Oops task

June 13, 2023

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):
        return f"the {self.name_of_vehicle} has a seating capacity of␣
↪{capacity}"
```

[3]:
```
car_details = car("alto",110,20)
```

[4]:
```
car_details.name_of_vehicle
```

[4]:
```
'alto'
```

[5]:
```
car_details.seating_capacity(4)
```

[5]:
```
'the alto has a seating capacity of 4'
```

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

ANS. when a class can be derived from more than one base class this type of inheritance is called multiple inheritance. in multiple inheritances all the feature of the base classes are inherited into the derived class .

[6]:
```
class mobile_number:

    def customer_number(self):
        return 9845474554

class name:

    def customer_name(self):
        return "raju"

class email_id:
    def customer_email(self):
        return "raju@gmail.com"

class customer_details(mobile_number, name, email_id):
    pass
```

[7]:
```
customer_details_obj=customer_details()
```

[8]:
```
customer_details_obj.customer_number()
```

[8]:
```
9845474554
```

[9]:
```
customer_details_obj.customer_name()
```

[9]:
```
'raju'
```

[10]:
```
customer_details_obj.customer_email()
```

[10]:
```
'raju@gmail.com'
```

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

ANS. getter- the getter is a method that is used in object oriented programming to access a class's private attributes.

setters - the setter is a method that is used to set the property's value. it is very useful in object oriented programming to set the value of private attributes in a class

in general , getters and setters are mainly used to ensure the data encapsulation in Oops

[11]:
```
class personal_details:

    def __init__(self, name_of_person ,age,phone_number):
        self.__name_of_person = name_of_person
        self.__age = age
        self.__phone_number = phone_number
```

```
    @property
    def personal_details_access(self):
        return self.__name_of_person , self.__age, self.__phone_number


    @personal_details_access.setter
    def person_details_set(self,age):
        if age<0 or age>200:
            pass
        else:
            self.__age=age
```

[12]:
```
personal_details_obj =personal_details("raju",25,7894561234)
```

[13]:
```
personal_details_obj.personal_details_access
```

[13]:
```
('raju', 25, 7894561234)
```

[14]:
```
personal_details_obj.person_details_set=205
```

[15]:
```
personal_details_obj.person_details_set
```

[15]:
```
('raju', 25, 7894561234)
```

[16]:
```
personal_details_obj.person_details_set=-12
```

[17]:
```
personal_details_obj.person_details_set
```

[17]:
```
('raju', 25, 7894561234)
```

[18]:
```
personal_details_obj.person_details_set=100
```

[19]:
```
personal_details_obj.person_details_set
```

[19]:
```
('raju', 100, 7894561234)
```

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

ANS. method overriding in python is subclass that inherits the superclass properties and methods, but can override the superclass methods .

[20]:
```
class animal:

    def sound(self):
        print("animals makes sounds")

class dog(animal):
```

```
    def sound(self):
        print("dog has a sound wof! wof!")
```

[21]:
```
dog =dog()
```

[22]:
```
dog.sound()
```

```
dog has a sound wof! wof!
```

[ ]:

[ ]: