In [1]:
```
import numpy as np
#Crea un array
a=np.arange(6)
print ('Arreglo a =',a,'\n')
print ('Tipo de a =',a.dtype,'\n')
print ('Dimension de a=',a.ndim,'\n')
print ('Número de elementos de a=',a.shape)
```

```
Arreglo a = [0 1 2 3 4 5]

Tipo de a = int32

Dimension de a= 1

Número de elementos de a= (6,)
```

In [2]:
```
#Arreglo multidimensional, se crea con la funcion array
m = np.array([np.arange(2),np.arange(2)])
print(m)
```

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

In [ ]:
```

```

In [4]:
```
a=np.array([[1,2],[3,4]])
print('a=\n',a,'\n')
print ('a[0,0]=',a[0,0],'\n')
print ('a[0,1]=',a[0,1],'\n')
print ('a[1,0]=',a[1,0],'\n')
print ('a[1,1]=',a[1,1])
```

```
a=
 [[1 2]
 [3 4]]

a[0,0]= 1

a[0,1]= 2

a[1,0]= 3

a[1,1]= 4
```

In [5]:
```
#Crea un array con 9 elementos, de 0 hasta 8
a = np.arange(9)
print ('a=',a,'\n')
print ('a[0:9]= ',a[0:9],'\n')
print ('a[3,7]=',a[3:7])
```

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

a[0:9]=  [0 1 2 3 4 5 6 7 8]

a[3,7]= [3 4 5 6]
```

In [ ]:
```

```

In [7]:
```
1 #Mostrar todos los elementos de 0 a 8 de uno en uno
      2 print('a[0:9:1]=',a[0:9:1],'\n')
----> 3 print('a[:9:1]=',a[:9:1],'\n'])
      4 print('a[0:9:2]=',a[0:9:2],'\n')
      5 print('a[0:9:3]=',a[0:9:3],'\n')
```

```
  File "<ipython-input-7-4bbee0582775>", line 2
    2 print('a[0:9:1]=',a[0:9:1],'\n')
    ^
IndentationError: unexpected indent
```

In [8]:
```
#Mostrar todos los elementos de 0 a 8 de uno en uno
print('a[0:9:1]=',a[0:9:1],'\n')
print('a[:9:1]=',a[:9:1],'\n'])
print('a[0:9:2]=',a[0:9:2],'\n')
print('a[0:9:3]=',a[0:9:3],'\n')
```

```
  File "<ipython-input-8-ce03aa235684>", line 3
    print('a[:9:1]=',a[:9:1],'\n'])
                                  ^
SyntaxError: closing parenthesis ']' does not match opening parenthesis '('
```

In [9]:
```
# de 0 a 8 de  uno en uno
print('a[0:9:1]=',a[0:9:1],'\n')
#mismo ejemplo omitiendo 0
print('a[:9:1]=',a[:9:1],'\n')
#de  2 en 2
print('a[0:9:2]=',a[0:9:2],'\n')
#de 3 en 3
print('a[0:9:3]=',a[0:9:3],'\n')
```

```
a[0:9:1]= [0 1 2 3 4 5 6 7 8]

a[:9:1]= [0 1 2 3 4 5 6 7 8]

a[0:9:2]= [0 2 4 6 8]

a[0:9:3]= [0 3 6]
```

In [10]:
```
#si utilizamos el incremento negativo el orden es inverso
print('a[9:0:-1]=',a[9:0:-1],'\n')
#Si se motien los valores de indice el resultado es preciso
print('a[::-1]=',a[::-1])
```

```
a[9:0:-1]= [8 7 6 5 4 3 2 1]

a[::-1]= [8 7 6 5 4 3 2 1 0]
```

In [11]:
```
#Arreglos mulridimensioinales
b=np.arange(24).reshape(2,3,4)
print('b=\n',b)
#reshape genera una matriz con 2 bloques, 3 filas
#y 4 columnas, numero total de elementos 24
```

```
b=
 [[[ 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 [13]:
```
#Acceso indivudal a los elemetnos del array
#Elemento en el bloque 1 fila,2,columna 3
print ('b[1,2,3]=',b[1,2,3],'\n')
#Elemento en el bloque 0 fila,2,columna 2
print ('b[0,2,2]=',b[0,2,2],'\n')
#Elemento en el bloque 0 fila,1,columna 1
print ('b[0,1,1]=',b[0,1,1],'\n')
```

```
b[1,2,3]= 23

b[0,2,2]= 10

b[0,1,1]= 5
```

In [14]:
```
#Mostraremos como generalizar una selección
#Elegimos componente fila 0 columna 0,bloque 0
print('b[0,0,0]=',b[0,0,0],'\n')
#lo mismo pero bloque 1
print('b[0,0,0]=',b[1,0,0],'\n')
#para elegir simultaneamente ambos elementos, lo hacemos utilizando 2puntos
print ('b[;,0,0]=',b[:0,0])
```

```
b[0,0,0]= 0

b[0,0,0]= 12

b[;,0,0]= []
```

In [15]:
```
#Si escribimos b[0] Habremos elegido el primer bloque
#Pero habriamos omitido las filas y las columnas
#En tal caso, numpy toma todas las filas y columnas
print ('b[0]=\n',b[0])
```

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

In [20]:
```
#Otra forma de b[0] es b[0,:,:]
#Los dos puntos sin ningun valor, indican que se utilizaran
#todos los terminos disponibles
#En este caso, filas y columnas
print ('b[0,:,:]=\n',b[0,:,:])
```

```
b[0,:,:]=
 [[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
```

In [18]:
```
#: a derecha o a izquierda se puede reemplazar por...
print ('b[0,...]=\n',b[0,...])
```

```
b[0,...]=
 [[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
```

In [19]:
```
#Si queremos la fila 1 en el bloque 0 sin importar las columnas
# Se tiene:
print ('b[0,1]=',b[0,1])
```

```
b[0,1]= [4 5 6 7]
```

In [21]:
```
#El resultado de una seleccion puede ser usado para después
#Se obtiene la fila 1 del blo que 0 y se asigna dicha respuesta
# a la varible z
z=b[0,1]
print ('z=',z,'\n')
#En este caso, Z = [4,5,6,7]
#si ahora queremos los valores de 2 en 2
print ('z[::2]=',z[::2])
```

```
z= [4 5 6 7]

z[::2]= [4 6]
```

In [22]:
```
#Imprimir todas las columnas sin importar bloques y fulas
print(b,'\n')
print('b[:,:,1]=\n',b[:,:,1],'\n')
#Variante de notación simplificado
print ('b[...,1] =\n',b[...,1])
```

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

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

b[:,:,1]=
 [[ 1  5  9]
 [13 17 21]]

b[...,1] =
 [[ 1  5  9]
 [13 17 21]]
```

In [24]:
```
#Para seeleccionar todas las filas 2, independiente de los bloques
#y columbas se tiene:
print(b,'\n')
print ('b[:,1]=',b[:,1])
```

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

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

b[:,1]= [[ 4  5  6  7]
 [16 17 18 19]]
```

In [25]:
```
#En el siguiente ejemplo seleccionamos la columna 1 del bloque 0
print(b,'\n')
print('b[0,:,1]=',b[0,:,1])
```

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

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

b[0,:,1]= [1 5 9]
```

In [27]:
```
#Si queremos seleccionar la ultima columna del primer bloque, tenemos:
print ('b[0,:,-1]',b[0,:,-1])
#0 = primer bloque
#-1 = ultima columna
#dos puntos, segunda posición, selecciona componentes de las filas
#que formarán parte de dicha columna
#en orden inverso seria..
print('b[0,::-1,-1]=',b[0,::-1,-1])
#::-1 invierte los valores que se hubieran seleccionado
#si en lugar de invertir quisieramos de 2 en 2
print ('b[0,::2,-1]=',b[0,::-1,-1])
```

```
b[0,:,-1] [ 3  7 11]
b[0,::-1,-1]= [11  7  3]
b[0,::2,-1]= [11  7  3]
```

In [28]:
```
#El array original
print(b,'\n------------------------\n')
#Invertir  bloques
print(b[::-1])
```

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

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

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

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

In [29]:
```
#La instrucción ravel(), de-constrruye el efecto de la instruccion:reshape
#Este es el array b en su estado matriz
print('Matriz b=\n',b,'\n--------------------\n')
#con ravel()generamos un vector desde la matriz
print('Vector b=\n',b.ravel())
```

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

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

Vector b=
 [ 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 [30]:
```
#La instrucción: flatten() es similar a ravel()
#La fierencia es que flatten genera un nuevo espacio de memoria
print('Vector b con flatten=\n',b.flatten())
```

```
Vector b con flatten=
 [ 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 [34]:
```
#Se puede cambiar la estructura de una matriz con la instrruccion shape
#Transformamos la matriz en 6 filas x4 columbas
b.shape=(6,4)
print('b(6x4)=\n',b)
```

```
b(6x4)=
 [[ 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 [35]:
```
#A partir de la matriz que acaba de ser generada, vamos a mostrar
#Como se construye la transpuesta de la matriz
#Matriz original
print ('b=\n',b,'\n----------------------\n')
#Transpuesta
print('Transpuesta de b=\n',b.transpose(),'\n------------\n')
```

```
b=
 [[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]
 [12 13 14 15]
 [16 17 18 19]
 [20 21 22 23]]
----------------------

Transpuesta de b=
 [[ 0  4  8 12 16 20]
 [ 1  5  9 13 17 21]
 [ 2  6 10 14 18 22]
 [ 3  7 11 15 19 23]]
------------
```

In [36]:
```
#Resize ejecuta una labora similar a reshape
#La diferencia está en que se resize altera la estructura del array
#reshape creea una copia del original
#por eso a reshape se debe asignar el resultado a una nueva variable

b.resize([2,12])
print('b=\n',b)
```

```
b=
 [[ 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 [ ]:
```

```