In [1]:
```
import numpy as np
```

In [8]:
```
#python has built in min and max functions for finding minimum elements of an array
a=np.array([2,1,4,5,3,6])
#different ways to find minimum element and finding the time taken for execution
%timeit a.min()
%timeit min(a)
%timeit np.min(a)
```

```
2.12 µs ± 291 ns per loop (mean ± std. dev. of 7 runs, 100,000 loops each)
1.27 µs ± 160 ns per loop (mean ± std. dev. of 7 runs, 1,000,000 loops each)
5.02 µs ± 564 ns per loop (mean ± std. dev. of 7 runs, 100,000 loops each)
```

In [9]:
```
#different ways to find minimum element and finding the time taken for execution
%timeit a.max()
%timeit max(a)
%timeit np.max(a)
```

```
2.42 µs ± 99.5 ns per loop (mean ± std. dev. of 7 runs, 1,000,000 loops each)
1.36 µs ± 121 ns per loop (mean ± std. dev. of 7 runs, 1,000,000 loops each)
4.62 µs ± 406 ns per loop (mean ± std. dev. of 7 runs, 100,000 loops each)
```

In [29]:
```
#aggregate funtions along rows and columns
#addition argument axis is given
b=np.random.randint(0,5,(3,3))
print(b)
print("Sum of all elements=",np.sum(b))#gives sum of all elements of the array
print("Sum of elements along each row=",np.sum(b,axis=1))#gives sum along rows only
print("Sum of elements along each column=",np.sum(b,axis=0))#gives sums along columns only
print("Minimum elements along each row=",b.min(axis=1))#gives minimum element along each row
print("Minimum elements along each column=",b.min(axis=0))#gives minimum element along each column
```

```
[[0 4 4]
 [2 3 0]
 [2 1 3]]
Sum of all elements= 19
Sum of elements along each row= [8 5 6]
Sum of elements along each column= [4 8 7]
Minimum elements along each row= [0 0 1]
Minimum elements along each column= [0 1 0]
```

In [7]:
```
#nansafe aggreagate functions will assign a nan(not a number)for invalid or missing data
```