Sprint 2_Matrix Structure

June 7, 2022

1 IT Academy - Data Science

1.1 S02 T02: Estructura d'una Matriu

1.1.1 Level 1

1.1.2 *Exercise 1*

Create a one-dimensional np.array, including at least 8 integers, data type int64. Displays the dimension and shape of the array.

[26]:
```
#import numpy library
import numpy as np

#create random column vector with 15 rows that takes values up to 100
arr1 = np.random.randint(100, size=(1,15), dtype="int64")
print(arr1)
```

```
[[35 40  3  0 37 27 99 40  4 76 71 96  4 46 66]]
```

[27]:
```
#display dimension
print(arr1.ndim)
```

```
2
```

[28]:
```
#display shape
print(arr1.shape)
```

```
(1, 15)
```

1.1.3 *Exercise 2*

From the matrix in Exercise 1, calculate the average of the values entered

[47]:
```
#calculate average of elements of the array
print(np.array(np.mean(arr1), dtype='f2'))
```

```
42.94
```

1.1.4 *Exercise 3*

Create a two-dimensional array with a shape of 5 x 5.

[37]:
```
#create a random 2D square matrix (5 rows and 5 columns)
arr2 = np.random.randint(100, size=(5,5))
print(arr2)
```

```
[[89 70 20 41  0]
 [15 86 95 10 54]
 [67 80 18 16 96]
 [30 68 91 31 42]
 [83 88 65 34 96]]
```

Extract the maximum value of the array.

[38]:
```
#maximum value of the array
print(np.amax(arr2))
```

```
96
```

Extract the maximum values of each of its axes.

[39]:
```
#find maximum value along axis = 0
arr2_max0 = np.amax(arr2, axis = 0)
print('Maximum values of the array along axis=0 is:', arr2_max0)
#find maximum value along axis=1
arr2_max1 = np.amax(arr2, axis=1)
print('Maximum values of the array along axis=1 is:', arr2_max1)
```

```
Maximum values of the array along axis=0 is: [89 88 95 41 96]
Maximum values of the array along axis=1 is: [89 95 96 91 96]
```

1.1.5 Level 2

1.1.6 *Exercise 4*

Show with examples of different arrays, the Broadcasting rule of thumb: "Arrays can be broadcast if their dimensions match or if one of the arrays has a size of 1."

[42]:
```
#show the broadcasting rule with matrixes of same dimension
arr3 = np.random.randint(100, size=(4,4))
arr4 = np.random.randint(100, size=(4,4))

print (arr3 + arr4)
```

```
[[132 113 122  40]
 [ 83 128  78  98]
 [ 73 155 102 111]
 [113 163  43  72]]
```

[43]:
```
#show the broadcasting rule with one matrix and one column vector with same␣
 ↪number of rows
arr5 = np.random.randint(100, size=(4,1))

print(arr3 * arr5)
```

```
[[6279 3549 6279 1911]
 [1288 7544 3312 4508]
 [2926 5005 1925 2310]
 [7161 7626 3627 1395]]
```

[44]:
```
#show the broadcasting rule with one matrix and one scalar
x = np.random.randint(100)

print(np.array(arr3/x, dtype='f2'))
```

```
[[1.15   0.65   1.15   0.35  ]
 [0.2333 1.366  0.6    0.817 ]
 [0.6333 1.083  0.4167 0.5   ]
 [1.283  1.366  0.65   0.25  ]]
```

1.1.7 *Exercise 5*

Use Indexing to extract the values of a column and a row from the array.

[48]:
```
#show array
print(arr3)
```

```
[[69 39 69 21]
 [14 82 36 49]
 [38 65 25 30]
 [77 82 39 15]]
```

[49]:
```
#show elements of column 3
y = arr3[:, 2]
print(y)
```

```
[69 36 25 39]
```

[50]:
```
#show elements of row 4
z = arr3[3, :]
print(z)
```

```
[77 82 39 15]
```

And add up their values.

[51]:
```
#sum by elements of column and row selected
print(y+z)
```

```
[146 118  64  54]
```

1.1.8 *Exercise 6*

Mask the above matrix, perform a vectorized Boolean calculation, taking each element and checking if it is evenly divided by four.

This returns a mask array in the same way as the elementary results of the calculation.

[52]:
```
#show array
print(arr3)
```

```
[[69 39 69 21]
 [14 82 36 49]
 [38 65 25 30]
 [77 82 39 15]]
```

[53]:
```
mask = (arr3 % 4 == 0)
print(mask)
```

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

1.1.9 *Exercise 7*

Then use this mask to index the original number array. This causes the array to lose its original shape, reducing it to one dimension, but you still get the data you are looking for.

