



**UNIVERSITÀ DEGLI STUDI
DELLA BASILICATA**

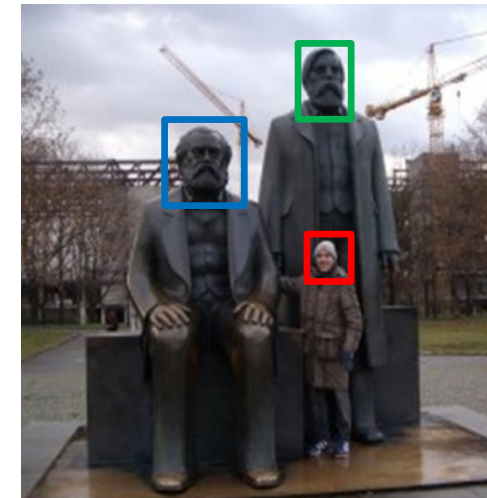
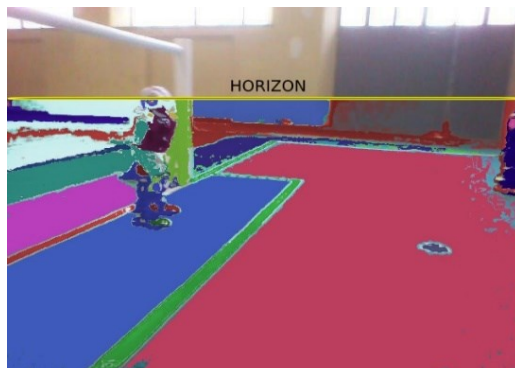
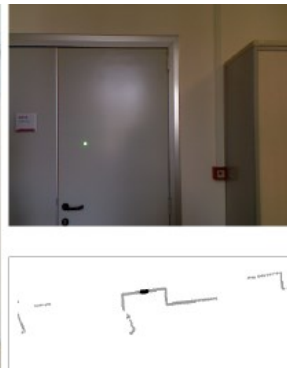
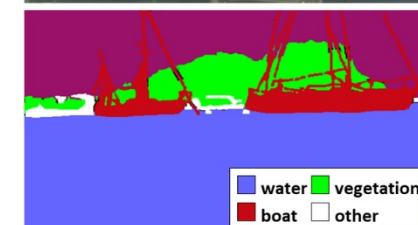
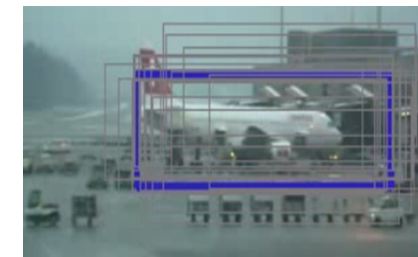
Corso di Visione e Percezione

Processamento delle immagini



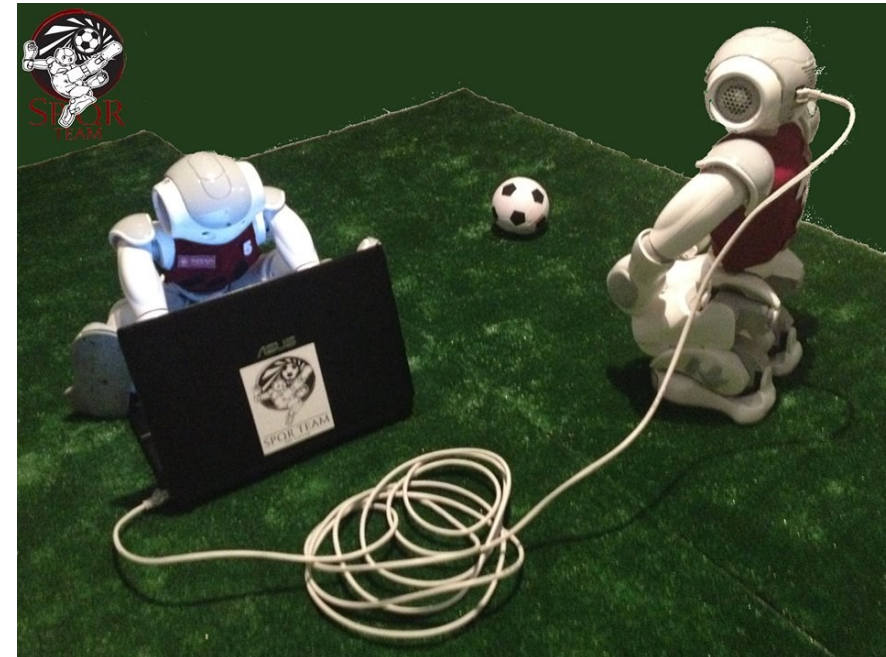
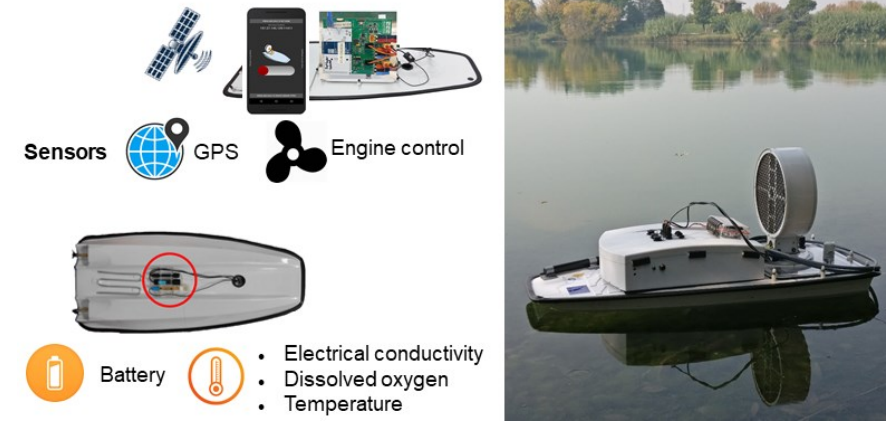
Docente

Domenico D. Bloisi



Domenico Daniele Bloisi

- Ricercatore RTD B
Dipartimento di Matematica, Informatica
ed Economia
Università degli studi della Basilicata
<http://web.unibas.it/bloisi>
- SPQR Robot Soccer Team
Dipartimento di Informatica, Automatica
e Gestionale Università degli studi di
Roma “La Sapienza”
<http://spqr.diag.uniroma1.it>



Informazioni sul corso

- Home page del corso
<http://web.unibas.it/bloisi/corsi/visione-e-percezione.html>
- Docente: Domenico Daniele Bloisi
- Periodo: **Il semestre** marzo 2021 – giugno 2021

Martedì 17:00-19:00 (Aula COPERNICO)

Mercoledì 8:30-10:30 (Aula COPERNICO)



Codice corso Google Classroom:

<https://classroom.google.com/c/NjI2MjA4MzgZNDFa?cjc=xgolays>

Ricevimento

- Su appuntamento tramite Google Meet

Per prenotare un appuntamento inviare
una email a

domenico.bloisi@unibas.it



Programma – Visione e Percezione

- Introduzione al linguaggio Python
- Elaborazione delle immagini con Python
- Percezione 2D – OpenCV
- Introduzione al Deep Learning
- ROS
- Il paradigma publisher and subscriber
- Simulatori
- Percezione 3D - PCL

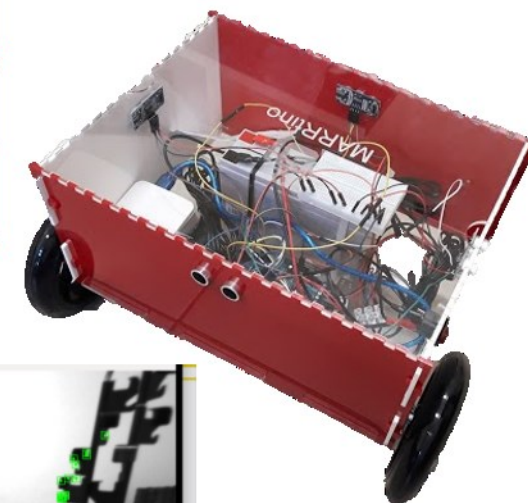
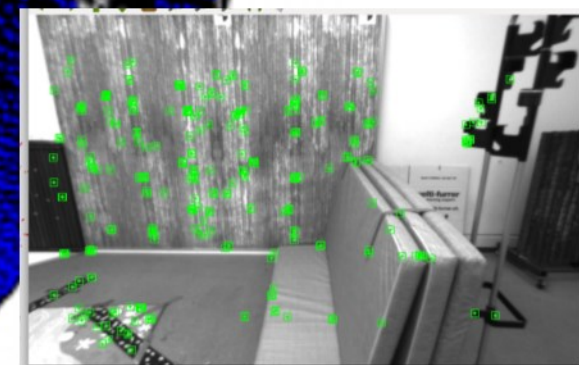
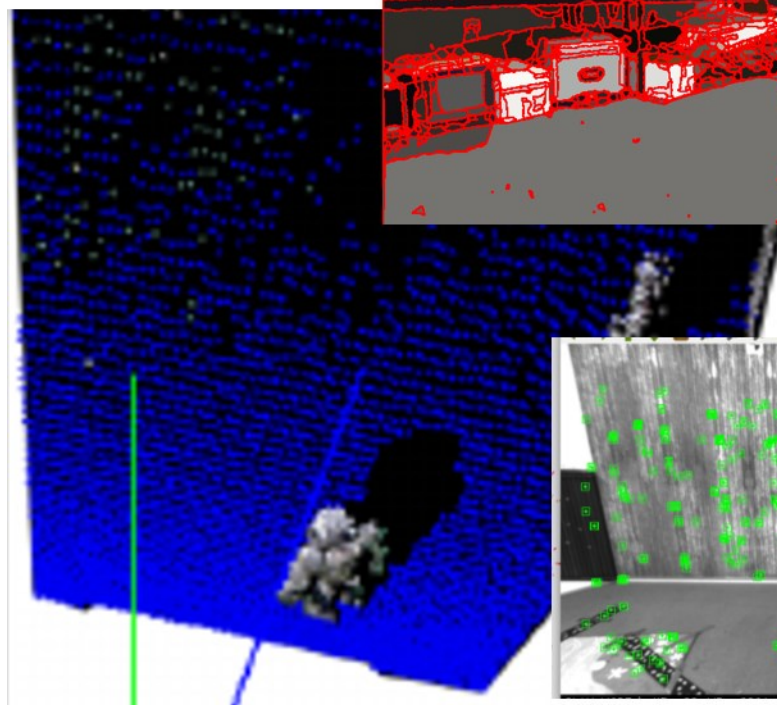


Immagine Digitale

- Una immagine digitale è una matrice di pixel
- Il termine pixel deriva da *picture element*
- Il pixel contiene l'informazione relativa alla rappresentazione della realtà che è stata catturata tramite uno scanner, una macchina fotografica o un frame grabber (per i video)



But the camera sees this:

194	210	201	212	199	213	215	195	178	158	182	209
180	189	190	221	209	205	191	167	147	115	129	163
114	126	140	188	176	165	152	140	170	106	78	88
87	103	115	154	143	142	149	153	173	101	57	57
102	112	106	131	122	138	152	147	128	84	58	66
94	95	79	104	105	124	129	113	107	87	69	67
68	71	69	98	89	92	98	95	89	88	76	67
41	56	68	99	63	45	60	82	58	76	74	65
20	41	69	75	56	41	51	73	55	70	63	44
50	50	57	69	75	75	73	74	53	68	59	37
72	59	53	66	84	92	84	74	57	72	63	42
67	61	58	65	75	78	76	73	59	75	69	50

Processamento delle immagini

- Per poter elaborare il contenuto di una immagine, avremo bisogno di caricarla in memoria per poter accedere ai suoi elementi e modificarli.
- Una volta terminate le modifiche, potremmo voler salvare l'immagine modificata su disco.
- Per poter processare le immagini utilizzeremo delle librerie esterne.

