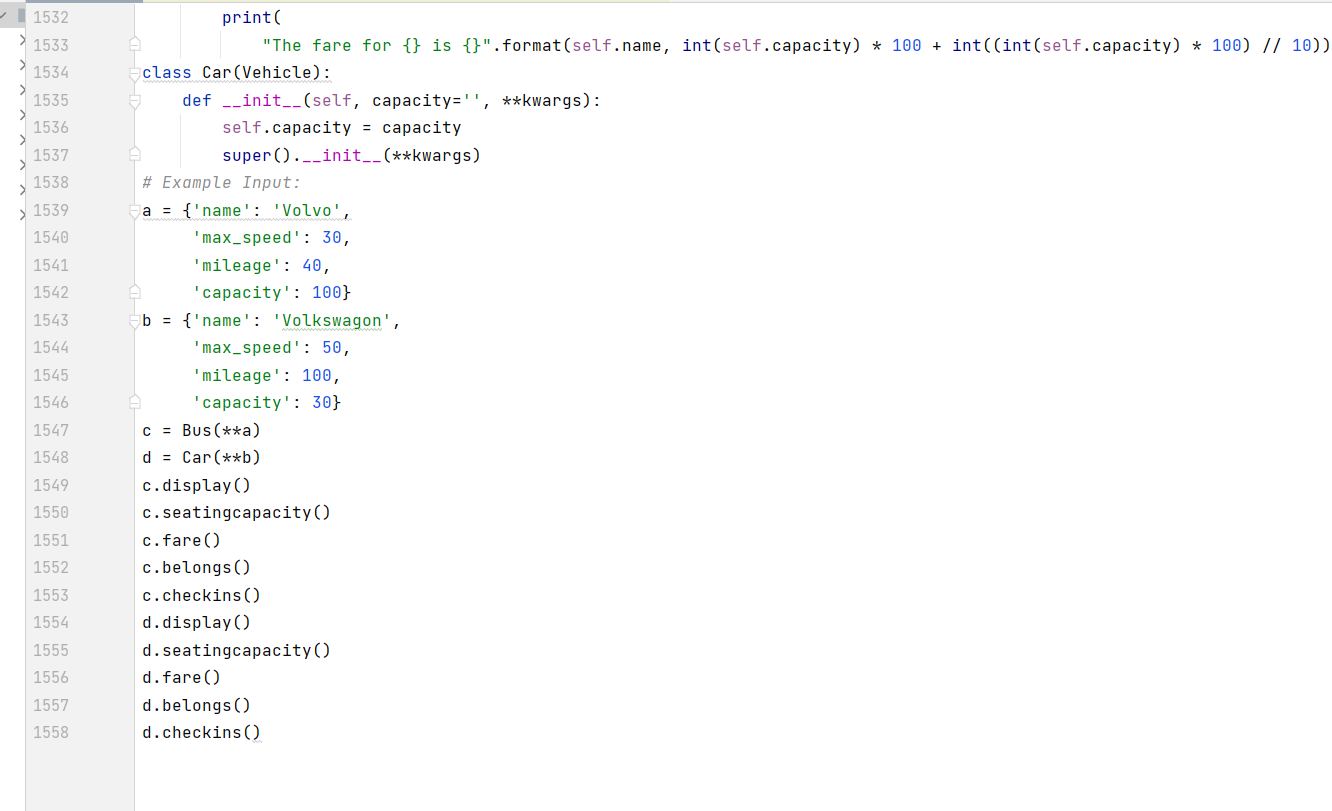
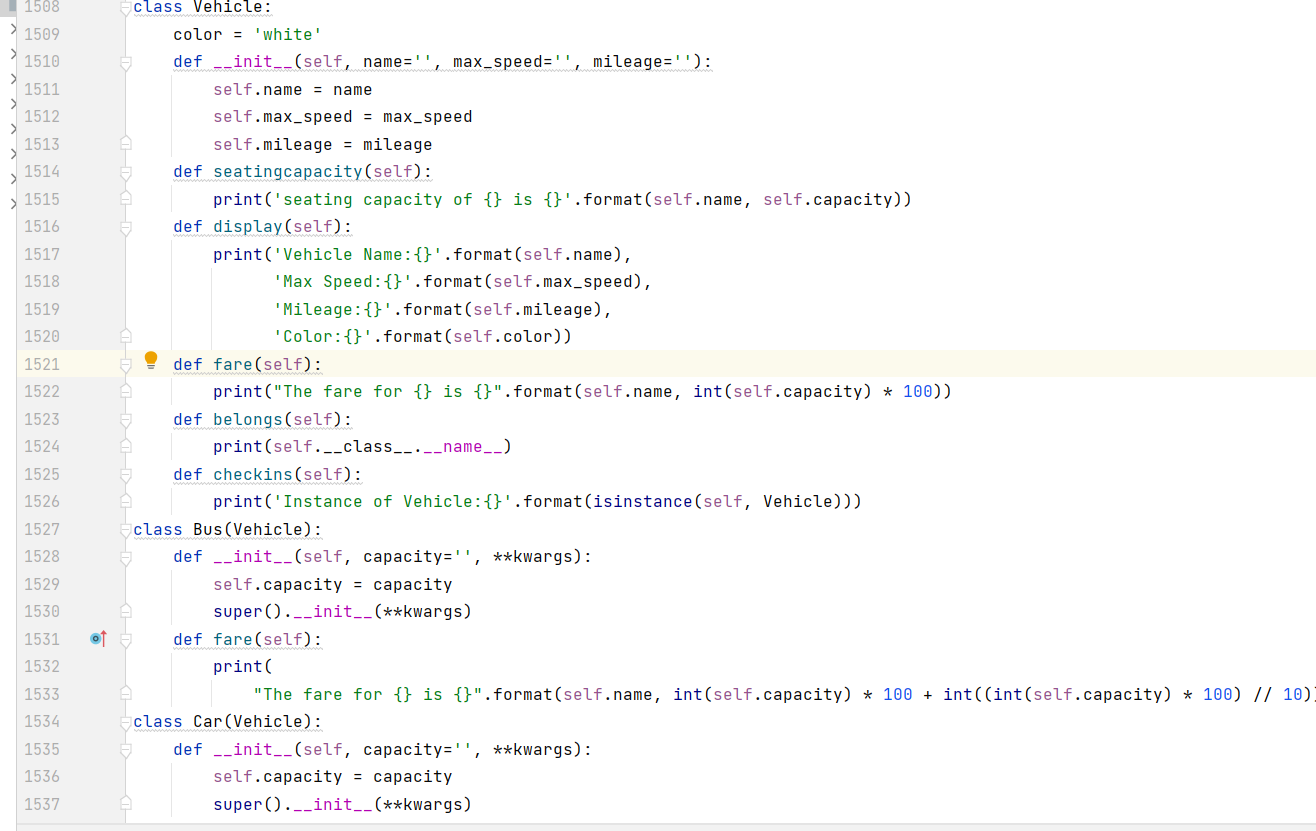
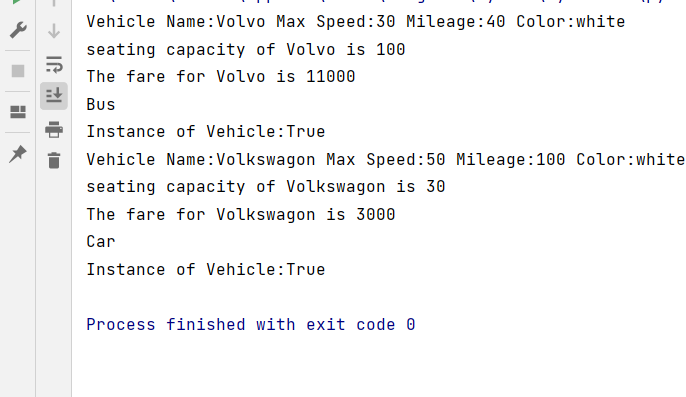
class Vehicle:  
 def \_\_init\_\_(self, name, mileage, capacity):  
 self.name = name  
 self.mileage = mileage  
 self.capacity = capacity  
 def fare(self):  
 return self.capacity \* 100  
class Bus(Vehicle):  
 def \_\_init\_\_(self, name, mileage, capacity=50):  
 super().\_\_init\_\_(name, mileage, capacity)  
 def fare(self):  
 fare = super().fare()  
 *# this is bus so we need to add an extra 10% on full fare as a maintenance charge* total\_fare = fare + (fare \* 0.10)  
 return total\_fare  
School\_bus = Bus("School Volvo", 12)  
print("Total Bus fare is:", School\_bus.fare())



class Vehicle:  
 def \_\_init\_\_(self, name, mileage, capacity):  
 self.name = name  
 self.mileage = mileage  
 self.capacity = capacity  
 def fare(self):  
 return self.capacity \* 100  
class Bus(Vehicle):  
 def \_\_init\_\_(self, name, mileage, capacity=50):  
 super().\_\_init\_\_(name, mileage, capacity)  
 def fare(self):  
 fare = super().fare()  
 *# this is bus so we need to add an extra 10% on full fare as a maintenance charge* total\_fare = fare + (fare \* 0.10)  
 return total\_fare  
School\_bus = Bus("School Volvo", 12)  
print("Total Bus fare is:", School\_bus.fare())

**Exercise 10 OOP Challenge Python advanced**

class Vehicle:  
 color = 'white'  
 def \_\_init\_\_(self, name='', max\_speed='', mileage=''):  
 self.name = name  
 self.max\_speed = max\_speed  
 self.mileage = mileage  
 def seatingcapacity(self):  
 print('seating capacity of {} is {}'.format(self.name, self.capacity))  
 def display(self):  
 print('Vehicle Name:{}'.format(self.name),  
 'Max Speed:{}'.format(self.max\_speed),  
 'Mileage:{}'.format(self.mileage),  
 'Color:{}'.format(self.color))  
 def fare(self):  
 print("The fare for {} is {}".format(self.name, int(self.capacity) \* 100))  
 def belongs(self):  
 print(self.\_\_class\_\_.\_\_name\_\_)  
 def checkins(self):  
 print('Instance of Vehicle:{}'.format(isinstance(self, Vehicle)))  
class Bus(Vehicle):  
 def \_\_init\_\_(self, capacity='', \*\*kwargs):  
 self.capacity = capacity  
 super().\_\_init\_\_(\*\*kwargs)  
 def fare(self):  
 print(  
 "The fare for {} is {}".format(self.name, int(self.capacity) \* 100 + int((int(self.capacity) \* 100) // 10)))  
class Car(Vehicle):  
 def \_\_init\_\_(self, capacity='', \*\*kwargs):  
 self.capacity = capacity  
 super().\_\_init\_\_(\*\*kwargs)  
*# Example Input:*a = {'name': 'Volvo',  
 'max\_speed': 30,  
 'mileage': 40,  
 'capacity': 100}  
b = {'name': 'Volkswagon',  
 'max\_speed': 50,  
 'mileage': 100,  
 'capacity': 30}  
c = Bus(\*\*a)  
d = Car(\*\*b)  
c.display()  
c.seatingcapacity()  
c.fare()  
c.belongs()  
c.checkins()  
d.display()  
d.seatingcapacity()  
d.fare()  
d.belongs()  
d.checkins()



import pyodbc  
import mysql.connector  
conn=pyodbc.connect('Driver={SQL Server};''Server=localhost\SQLEXPRESS;''Database=db2;''Trusted\_connection=yes;')  
cursor=conn.cursor()  
cursor.execute('Select \* from dbo.player')  
for row in cursor:  
 print(row)

////////////////////////////////////////////////////////////////////////////////////

import pyodbc  
import mysql.connector  
conn=pyodbc.connect('Driver={SQL Server};'  
 'Server=localhost\SQLEXPRESS;''Database=db2;'  
 'Trusted\_connection=yes;')  
cursor=conn.cursor()  
cursor.execute('Select \* from dbo.sales2')  
for row in cursor:  
 print(row)

