In [1]:
```
# How to count number of instances of a class in Python?
```

In [1]:
```
class data:
    count=0
    def __init__(self):
        data.count+=1
```

In [2]:
```
d1=data()
d2=data()
d3=data()
print(data.count)
```

```
3
```

In [3]:
```
# How to create a list of object in Python class
class data:
    def __init__(self,fname,lname,stud_id):
        self.fname=fname
        self.lname=lname
        self.id=stud_id
```

In [4]:
```
l=[]
l.append(data("Rahul","sharma",1))
l.append(data("Riya","Shukla",2))
l.append(data("Ritu","mehta",3))
```

In [5]:
```
print(l[0].fname)
```

```
Rahul
```

In [6]:
```
print(l[0].lname)
```

```
sharma
```

In [7]:
```
print(l[2].lname)
```

```
mehta
```

In [10]:
```
# Function Inside the Function
```

In [11]:
```
def outerFunction(text):
    text = text

    def innerFunction():
        print(text)

    innerFunction()
```

In [12]:
```
outerFunction("Hello")
```

```
Hello
```

In [29]:
```
a=int(input("Enter a : "))
b=int(input("Enter b : "))
def calculation(a,b):
    def add():
        print("addition is: ",a+b)
    add()
    def subs():
        print("substraction is: ",a-b)
    subs()
    def mul():
        print("multiplacation is : ",a*b)
    mul()
```

```
Enter a : 23
Enter b : 3
```

In [30]:
```
calculation(a,b)
```

```
addition is:  26
substraction is:  20
multiplacation is :  69
```

```
Creating a Person Class with name, age, and profession instance variables.
```

In [31]:
```
class person:
    def __init__(self,name,age,profession):
        self.name=name
        self.age=age
        self.profession=profession
    def display(self):
        print(f"{self.name} is {self.age} years old and profession is {self.prc
    def work(self):
        print("{} is workinig as as a {}".format(self.name,self.profession))
```

In [32]:
```
p=person("Siya",24,"software Developer")
```

In [33]:
```
p.display()
p.work()
```

```
Siya is 24 years old and profession is software Developer
Siya is workinig as as a software Developer
```

In [36]:
```
# Instance and class variable
class person:
    '''company and school are class variable.
    name, age, profession are instance variable'''
    company="ABC"
    school="XYZ"
    def __init__(self,name,age,profession):
        self.name=name
        self.age=age
        self.profession=profession
    def display(self):
        print(f"{self.name} is {self.age} years old and profession is {self.pro
    def work(self):
        print("{} is workinig as as a {}".format(self.name,self.profession))
```

In [37]:
```
p=person("Siya",24,"software Developer")
```

In [38]:
```
p.name # access instance variable
```

Out[38]: 'Siya'

In [39]:
```
p.company  # access class variable
```

Out[39]: 'ABC'

In [44]:
```
# update / change company
person.company=" PQR"
```

In [45]:
```
p=person("Vaibhav",24,"software Developer")
```

In [46]:
```
p.company
```

Out[46]: ' PQR'

In [67]:
```
# Instance and class variable
class person:
    '''company and school are class variable.
    name, age, profession are instance variable'''
    company="ABC"
    school="XYZ"
    def __init__(self,name,age,profession):
        self.name=name
        self.age=age
        self.profession=profession
    def display(self):
        print(f"{self.name} is {self.age} years old and profession is {self.pro
    def work(self):
        print("{} is workinig as as a {}".format(self.name,self.profession))
    def students(self,new_name,new_age,new_prof):
        self.name=new_name
        self.age=new_age
        self.profession=new_prof
        print("name of student is : {} and \nage : {} \nprofession {}".format(r
    def new_sch(self,new_school):
        person.school=new_school
        print("new school is: ",new_school)
```

In [68]:
```
p=person("Vaibhav",24,"software Developer")
```

In [69]:
```
p.work()
p.company
```

```
Vaibhav is workinig as as a software Developer
```

Out[69]:
```
'ABC'
```

In [71]:
```
p.students("Riya",12,"student")
```

```
name of student is : Riya and
age : 12
profession student
```

