oops assesment

June 9, 2023

```
[ ]: #Q1
```

```
[19]: class vehicle :
          def __init__(self, name_of_vehicle, max_speed, average_of_vechile):
              self. name_of_vehicle = name_of_vehicle
              self. max_speed = max_speed
              self. average_of_vehicle = average_of_vechile
          def return_vehicle_details(self):
              return self.name_of_vehicle,self.max_speed,self.average_of_vehicle
```

```
[20]: my_car = vehicle ('B_m_w',420 ,42)
```

```
[21]: print(my_car.name_of_vehicle)
```

```
B_m_w
```

```
[22]: print(my_car.max_speed)
```

```
420
```

```
[23]: print(my_car.average_of_vehicle)
```

```
42
```

```
[ ]: #Q2
```

```
[42]: class car(vehicle) :
            def __init__ (self, seating_capacity) :
              self.seating_capacity = seating_capacity
              def return_car_vehcile(self):
                  return self.name_of_vehicle , self.seating_capacity
```

```
[43]: my_car_seating = car (5)
```

```
[46]: print(my_car.name_of_vehicle , my_car_seating.seating_capacity)
```

```
B_m_w 5
```

[ ]:
```
#Q3
```

[ ]:
```
'multiple inheritance we can use parent class inside the sub class this sub␣
 ↪class used inside another sub class'
```

[51]:
```
class class1():
    def test_class1(self):
        return 'I am getting attract towoards coding'
```

[52]:
```
class class2(class1):
    def test_class2(self):
        return' oh!! my god help him to get sucess in his life'
```

[53]:
```
class class3(class2):
    def test_class3(self):
        return 'thanks man'

```

[54]:
```
obj_class3=class3()
```

[55]:
```
obj_class3.test_class1()
```

[55]:
```
'I am getting attract towoards coding'
```

[56]:
```
obj_class3.test_class2()
```

[56]:
```
' oh!! my god help him to get sucess in his life'
```

[57]:
```
obj_class3.test_class3()
```

[57]:
```
'thanks man'
```

[ ]:
```
#Q4
```

[ ]:
```
'getter is used to return  the value under the created class inside the object'
'setter is used to modified the value under the created class inside the object'
```

[81]:
```
class person():
    def __int__(self,name):
        self.name=name
    def get_name(self):
        return self.name
    def set_name(self,name):
        self.name = name
```

[90]:
```
person=Person('chirag')
print(person.get_name())
```

```
person.set_name('shashank')
print(person.get_name())
```

```
chirag
shashank
```

[ ]:
```
#Q5
```

[ ]:
```
'method overriding in python is a fetaure which help to create different␣
 ↪implementation of subclass which is already present in the parent class'
```

[96]:
```
class animal:
    def sound (self):
        print('the sound effect of the animal')
class cat(animal):
    def sound (self):
        print ("meow")
class tiger(animal):
    def sound(self):
        print('baaaaaaaa')
```

[97]:
```
cat=cat()
tiger=tiger()
```

[98]:
```
cat.sound()
```

```
meow
```

[99]:
```
tiger.sound()
```

```
baaaaaaaa
```

[ ]:
```

```

[ ]:
```

```