In [1]:
```
#MANUEL FELIPE VALENCIA CEBALLOS
#1004768150
import numpy as np

a=np.arange(15).reshape(5,3)
print('a=\n', a,'\n')
#matriz b desde a
b=a*2
print('b=\n',b)
```

```
a=
 [[ 0  1  2]
 [ 3  4  5]
 [ 6  7  8]
 [ 9 10 11]
 [12 13 14]]

b=
 [[ 0  2  4]
 [ 6  8 10]
 [12 14 16]
 [18 20 22]
 [24 26 28]]
```

In [3]:
```
#Apilamiento horizontal
print('Apilamiento horizontal=\n', np.hstack((a,b)))
```

```
Apilamiento horizontal=
 [[ 0  1  2  0  2  4]
 [ 3  4  5  6  8 10]
 [ 6  7  8 12 14 16]
 [ 9 10 11 18 20 22]
 [12 13 14 24 26 28]]
```

In [7]:
```
#Apilamiento horizontal-variable
print('Apilamiento horizontal con concatenate= \n',np.concatenate((a,b),axis=1
))
```

```
Apilamiento horizontal con concatenate=
 [[ 0  1  2  0  2  4]
 [ 3  4  5  6  8 10]
 [ 6  7  8 12 14 16]
 [ 9 10 11 18 20 22]
 [12 13 14 24 26 28]]
```

In [9]:

```
#Apilamiento vertical
print('Apilamient vertical=\n',np.vstack((a,b)))
```

```
Apilamient vertical=
 [[ 0  1  2]
 [ 3  4  5]
 [ 6  7  8]
 [ 9 10 11]
 [12 13 14]
 [ 0  2  4]
 [ 6  8 10]
 [12 14 16]
 [18 20 22]
 [24 26 28]]
```

In [11]:

```
#Apilamiento vertical-variante
print('Apilamiento vertical con concatenate=\n',np.concatenate((a,b),axis=0))
```

```
Apilamiento vertical con concatenate=
 [[ 0  1  2]
 [ 3  4  5]
 [ 6  7  8]
 [ 9 10 11]
 [12 13 14]
 [ 0  2  4]
 [ 6  8 10]
 [12 14 16]
 [18 20 22]
 [24 26 28]]
```

In [12]:

```
#Apilamiento en profundidad
#Se crean bloques utilizadno parejas de datos
print('Apilamiento en profundidad=\n',np.dstack((a,b)))
```

```
Apilamiento en profundidad=
 [[[ 0  0]
  [ 1  2]
  [ 2  4]]

 [[ 3  6]
  [ 4  8]
  [ 5 10]]

 [[ 6 12]
  [ 7 14]
  [ 8 16]]

 [[ 9 18]
  [10 20]
  [11 22]]

 [[12 24]
  [13 26]
  [14 28]]]
```

In [13]:

```
#Apilaimento por columnas
#orginales
print('a=\n', a,'\n')
print('b=\n',b)
#Apilamiento vertical
print('Apilamiento por columnas=\n',np.column_stack((a,b)))
```

```
a=
 [[ 0  1  2]
 [ 3  4  5]
 [ 6  7  8]
 [ 9 10 11]
 [12 13 14]]

b=
 [[ 0  2  4]
 [ 6  8 10]
 [12 14 16]
 [18 20 22]
 [24 26 28]]
Apilamiento por columnas=
 [[ 0  1  2  0  2  4]
 [ 3  4  5  6  8 10]
 [ 6  7  8 12 14 16]
 [ 9 10 11 18 20 22]
 [12 13 14 24 26 28]]
```

In [14]:

```
#Apilaiento pr filas
print('Apilamiento por filas=\n',np.row_stack((a,b)))
```

```
Apilamiento por filas=
 [[ 0  1  2]
 [ 3  4  5]
 [ 6  7  8]
 [ 9 10 11]
 [12 13 14]
 [ 0  2  4]
 [ 6  8 10]
 [12 14 16]
 [18 20 22]
 [24 26 28]]
```

In [22]:

```
#Division horizontal
print('Array con division horizontal=\n', np.hsplit(a,3),'\n')

print('Array con division horizontal, uso de split()=\n',np.split(a,3,axis=1))
```

```
Array con division horizontal=
 [array([[ 0],
       [ 3],
       [ 6],
       [ 9],
       [12]]), array([[ 1],
       [ 4],
       [ 7],
       [10],
       [13]]), array([[ 2],
       [ 5],
       [ 8],
       [11],
       [14]])]

Array con division horizontal, uso de split()=
 [array([[ 0],
       [ 3],
       [ 6],
       [ 9],
       [12]]), array([[ 1],
       [ 4],
       [ 7],
       [10],
       [13]]), array([[ 2],
       [ 5],
       [ 8],
       [11],
       [14]])]
```

In [24]:

```
#Division vertical
print('Division vertical=\n',np.vsplit(a,5),'\n')
print('Array con division vertical, uso de split()=\n',np.split(a,5,axis=0))
```

```
Division vertical=
 [array([[0, 1, 2]]), array([[3, 4, 5]]), array([[6, 7, 8]]), array([[ 9, 10,
11]]), array([[12, 13, 14]])]

Array con division vertical, uso de split()=
 [array([[0, 1, 2]]), array([[3, 4, 5]]), array([[6, 7, 8]]), array([[ 9, 10,
11]]), array([[12, 13, 14]])]
```

In [25]:

```
#Division de profundidad
c=np.arange(27).reshape(3,3,3)
print(c,'\n')
print('Division en profundidad=\n', np.dsplit(c,3),'\n')
```

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

 [[ 9 10 11]
  [12 13 14]
  [15 16 17]]

 [[18 19 20]
  [21 22 23]
  [24 25 26]]]

Division en profundidad=
 [array([[[ 0],
        [ 3],
        [ 6]],

       [[ 9],
        [12],
        [15]],

       [[18],
        [21],
        [24]]]), array([[[ 1],
        [ 4],
        [ 7]],

       [[10],
        [13],
        [16]],

       [[19],
        [22],
        [25]]]), array([[[ 2],
        [ 5],
        [ 8]],

       [[11],
        [14],
        [17]],

       [[20],
        [23],
        [26]]])]
```

In [26]:

```
#ndim calcula el nuero de dimensiones
print(b,'\n')
print('ndim: ',b.ndim)
```

```
[[ 0  2  4]
 [ 6  8 10]
 [12 14 16]
 [18 20 22]
 [24 26 28]]

ndim:  2
```

In [27]:

```
#size calcula el numero de elementos
print('size: ',b.size)
```

```
size:  15
```

In [28]:

```
#itemsize obtiene el numero de bytes por cada elemento en el array
print('itemsize: ', b.itemsize)
```

```
itemsize:  4
```

In [29]:

```
#nbytes numero total de bytes delarray
print('nbytes: ', b.nbytes,'\n')
#Es equivalente a:
print('nbytes equivalente: ', b.size*b.itemsize)
```

```
nbytes:  60

nbytes equivalente:  60
```

In [30]:

```
#T=transpuesta
print('Transpuesta: ', b.T)
```

```
Transpuesta:  [[ 0  6 12 18 24]
 [ 2  8 14 20 26]
 [ 4 10 16 22 28]]
```

In [32]:

```
#numreos complejos en numoy (j)
b = np.array([1.j + 1, 2.j + 3])
print('Complejo:\n',b)
```

```
Complejo:
 [1.+1.j 3.+2.j]
```

In [33]:

```
#numeros reales
print('real: ',b.real,'\n')
#imginarios
print('imaginarios: ',b.imag)
```

```
real:  [1. 3.]

imaginarios:  [1. 2.]
```

In [34]:

```
print(b.dtype)
```

```
complex128
```

In [38]:

```
b=np.arange(4).reshape(2,2)
print(b,'\n')

f=b.flat
print(f,'\n')

for item in f: print(item)
#seleccion e un elemento
print('\n')
print('Elemento 2: ',b.flat[2])

#Operaciones directas con flat
b.flat=7
print(b,'\n')

b.flat[[1,3]]=1
print(b,'\n')
```

```
[[0 1]
 [2 3]]

<numpy.flatiter object at 0x000002B111BF2160>

0
1
2
3


Elemento 2:  2
[[7 7]
 [7 7]]

[[7 1]
 [7 1]]
```

In [ ]: