S03_T02_Estructura_Matriu

October 31, 2021

1 *S03 T02: Estructura d'una Matriu*

Anem a practicar i a familiaritzar-nos amb l'estructura de Matrius, dimensió, forma, vectorització i Broadcasting

Documentación:

- Numpy: https://www.w3schools.com/python/numpy/numpy_intro.asp
- Broadcasting: https://numpy.org/doc/stable/user/basics.broadcasting.html?highlight=broadcasting
- Indexing: https://numpy.org/doc/stable/user/basics.indexing.html?highlight=indexing
- Masking: https://numpy.org/doc/stable/reference/maskedarray.generic.html
- Matplotlib Image: https://matplotlib.org/stable/tutorials/introductory/images.html

Librerias:

[1]:
```
import numpy as np
from numpy import random
import numpy.ma as ma
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
```

---

2 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.

2.1 Exercici 1

- Crea un np.array d'una dimensió, que inclogui l'almenys 8 nombres sencers, data type int64. Mostra la dimensió i la forma de la matriu.

[2]:
```
num = random.randint(8,15) # definimos un numero entre 8 y 15.
array2=random.randint(100, size=(num)) #definimos array con numero d 0 a 99 con␣
 ↪un tamoño de num. y una dimensión.

print(array2)
```

```
print(array2.dtype) # comprobamos el tipo

array64=array2.astype("int64") #lo convertimos a int64

print(array64.dtype)

print(f'Cabiamos el tipo de la matriz a cambiado de int32 a int 64')

print(f'\nLa matriz tiene {array64.ndim} dimensiones')
print(f'La matriz tiene la forma {array64.shape}, {len(array64)} elementos en␣
↪{array64.ndim} dimensión')
```

```
[61 49  9 46  8 40 75 15]
int32
int64
Cabiamos el tipo de la matriz a cambiado de int32 a int 64

La matriz tiene 1 dimensiones
La matriz tiene la forma (8,), 8 elementos en 1 dimensión
```

2.2 Exercici 2

- De la matriu de l'exercici 1, calcula el valor mitjà dels valors introduïts i resta la mitjana resultant de cada un dels valors de la matriu.

[3]:
```
mean= array64.mean() #definimos la media

# dos formaas de realizar la resta

print(np.subtract(array64, mean))
print("")
print(array64-mean)
```

```
[ 23.125  11.125 -28.875   8.125 -29.875   2.125  37.125 -22.875]

[ 23.125  11.125 -28.875   8.125 -29.875   2.125  37.125 -22.875]
```

2.3 Exercici 3

- Crea una matriu bidimensional amb una forma de 5 x 5. Extreu el valor màxim de la matriu, i els valors màxims de cadascun dels seus eixos.

[4]:
```
myArray=random.randint(100,size=(5,5))

print(f'Creamos una matriz con forma {myArray.shape} y {myArray.ndim}␣
↪dimensiones. \n')

print(myArray) # imprimimos array.
```

```
print(f'\nEl valor máximo de la matriz es {myArray.max()}.')
print(f'\nLos valores máximos en el eje vertical son {myArray.max(axis=0)}.')
print(f'\nLos valores máximos en el eje horizontal son {myArray.max(axis=1)}.')
```

```
Creamos una matriz con forma (5, 5) y 2 dimensiones.

[[77 41 87 66 99]
 [85 24 30 76 87]
 [88 86 34 51 56]
 [63 60 28 58 53]
 [57 80  6 85 38]]

El valor máximo de la matriz es 99.

Los valores máximos en el eje vertical son [88 86 87 85 99].

Los valores máximos en el eje horizontal son [99 87 88 63 85].
```

---

3 Nivell 2

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

3.1 Exercici 4

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".

Les matrius es poden trasnmetre si les seves dimension 1. Les seves dimension són iguals. 2. Una de les dimensions és 1.

Les matrius tenen una dimension igual i la altra es 1

[5]:
```
x=np.random.randint(0,10, size=(3,3))
y=np.random.randint(0,10, size=(3,1))

print(x)
print("")
print(y)

print('Realizamos suma y producto')
suma=x+y
prod=x*y
print("")
```

```
print(suma)
print("")
print(prod)
```

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

[[2]
 [6]
 [7]]
Realizamos suma y producto

[[ 8  6  2]
 [ 6  9  7]
 [12 15  8]]

[[12  8  0]
 [ 0 18  6]
 [35 56  7]]
```

[6]:
```
z=np.random.randint(0,10, size=(1,3))
print(x)
print("")
print(z)

print('Realizamos suma y producto')
suma=x+z
prod=x*z
print("")
print(suma)
print("")
print(prod)
```

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

[[4 0 4]]
Realizamos suma y producto

[[10  4  4]
 [ 4  3  5]
 [ 9  8  5]]

[[24  0  0]
 [ 0  0  4]
 [20  0  4]]
```

[7]:
```
z=np.random.randint(0,10, size=(1,3))
print(x)
print("")
print(z)

print('Realizamos suma y producto')
suma=x+z
prod=x*z
print("")
print(suma)
print("")
print(prod)
```

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

[[4 9 0]]
Realizamos suma y producto

[[10 13  0]
 [ 4 12  1]
 [ 9 17  1]]

[[24 36  0]
 [ 0 27  0]
 [20 72  0]]
```

[8]:
```
w=np.random.randint(0,10, size=(3,2))
print(x)
print("")
print(w)

print('Realizamos suma y producto')

try:
    suma=x+w
    prod=x*w
    print("")
    print(suma)
    print("")
    print(prod)

except:
    print("ValueError: operands could not be broadcast together with shapes␣
 ↪(3,3) (3,2)")
```

```
[[6 4 0]
```

```
 [0 3 1]
 [5 8 1]]

[[2 5]
 [5 7]
 [0 6]]
Realizamos suma y producto
ValueError: operands could not be broadcast together with shapes (3,3) (3,2)
```

Es produeix un error pel fet que les dimensions no són iguals entre si i diferents de la unitat. No es poden transmetre matrius amb diferents formes.

ValueError: operands could not be broadcast together with shapes (3,3) (3,2)

3.2 Exercici 5

Utilitza la Indexació per extreure els valors d'una columna i una fila de la matriu. I suma els seus valors.

[9]:
```
print(myArray)

print("")
print(myArray[0])

print(f'La primera fila de la matriz es {myArray[0]}.\n')
print(myArray[:,0])
print(f'La primera columna de la matriz es {myArray[:,0]}.\n')

suma=myArray[0]+myArray[:,0]

print(f'La suma de la primera columna con la primera fila de la matriz es␣
 ↪{suma}, una matriz de una dimensión.')
```

```
[[77 41 87 66 99]
 [85 24 30 76 87]
 [88 86 34 51 56]
 [63 60 28 58 53]
 [57 80  6 85 38]]

[77 41 87 66 99]
La primera fila de la matriz es [77 41 87 66 99].

[77 85 88 63 57]
La primera columna de la matriz es [77 85 88 63 57].

La suma de la primera columna con la primera fila de la matriz es [154 126 175
129 156], una matriz de una dimensión.
```

3.3 Exercici 6

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.

[10]:
```
# Utilizamos la matriz myArray
print(myArray)
print("")
mask=(myArray % 4 ==0)

print(mask)
```

```
[[77 41 87 66 99]
 [85 24 30 76 87]
 [88 86 34 51 56]
 [63 60 28 58 53]
 [57 80  6 85 38]]

[[False False False False False]
 [False  True False  True False]
 [ True False False False  True]
 [False  True  True False False]
 [False  True False False False]]
```

3.4 Exercici 7

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.

[11]:
```
print(myArray[mask])
div4=(myArray[mask])

print(f'Los numeros divisibles por 4 de la matriz myArray son {div4}.')
```

```
[24 76 88 56 60 28 80]
Los numeros divisibles por 4 de la matriz myArray son [24 76 88 56 60 28 80].
```

3.4.1 *alternativamente*

Alternativamente podemos utilizar el modulo numpy Maskered arrays donde la forma se conserva.

[12]:
```
myArray2=ma.masked_array(myArray,mask=myArray % 4) # el valor 0 se considerará␣
 ↪booliano false

print('Imprimimos los valores divisibles entre 4 con las misma forma que la␣
 ↪matriz original: \n')
print(myArray2)
```

```
print('\n La mascara:')
print(mask)
print('\nPodemos recuperar los valores que cumplen la condicion en una matriz␣
 ↪de una dimensión (lista).')
print(myArray2.compressed())
```

```
Imprimimos los valores divisibles entre 4 con las misma forma que la matriz
original:

[[-- -- -- -- --]
 [-- 24 -- 76 --]
 [88 -- -- -- 56]
 [-- 60 28 -- --]
 [-- 80 -- -- --]]

 La mascara:
[[False False False False False]
 [False  True False  True False]
 [ True False False False  True]
 [False  True  True False False]
 [False  True False False False]]

Podemos recuperar los valores que cumplen la condicion en una matriz de una
dimensión (lista).
[24 76 88 56 60 28 80]
```

---

4 Nivel 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. estudieu el metodde mpimg.imread(()

4.1 Exercici 8

Mostreu-me a veure que passa quan eliminem el canal G Verd o B Blau.

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.

Utilitzar el mètode, mpimg.imsave () de la llibreria importada, per guardar les imatges modificades i que haureu de pujar al vostre repositori a github.

[13]:
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
```

[14]:
```
img = mpimg.imread('./imagen.jpeg')
```

[15]:
```
img2=img.copy()
```

[16]:
```
img2
```

[16]:
```
array([[[ 13,  11,  12],
        [ 17,  15,  16],
        [ 23,  21,  22],
        …,
        [ 69,  69,  69],
        [ 68,  68,  68],
        [ 68,  68,  68]],

       [[ 12,  10,  11],
        [ 17,  15,  16],
        [ 21,  19,  20],
        …,
        [ 69,  69,  69],
        [ 69,  69,  69],
        [ 69,  69,  69]],

       [[ 12,  10,  11],
        [ 16,  14,  15],
        [ 20,  18,  19],
        …,
        [ 69,  69,  69],
        [ 69,  69,  69],
        [ 69,  69,  69]],

       …,

       [[247, 253, 253],
        [247, 253, 253],
        [247, 253, 253],
        …,
        [150, 144, 122],
        [107, 101,  79],
        [ 71,  65,  43]],

       [[247, 253, 253],
        [247, 253, 253],
        [247, 253, 253],
        …,
```

```
        [123, 117,  95],
        [134, 128, 106],
        [ 90,  84,  62]],

       [[247, 253, 253],
        [247, 253, 253],
        [247, 253, 253],
        …,
        [109, 103,  81],
        [126, 120,  98],
        [121, 115,  93]]], dtype=uint8)
```

[17]:
```
img2.shape
```

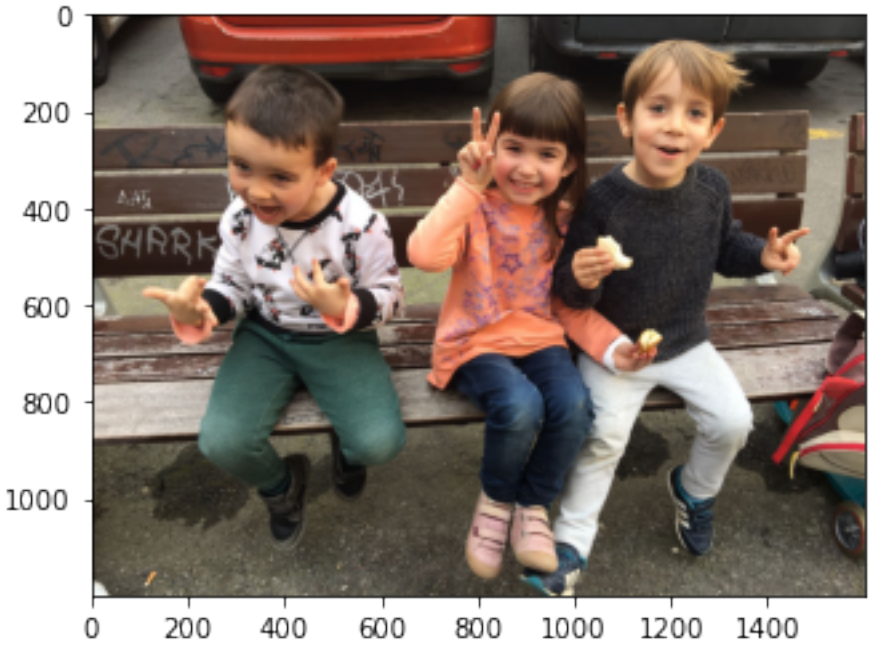
[17]:
```
(1200, 1600, 3)
```

pruebas

[18]:
```
# primera
print(img2[0,0,:])
# ultima
print(img2[-1,-1,:])
# medio
print(img2[599,799,:])
```

```
[13 11 12]
[121 115  93]
[185 111  98]
```

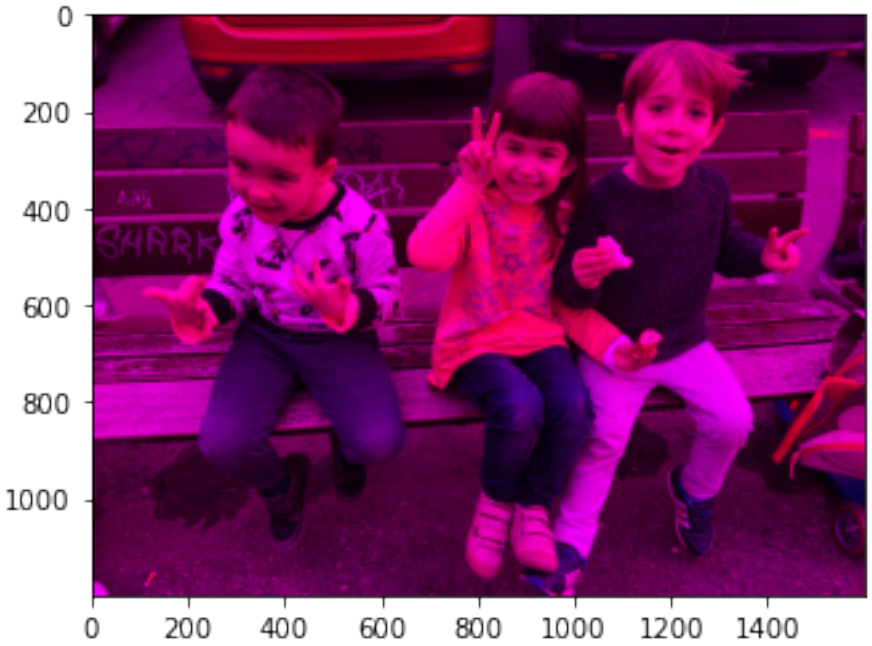
[19]:
```
imgplot = plt.imshow(img2)
```



Eliminem el canal G Verd:

[20]:
```
img2[:,:,1]=0
plt.imshow(img2)
```

[20]:
```
<matplotlib.image.AxesImage at 0x18f480b35b0>
```



[21]:
```
mpimg.imsave ("imagen_sin_Verde.jpg",img2) # grabamos la imagen
```

Eliminem el canal B blau:

[22]:
```
img3=img.copy()
img3[:,:,2]=0
plt.imshow(img3)
```

[22]:
```
<matplotlib.image.AxesImage at 0x18f4810ef10>
```

[23]:
```
mpimg.imsave ("imagen_sin_Azul.jpg",img3) # grabamos la imagen
```