

1 Nivell 1

Treballem els conceptes de l'estructura d'una matriu, dimensió, eixos i la vectorització que ens permet reduir l'ús de for loops en operacions aritmètiques o matemàtiques..

```
In [1]: import numpy as np
        executed in 223ms, finished 00:07:29 2021-04-09

In [2]: ### Crea un np.array d'una dimensió, que inclogui l'almenys 8 nombres sencers, data type int64.
        matriu = np.array([4,51,6,34,7,4,8,5,23,46], dtype="int64")
        executed in 10ms, finished 00:07:29 2021-04-09

In [3]: #Mostra La dimensió
        matriu.ndim
        executed in 129ms, finished 00:07:29 2021-04-09

Out[3]: 1

In [4]: # i La forma de La matriu.
        matriu.shape
        executed in 233ms, finished 00:07:29 2021-04-09

Out[4]: (10,)
```

```
In [5]: #De La matriu de L'exercici 1, calcula el valor mitjà dels valors introduïts
        np.mean(matriu)
        executed in 244ms, finished 00:07:29 2021-04-09

Out[5]: 18.8

In [6]: #i resta La mitjana resultant de cada un dels valors de La matriu
        matriu - np.median(matriu)
        executed in 152ms, finished 00:07:30 2021-04-09

Out[6]: array([-3.5, 43.5, -1.5, 26.5, -0.5, -3.5,  0.5, -2.5, 15.5, 38.5])

In [7]: #Crea una matriu bidimensional amb una forma de 5 x 5. Extreu el valor màxim de La matriu,
        matriu = np.array([[45, 17, 23, 5, 4], [37, 24, 1, 73, 6]])
        np.amax(matriu)
        executed in 121ms, finished 00:07:30 2021-04-09

Out[7]: 73

In [8]: #i els valors màxims de cadascun dels seus eixos.
        np.amax(matriu, axis=0), np.amax(matriu, axis=1)
        executed in 122ms, finished 00:07:30 2021-04-09

Out[8]: (array([45, 24, 23, 73,  6]), array([45, 73]))
```

2 Nivell 2

Treballem els conceptes de l'estructura d'una matriu, Broadcasting, indexació, Mask.

```
In [9]: #Mostreu-me amb exemples de diferents matrius, la regla fonamental de Broadcasting que diu :
        #"Les matrius es poden transmetre / broadcast si les seves dimensions coincideixen o si una de les matrius té una mida d'1"

        matriu1 = np.arange(1, 13).reshape(4,3)
        executed in 116ms, finished 00:07:30 2021-04-09

In [10]: #matriu2 de les mateixes dimensions:
        matriu2 = np.linspace(3, 25, 12).reshape(4,3)
        matriu1, matriu2
        executed in 122ms, finished 00:07:30 2021-04-09

Out[10]: (array([[ 1,  2,  3],
                  [ 4,  5,  6],
                  [ 7,  8,  9],
                  [10, 11, 12]]),
          array([[ 3.,  5.,  7.],
                  [ 9., 11., 13.],
                  [15., 17., 19.],
                  [21., 23., 25.]])

In [11]: matriu2 * matriu1
        executed in 140ms, finished 00:07:30 2021-04-09

Out[11]: array([[ 3., 10., 21.],
                 [36., 55., 78.],
                 [105., 136., 171.],
                 [210., 253., 300.]])
```

```
In [12]: #matriu2 amb mateixa longitud de files
matriu2 = np.linspace(3, 25, 4).reshape(4,1)
matriu1, matriu2
executed in 120ms, finished 00:07:30 2021-04-09
```

```
Out[12]: (array([[ 1,  2,  3],
                  [ 4,  5,  6],
                  [ 7,  8,  9],
                  [10, 11, 12]]),
          array([[ 3.          ],
                  [10.33333333],
                  [17.66666667],
                  [25.          ]]))
```

```
In [13]: matriu2 / matriu1
executed in 120ms, finished 00:07:30 2021-04-09
```

```
Out[13]: array([[3.          , 1.5          , 1.          ],
                  [2.58333333, 2.06666667, 1.72222222],
                  [2.52380952, 2.20833333, 1.96296296],
                  [2.5          , 2.27272727, 2.08333333]])
```

```
In [14]: #matriu2 de mida d'1 (amb tres dimensions)
matriu2 = np.array([[[4]]])
matriu1, matriu2
executed in 121ms, finished 00:07:31 2021-04-09
```

```
Out[14]: (array([[ 1,  2,  3],
                  [ 4,  5,  6],
                  [ 7,  8,  9],
                  [10, 11, 12]]),
          array([[[4]]]))
```

```
In [15]: matriu2 + matriu1
executed in 119ms, finished 00:07:31 2021-04-09
```

```
Out[15]: array([[[ 5,  6,  7],
                  [ 8,  9, 10],
                  [11, 12, 13],
                  [14, 15, 16]])])
```

```
In [16]: #Utilitza La Indexació per extreure els valors d'una columna i una fila de la matriu. I suma els seus valors
matriu = np.arange(2, 11).reshape(3,3)
matriu
executed in 133ms, finished 00:07:31 2021-04-09
```

```
Out[16]: array([[ 2,  3,  4],
                  [ 5,  6,  7],
                  [ 8,  9, 10]])
```

```
In [17]: columna2 = matriu[:, 1]
fila3 = matriu[2, :]
columna2 + fila3
executed in 105ms, finished 00:07:31 2021-04-09
```

```
Out[17]: array([11, 15, 19])
```

```
In [18]: #Mask la matriu anterior, realitzeu un càlcul booleà vectoritzat, agafant cada element i
#comprovant si es divideix uniformement per quatre.
#Això retorna una matriu de mask de la mateixa forma amb els resultats elementals del càlcul.

mask = (matriu % 4 == 0)
mask
executed in 129ms, finished 00:07:31 2021-04-09
```

```
Out[18]: array([[False, False,  True],
                  [False, False, False],
                  [ True, False, False]])
```

```
In [19]: #A continuació, utilitzeu aquesta màscara per indexar a la matriu de números original.
#Això fa que la matriu perdi la seva forma original, reduint-la a una dimensió, però encara obteniu les dades que esteu cercant
matriu[mask]
executed in 121ms, finished 00:07:31 2021-04-09
```

```
Out[19]: array([4, 8])
```

3 Nivell 3

Manipulació d'imatges amb Matplotlib.

Carregareu qualsevol imatge (jpg, png ..) amb Matplotlib. adoneu-vos que les imatges RGB (Red, Green, Blue) són realment només amplades × alçades × 3 matrius (tres canals Vermell, Verd i Blau), una per cada color de nombres enters int8,

manipuleu aquests bytes i torneu a utilitzar Matplotlib per desar la imatge modificada un cop hàgiu acabat.

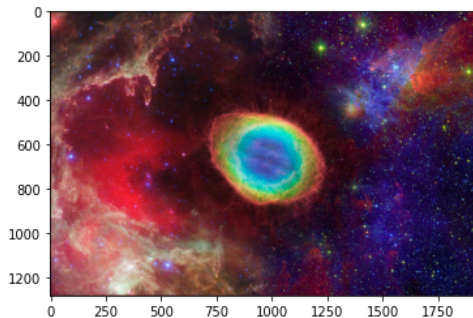
Ajuda:Importeu, import matplotlib.image as mpimg. estudeu el metodelle mpimg.imread()

```
In [20]: import matplotlib.pyplot as plt
import matplotlib.image as mpimg
executed in 532ms, finished 00:07:32 2021-04-09
```

```
In [21]: img = mpimg.imread('imatge.jpg')
img.shape, img.dtype
executed in 142ms, finished 00:07:32 2021-04-09
```

Out[21]: ((1280, 1920, 3), dtype('uint8'))

```
In [22]: imgplot = plt.imshow(img)
executed in 1.04s, finished 00:07:33 2021-04-09
```

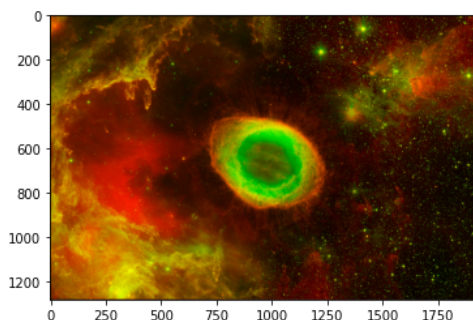


```
In [23]: #Mostreu-me a veure què passa quan eliminem el canal G Verd o B Blau.
#Hauries d'utilitzar la indexació per seleccionar el canal que voleu anul·lar.

img_modif = img.copy()

#eliminaré el blau b=0
img_modif[:, :, 2]=0
executed in 15ms, finished 00:07:33 2021-04-09
```

```
In [24]: imgplot = plt.imshow(img_modif)
executed in 1.13s, finished 00:07:34 2021-04-09
```



```
In [25]: #Utilitzar el mètode, mpimg.imsave () de la llibreria importada, per guardar les imatges modificades
#i que haureu de pujar al vostre repositori a github.
plt.imsave("imatge_modif.jpg", img_modif)
executed in 199ms, finished 00:07:34 2021-04-09
```