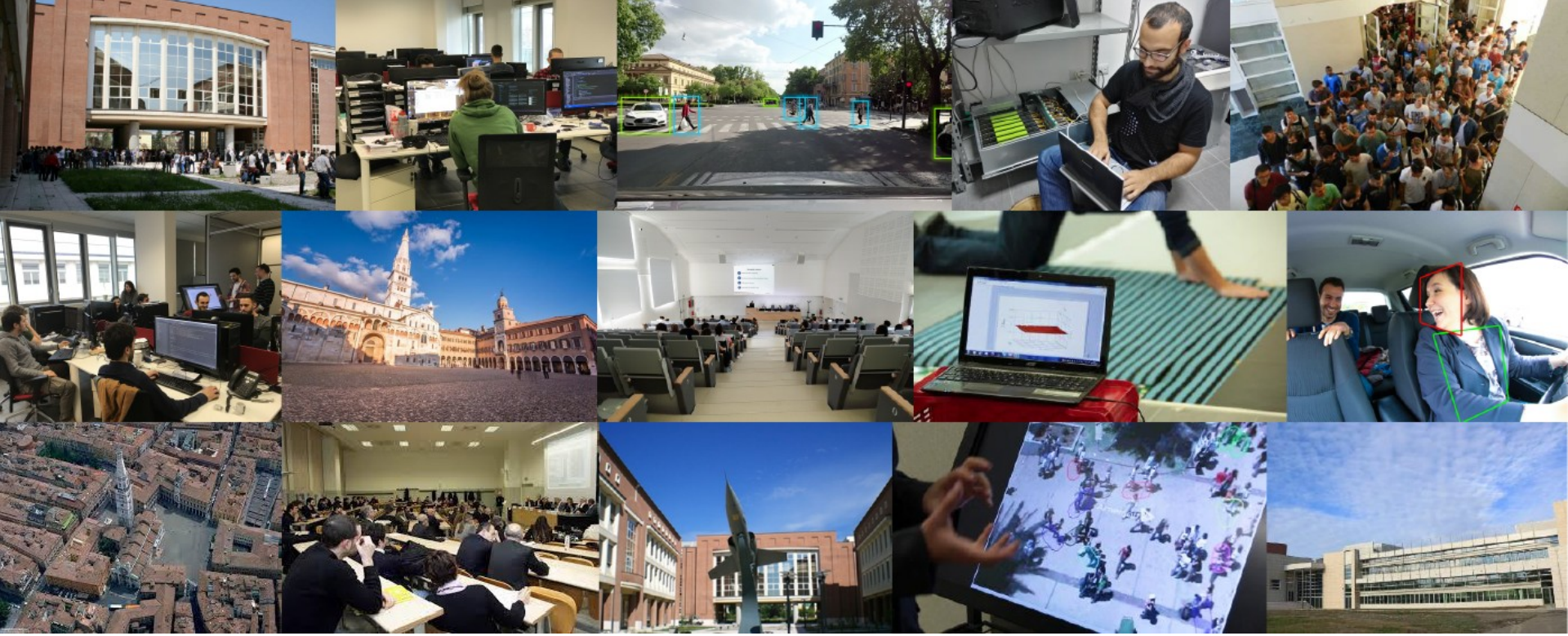


MACHINE LEARNING
Numpy -

AImage Lab

UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Numpy

numpy è un pacchetto per operazioni numeriche (rende Python più simile a Matlab)

```
import numpy as np
```

Gli oggetti *ndarray* di numpy rappresentano array multidimensionali e *omogenei* di oggetti di dimensione fissa

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

```
array([1, 2, 3, 4, 5])
```

Numpy Array vs List

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

A differenza delle liste Python, gli *ndarray* si basano su un'allocazione in memoria di un blocco contiguo di dati, quindi sono più efficienti:

```
l = list(range(10000))
a = np.array(range(10000))
def somma_numpy(a):
    return np.sum(a)

def somma_python(l):
    return sum(l)
```

```
timeit somma_python(l)
```
568 µs ± 41.1 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)

```
timeit somma_numpy(a)
```
9.52 µs ± 24.8 ns per loop (mean ± std. dev. of 7 runs, 100000 loops each)

Indexing e Slicing in Python

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Indexing: Accesso ai singoli elementi di una lista (e di una sequenza in generale):

- `s[i]` ritorna l'`i`−esimo elemento di `s` (l'indice `i` deve essere compreso tra `0` e `len(s)-1`)

Slicing: Accesso a più elementi di una lista (e di una sequenza in generale):

- `s[i:j]` ritorna la sotto-lista costituita dagli elementi `s[i],s[i+1],...,s[j-1]` di `s`, ma `s[j]` è escluso.
- Si può usare il parametro step, che determina l'incremento tra ogni indice (a.k.a. stride). Ese.: `s[start:stop:step]`

Sia nello Slicing che nell'Indexing un valore negativo (`-i`) usato come indice viene interpretato come `len(s)-i`. Ese.: `s[-1] == s[len(s)-1], s[0:-2] == s[0:len(s)-2]`

Indexing e Slicing in Numpy

Si può accedere agli elementi di un ndarray con:

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

a[0]
1
a[-1]
5
```

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

    a[...,1]   oppure a[:,1]
    array([2, 4, 5])

    a[2,...] oppure a[2,:]
    array([4, 5, 6])

    a[0:2,0:2]
    array([[1, 2],
           [3, 4]])
```

Indexing e Slicing in Numpy

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

```
l = [1,2,3,4,5]  # lista Python
a = np.array([1,2,3,4,5])  # array Numpy
```

```
l[0] == a[0]
```
True
```
l[:-3]
```
[1, 2]
```
a[:-3]
```
array([1, 2])

Numpy Array vs List

Ma attenzione: il confronto su più elementi (indicizzati con lo slicing) su un ndarray è eseguito sui singoli valori. Per un confronto “complessivo” si possono usare le funzioni Python *all()* o *any()*:

```
l = [1,2,3,4,5]  # lista Python
a = np.array([1,2,3,4,5])  # array Numpy
l[:-3] == a[:-3]
```

array([True, True], dtype=bool)

```
a[0] = 100
```

```
all(l == a)
```

False

```
any(l == a)
```

True

Numpy Array vs List

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

A differenza delle liste, gli ndarray non possono contenere elementi eterogenei:

```
l = [1, "ciao", False]
a = np.array([1, "ciao", False])

array(['1', 'ciao', 'False'], dtype='<U11')
```

Numpy Array vs List

Le liste di Python sono eterogenee (cioè possono contenere oggetti di qualsiasi tipo e dimensione), facilmente ridimensionabili
→ versatili

```
l = [1,2,3,4,5]  # lista Python
```

Gli array sono omogenei (cioè, sono ammessi solo alcuni tipi di oggetti), invariabili nelle dimensioni, orientati alle operazioni matematiche e non inclusi in Python (è necessario importare NumPy)
→ efficienti

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

Numpy Array - Attributi

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

I principali attributi di un ndarray sono il tipo dei suoi elementi, la sua shape, la sua size e le sue dimensioni:

```
a = np.array(range(10))
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

```
a.dtype
```
dtype('int64')

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

```
X.shape
```
(3, 3)
```
X.size
```
9
```
X.ndim
```
2

Numpy Array - Creazione

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Si possono creare ndarray con caratteristiche “speciali”:

```
np.arange(10)
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
np.empty((4,2), dtype=float)
 array([[ 6.92785084e-310,  4.67938787e-310],
        [ 0.00000000e+000,  0.00000000e+000],
        [ 0.00000000e+000,  0.00000000e+000],
        [ 0.00000000e+000,                nan]])
```

```
np.zeros((4,2), dtype=np.int32)
 array([[0, 0],
        [0, 0],
        [0, 0],
        [0, 0]], dtype=int32)
```

```
np.ones((4,2), dtype=np.int32)
 array([[1, 1],
        [1, 1],
        [1, 1],
        [1, 1]], dtype=int32)
```

```
np.identity(3) oppure np.eye(3)
 array([[ 1.,  0.,  0.],
        [ 0.,  1.,  0.],
        [ 0.,  0.,  1.]])
```

```
np.full((2,2), 7.0)
 array([[ 7.,  7.],
        [ 7.,  7.]])
```

```
np.random.uniform(0, 1, (2,2))
array([[0.81568445, 0.14143954],
       [0.32903066, 0.7071234 ]])
```

Numpy Array - Operazioni

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Gli ndarray possono essere moltiplicati (per degli scalari o tra loro) con:

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

```
Y = np.array([[0,0,0], [1,1,1], [0,0,0]])
```
```
array([[0, 0, 0],
       [1, 1, 1],
       [0, 0, 0]])
```

```
2 * X
```
```
array([[ 2,  4,  6],
       [ 8, 10, 12],
       [14, 16, 18]])
```

```
np.dot(X, Y) # prodotto matriciale
```
```
array([[2, 2, 2],
       [5, 5, 5],
       [8, 8, 8]])
```

```
np.dot(X, 2)
```
```
array([[ 2,  4,  6],
       [ 8, 10, 12],
       [14, 16, 18]])
```

```
X * Y      # prodotto elemento per elemanto
```
```
array([[0, 0, 0],
       [4, 5, 6],
       [0, 0, 0]])
```

Numpy Array - Operazioni

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Altri esempi di operazioni elemento per elemanto (“element-wise”):

- `x + y`
- `x – y`
- `x / y`
- `np.sqrt(x)`

Sommatoria:

```
x = np.array([[1,2],[3,4]])
print(np.sum(x))  # calcola la somma di tutti gli elementi; ouput:
"10"
print(np.sum(x, axis=0))  # calcola la somma di ogni colonna; output:
"[4 6]"
print(np.sum(x, axis=1))  # calcola la somma di ogni riga; output:
"[3 7]“
```

Numpy Array - Operazioni

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Gli ndarray possono essere trasposti con:

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

```
X.T
  array([[1, 4, 7],
         [2, 5, 8],
        [3, 6, 9]])
```

```
np.transpose(X)
 array([[1, 4, 7],
        [2, 5, 8],
       [3, 6, 9]])
```

Numpy Array – Indicizzazione booleana

Il boolean indexing consente di selezionare elementi arbitrari di un array.

Spesso questo tipo di indicizzazione viene utilizzata per selezionare gli elementi di una matrice che soddisfano alcune condizioni. Ad esempio:

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

a[bool_idx]
array([3, 4, 5, 6])

a[a > 2]
array([3, 4, 5, 6])
```

Numpy Array - Iterazione

Si può accedere agli elementi di un ndarray, uno ad uno, con dei cicli for:

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

```
for i, elemento in enumerate(X):
    print(i, elemento)
```
0 1
1 2
2 3
3 4
4 5
5 6
6 7
7 8
8 9

```
X.shape = (3,3)
```

```
for i, elemento in enumerate(X):
    print(i, elemento)
```
0 [1 2 3]
1 [4 5 6]
2 [7 8 9]

Array: copia

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

- Quando si opera con gli array è essenziale sapere se i dati vengono copiati in un nuovo array o no:
  - **Nessuna copia**: Le istruzioni di assegnazione come `arr2 = arr1` non copiano i dati, creano solo un alias
  - **Copia profonda**: Il metodo *copy()* crea una copia completa dell'array e dei dati
  - **Vista o copia superficiale**: lo slicing produce una vista, cioè un nuovo array con dati condivisi

Array: copia


```
a = np.array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])

b = a[:2, 1:3]
array([[2, 3],
       [6, 7]])

b[0,0] = 99

array([[ 1, 99,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
```

Array: modifica delle dimensioni

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Come per le liste, possiamo aggiungere elementi a un ndarray con *append()*:

```
l = [1,2,3,4,5]  # lista Python
a = np.array([1,2,3,4,5])  # array Numpy
l.append(6)
```

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

```
np.append(a, 6)
```

```
array([1, 2, 3, 4, 5, 6])
```

append() su un ndarray **ritorna un nuovo ndarray**, non altera quello originale!
Per alterare l'ndarray originale:

```
a = np.append(a, 6)
```

Attenzione: è un'operazione computazionalmente molto costosa, perché comporta la copia dell'array!

Metodo Where

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

numpy.where

`numpy.where(condition, [x, y, ]/)`

Return elements chosen from *x* or *y* depending on *condition*.

Note

When only *condition* is provided, this function is a shorthand for `np.asarray(condition).nonzero()`. Using `nonzero` directly should be preferred, as it behaves correctly for subclasses. The rest of this documentation covers only the case where all three arguments are provided.

Parameters: **condition** : *array_like, bool*

Where True, yield *x*, otherwise yield *y*.

x, y : *array_like*

Values from which to choose. *x*, *y* and *condition* need to be broadcastable to some shape.

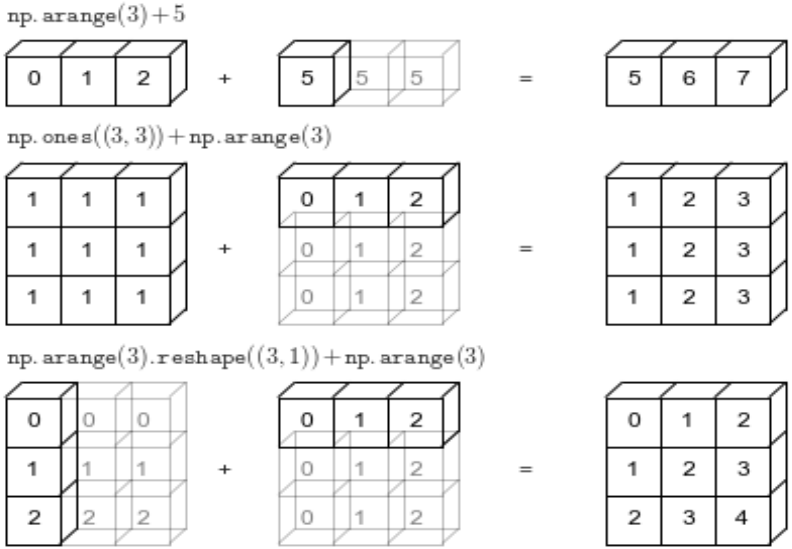
Returns: **out** : *ndarray*

An array with elements from *x* where *condition* is True, and elements from *y* elsewhere.

Broadcasting

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Il broadcasting è il meccanismo di NumPy per eseguire operazioni aritmetiche su array di forma diversa in maniera ottimizzata
L'array più piccolo viene "diffuso" sull'array più grande in modo che abbiano dimensioni compatibili.



Riferimenti

- **A tu per tu col Machine Learning. L'incredibile viaggio di un developer nel favoloso mondo della Data Science**, Alessandro Cucci, The Dot Company, 2017 [cap.2]
- **NumPy user guide**, https://numpy.org/doc/stable/user/index.html

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Proviamo…

Esercizi su NumPy (suggerimento: consultate la documentazione on-line)

1. Importa numpy come np e stampa il numero di versione

2. Crea un array 1D di numeri da 0 a 99

3. Crea un array numpy 3×3 di tutti i True

4. Estrai i numeri dispari da arr

Input:

```
arr = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

Output:

```
array([1, 3, 5, 7, 9])
```

Esercizi su NumPy

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

5. Sostituisci tutti i numeri dispari in arr con -1

Input:
```
arr = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```
Output:
```
array([ 0, -1,  2, -1,  4, -1,  6, -1,  8, -1])
```

6. Sostituisci tutti i numeri dispari in arr con -1 senza modificare arr

Input:
```
arr = np.array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```
Output:
```
out
#>  array([ 0, -1,  2, -1,  4, -1,  6, -1,  8, -1])
arr
#>  array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

Esercizi su NumPy

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

7. Converti un array 1D in un array 2D con 2 righe

Input:
```
np.arange(10)
```
Output:
```
array([[0, 1, 2, 3, 4],
       [5, 6, 7, 8, 9]])
```

8. Impila gli array a e b verticalmente

Input
```
a = np.arange(10).reshape(2,-1)
b = np.repeat(1, 10).reshape(2,-1)
```
Output:
```
array([[0, 1, 2, 3, 4],
       [5, 6, 7, 8, 9],
       [1, 1, 1, 1, 1],
       [1, 1, 1, 1, 1]])
```

Esercizi su NumPy

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

9. Impila gli array a e b orizzontalmente

Input
```
a = np.arange(10).reshape(2,-1)
b = np.repeat(1, 10).reshape(2,-1)
```
Output:
```
array([[0, 1, 2, 3, 4, 1, 1, 1, 1, 1],
       [5, 6, 7, 8, 9, 1, 1, 1, 1, 1]])
```

10. Ricava le posizioni in cui gli elementi di a e b coincidono

Input:
```
a = np.array([1,2,3,2,3,4,3,4,5,6])
b = np.array([7,2,10,2,7,4,9,4,9,8])
```
Output:
```
(array([1, 3, 5, 7]), dtype=int64)
```

Esercizi su NumPy

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

11. Estrai gli elementi tra 5 e 10 da a

Input:

```
a = np.array([2, 6, 1, 9, 10, 3, 27])
```

Output:

```
(array([6, 9, 10]),)
```

12.Scambia le colonne 1 e 2 in arr

Input:

```
arr = np.arange(9).reshape(3,3)
```

Output:

```
array([[1, 0, 2],
       [4, 3, 5],
       [7, 6, 8]])
```

Esercizi su NumPy

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

13. Scambia le righe 1 e 2 in arr

Input:

```
arr = np.arange(9).reshape(3,3)
```

Output:

```
array([[3, 4, 5],
       [0, 1, 2],
       [6, 7, 8]])
```

14. Scopri se iris_2d ha valori mancanti (nan)

Input

```
url = 'https://archive.ics.uci.edu/ml/machine-learning-databases/iris/iris.data'
iris_2d = np.genfromtxt(url, delimiter=',', dtype='float',
usecols=[0,1,2,3])
```

Esercizi su NumPy

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

15. Converti array_of_arrays in un array 1D flat linear

Input:

```
arr1 = np.arange(3)
arr2 = np.arange(3,7)
arr3 = np.arange(7,10)
array_of_arrays = np.array([arr1, arr2, arr3])
#> array([array([0, 1, 2]), array([3, 4, 5, 6]), array([7, 8, 9])], dtype=object)
```

Output:

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

16. Calcola il massimo di ciascuna riga dell'array a

Input:

```
np.random.seed(100)
a = np.random.randint(1,10, [5,3])
#> array([[9, 9, 4],[8, 8, 1],[5, 3, 6],[3, 3, 3],[2, 1, 9]])
```

Esercizi su NumPy

AImage Lab
UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

17. Rimuovi tutti i valori nan da un array 1D

Input:
```
np.array([1,2,3,np.nan,5,6,7,np.nan])
```
Output:
```
array([ 1.,  2.,  3.,  5.,  6.,  7.])
```

18. Sottrai l'array 1D b_1d dall'array 2D a_2d, in modo che b_1d sia sottratto da ciascuna riga di a_2d

Input:
```
a_2d = np.array([[3,3,3],[4,4,4],[5,5,5]])
b_1d = np.array([1,2,3])
```
Output:
```
#> [[2 1 0]
#>  [3 2 1]
#>  [4 3 2]]
```