9th Feb Assignment

February 13, 2023

1 Assignment 9

Q1. Create a vehicle class with an init method having instance variables as name_of_vehicle, max_speed and average_of_vehicle.

[1]:
```
class vehicle:
    def __init__(self,name_of_vehicle,max_speed,average_of_vehicle):
        self.name_of_vehicle=name_of_vehicle
        self.max_speed=max_speed
        self.average_of_vehicle=average_of_vehicle
```

Q2. Create a child class car from the vehicle class created in Que 1, which will inherit the vehicle class. Create a method named seating_capacity which takes capacity as an argument and returns the name of the vehicle and its seating capacity.

[2]:
```
class car(vehicle):
    def seating_capacity(self,capacity):
        self.capacity=capacity
        return self.name_of_vehicle,self.capacity
```

Q3. What is multiple inheritance? Write a python code to demonstrate multiple inheritance.

Ans.Multiple inheritance is a type of inheritance which have 2 parent or base class and one child class. The single child class inherit the property of the base classes.

[3]:
```
#example

#create parent class 1
class parent1:
    def test1(self):
        print('This is parent class 1')

#create parent class 2
class parent2:
    def test2(self):
        print('This is parent class 2')

#create child class
```

```
class child(parent1,parent2):
    def test(self):
        print('This is child class')
```

[4]:
```
#create an object of child class
child_object=child()
```

[5]:
```
child_object.test2()
```

```
This is parent class 2
```

[6]:
```
child_object.test1()
```

```
This is parent class 1
```

[7]:
```
child_object.test()
```

```
This is child class
```

Q4. What are getter and setter in python? Create a class and create a getter and a setter method in this class.

Ans. Getter and setters are used as property decorator to access the private varibale and modify that variable.

[50]:
```
class student:
    def __init__(self,name):
        self.__name=name

    @property
    def student_name_access(self):
        return self.__name

    @student_name_access.setter
    def set_name(self,new_name):
        self.__name=new_name
```

[51]:
```
jp=student('Jp')
```

[53]:
```
jp.student_name_access
```

[53]:
```
'Jp'
```

[55]:
```
jp.set_name='Vj'
```

[56]:
```
jp.student_name_access
```

[56]:
```
'Vj'
```

Q5.What is method overriding in python? Write a python code to demonstrate method overriding.

Ans. Method overriding in python is a property of oops in which a method which is defined in parent class,this method redefines in child class as different implementation.

[62]:
```python
#base class or parent class
class animal:
    def speak(self):
        print('Animal speaks')

#child class of animal
class dog(animal):
    def speak(self):
        print('Dog barks')

#object of dog or child class
d=dog()
```

[63]:
```python
d.speak()
```

```
Dog barks
```