In [5]:

```
class operatoroverload :
    def __init__(self,a,b):
        self.x=a
        self.y=b
    def __add__(self,other):
        return self.x+other.x , self.y+other.y

obj1=operatoroverload(2,3)
obj2=operatoroverload(2,3)
print(obj1+obj2)
```

```
(4, 6)
```

In [15]:

```
class Vector :
    def __init__(self,a,b,c):
        self.x=a
        self.y=b
        self.z=c
    def __sub__(self,other):
        return self.x-other.x , self.y-other.y , self.z-other.z

vec1 = Vector(1, 2, 2)
vec2 = Vector(0, 1, 2)
print(vec1-vec2)
```

```
(1, 1, 0)
```

In [17]:

```
class Vector :
    def __init__(self,a,b,c):
        self.x=a
        self.y=b
        self.z=c
    def __add__(self,other):
        return self.x+other.x , self.y+other.y , self.z+other.z

vec1 = Vector(1, 2, 2)
vec2 = Vector(0, 1, 2)
print(vec1+vec2)
```

```
(1, 3, 4)
```

In [18]:

```
class Vector :
    def __init__(self,a,b,c):
        self.x=a
        self.y=b
        self.z=c
    def __mul__(self,other):
        return self.x*other.x , self.y*other.y , self.z*other.z

vec1 = Vector(1, 2, 2)
vec2 = Vector(0, 1, 2)
print(vec1*vec2)
```

```
(0, 2, 4)
```

In [20]:

```
class Vector :
    def __init__(self,a,b,c):
        self.x=a
        self.y=b
        self.z=c
    def __pow__(self,other):
        return self.x**other.x , self.y**other.y , self.z**other.z

vec1 = Vector(1, 2, 2)
vec2 = Vector(0, 1, 7)
print(vec1**vec2)
```

```
(1, 2, 128)
```

In [23]:

```
class Vector :
    def __init__(self,a,b,c):
        self.x=a
        self.y=b
        self.z=c
    def __ge__(self,other):
        return self.x>=other.x , self.y>=other.y , self.z>=other.z

vec1 = Vector(1, 2, 2)
vec2 = Vector(0, 6, 2)
print(vec1>=vec2)
```

```
(True, False, True)
```

In [24]:

```
class Vector :
    def __init__(self,a,b,c):
        self.x=a
        self.y=b
        self.z=c
    def __le__(self,other):
        return self.x<=other.x , self.y<=other.y , self.z<=other.z

vec1 = Vector(1, 2, 2)
vec2 = Vector(0, 6, 2)
print(vec1<=vec2)
```

```
(False, True, True)
```

In [ ]:

```

```