In [3]:
```
#overloading by default argument
class show:
    def display(self,name=None):
        if name:
            print(f"Hello{name}")
        else:
            print("Hello There!")
cal=show()
cal.display()
cal.display("Amrren")
```

```
Hello There!
HelloAmrren
```

In [5]:
```
#overloading by variable argument
class calculator:
    def add(self,*args):
        return sum(args)
c=calculator()
print(c.add(23,45))
print(c.add(24,46,78,90))
```

```
68
238
```

In [7]:
```
class Notification:
    def send(self,message,user=None):
        if user:
            print(f"Sending message '{message}' to {user}")
        else:
            print(f"Broadcasting '{message}' to all users")
obj=Notification()
obj.send("Live session will start from 7 pm")
obj.send("Activity completed","Ananya")
```

```
Broadcasting 'Live session will start from 7 pm' to all users
Sending message 'Activity completed' to Ananya
```

In [11]:
```
#operator overloading-->we can overload operators like +,- etc using   special meth
class Employee:
    def __init__(self,name,salary):
        self.name=name
        self.salary=salary
    def __add__(self,other):
        return self.salary+other.salary
e1=Employee("Pooja",25000)

e2=Employee("Pavitra",40000)
total_salary=e1+e2
print("Total salary to be paid:",total_salary)
```

```
Total salary to be paid: 65000
```

In [21]:
```
class Cart:
    def __init__(self,items):
        self.items=items

    def __add__(self,other):
        return Cart(self.items+other.items)
    def display(self):
        print("Cart items:",self.items)
c1=Cart(["Tomato","Choclate","Bread"])

c2=Cart(["Milk","egg"])
c3=c1+c2#operator overloading
c3.display()
```

```
Cart items: ['Tomato', 'Choclate', 'Bread', 'Milk', 'egg']
```

In [27]:
```
#Constructor overloading-->python does not support constructor overloading but we c
class Student:
    def __init__(self,name=None,rollNo=None,Branch="Not Assigned"):
        if name and rollNo:
            self.rollNo=rollNo
            self.name=name
            self.Branch=Branch
        elif name:
            self.name=name
            self.rollNo="Pending"
            self.Branch="Not Assigned"
        else:
            self.name="Unknown"
            self.rollNo="Pending"
            self.Branch="Not Assigned"

    def show(self):
        print(f"Name:{self.name},Roll No:{ self.rollNo},Branch:{self.Branch}")
#constructor Overloading(simulated)
s1=Student("Sahil",101,"CSE")
s2=Student("Ananya")
s3=Student()
s1.show()
s2.show()
s3.show()
```

```
Name:Sahil,Roll No:101,Branch:CSE
Name:Ananya,Roll No:Pending,Branch:Not Assigned
Name:Unknown,Roll No:Pending,Branch:Not Assigned
```

In [ ]:
```
```