La libreria NumPy

- NumPy è una libreria per il calcolo scientifico in Python



<http://www.numpy.org/>

- NumPy è inclusa in Google Colab e viene rilasciata sotto la [BSD license](#)
- La utilizzeremo principalmente per la gestione degli array N-dimensionali e per la definizione di nuovi tipi di dato

Array in NumPy

- Un array in NumPy è una griglia di valori, tutti dello stesso tipo
- Gli array sono indicizzati
- La classe array in NumPy è chiamata `ndarray`. Per creare un ndarray viene utilizzata la funzione `array()`

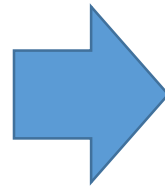
Array in NumPy

```
import numpy as np  
  
a = np.array([1,2,3])  
  
print(a)
```

```
[1 2 3]
```

```
print(type(a))
```

```
<class 'numpy.ndarray'>
```



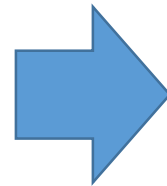
a

1
2
3

Array multidimensionali

```
import numpy as np  
  
b = np.array([[1,2],[3,4]])  
  
print(b)
```

```
[[1 2]  
 [3 4]]
```



b

1	2
3	4

Rank e shape in NumPy

- In NumPy le dimensioni di un array sono chiamate *axes*
- Il numero di axes è chiamato *rank*. Il *rank* (che è memorizzato nella variabile `ndim`) rappresenta la dimensione dell'array
- La *shape* è una tupla di interi che fornisce la lunghezza dell'array lungo ogni dimensione

Rank e shape



```
import numpy as np

a = np.array([1, 2, 3])
print(a.ndim) #rank
print(a.shape)

b = np.array([[1,2,3],[4,5,6]])
print(b.ndim)
print(b.shape)
```

```
1
(3,)
2
(2, 3)
```

a

1
2
3

Rank: 1
Shape: (3,)

Si usa (3,) per indicare che si tratta di una tupla con un solo elemento, diversa da (3) che è il numero 3

b

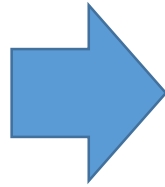
1	2	3
4	5	6

Rank: 2
Shape: (2, 3)

Funzioni per inizializzare array

```
▶ z = np.zeros(3)  
print(z)
```

```
☐➔ [0. 0. 0.]
```



z

0.
0.
0.

```
▶ o = np.ones(3)  
print(o)
```

```
☐➔ [1. 1. 1.]
```



o

1.
1.
1.

Funzioni per inizializzare array

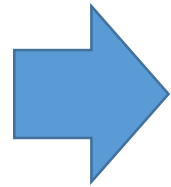
```
▶ e = np.eye(3)
```

```
print(e)
```

```
↳ 

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


```



e

1.	0.	0.
0.	1.	0.
0.	0.	1.

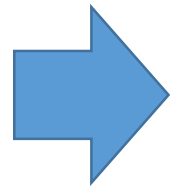
```
▶ f = np.full(3, 2)
```

```
print(f)
```

```
↳ 

```
[2 2 2]
```


```



f

2
2
2

numpy.full(*shape, fill_value, dtype=None, order='C', *, like=None*)
Return a new array of given shape and type, filled with *fill_value*.

Valori (pseudo)random



```
import numpy as np
```

```
r_1 = np.random.random((5,))  
print(r_1)
```

```
r_2 = np.random.random((2,3))  
print(r_2)
```

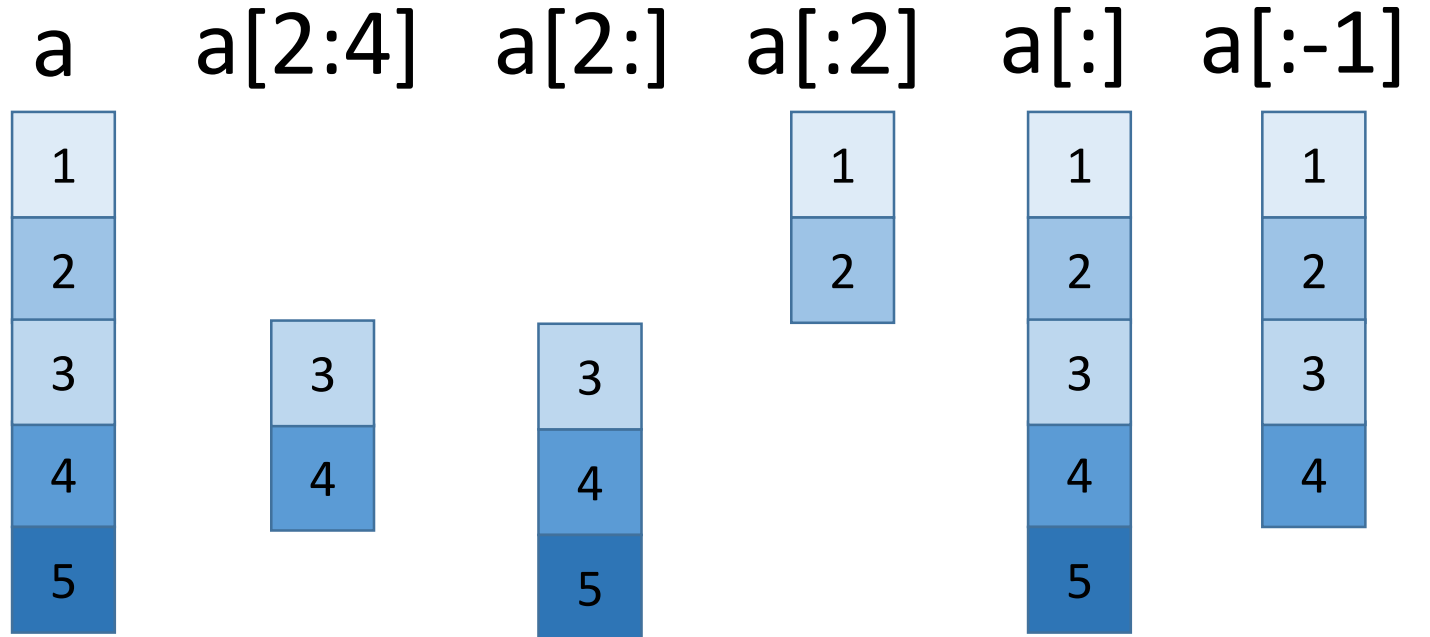


```
[0.37389921 0.79844257 0.35715868 0.54446747 0.51375202]  
[[0.56753889 0.77342533 0.85682621]  
 [0.33778405 0.41373532 0.2560605 ]]
```

Slicing mono-dimensionale

```
▶ a = np.array([1,2,3,4,5])  
  
print(a[2:4])  
print(a[2:])  
print(a[:2])  
print(a[:])  
print(a[:-1])
```

```
↳ [3 4]  
[3 4 5]  
[1 2]  
[1 2 3 4 5]  
[1 2 3 4]
```



Slicing bi-dimensionale

```
▶ b = np.array([range(1,5), range(5,9), range(9,13)])  
  
print(b)  
print(b[1:,2:])  
print(b[:, :2])
```

```
[[ 1  2  3  4]  
 [ 5  6  7  8]  
 [ 9 10 11 12]]  
[[ 7  8]  
 [11 12]]  
[[ 1  2]  
 [ 5  6]  
 [ 9 10]]
```

b

1	2	3	4
5	6	7	8
9	10	11	12

b[1:,2:]

7	8
11	12

b[:, :2]

1	2
5	6
9	10

range è una funzione python built-in che crea una lista di interi

Indexing

```
a = np.array([[1,2], [3, 4], [5, 6]])
```

```
print(a[0])  
print(a[0,1])  
print(a[[0,1]])  
print(a[[0,1,2]])  
print(a[[0,1,2],[0]])  
print(a[[0,1,2],[0,1,1]])
```

```
➞ [1 2]  
2  
[[1 2]  
 [3 4]]  
[[1 2]  
 [3 4]  
 [5 6]]  
[1 3 5]  
[1 4 6]
```

a

1	2
3	4
5	6

Boolean indexing

```
b = np.array([[1,2], [3, 4], [5, 6]])  
  
print(b)  
  
print(b > 3)  
  
print(b[b > 3])
```

```
[[1 2]  
 [3 4]  
 [5 6]]  
[[False False]  
 [False  True]  
 [ True  True]]  
[4 5 6]
```

b

1	2
3	4
5	6

Tipi di dato in Numpy

```
a = np.array([22, 33, 44])
print(a)
print(a.dtype)

b = np.array([22.3, 44.5])
print(b)
print(b.dtype)

c = np.array([22, 33, 44], dtype=np.float64)
print(c)
print(c.dtype)
```

```
☞ [22 33 44]
   int64
   [22.3 44.5]
   float64
   [22. 33. 44.]
   float64
```

Tipi di dato in Numpy vs. C

Numpy type	C type	Description
<code>np.int8</code>	<code>int8_t</code>	Byte (-128 to 127)
<code>np.int16</code>	<code>int16_t</code>	Integer (-32768 to 32767)
<code>np.int32</code>	<code>int32_t</code>	Integer (-2147483648 to 2147483647)
<code>np.int64</code>	<code>int64_t</code>	Integer (-9223372036854775808 to 9223372036854775807)
<code>np.uint8</code>	<code>uint8_t</code>	Unsigned integer (0 to 255)
<code>np.uint16</code>	<code>uint16_t</code>	Unsigned integer (0 to 65535)
<code>np.uint32</code>	<code>uint32_t</code>	Unsigned integer (0 to 4294967295)
<code>np.uint64</code>	<code>uint64_t</code>	Unsigned integer (0 to 18446744073709551615)
<code>np.intp</code>	<code>intptr_t</code>	Integer used for indexing, typically the same as <code>ssize_t</code>
<code>np.uintp</code>	<code>uintptr_t</code>	Integer large enough to hold a pointer
<code>np.float32</code>	<code>float</code>	Note that this matches the precision of the builtin python <i>float</i> .
<code>np.float64 / np.float_</code>	<code>double</code>	
<code>np.complex64</code>	<code>float complex</code>	Complex number, represented by two 32-bit floats (real and imaginary components)
<code>np.complex128 / np.complex_</code>	<code>double complex</code>	Note that this matches the precision of the builtin python <i>complex</i> .

Operazioni con gli array

```
a = np.array([1,2,3,4])
b = np.array([5,6,7,8])

print(a + b)
print(np.add(a, b))

c = np.array([[1,2], [3,4]])
d = np.array([[5,6], [7,8]])

print(c + d)
print(np.add(c, d))
```

```
↳ [ 6  8 10 12]
   [ 6  8 10 12]
   [[ 6  8]
    [10 12]]
   [[ 6  8]
    [10 12]]
```


ValueError

```
a = np.array([1,2,3,4])
b = np.array([5,6,7,8])

print(a + b)
print(np.add(a, b))

c = np.array([[1,2,4], [3,4,4]])
d = np.array([[5,6], [7,8]])

print(c + d)
print(np.add(c, d))
```

```
[ 6  8 10 12]
[ 6  8 10 12]
```

ValueError

Traceback (most recent call last)

[<ipython-input-9-25beb093c60b>](#) in <module>()

10 d = np.array([[5,6], [7,8]])

11

---> 12 print(c + d)

13 print(np.add(c, d))

14

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

Sottrazione

```
a = np.array([1,2,3,4])  
b = np.array([5,6,7,8])
```

```
print(a - b)  
print(np.subtract(a, b))
```

```
c = np.array([[1,2], [3,4]])  
d = np.array([[5,6], [7,8]])
```

```
print(c - d)  
print(np.subtract(c, d))
```

```
[-4 -4 -4 -4]  
[-4 -4 -4 -4]  
[[-4 -4]  
 [-4 -4]]  
[[-4 -4]  
 [-4 -4]]
```

Divisione (elemento per elemento)

```
a = np.array([1,2,3,4])
b = np.array([5,6,7,8])

print(a / b)
print(np.divide(a, b))

c = np.array([[1,2], [3,4]])
d = np.array([[5,6], [7,8]])

print(c / d)
print(np.divide(c, d))
```

```
↳ [0.2      0.33333333 0.42857143 0.5      ]
   [0.2      0.33333333 0.42857143 0.5      ]
   [[0.2      0.33333333]
    [0.42857143 0.5      ]]
   [[0.2      0.33333333]
    [0.42857143 0.5      ]]
```

Moltiplicazione (elemento per elemento)

```
a = np.array([1,2,3,4])
b = np.array([5,6,7,8])

print(a * b)
print(np.multiply(a, b))

c = np.array([[1,2], [3,4]])
d = np.array([[5,6], [7,8]])

print(c * d)
print(np.multiply(c, d))
```

```
↳ [ 5 12 21 32]
   [ 5 12 21 32]
   [[ 5 12]
    [21 32]]
   [[ 5 12]
    [21 32]]
```

Prodotto scalare

```
a = np.array([1,2,3,4])
b = np.array([5,6,7,8])
print(a.dot(b))
print(np.dot(a, b))

c = np.array([[1,2], [3,4]])
d = np.array([[5,6], [7,8]])
print(c.dot(d))
print(np.dot(c, d))

e = np.array([[1,2], [3,4], [5,6]])
print(e.dot(d))
print(np.dot(e, d))
```

```
70
70
[[19 22]
 [43 50]]
[[19 22]
 [43 50]]
[[19 22]
 [43 50]
 [67 78]]
[[19 22]
 [43 50]
 [67 78]]
```


Trasposta

```
a = np.array([[1,2], [3,4]])  
print(a)  
  
print(a.T)
```

```
↳ [[1 2]  
   [3 4]]  
   [[1 3]  
   [2 4]]
```

Broadcasting

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix} \quad \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \quad (1 \quad 2 \quad 3)$$

A

X

B

$$\mathbf{A.X + B}$$



broadcast

Broadcasting

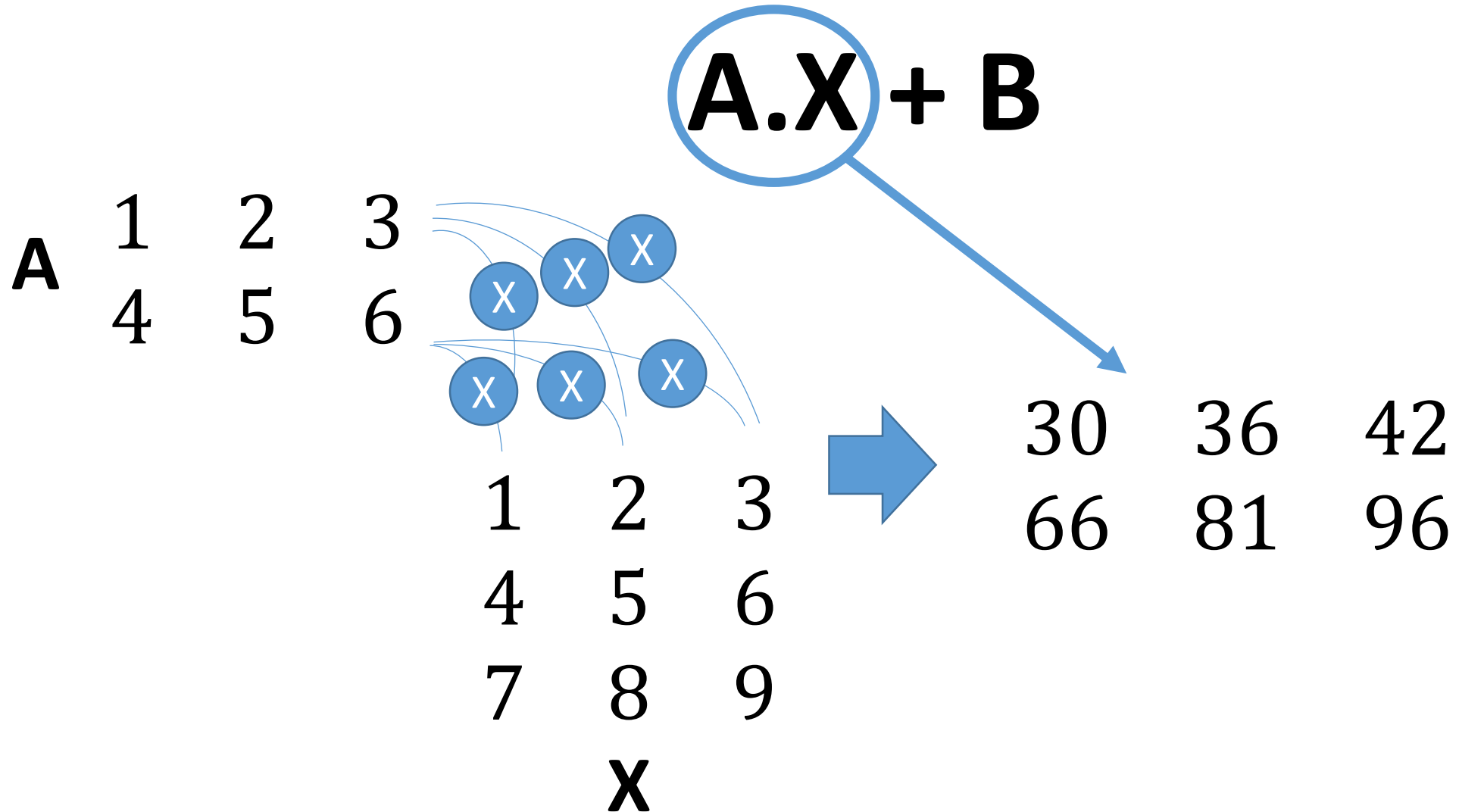
broadcast



$$\mathbf{A}.\mathbf{X} + \mathbf{B}$$

$$\begin{pmatrix} A_{0,0} & A_{0,1} & A_{0,2} \\ A_{1,0} & A_{1,1} & A_{1,2} \end{pmatrix} \cdot \begin{pmatrix} X_{0,0} & X_{0,1} & X_{0,2} \\ X_{1,0} & X_{1,1} & X_{1,2} \\ X_{2,0} & X_{2,1} & X_{2,2} \end{pmatrix} + \begin{pmatrix} B_0 & B_1 & B_2 \\ B_0 & B_1 & B_2 \end{pmatrix}$$


Broadcasting




Broadcasting


broadcast 1 2 3

A.X + B





$$\begin{pmatrix} 30 & 36 & 42 \\ 66 & 81 & 96 \end{pmatrix} + \begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \end{pmatrix}$$



$$\begin{pmatrix} 31 & 38 & 45 \\ 67 & 83 & 99 \end{pmatrix}$$

Broadcasting



```
A = np.array([[1,2,3],[4,5,6]])  
X = np.array([[1,2,3], [4,5,6], [7,8,9]])  
B = np.array([1,2,3])  
print(A.dot(X))  
print(A.dot(X) + B)
```



```
[[30 36 42]  
 [66 81 96]]  
[[31 38 45]  
 [67 83 99]]
```

Matplotlib

Matplotlib è una libreria Python per il plotting in 2D

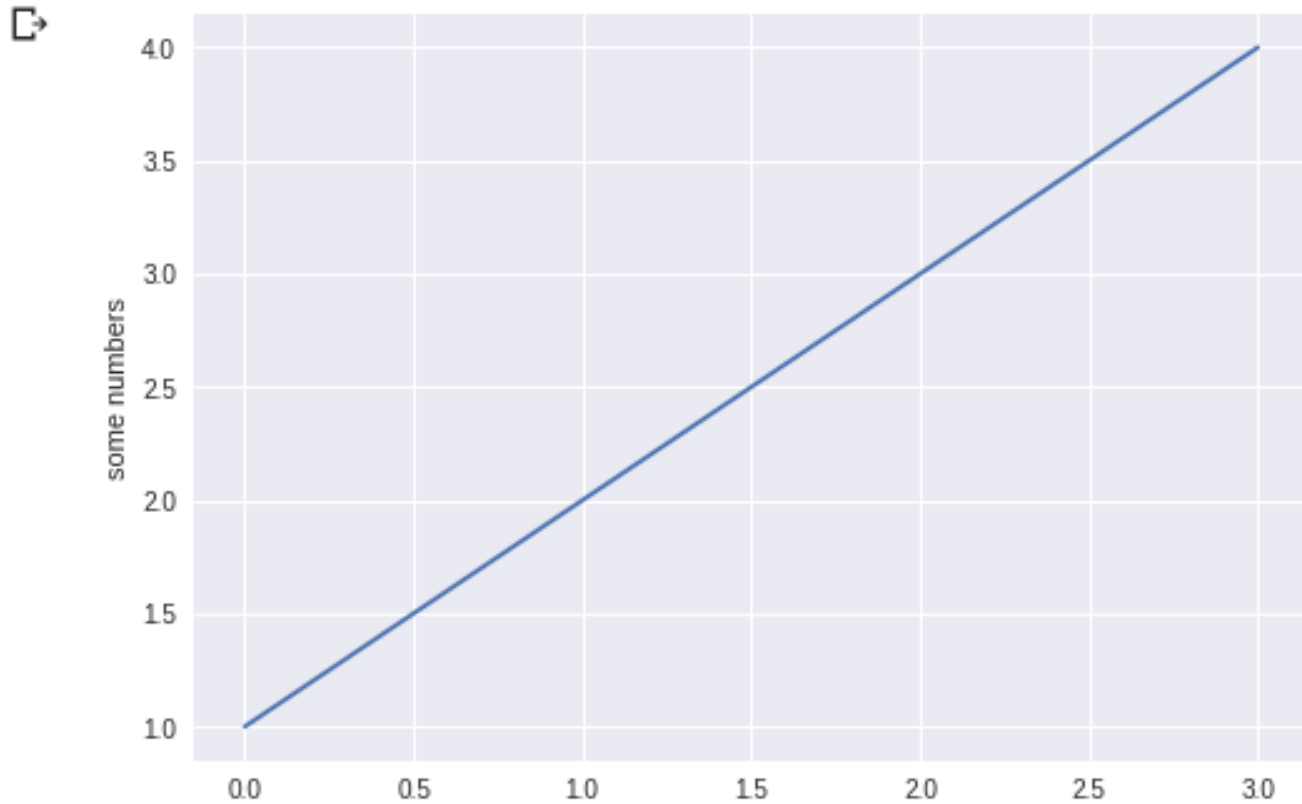


<https://matplotlib.org/>

Con matplotlib è possibile generare grafici, istogrammi, spettri, diagrammi a barre, grafici di dispersione e altro ancora usando una interfaccia tipo MATLAB

plot

```
import matplotlib.pyplot as plt  
  
plt.plot([1,2,3,4])  
plt.ylabel('some numbers')  
plt.show()
```



https://matplotlib.org/users/pyplot_tutorial.html


Visualizzare una immagine con matplotlib

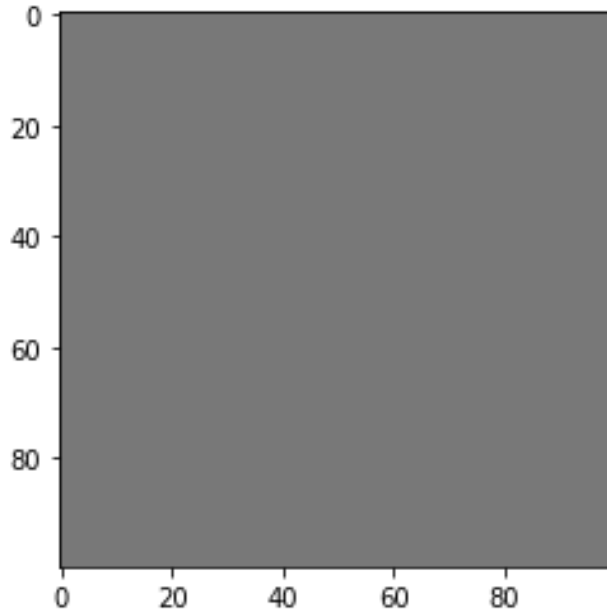
```
import numpy as np

import matplotlib.pyplot as plt

img = np.ones([100,100,3],dtype=np.uint8)*120

plt.imshow(img)
```

 <matplotlib.image.AxesImage at 0x7fe1b6fa4a58>

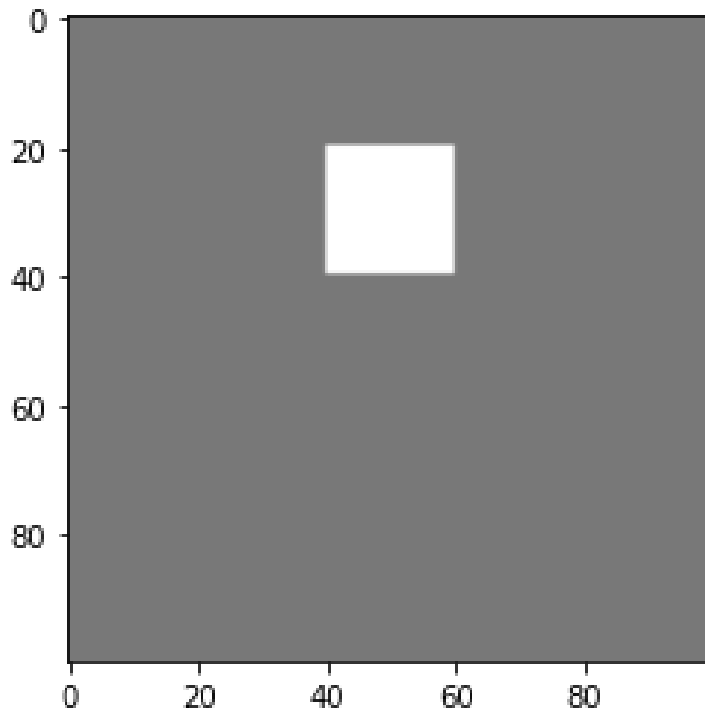


Modifica dell'immagine

```
img[20:40,40:60,:] = 255
```


```
plt.imshow(img)
```

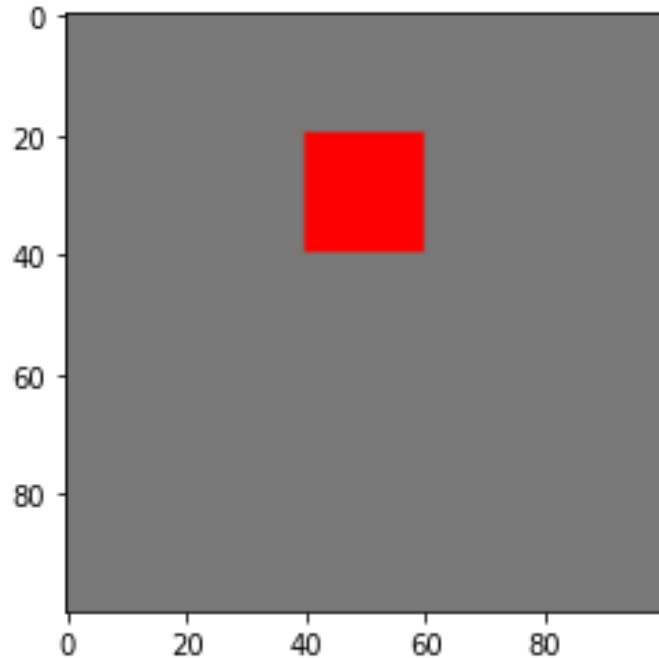
↳ <matplotlib.image.AxesImage at 0x7fe1b6e10cf8>



Modifica dell'immagine


```
img[20:40,40:60,0] = 255  
img[20:40,40:60,1] = 0  
img[20:40,40:60,2] = 0  
  
plt.imshow(img)
```

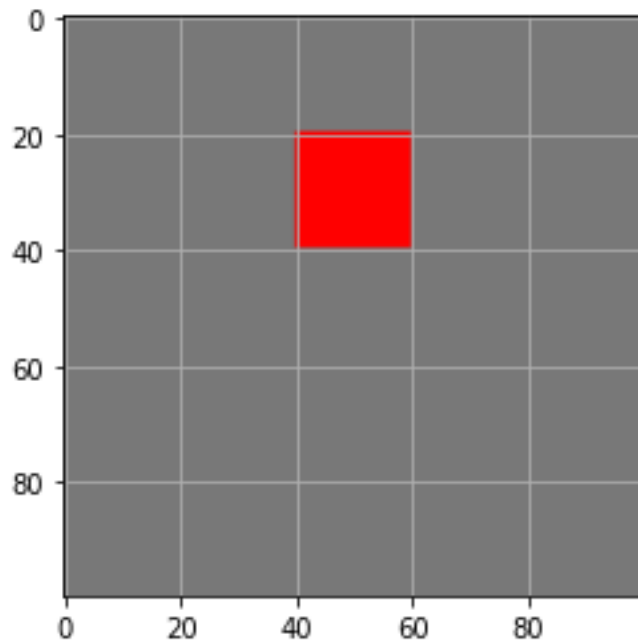
 <matplotlib.image.AxesImage at 0x7fe1b6de8eb8>



Grid on

```
img[20:40,40:60,0] = 255  
img[20:40,40:60,1] = 0  
img[20:40,40:60,2] = 0  
  
plt.grid(True)  
plt.imshow(img)
```

 <matplotlib.image.AxesImage at 0x7fe1b6dbdf28>



Axis off

```
img[20:40,40:60,0] = 255  
img[20:40,40:60,1] = 0  
img[20:40,40:60,2] = 0
```

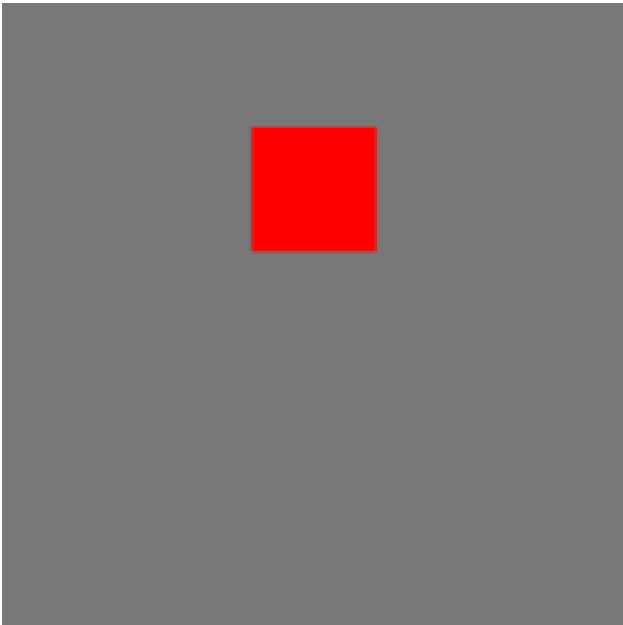
```
plt.axis(False)  
plt.imshow(img)
```

↳ <matplotlib.image.AxesImage at 0x7fe1b6d1c4e0>



Evitare la stampa a video

```
▶ img[20:40,40:60,0] = 255  
img[20:40,40:60,1] = 0  
img[20:40,40:60,2] = 0  
  
plt.axis(False)  
_ = plt.imshow(img)
```



Salvare l'immagine

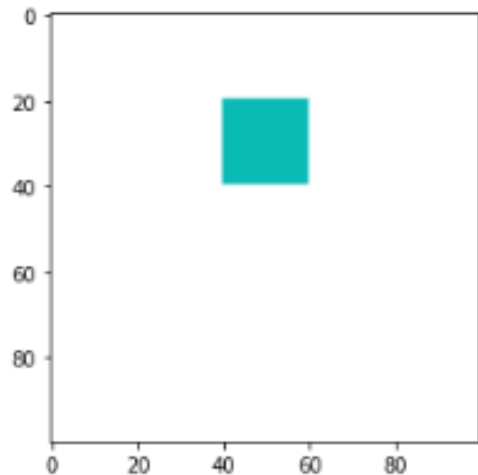
```
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image

img = np.ones([100,100,3],dtype=np.uint8)*255
img[20:40,40:60,0] = 10
img[20:40,40:60,1] = 186
img[20:40,40:60,2] = 181

_ = plt.imshow(img)

!ls
pil_img = Image.fromarray(img)
pil_img.save("img1.png")
!ls
```

sample_data
img1.png sample_data



Pillow

Pillow è una libreria open source per aprire, elaborare e salvare immagini derivata dalla Python Imaging Library (PIL)



Homepage: <https://python-pillow.org/>

Source code: <https://github.com/python-pillow/Pillow>

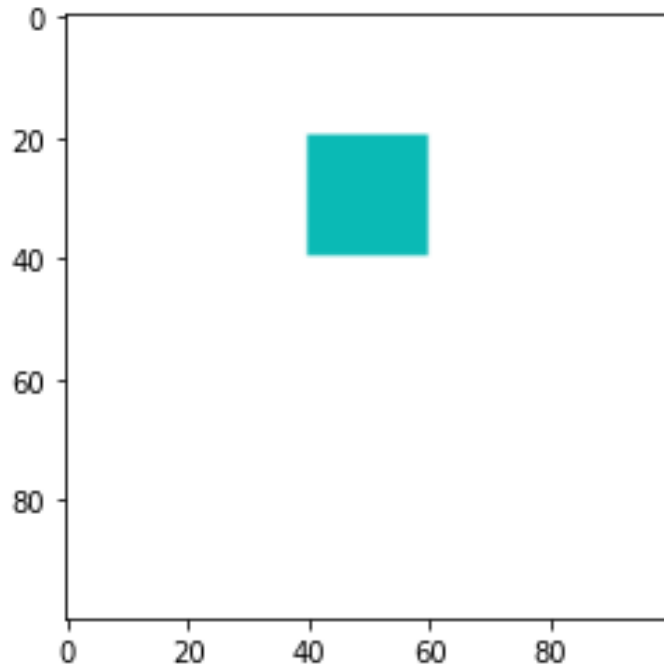
Documentation: <https://pillow.readthedocs.io/>

Aprire l'immagine con Pillow

```
▶ from PIL import Image
import matplotlib.pyplot as plt

my_img = Image.open("img1.png")

_ = plt.imshow(my_img)
```



Immagini: occupazione di memoria

L'occupazione di memoria è data dal prodotto tra la dimensione dell'immagine e la profondità di colore del singolo pixel

Occupazione = (Dimensione) x (Profondità di colore)

Esempio:

Una immagine a colori (RGB) 640x480 occupa in memoria 9830400 bit pari a circa 1.23 MB

Compressione

Per diminuire la dimensione di una immagine in memoria è possibile utilizzare degli opportuni metodi di compressione



Tipi di Compressione

La compressione può essere "lossless" o "lossy"

- Compressione "lossless": **reversibile**
Ad esempio, file PNG e file ZIP
- Compressione "lossy": **ricostruzione approssimata**, dove maggiore è il rapporto di compressione, maggiore è l'errore
Ad esempio, file JPEG e file MP3

Formato BMP

- Il formato BMP (bitmap) è stato sviluppato da Microsoft per la gestione dei file in Windows
- Si tratta di un formato piuttosto datato (anni 90) che permette di salvare immagini in grayscale e a colori
- Viene usato di solito per salvare immagini senza compressione (**lossless**)

<https://docs.microsoft.com/en-us/windows/desktop/gdi/bitmap-storage>

Formato PNG

Il formato PNG (Portable Network Graphics) utilizza un algoritmo di compressione **lossless** che permette di preservare dettagli e sfumature di colore nell'immagine

<http://www.libpng.org/pub/png/>

Compressione lossy

L'occhio umano è meno sensibile alle ALTE frequenze spaziali:

- Se l'ampiezza di una componente ad ALTA frequenza cade sotto una certa soglia, l'occhio umano NON LA RILEVA



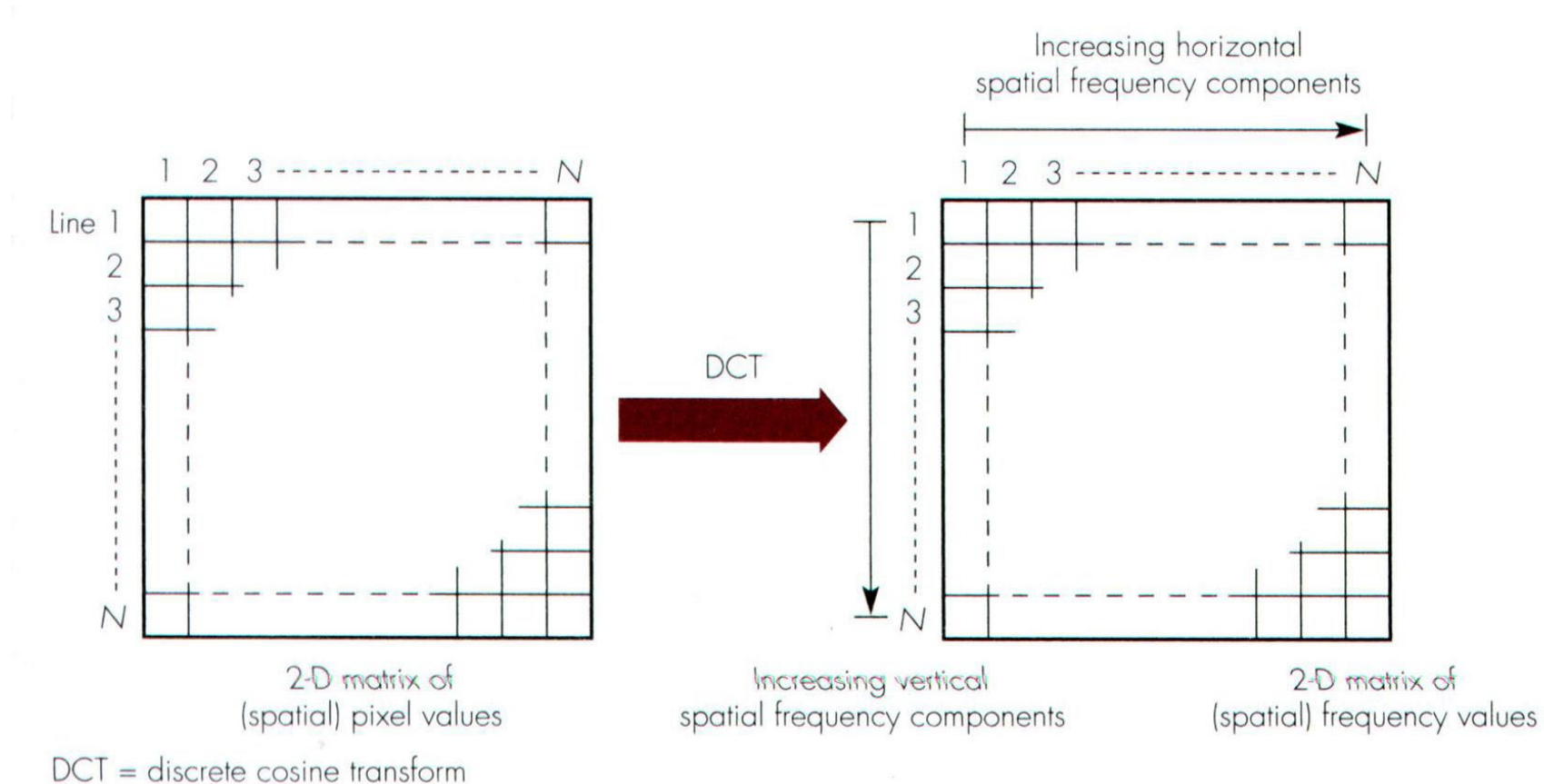
La quantizzazione può essere
meno accurata alle alte frequenze

Possono essere utilizzati meno bit



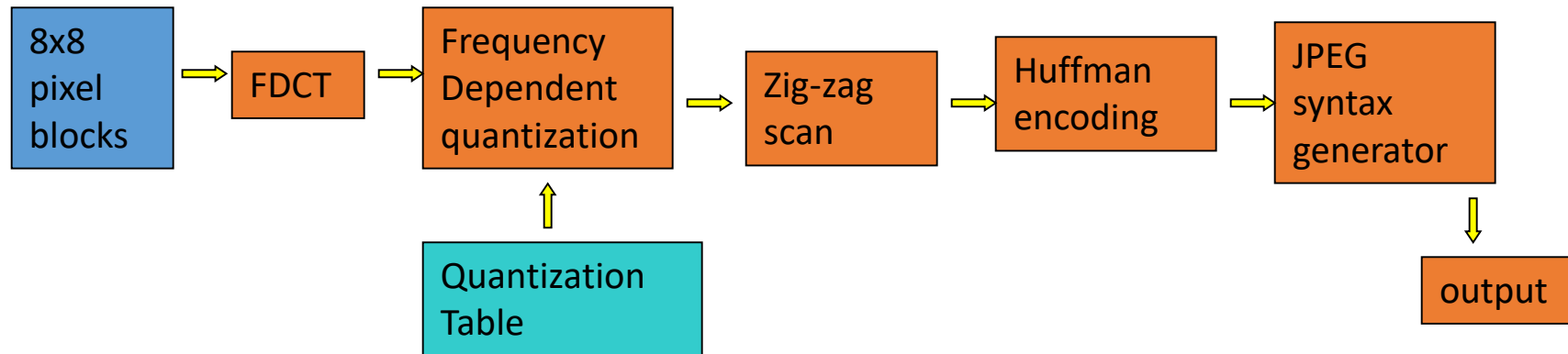
Discrete Cosine Transform

La DCT trasforma una matrice bi-dimensionale di pixel in una matrice equivalente di "spatial frequency components"



Compressione JPEG

- L'immagine viene divisa in blocchi 8x8
- Si applica la 2D Fourier Discrete Cosine Transform (FDCT)
- I componenti ad alta frequenza spaziale vengono quantizzati più grossolanamente
- I dati risultanti dalla quantizzazione vengono compressi con un meccanismo senza perdita di informazione



Quantizzazione in frequenza

Spatial domain

128	128	128	128	128	128	128	128
118	111	112	117	120	123	123	122
125	121	115	111	119	119	118	117
120	121	113	113	125	124	115	108
120	120	116	119	124	120	115	110
117	113	111	122	120	110	116	119
109	113	111	122	120	110	116	119
111	121	124	118	115	121	117	113

Frequency domain

-80	4	-6	6	2	-2	-2	0
24	-8	8	12	0	0	0	2
10	-4	0	-12	-4	4	4	-2
8	0	-2	-6	10	4	-2	0
18	4	-4	6	-8	-4	0	0
-2	8	6	-4	0	-2	0	0
12	0	6	0	0	0	-2	-2
0	8	0	-4	-2	0	0	0

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Quantization Matrix to divide by

-5	0	0	0	0	0	0	0
2	-1	1	1	0	0	0	0
1	0	0	-1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Quantized spatial frequency values

Immagine tratta da M. Moewe
“Media Compression
Techniques”

Scanning e Huffman Encoding

- Si usa un percorso a zig-zag per scansionare le frequenze spaziali
- Le alte frequenze valgono quasi sempre zero
- La Huffman encoding è usata per immagazzinare con compressione lossless i valori

-5	0	0	0	0	0	0	0
2	-1	1	1	0	0	0	0
1	0	0	-1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Usando la codifica di Huffman i valori

0,2,1,-1,0,0,1,0,1,1,0,0,1,0,0,0,-1,0,0,... 0

vengono memorizzati come

(1,2),(0,1),(0,-1),(2,1),(1,1),(0,1),(0,1),(2,1),(3,1),EOB

Vari livelli di compressione JPEG



500KB image, minimum compression



40KB image, half compression



11KB image, max compression

Perdita di dettagli



Uncompressed image
(roughness between
pixels still visible)



Half compression,
blurring & halos around
sharp edges



Max compression, 8-
pixel blocks apparent,
large distortion in high-
frequency areas

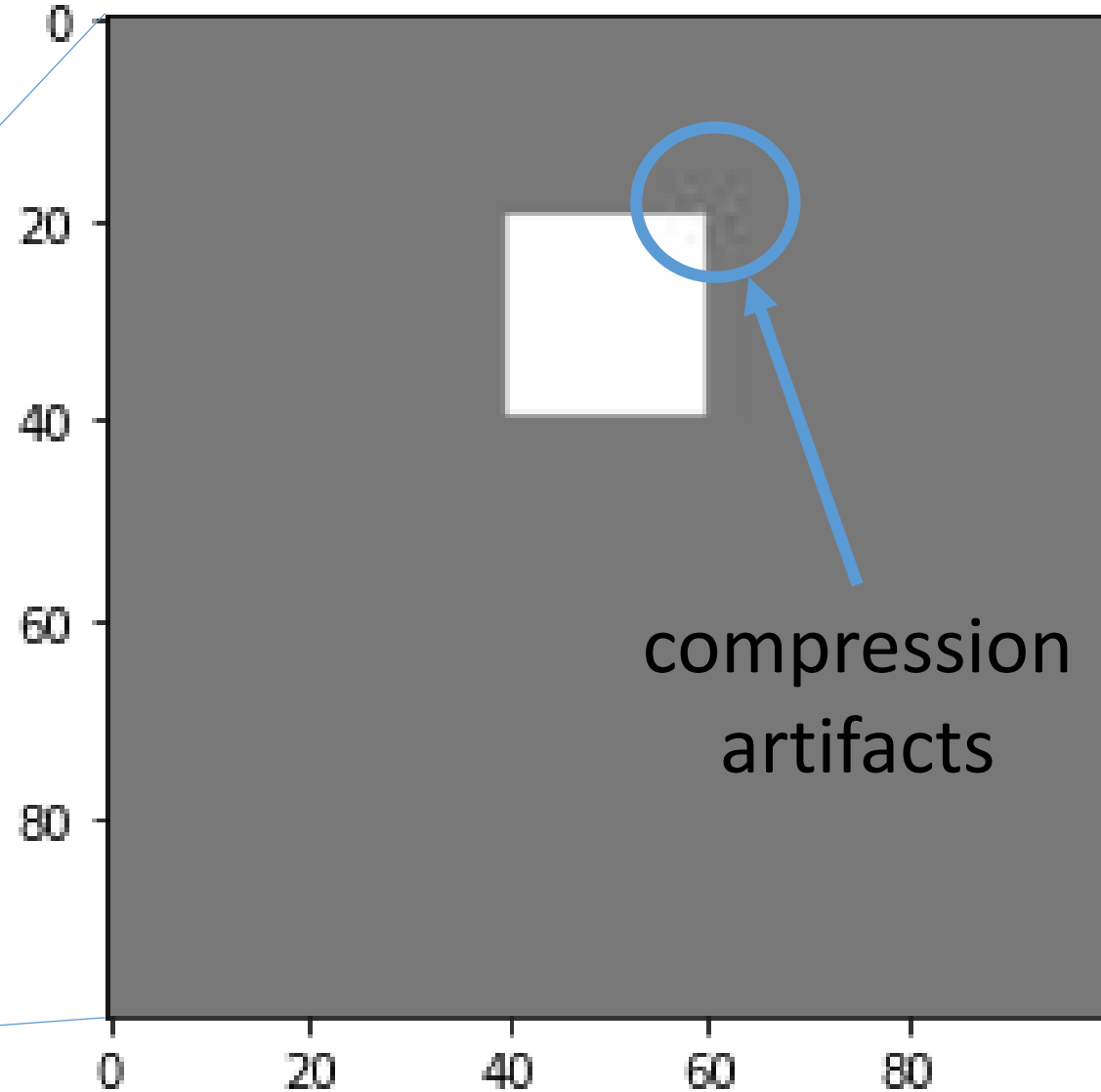
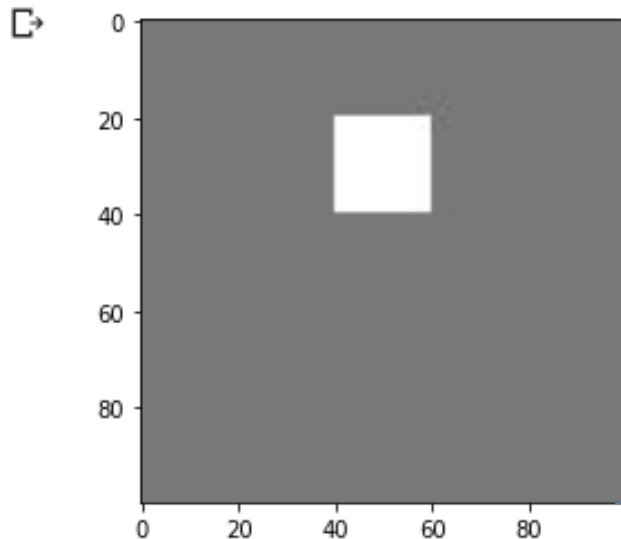
Immagini in Jpeg con Pillow

```
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image

img = np.ones([100,100,3],dtype=np.uint8)*120
img[20:40,40:60,:] = 255

pil_img = Image.fromarray(img)
pil_img.save("my_img.jpg")

jpg_img = Image.open("my_img.jpg")
_ = plt.imshow(jpg_img)
```



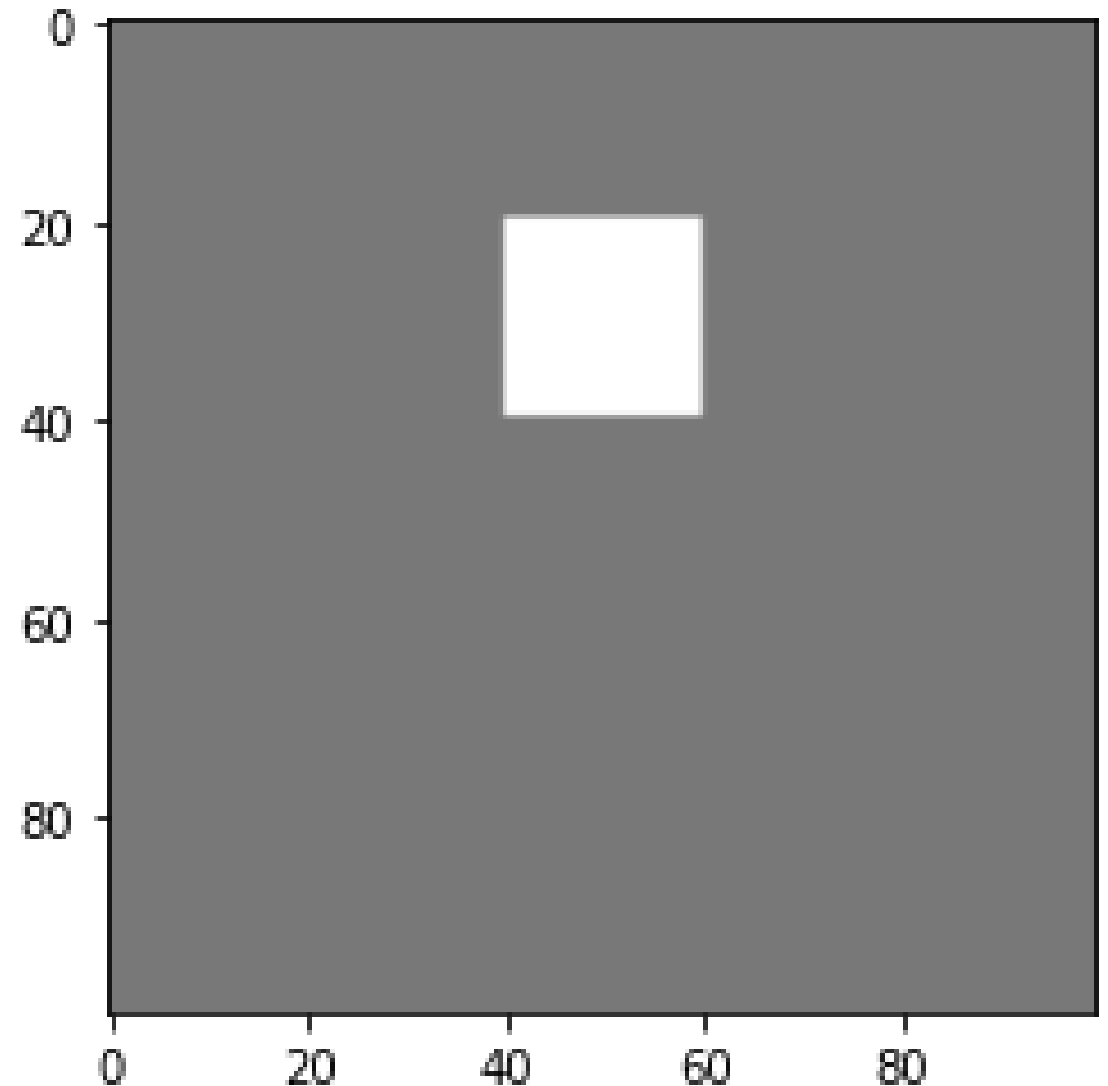
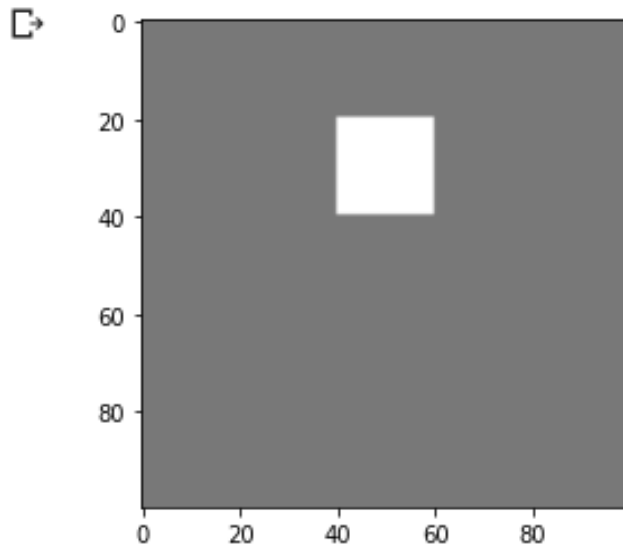
Cambiare Jpeg quality

```
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image

img = np.ones([100,100,3],dtype=np.uint8)*120
img[20:40,40:60,:] = 255

pil_img = Image.fromarray(img)
pil_img.save("my_img.jpg", quality=100)

jpg_img = Image.open("my_img.jpg")
_ = plt.imshow(jpg_img)
```



How to read an image from url



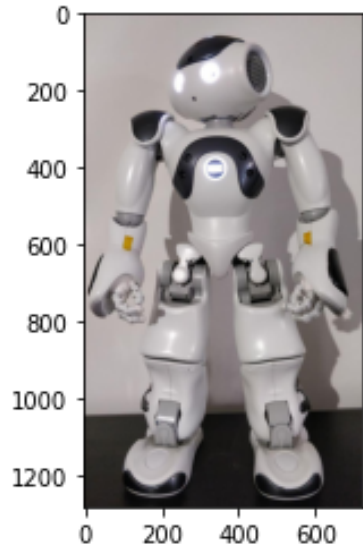
```
from PIL import Image
import matplotlib.pyplot as plt
from urllib.request import urlopen

url = "https://web.unibas.it/bloisi/corsi/images/nao-v6-spqr.jpg"

img = Image.open(urlopen(url))

plt.grid(False)

_ = plt.imshow(img)
```



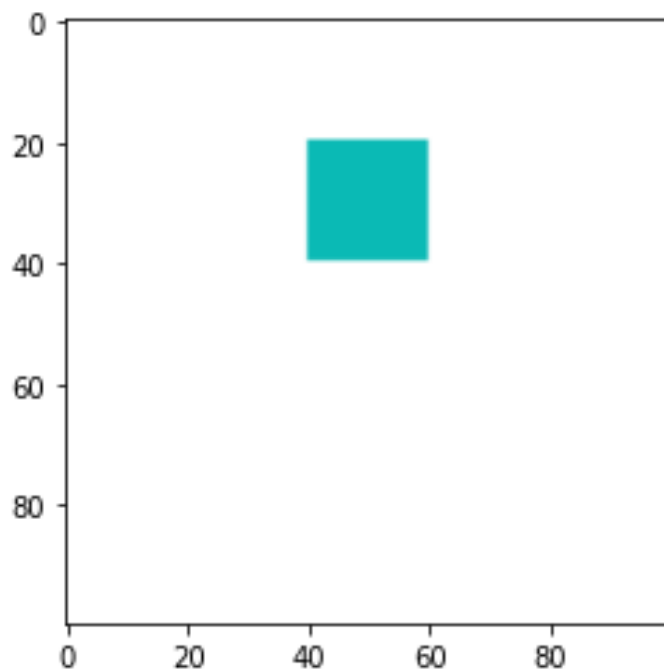
Trasformazioni

- Rotating
- Flipping
- Resizing
- Cropping

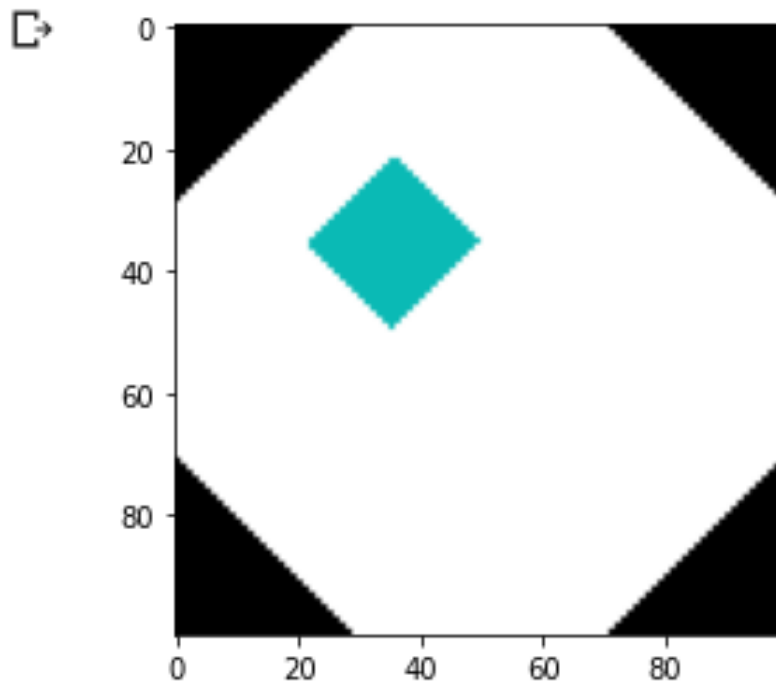
Rotate

```
▶ from PIL import Image
import matplotlib.pyplot as plt

my_img = Image.open("img1.png")
_ = plt.imshow(my_img)
```



```
▶ rotated_img = my_img.rotate(45)
_ = plt.imshow(rotated_img)
```



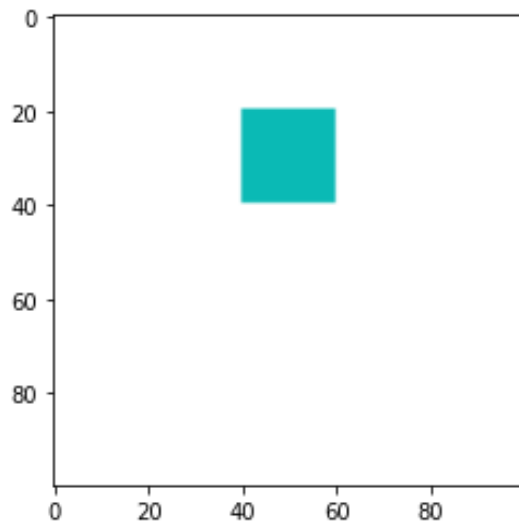
Flip

Pillow (PIL) provides

- `flip()` to flip the image upside down (vertically)
- `mirror()` to flip the left and right (horizontally)

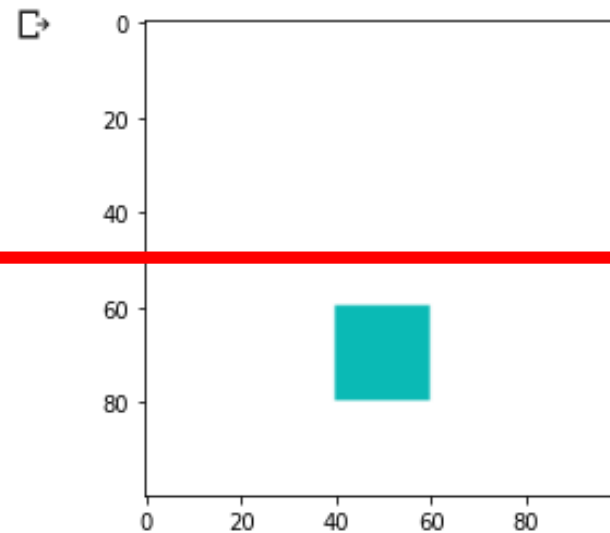
```
from PIL import Image
import matplotlib.pyplot as plt

my_img = Image.open("img1.png")
_ = plt.imshow(my_img)
```



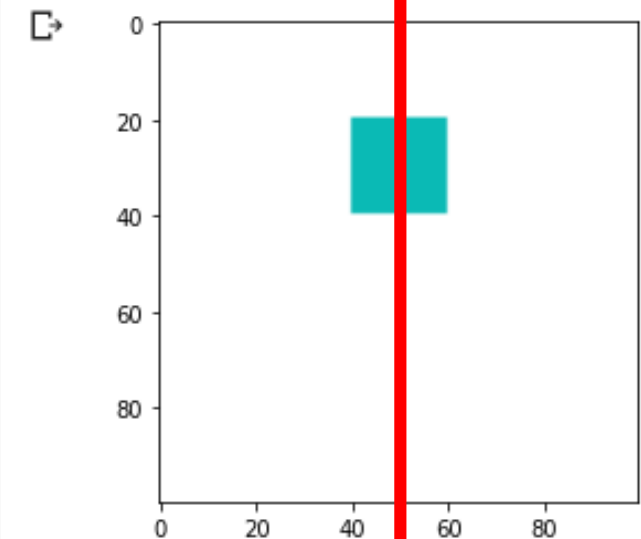
```
from PIL import ImageOps

v_flip_img = ImageOps.flip(my_img)
_ = plt.imshow(v_flip_img)
```



```
from PIL import ImageOps

h_flip_img = ImageOps.mirror(my_img)
_ = plt.imshow(h_flip_img)
```



SPL realistic black and white ball

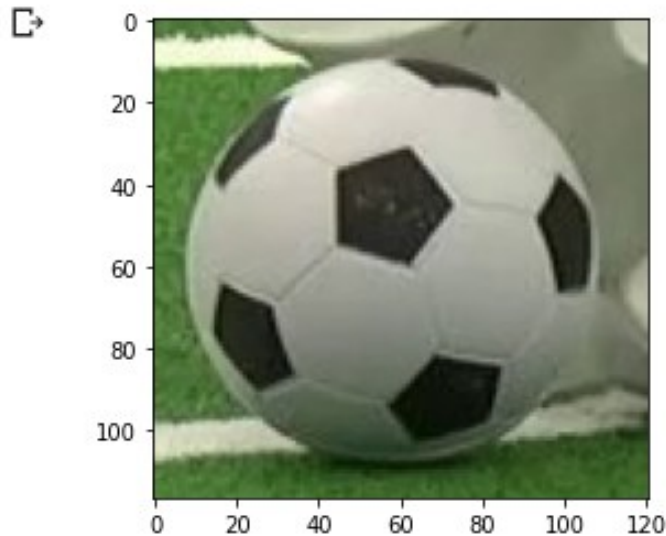
```
from PIL import Image
import matplotlib.pyplot as plt
from urllib.request import urlopen

url = "https://web.unibas.it/bloisi/corsi/images/ball.jpg"

ball = Image.open(urlopen(url))

plt.grid(False)

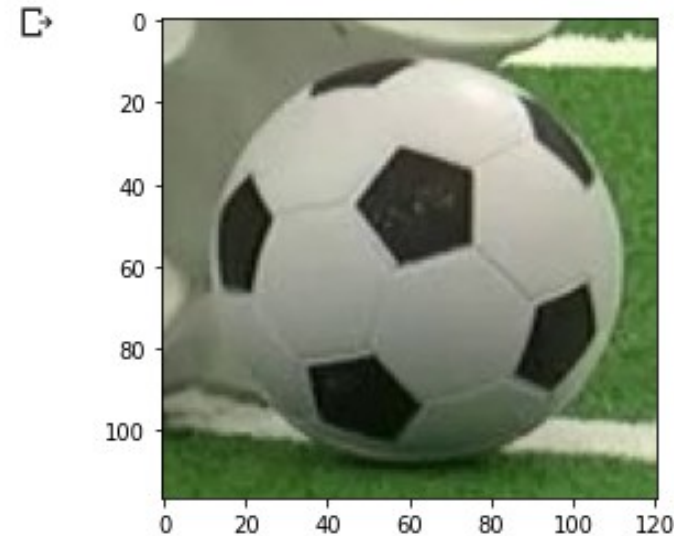
_ = plt.imshow(ball)
```



```
from PIL import ImageOps

h_flip_ball = ImageOps.mirror(ball)

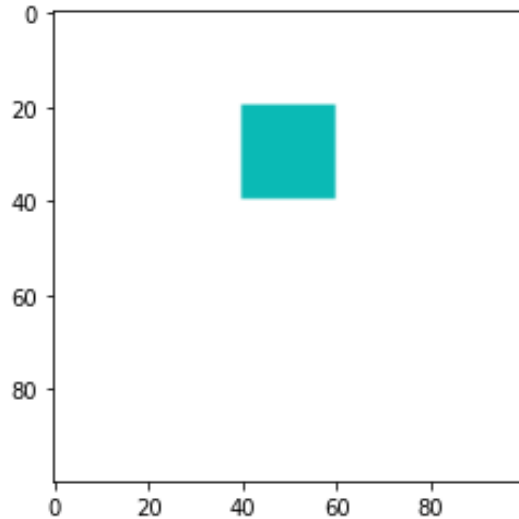
_ = plt.imshow(h_flip_ball)
```



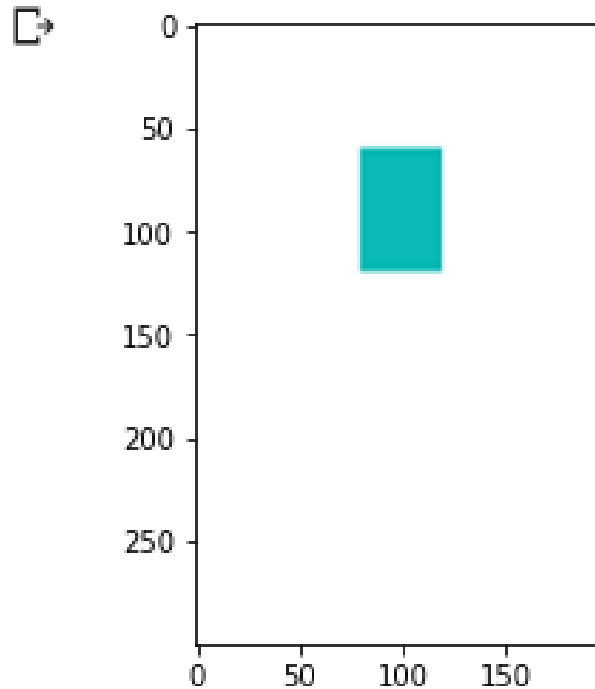
Resize

```
from PIL import Image
import matplotlib.pyplot as plt

my_img = Image.open("img1.png")
_ = plt.imshow(my_img)
```



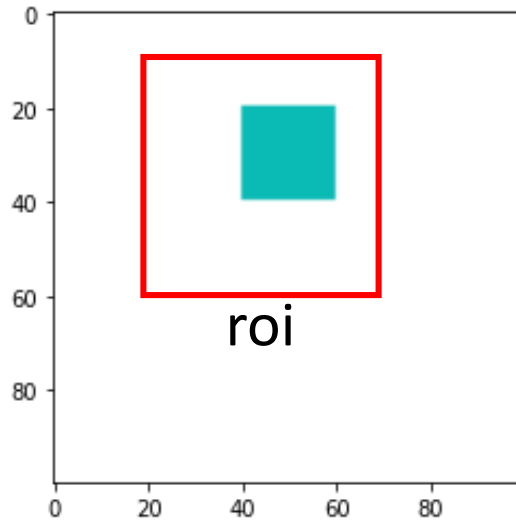
```
resized_img = my_img.resize((my_img.size[0]*2, my_img.size[1]*3))
_ = plt.imshow(resized_img)
```



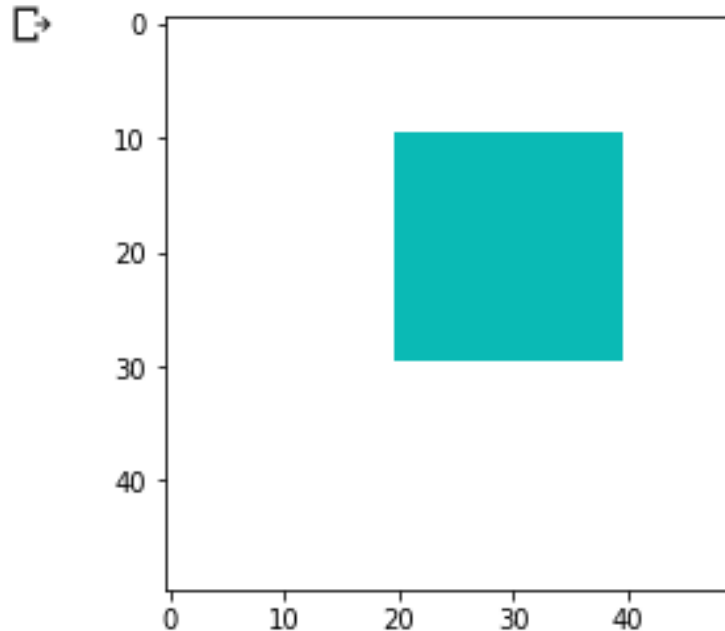
Crop

```
from PIL import Image
import matplotlib.pyplot as plt

my_img = Image.open("img1.png")
_ = plt.imshow(my_img)
```

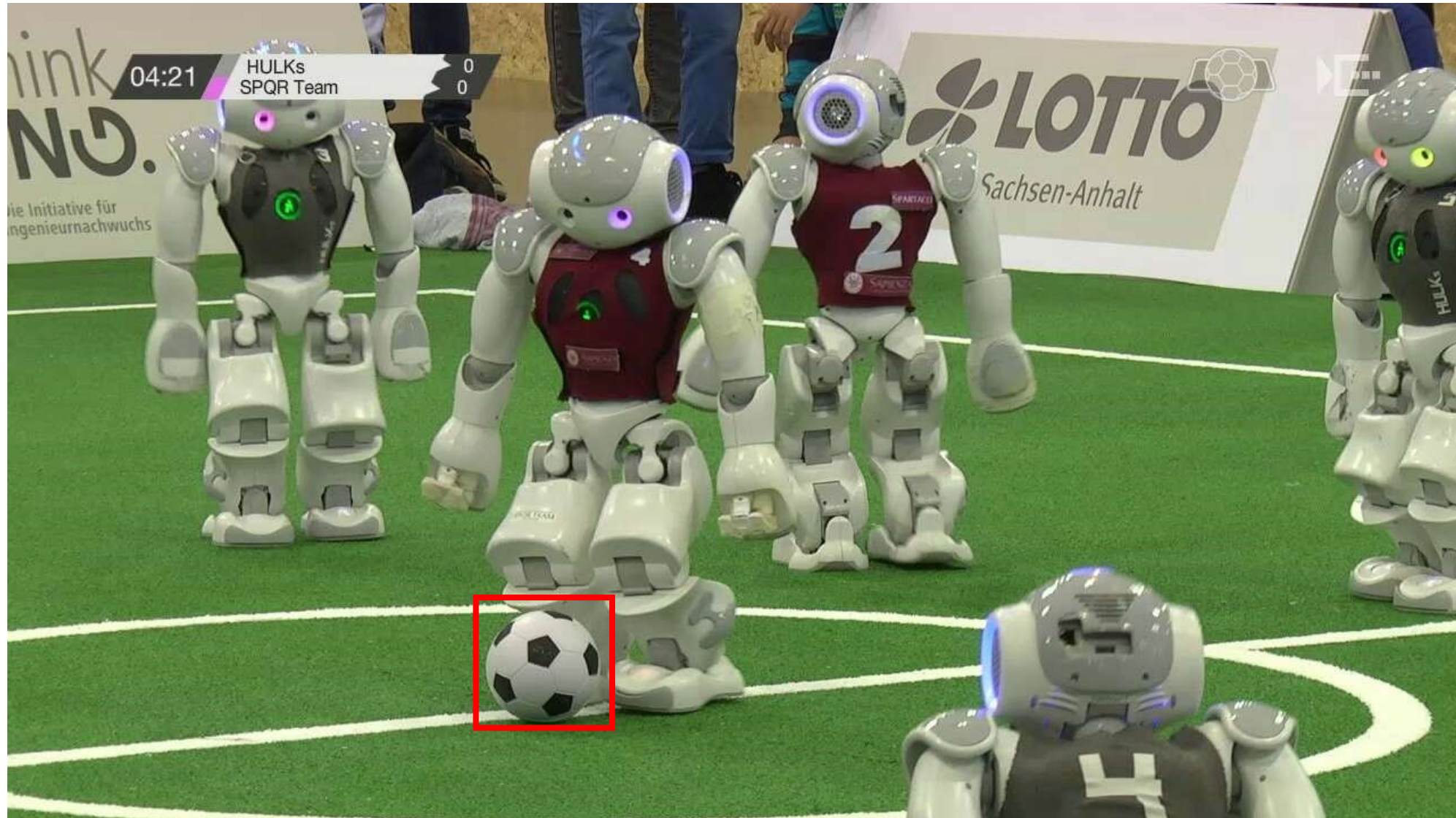


```
roi = (20,10,70,60)
crop_img = my_img.crop(roi)
_ = plt.imshow(crop_img)
```



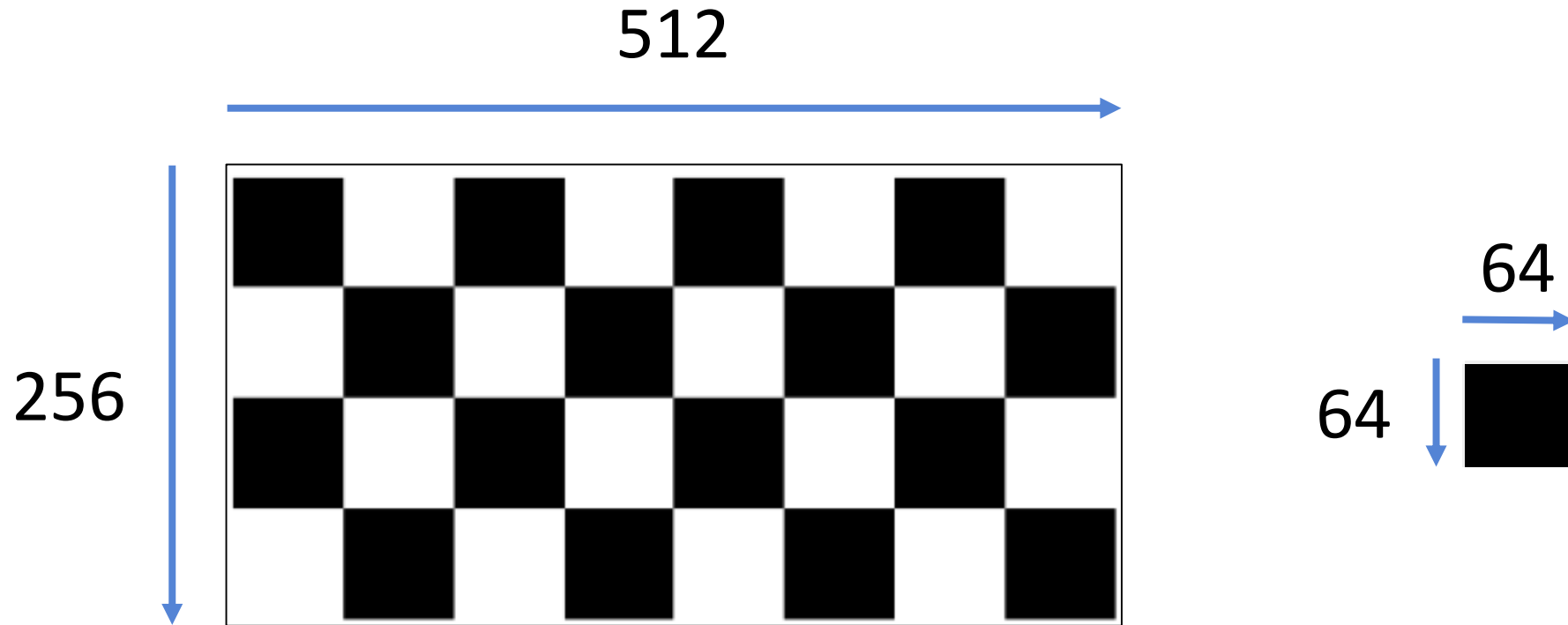
(left, upper, right, lower)-tuple

Region of interest (ROI)



Esercizio 1

Realizzare una immagine come quella in figura utilizzando le librerie NumPy, Matplotlib e Pillow



Esercizio 1: soluzione



```
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image

width = 512
height = 256
l = 64

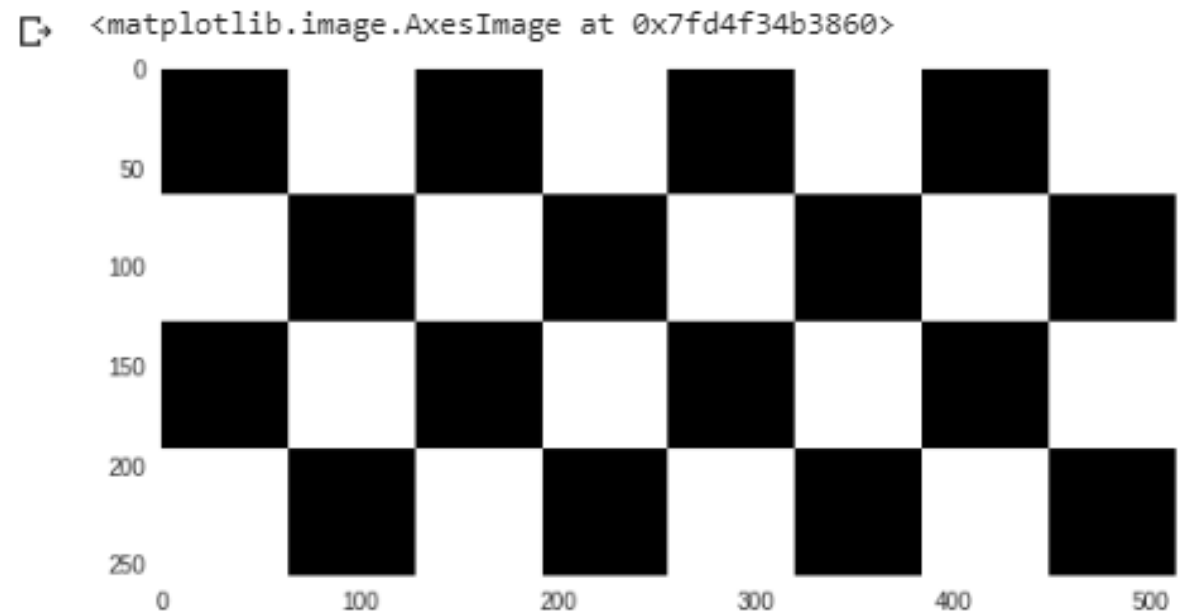
numpy_array = np.ones([height,width,3], dtype=np.uint8)*255

righe = numpy_array.shape[0] // l
colonne = numpy_array.shape[1] // l

for i in range(0, righe*l, l):
    for j in range((i // l) % 2, colonne, 2):
        c = j * l
        numpy_array[i:i+l,c:c+l] = 0

chessboard = Image.fromarray(numpy_array)

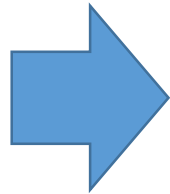
plt.grid(b=False)
plt.imshow(chessboard)
```



Esercizio 2

Utilizzare le librerie NumPy, Matplotlib e Pillow per effettuare il cropping dell'immagine

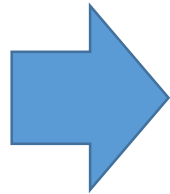
<https://web.unibas.it/bloisi/corsi/images/nao-v6-spqr.jpg>



Esercizio 3

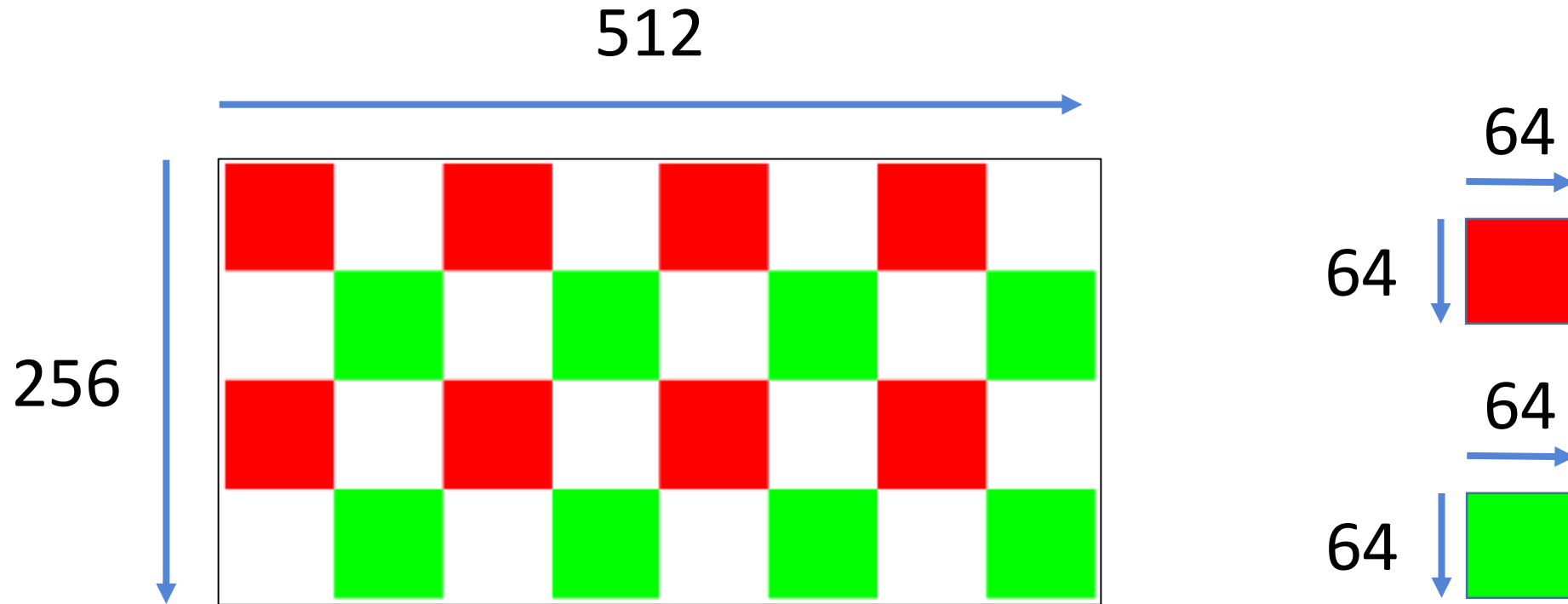
Utilizzare le librerie NumPy, Matplotlib e Pillow per effettuare la rotazione di 30° dell'immagine

<https://web.unibas.it/bloisi/corsi/images/nao-v6-spqr.jpg>



Esercizio 4

Realizzare una immagine come quella in figura utilizzando le librerie NumPy, Matplotlib e Pillow





**UNIVERSITÀ DEGLI STUDI
DELLA BASILICATA**

Corso di Visione e Percezione

Processamento delle immagini



Docente

Domenico D. Bloisi

