In [73]:

```
class Matrix:
    def __init__(self,row,col):
        self.row=row
        self.col=col
        self.data=[0 for i in range(row*col)]
    def setValues(self,i,j,value):
        self.data[self.Location(i,j)]=value
    def getValues(self,i,j):
        return self.data[self.Location(i,j)]
    def Location(self,i,j):
        loc=i+self.row*j
        return loc
    def Print(self):
        for i in range(self.row):
            for j in range(self.col):
                print(self.getValues(i,j),end=" ")
            print("\r")
    def multValues(self,a,b):
        for i in range(a.row):
            for j in range(b.col):
                Sum=0
                for k in range(b.row):
                    x=a.getValues(i,k)
                    y=b.getValues(k,j)
                    Sum+=x*y
                    self.setValues(i,j,Sum)
    def transpose(self,a):
        for i in range(a.row):
            for j in range(a.col):
                self.setValues(j,i,a.getValues(i,j))
row=2
col=2
a=Matrix(row,col)
print("***************Matrix A***************")
for i in range(row):
    for j in range(col):
        a.setValues(i,j,i+j)
a.Print()
row=2
col=2
b=Matrix(row,col)
print("***************Matrix A***************")
for i in range(row):
    for j in range(col):
        b.setValues(i,j,i+i+j)
b.Print()
row=2
col=2
c=Matrix(row,col)
c.multValues(a,b)
print("*******Product of Matrix A and B*******")
c.Print()
row=2
col=2
d=Matrix(row,col)
d.transpose(c)
print("***Transpose of the Product of Matrix A and B***")
d.Print()
```

```
***************Matrix A***************
0 1
1 2
***************Matrix A***************
0 1
2 3
*******Product of Matrix A and B*******
2 3
4 7
***Transpose of the Product of Matrix A and B***
```

```
2 4
3 7
```

In [1]:
```
import numpy as np
array1 = np.array([[1,2,3,4],[5,6,7,8]], dtype=np.int64)
print(array1)
```

```
[[1 2 3 4]
 [5 6 7 8]]
```

In [2]:
```
x = np.ones((3,4),dtype=np.int64)
print(x)
```

```
[[1 1 1 1]
 [1 1 1 1]
 [1 1 1 1]]
```

In [3]:
```
y = np.zeros((2,3,4),dtype=np.int16)
print(y)
```

```
[[[0 0 0 0]
  [0 0 0 0]
  [0 0 0 0]]

 [[0 0 0 0]
  [0 0 0 0]
  [0 0 0 0]]]
```

In [4]:
```
array2 = np.random.random((2,2))
print(array2)
```

```
[[0.61216521 0.54678502]
 [0.89942403 0.90670996]]
```

In [5]:
```
array3 = np.full((3,3),7)
print(array3)
```

```
[[7 7 7]
 [7 7 7]
 [7 7 7]]
```

In [6]:
```
array4 = np.identity(3,dtype=np.int64)
print(array4)
```

```
[[1 0 0]
 [0 1 0]
 [0 0 1]]
```

In [7]:
```
add = np.add(x,y)
print(add)
```

```
[[[1 1 1 1]
  [1 1 1 1]
  [1 1 1 1]]

 [[1 1 1 1]
  [1 1 1 1]
  [1 1 1 1]]]
```

In [8]:
```
diff = np.subtract(x,y)
print(diff)
```

```
[[[1 1 1 1]
  [1 1 1 1]
  [1 1 1 1]]

 [[1 1 1 1]
  [1 1 1 1]
  [1 1 1 1]]]
```

In [9]:
```
mult = np.multiply(x,y)
print(mult)
```

```
[[[0 0 0 0]
  [0 0 0 0]
  [0 0 0 0]]

 [[0 0 0 0]
  [0 0 0 0]
  [0 0 0 0]]]
```

In [10]:
```
div = np.divide(y,x)
print(div)
```

```
[[[0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]]

 [[0. 0. 0. 0.]
  [0. 0. 0. 0.]
  [0. 0. 0. 0.]]]
```

In [11]:
```
rem = np.remainder(y,x)
print(rem)
```

```
[[[0 0 0 0]
  [0 0 0 0]
  [0 0 0 0]]

 [[0 0 0 0]
  [0 0 0 0]
  [0 0 0 0]]]
```

In [13]:
```
result = np.array_equal(x,y)
print(result)
```

```
False
```