[54]:
```
print(arr3[mask])
```

```
[36]
```

1.1.10 Level 3

1.1.11 *Exercise 8*

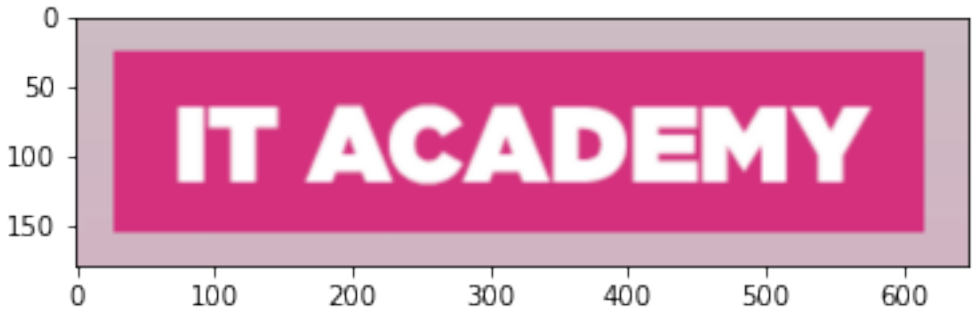
Carregareu qualsevol imatge (jpg, png ..) amb Matplotlib.

[55]:
```
# importing required libraries
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
```

[57]:
```
# reading the image
image = mpimg.imread('ITacademy.png')
```

[58]:
```
# displaying the image
plt.imshow(image)
```

[58]: <matplotlib.image.AxesImage at 0x7fa3c35e8160>



[59]:
```
# displaying the image as an array
print(image)
```

```
[[[0.8039216  0.7372549  0.7647059  1.        ]
  [0.8039216  0.7372549  0.7647059  1.        ]
  [0.8039216  0.7372549  0.7647059  1.        ]
  …
  [0.8039216  0.7372549  0.7647059  1.        ]
  [0.8039216  0.7372549  0.7647059  1.        ]
  [0.8039216  0.7372549  0.7647059  1.        ]]

 [[0.8039216  0.7372549  0.7647059  1.        ]
  [0.8039216  0.7372549  0.7647059  1.        ]
  [0.8039216  0.7372549  0.7647059  1.        ]
  …
  [0.8039216  0.7372549  0.7647059  1.        ]
  [0.8039216  0.7372549  0.7647059  1.        ]
  [0.8039216  0.7372549  0.7647059  1.        ]]

 [[0.8039216  0.7372549  0.7647059  1.        ]
  [0.8039216  0.7372549  0.7647059  1.        ]
  [0.8039216  0.7372549  0.7647059  1.        ]
  …
  [0.8039216  0.7372549  0.7647059  1.        ]
  [0.8039216  0.7372549  0.7647059  1.        ]
  [0.8039216  0.7372549  0.7647059  1.        ]]

 …

 [[0.81960785 0.7058824  0.7607843  1.        ]
  [0.81960785 0.7058824  0.7607843  1.        ]
  [0.81960785 0.7058824  0.7607843  1.        ]
  …
```

```
  [0.81960785 0.7058824  0.7607843  1.        ]
  [0.81960785 0.7058824  0.7607843  1.        ]
  [0.81960785 0.7058824  0.7607843  1.        ]]

 [[0.81960785 0.7058824  0.7607843  1.        ]
  [0.81960785 0.7058824  0.7607843  1.        ]
  [0.81960785 0.7058824  0.7607843  1.        ]
  …
  [0.81960785 0.7058824  0.7607843  1.        ]
  [0.81960785 0.7058824  0.7607843  1.        ]
  [0.81960785 0.7058824  0.7607843  1.        ]]

 [[0.81960785 0.7058824  0.7607843  1.        ]
  [0.81960785 0.7058824  0.7607843  1.        ]
  [0.81960785 0.7058824  0.7607843  1.        ]
  …
  [0.81960785 0.7058824  0.7607843  1.        ]
  [0.81960785 0.7058824  0.7607843  1.        ]
  [0.81960785 0.7058824  0.7607843  1.        ]]]
```

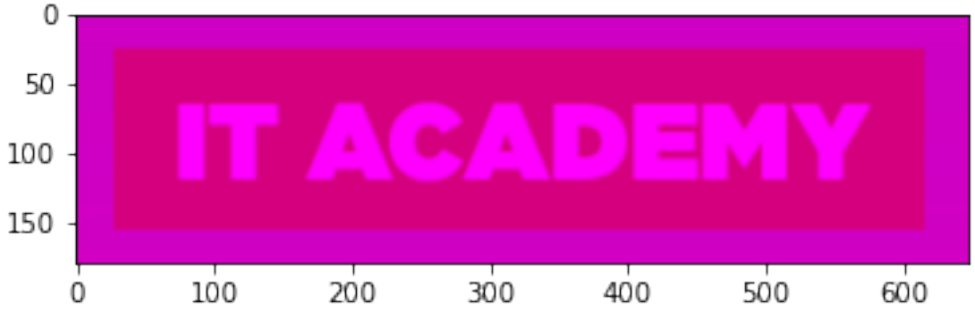
[60]:
```
# display the shape of the image: height, width, mode
print(image.shape)
```

```
(180, 648, 4)
```

[61]:
```
# create copy of the image to modify the mode
new_image = image.copy()
```

[62]:
```
#Show what happens when the Green G or Blue B channel is removed.
# mode of the image green modified
new_image[:, :, 1]=0
plt.imshow(new_image)
```

[62]: <matplotlib.image.AxesImage at 0x7fa3c369a1c0>



[63]:
```
#Use the mpimg.imsave () method of the imported library to save the modified␣
 ↪images
#and upload them to your repository on github.
plt.imsave("new_image.jpg", new_image)
```