In [1]:

```python
import numpy as np
import time
import sys

a=range(100)
print(sys.getsizeof(10)*len(a))

b=np.arange(100)
print(b.size*b.itemsize)
```

```
2800
400
```

In [4]:

```python
import numpy as np
import time
import sys

size=1000000

A1 = range(size)
A2 = range(size)

B1=np.arange(size)
B2=np.arange(size)

start=time.time()
result=[(x,y) for x,y in zip(A1,A2)]
print((time.time()-start)*1000)

start =time.time()
result=B1 +B2
print((time.time()-start)*1000)
```

```
206.6967487335205
64.1481876373291
```

1. Array Attributes

In [7]:

```python
import numpy as np
a=np.array([[1,2,3],[4,5,6]])
print(a.shape)
```

```
(2, 3)
```

In [13]:

```python
a=np.array([[1, 2,3,4,5, 6]])
a.shape=(3,2)
print(a)
```

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

In [10]:

```
import numpy as np
a=np.arange(24)
b=a.reshape(2,4,3)
print(b.ndim)
print(b)
```

```
3
[[[ 0  1  2]
  [ 3  4  5]
  [ 6  7  8]
  [ 9 10 11]]

 [[12 13 14]
  [15 16 17]
  [18 19 20]
  [21 22 23]]]
```

In [14]:

```
import numpy as np
x=np.array([1,2,3,4,5],dtype=np.int8)
print(x.itemsize)
```

```
1
```

In [16]:

```
import numpy as np
x=np.array([1,2,3,4,5],dtype=np.float32)
print(x.itemsize)
```

```
4
```

2. Indexing & Slicing

In [25]:

```
import numpy as np
a=np.arange(20)
s=slice(1,10,3)
print(a[s])
```

```
[1 4 7]
```

In [19]:

```
import numpy as np
a=np.arange(10)
b=a[2:7:2]
print(b)
```

```
[2 4 6]
```

In [20]:

```
import numpy as np
a=np.arange(10)
print(a[5:])
```

```
[5 6 7 8 9]
```

3. Iterating over array

In [32]:

```
import numpy as np
a=np.arange(0,60,5)
a=a.reshape(3,4)

print(a,"\n")

for x in np.nditer(a):
    print(x,end=' ')
```

```
[[ 0  5 10 15]
 [20 25 30 35]
 [40 45 50 55]]

0 5 10 15 20 25 30 35 40 45 50 55
```

In [33]:

```
import numpy as np
a=np.arange(0,60,5)
a=a.reshape(3,4)

print(a,"\n")

for x in np.nditer(a, op_flags=['readwrite']):
    x[...]=2*x
print(a)
```

```
[[ 0  5 10 15]
 [20 25 30 35]
 [40 45 50 55]]

[[  0  10  20  30]
 [ 40  50  60  70]
 [ 80  90 100 110]]
```

4. Array maipulation

In [43]:

```
import numpy as np
a=np.arange(12).reshape(3,4)

print(a,"\n")

print('ravel',a.ravel(),'\n')

print('flatten',a.flatten(),'\n')

print('transpose',np.transpose(a),'\n')

print('T',a.T,'\n')
```

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]

ravel [ 0  1  2  3  4  5  6  7  8  9 10 11]

flatten [ 0  1  2  3  4  5  6  7  8  9 10 11]

transpose [[ 0  4  8]
 [ 1  5  9]
 [ 2  6 10]
 [ 3  7 11]]

T [[ 0  4  8]
 [ 1  5  9]
 [ 2  6 10]
 [ 3  7 11]]
```

In [45]:

```
a=np.array([[1,2],[3,4]])

b=np.array([[5,6],[7,8]])

print(np.concatenate((a,b)))
print('\n')

print(np.stack((a,b),0))
```

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


[[[1 2]
  [3 4]]

 [[5 6]
  [7 8]]]
```

5.Binary operators

In [46]:

```
import numpy as np
a=np.array([[1,2,3],[4,5,6]])

print(np.append(a,[7,8,9]))
print('\n')

print(np.insert(a,6,[7,8,9]))
print('\n')

print(np.delete(a,5))
print('\n')
```

```
[1 2 3 4 5 6 7 8 9]


[1 2 3 4 5 6 7 8 9]


[1 2 3 4 5]
```

In [48]:

```
import numpy as np
a,b=13,17
print(bin(13),bin(17),'\n')
print(np.bitwise_and(13,17),'\n')
print(np.bitwise_or(13,17),'\n')
print(np.invert(np.array([13],dtype=np.uint8)))
```

```
0b1101 0b10001

1

29

[242]
```

6.Statistical Functions

In [52]:

```
import numpy as np
a=np.array([[30,65,70],[80,95,10],[50,90,60]])

print(np.amin(a))
print(np.amax(a))
print(np.median(a))
print(np.mean(a))
```

```
10
95
65.0
61.111111111111114
```

In [54]:

```
import numpy as np
a=np.array([1,2,3,4])
print(np.average(a,weights=[4,3,2,1]))

print(np.std([1,2,3,4]))
print(np.var([1,2,3,4]))
```

```
2.0
1.118033988749895
1.25
```

7.Sorting & Searching

In [55]:

```
import numpy as np
a=np.array([3,7,9,1])

print(np.sort(a))
print(np.argsort(a))
```

```
[1 3 7 9]
[3 0 1 2]
```

In [60]:

```
import numpy as np
x=np.arange(9.).reshape(3,3)

print(x,'\n')
y=np.where(x>3)
print(x[y])
```

```
[[0. 1. 2.]
 [3. 4. 5.]
 [6. 7. 8.]]

[4. 5. 6. 7. 8.]
```

8. Counting functions

In [63]:

```
import numpy as np
x=np.arange(9).reshape(3,3)

condition=np.mod(x,2)==0
print(condition)
print(np.extract(condition,x))
```

```
[[ True False  True]
 [False  True False]
 [ True False  True]]
[0 2 4 6 8]
```