In [1]:
```python
class Bkdet:
    name = "Omsakthi"
    name1 = "Parasakthi"
def frt(self):
    print("Hi")

myclass = Bkdet()
print(myclass.name)
print(myclass.name1)
frt('Om')
frt('')
```

```
Omsakthi
Parasakthi
Hi
Hi
```

In [55]:
```python
class Restaurantdt:

        def __init__(self, restname, cusintype):
            self.restname = restname
            self.cusintype = cusintype
        def describe_restaurant(self):
            print(f"The restaurant name is Indian Restaurant {self.restname}")
        def open_restaurant(self):
            print(f"The restaurant opens from 9:00 am into 10:00pm {self.cusintype}")

        restdet = Restaurantdt('Tajmahal Restaurant','Indian Cusine')
        restother = Restaurantdt('Taj Coramandal','Indian')
        restaurant3 = Restaurantdt('Hotel Mariot','Mexican food')
        rest4 = Restaurantdt('Appakadai','South Indian Food')

        print(f"The name of the restaurant is {restdet.restname}")
        print(f"The type of cusine is {restdet.cusintype}")

        print(f"The name of the restaurant is {restother.restname}")
        print(f"The type of cusine is {restother.cusintype}")

        print(f"The name of the restaurant is {restother.restname}")
        print(f"The name of the restaurant is {restaurant3.restname}")
        print(f"The name of the restaurant is {rest4.restname}")

        restdet.describe_restaurant()
        restdet.open_restaurant()
        restaurant3.describe_restaurant()
        rest4.describe_restaurant()
        restother.describe_restaurant()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-55-ae4e32f2c04c> in <module>
----> 1 class Restaurantdt:
      2 
      3         def __init__(self, restname, cusintype):
      4             self.restname = restname
      5             self.cusintype = cusintype

<ipython-input-55-ae4e32f2c04c> in Restaurantdt()
      9             print(f"The restaurant opens from 9:00 am into 10:00pm {self.cusintype}")
     10 
---> 11         restdet = Restaurantdt('Tajmahal Restaurant','Indian Cusine')
     12         restother = Restaurantdt('Taj Coramandal','Indian')
     13         restaurant3 = Restaurantdt('Hotel Mariot','Mexican food')

NameError: name 'Restaurantdt' is not defined
```

In [ ]:
```python
class Dog:
#  """A simple attempt to model a dog."""

    def __init__(self, name, age):
        self.name = name
        self.age = age
    def sit(self):
        print(f"{self.name} is now sitting.")

    def roll_over(self):
        print(f"{self.name} rolled over!")

    my_dog = Dog('Willie', 6)


    print(f"My dog's name is {my_dog.name}.")
    print(f"My dog is {my_dog.age} years old.")
```

In [ ]:
```python
class User:
    def __init__(self, f_name, l_name, date_birth,qualification)
```

In [2]:
```python
class Car:


    def __init__(self, make, model, year,odometer):
        self.make = make
        self.model = model
        self.year = year
        self.odometer = odometer
        #setting a default value for an attribute.
        self.speed = 0
        self.spdLimit=0
        self.spdLimit1=0
        self.shlspeed =40
        self.incrlt=0

    def get_descriptive_name(self):
        long_name = f"{self.year} {self.make} {self.model}"
        return long_name.title()

    def odometer_reading(self):
        print(f"Odometer reading is {self.odometer}")

#modyfying an arrtribute value directly
    def speed_limit(self):
        print(f"The speed limit at the school area {self.speed}")

    def school_spdlt(self):
        print(f"The school speed limit is {self.shlspeed}")

    def user_input(self):
        spdLimit1 = int(input("Enter the school area speed limit here to know is it correctly define
d or not:"))
        print(spdLimit1)
        self.spdLimit= spdLimit1
        print(f"Decleard speed limit is {self.shlspeed}")
        if spdLimit1<= self.shlspeed:
            print(f"The entered {spdLimit1}, and this speed limit is not exceeds the defined speed l
imit ")
        else:
            print(f"You entered {spdLimit1}, and this speed limit exceeds the oringinal limit ")

    def increment_splimit(self,incrlt):
        self.speed +=incrlt

    def capacity_fill_gas(self):
        print("The gas tank cacity is: 40 Gallons")




my_new_car = Car('audi', 'a4', 2019,20)

print(my_new_car.get_descriptive_name())

print(my_new_car.odometer_reading())

sp = my_new_car.speed_limit = 30
splt = my_new_car.school_spdlt = 25
my_new_car.user_input()
print(f"Before increment {my_new_car.speed}")
increment = my_new_car.increment_splimit(10)
print(f"After increment {my_new_car.speed}")

my_new_car.capacity_fill_gas()
```

```
2019 Audi A4
Odometer reading is 20
None
Enter the school area speed limit here to know is it correctly defined or not:89
89
Decleard speed limit is 40
You entered 89, and this speed limit exceeds the oringinal limit
Before increment 0
After increment 10
The gas tank cacity is: 40 Gallons
```

In [6]:
```python
class Restnt:


    def __init__(self,number_of_table,number_served):
        self.number_served = number_served
        self.number_of_table = number_of_table
        self.additionalsrd = 0

    def numofpl_served(self):
        print(f"The number of people served {self.number_served}")

    def number_oftable(self):
        print(f"The number of table ordered {self.number_of_table}")

    def add_numserved(self,additionalsrd):
        self.number_served += additionalsrd
        print(f"After adding the number of people is {self.number_served}")




restnser = Restnt(303,23)
print(restnser.number_served)
print(restnser.number_of_table)
restnser.numofpl_served()
restnser.number_oftable()

restnser.add_numserved(503)
print(f"After adding the people is {restnser.number_served}")
```

```
23
303
The number of people served 23
The number of table ordered 303
After adding the number of people is 526
After adding the people is 526
```

In [34]:
```python
class UserDet:

    def __init__(self, login_attempts):
        self.login_attempts= login_attempts
        incrattemp=0
        decattempt=0

    def user_login(self):
        print(login_attempts)


    def increment_login_attempts(self,incrattemp):
        self.login_attempts += incrattemp
        print(f"After increment the login attempts are : {self.login_attempts}")

    def reset_login_attempts(self,decattempt):
        self.login_attempts-= decattempt
        self.login_attempts=0
        print(f"After decrement the login attempt is : {self.login_attempts}")

log = UserDet(10)
log.increment_login_attempts(11)
log.reset_login_attempts(10)

log1 = UserDet(20)
log1.increment_login_attempts(9)
log1.reset_login_attempts(0)
```

```
After increment the login attempts are : 21
After decrement the login attempt is : 0
After increment the login attempts are : 29
After decrement the login attempt is : 0
```

In [3]:
```python
#INHERITANCE

class ElectriCar1(Car):

    def __init__(self, make, model, year,odometer):
        super().__init__(make, model, year,odometer)
    def get_range(self):
        ent_battry_size = int(input("Entered the battery size"))
        print(ent_battry_size)
        self.battry_sze  = ent_battry_size
        if ent_battry_size >=100:
            print(f"{ent_battry_size}The range is = 250")
        else:
            print(f"{ent_battry_size}The range is = 150")


Tesla = ElectriCar1('Tesla','Hyprid',2019,23)
print(Tesla.get_descriptive_name())

Tesla.get_range()
```

```
2019 Tesla Hyprid
Entered the battery size89
89
89The range is = 150
```

In [15]:
```python
class ElectriCar(Car):


    def __init__(self, make, model, year,odometer):
        #spdLimit1=0
        super().__init__(make, model, year,odometer)
        self.battery_size = 75
        self.batterycar = Battery()

    def user_input(self):
        print(self.spdLimit1)
        print(self.shlspeed)

    def car_battery_size(self):
        print(self.battery_size)

#OVER RIDDIG A METHOD FROM THE PARENT CLASS.
    def capacity_fill_gas(self):
        print("This Tesla car is hybrid,so no need any gas to fill now")



Tesla = ElectriCar('Tesla','Hyprid',2019,23)
print(Tesla.get_descriptive_name())

Tesla.car_battery_size()
Tesla.user_input()
Tesla.capacity_fill_gas()

Tesla.batterycar.battery_req()
Tesla.batterycar.get_range()
```

```
2019 Tesla Hyprid
75
0
40
This Tesla car is hybrid,so no need any gas to fill now
Baterry charge required for the electric car and the battery size is 75
Entered the battery size is : 89
89
89 so, the range is = 150
```

In [14]:
```python
class Battery:

    def __init__(self,battery_charge=75, ugdbattery_size=100):
        self.battery_charge = battery_charge
        self.battry_sze =0
        self.ent_battry_size =0
        self.upg1_battry_size= 0
        self.upg_battry_size = 0


    def battery_req(self):
        print(f"Baterry charge required for the electric car and the battery size is {self.battery_c
harge}")

    def get_range(self):
        ent_battry_size = int(input("Entered the battery size is : "))
        print(ent_battry_size)
        self.battry_sze  = ent_battry_size
        if ent_battry_size >=100:
            print(f"{ent_battry_size} so, the range is = 250")
        else:
            print(f"{ent_battry_size} so, the range is = 150")
    def upgrade_battery(self):
        upg_battry_size = int(input("Entered the battery size is : "))
        print(upg_battry_size)
        self.upg1_battry_size = upg_battry_size
        if upg_battry_size >= 100:
            print("This battery size will suitable for the electri car")
        else:
            print("It is not suitable")

bat = Battery()
print(bat.battery_req())
```

```
Baterry charge required for the electric car and the battery size is 75
None
```

In [13]:
```python
class IceCreamStand(Restnt):

    def __init__(self,flavors,number_of_table, number_served):
        super().__init__(number_of_table,number_served)
        self.flavors = flavors

    def name_flavors(self):
        print(f"The name of the flavor is : {self.flavors}")

    def numofpl_served(self):
        print(f"The number of people served {self.number_served}")

    def number_oftable(self):
        print(f"The number of table ordered {self.number_of_table}")

ice = IceCreamStand('Vennila',89,22)
print(ice.numofpl_served())
print(ice.number_oftable())
print(ice.name_flavors())
```

```
The number of people served 22
None
The number of table ordered 89
None
The name of the flavor is : Vennila
None
```

In [39]:
```python
#9.7
class Admin(UserDet):

    def __init__(self,privileges):
        self.privileges = privileges
        login_attempts= 100
        super().__init__(login_attempts)
       # self.privinher= Privileges()
        incrattemp=0
        decattempt=0

    def show_privileges(self):
        print(f"Can add post")
        print(f"Can delete post {self.privileges}")
        print(f"Can ban post {self.privileges}")
        print(self.privileges)

    def user_login(self):
        print(f"Number of login time is {self.login_attempts}")


    def increment_login_attempts(self,incrattemp):
        self.login_attempts += incrattemp
        print(f"After increment the login attempts are : {self.login_attempts}")


admindt = Admin('- Adminstraotor')
admindt1 = Admin('I can add post')
admindt2 = Admin('I can delete post')
print(admindt1.privileges)
print(admindt2.privileges)
admindt.show_privileges()

admindt.user_login()
admindt.increment_login_attempts(10)

#admindt.privinher.password_change()
```

```
I can add post
I can delete post
Can add post
Can delete post - Adminstraotor
Can ban post - Adminstraotor
- Adminstraotor
Number of login time is 100
After increment the login attempts are : 110
```

In [40]:
```python
#9.8

class Privileges(Admin):


    def __init__(self,pw):
        self.pw =pw
       # super().__init__(privileges)

       # self.reuser = UserDet()
    def password_change(self):
        print(f"Hi {self.pw}")



privs = Privileges('jk')
#privs.show_privileges()
#privs.reuser.user_login()
privs.password_change()
```

```
Hi jk
```

In [8]:
```python
from car import ElectriCar,Car
Tesla = ElectriCar('Tesla','Hyprid',2019,23)
print(Tesla.get_descriptive_name())
my_new_car = Car('audi1', 'a4', 2019,20)
print(my_new_car.get_descriptive_name())
```

```
2019 Audi A4
Odometer reading is 20
None
Enter the school area speed limit here to know is it correctly defined or not:8
8
Decleard speed limit is 40
The entered 8, and this speed limit is not exceeds the defined speed limit
Before increment 0
After increment 10
The gas tank cacity is: 40 Gallons
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-8-a7681c13925d> in <module>
----> 1 from car import ElectriCar,Car
      2 Tesla = ElectriCar('Tesla','Hyprid',2019,23)
      3 print(Tesla.get_descriptive_name())
      4 my_new_car = Car('audi1', 'a4', 2019,20)
      5 print(my_new_car.get_descriptive_name())

~\car.py in <module>
     88 
     89 
---> 90 Tesla = ElectriCar('Tesla','Hyprid',2019,23)
     91 print(Tesla.get_descriptive_name())
     92 

~\car.py in __init__(self, make, model, year, odometer)
     74         super().__init__(make, model, year,odometer)
     75         self.battery_size = 75
---> 76         self.batterycar = Battery()
     77 
     78     def user_input(self):

NameError: name 'Battery' is not defined
```

In [11]:
```python
import car

capscar = car.Car('audi1', 'a4', 2019,20)
print(cascar.get_descriptive_name())

Tesla1 = car.ElectriCar('Tesla','Hyprid',2019,23)
print(Tesla1.get_descriptive_name())
```

```
2019 Audi A4
Odometer reading is 20
None
Enter the school area speed limit here to know is it correctly defined or not:7
7
Decleard speed limit is 40
The entered 7, and this speed limit is not exceeds the defined speed limit
Before increment 0
After increment 10
The gas tank cacity is: 40 Gallons
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-11-f1ce3f39e363> in <module>
----> 1 import car
      2 
      3 capscar = car.Car('audi1', 'a4', 2019,20)
      4 print(cascar.get_descriptive_name())
      5 

~\car.py in <module>
     88 
     89 
---> 90 Tesla = ElectriCar('Tesla','Hyprid',2019,23)
     91 print(Tesla.get_descriptive_name())
     92 

~\car.py in __init__(self, make, model, year, odometer)
     74         super().__init__(make, model, year,odometer)
     75         self.battery_size = 75
---> 76         self.batterycar = Battery()
     77 
     78     def user_input(self):

NameError: name 'Battery' is not defined
```

In [18]:
```python
from car import Car
from Electric_car import ElectriCar
from Electric_car import Battery

capscar = Car('audi1', 'a4', 2019,20)
print(cascar.get_descriptive_name())

Tesla1 = ElectriCar('Tesla','Hyprid',2019,23)
print(Tesla1.get_descriptive_name())

bat1 = Battery()
print(bat.battery_req())
```

```
2019 Audi A4
Odometer reading is 20
None
Enter the school area speed limit here to know is it correctly defined or not:9
9
Decleard speed limit is 40
The entered 9, and this speed limit is not exceeds the defined speed limit
Before increment 0
After increment 10
The gas tank cacity is: 40 Gallons
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-18-ea636ce05626> in <module>
----> 1 from car import Car
      2 from Electric_car import ElectriCar
      3 from Electric_car import Battery
      4 
      5 capscar = Car('audi1', 'a4', 2019,20)

~\car.py in <module>
     88 
     89 
---> 90 Tesla = ElectriCar('Tesla','Hyprid',2019,23)
     91 print(Tesla.get_descriptive_name())
     92 

~\car.py in __init__(self, make, model, year, odometer)
     74         super().__init__(make, model, year,odometer)
     75         self.battery_size = 75
---> 76         self.batterycar = Battery()
     77 
     78     def user_input(self):

NameError: name 'Battery' is not defined
```

In [20]:
```python
from User_infor import UserDet
log1 = UserDet(20)
log1.increment_login_attempts(9)
log1.reset_login_attempts(0)
```

```
After increment the login attempts are : 29
After decrement the login attempt is : 0
```

In [22]:
```python
from User_infor import UserDet, Admin
log1 = UserDet(20)
log1.increment_login_attempts(9)
log1.reset_login_attempts(0)
admindt = Admin('- Adminstraotor')
admindt1 = Admin('I can add post')
admindt2 = Admin('I can delete post')
print(admindt1.privileges)
print(admindt2.privileges)
admindt.show_privileges()
```

```
After increment the login attempts are : 29
After decrement the login attempt is : 0
I can add post
I can delete post
Can add post
Can delete post - Adminstraotor
Can ban post - Adminstraotor
- Adminstraotor
```

In [39]:
```python
from random import randint

randint(1,5)
randint(2,3)
```

Out[39]: 3

In [40]:
```python
from random import choice
players = ['akila','kamala','ravi']
cho = choice(players)
print(cho)
```

```
kamala
```

In [14]:
```python
from random import randint
class Die:

    def __init__(self,rl=0,sides=6,sides1=6,sides10=10,sides20=20):
        self.sides = sides
        self.rl= rl
        self.sides1 = sides1
        self.sides10 = sides10
        self.sides20 = sides20


    def rol_die(self,sides):
        dies =randint(1,sides)
        print(dies)


    def troll(self,rl):
        rl=1
        while rl<=10:
            dies1 =randint(1,self.sides1)
           # print(dies1)
            print(f"The number {rl} roll and the value in the die is: {dies1}")
            rl+=1

    def ten_sided_die(self):
        self.rl=1
        while self.rl<=10:
            times10 = randint(1,self.sides10)
            print(f"The number {self.rl} roll and the value in the die is: {times10}")
            self.rl+=1

    def twent_sided_die(self):
        self.rl = 1
        while self.rl <= 10:
            times20 = randint(1,self.sides20)
            print(f"The number {self.rl} roll and the value in the die is: {times20}")
            self.rl += 1



die6 = Die()
res = die6.rol_die(6)
res1 = die6.rol_die(6)
die6.rol_die(6)

die6.troll(1)

die6.ten_sided_die()

die6.twent_sided_die()
```

```
1
1
5
The number 1 roll and the value in the die is: 6
The number 2 roll and the value in the die is: 6
The number 3 roll and the value in the die is: 2
The number 4 roll and the value in the die is: 3
The number 5 roll and the value in the die is: 3
The number 6 roll and the value in the die is: 1
The number 7 roll and the value in the die is: 1
The number 8 roll and the value in the die is: 1
The number 9 roll and the value in the die is: 2
The number 10 roll and the value in the die is: 6
The number 1 roll and the value in the die is: 9
The number 2 roll and the value in the die is: 10
The number 3 roll and the value in the die is: 3
The number 4 roll and the value in the die is: 2
The number 5 roll and the value in the die is: 8
The number 6 roll and the value in the die is: 6
The number 7 roll and the value in the die is: 9
The number 8 roll and the value in the die is: 9
The number 9 roll and the value in the die is: 6
The number 10 roll and the value in the die is: 8
The number 1 roll and the value in the die is: 20
The number 2 roll and the value in the die is: 12
The number 3 roll and the value in the die is: 20
The number 4 roll and the value in the die is: 10
The number 5 roll and the value in the die is: 9
The number 6 roll and the value in the die is: 7
The number 7 roll and the value in the die is: 12
The number 8 roll and the value in the die is: 9
The number 9 roll and the value in the die is: 10
The number 10 roll and the value in the die is: 14
```

In [66]:
```python
from random import choice
class Lottery:

#This  class used 2 methods for choosing the values from the list and tuple using choice() method.
#It is imported random library.


    def ___init___(self,names,my_ticket,lott_tuple,my_ticket1):
        self.names = names
        self.my_ticket = my_ticket
        self.lott_tuple = lott_tuple
        self.my_ticket1 = my_ticket1

    def lott_list(self):
        self.names = ['kavi','loga','uma','krishna','adhi','1','2','3','4','5']
        relt = choice(self.names)
        relt1= choice(self.names)
        relt2= choice(self.names)
        relt3= choice(self.names)
        self.my_ticket = relt+ relt1 + relt2 + relt3
        print(self.my_ticket)

        print(f"Any ticket matching for the following numbers and names wins a prize {relt,relt1,rel
t2,relt3}")
      # print(self.my_ticket1)

#TUPLE VALUE IS IMMUTABLE IT MEANS IT CAN'T CHANGE ONCE WE DECLARED IN TUPLE BUT LIST CAN CHANGE.

    def lott_tuplefn(self):
        self.lott_tuple =('kavi','loga','uma','krishna','adhi','1','2','3','4','5')


        relt4 = choice(self.lott_tuple)
        print(f"result {relt4}")


        for lottpl in choice(self.lott_tuple):
            if lottpl == relt4:
                print(f"Equal  1st value {lottpl}")
            else:
                print(relt4)


        relt5= choice(self.lott_tuple)
        print(f"result {relt5}")
        for lottpl in choice(self.lott_tuple):
            if lottpl == relt5:
                print(f"Equal 2nd value {lottpl}")
                ct=count(lottpl)
            else:
                print(relt5)

        relt6= choice(self.lott_tuple)
        print(f"result {relt6}")
        for lottpl in choice(self.lott_tuple):
            if lottpl == relt6:
                print(f"Equal 3rd value {lottpl}")
            else:
                print(relt6)

        relt7= choice(self.lott_tuple)
        print(f"result {relt7}")
        for lottpl in choice(self.lott_tuple):
            if lottpl == relt7:
                print(f"Equal 4th value {lottpl}")
            else:
                print(relt7)


        self.my_ticket1 = relt4 + relt5 + relt6 + relt7
        print(f" The choice of value from the tuple {self.my_ticket1}")


        for lottp in self.lott_tuple:
            if lottp == self.my_ticket1:
                print(f"Equal value {lottp}")
            else:
                print(f"From my ticket1 {self.my_ticket1}")




lotrobj =  Lottery()
lotrobj.lott_list()
lotrobj.lott_tuplefn()
```

```
1krishna44
Any ticket matching for the following numbers and names wins a prize ('1', 'krishna', '4', '4')
result 4
4
result krishna
krishna
result uma
uma
result krishna
krishna
 The choice of value from the tuple 4krishnaumakrishna
From my ticket1 4krishnaumakrishna
From my ticket1 4krishnaumakrishna
From my ticket1 4krishnaumakrishna
From my ticket1 4krishnaumakrishna
From my ticket1 4krishnaumakrishna
From my ticket1 4krishnaumakrishna
From my ticket1 4krishnaumakrishna
From my ticket1 4krishnaumakrishna
```