In [72]:
```
p.new_sch("PORS School")
```

```
new school is:  PORS School
```

In [73]:
```
p.display()
```

```
Riya is 12 years old and profession is student
```

In [74]:
```
# In above code display, work,students is instance method and new_sch is class
```

In [75]:
```
class data:
```

```
  Cell In[75], line 1
    class data:
               ^
SyntaxError: incomplete input
```

In [76]:
```
def fun():
```

```
  Cell In[76], line 1
    def fun():
              ^
SyntaxError: incomplete input
```

In [77]:
```
# empty loops, functions, class not allowed. to give empty block we use pass st
```

In [78]:
```
class data:
    pass
```

In [79]:
```
def fun():
    pass
```

In [ ]:
```
# create a class for car properties
```

In [165]:
```
class car:
    def __init__(self,name,color,sunroof,fuel):
        self.name=name
        self.color=color
        self.sunroof=sunroof
        self.fuel=fuel
    def budget(self,price):
        self.price=price
        print("Is car in budget? :")
        if self.price < 600000:
            print("Yes")
            def purchase(self):
                    print("Car price {} is in budget. we can buy this car".form
            purchase(self)
        else:
            print("No")
            print("Car price {} is not in budget. we can not buy this car".form
```

In [166]:
```
c=car("Safari","red","No","Disel")
```

In [167]:
```
c.budget(500000)
```

```
Is car in budget? :
Yes
Car price 500000 is in budget. we can buy this car
```

In [168]:
```
c1=car("BMW","Blue","No","Disel")
c1.budget(70000000)
```

```
Is car in budget? :
No
Car price 70000000 is not in budget. we can not buy this car
```

In [ ]:
```

```

```

```

In [61]:
```
class vehicle:
    def __init__(self,name,wheels,windows,engine_type,fuel):
        self.name=name
        self.wheels=wheels
        self.windows=windows
        self.engine_type=engine_type
        self.fuel=fuel
    def Type(self):
        if  self.windows >4 and self.wheels==4:
            print("vehicle is Bus")
            class bus(vehicle):
                def properties(self):
                    print("It is {},and it has \n wheels: {} \n windows: {} \n
                properties(self)
#             B=bus(self)

        else:
            print("vehicle is not Bus")
```

In [62]:
```
V=vehicle("Alto",4,4,"disel","disel")
```

In [63]:
```
V.Type()
```

```
vehicle is not Bus
```

In [64]:
```
V1=vehicle("Truck",4,2,"disel","disel")
```

In [65]:
```
V1.Type()
```

```
vehicle is not Bus
```

In [66]:
```
V1=vehicle("Bus",4,16,"CNG","CNG")
V1.Type()
# B=bus("Bus",4,16,"CNG","CNG")
# B.properties()
```

```
vehicle is Bus
It is Bus,and it has
 wheels: 4
 windows: 16
 engine type : CNG
 fuel : CNG
```

In [76]:
```
# Create a child class Bus that will inherit all of the variables and methods o
class vehicle:
    def __init__(self,name,speed,capacity):
        self.name=name
        self.speed=speed
        self.capacity=capacity
class bus(vehicle):
    def info(self):
        print("Vehicle Name:", self.name, "Speed:", self.speed, "Mileage:", sel
```

In [82]:
```
v=bus("Mini_bus",60,30)
```

In [83]:
```
v.info()
```

```
Vehicle Name: Mini_bus Speed: 60 Mileage: 30
```

```
The super() function is used to give access to methods and properties of a
parent or sibling class.

The super() function returns an object that represents the parent class.
```

Create a Bus class that inherits from the Vehicle class. Give the capacity argument of Bus.seating_capacity() a default value of 50.

In [108]:
```
class vehicle:
    def __init__(self,name,speed):
        self.name=name
        self.speed=speed
    def seating_capacity(self,capacity):
        self.capacity=capacity
        print("Vehicle Name:", self.name, "Speed:", self.speed, "Capacity:", se
class bus(vehicle):
    def seating_capacity(self,capacity=50):
        return super().seating_capacity(capacity=50)
```

In [112]:
```
v=bus("Mini_bus",80)
v.seating_capacity()
```

```
Vehicle Name: Mini_bus Speed: 80 Capacity: 50
```

In [ ]: