DEFINIZIONE ARRAY

`a = np.array([2000, 4000, 8000])`	Array monodimensionale (1D)
`a = np.array([(10,5,3),(4,9,6)])`	Array bidimensionale (2D)
`a = np.zeros(2)`	Array 1D di lunghezza 2 con valori 0
`a = np.zeros((2,3))`	Array 2D 2x3 con valori 0
`a = np.ones(2)`	Array 1D di lunghezza 2 con valori 1
`a = np.ones((2,3))`	Array 2D 2x3 con valori 1
`a = np.eye(4)`	Array 4x4 di 0 con diagonale principale a 1
`a = np.linspace(0,10,5)`	Array di lung. 5 con i valori equamente distribuiti tra 0-10
`a = np.arange(0,10,3)`	Array di valori compresi tra 0-10 con passo 3 – [0, 3, 6, 9]
`a c np.random.rand(2,3)`	Array 2x3 di valori casuali tra 0-1
`a = np.random.randint(6,size=(4,3))`	Array 4x3 di valori casuali interi tra 0-5

OPERATORI ARITMETICI E FUNZIONI MATEMATICHE

`x = b + a` `x = np.add(b,a)`	addizione
`x = a - b` `x = np.subtract(a,b)`	sottrazione
`x = b * a` `x = np.multiply(b,a)`	moltiplicazione
`x = a / b` `x = np.divide(a,b)`	divisione
`x = np.exp(a)`	esponenziale
`x = np.power(a,b)`	elevamento a potenza
`x = np.sqrt(a)`	radice quadrata
`x = np.absolute(a)`	valore assoluto
`x = np.log(a)`	funzione logaritmo
`x = np.log10(a)`	funzione logaritmo base 10
`x = np.sin(a)`	funzione seno
`x = np.cos(a)`	funzione coseno
`x = np.tan(a)`	funzione tangente
`x = np.sinh(a)`	funzione seno iperbolico
`x = np.cosh(a)`	funzione coseno iperbolico
`x = np.tanh(a)`	funzione tangente iperbolica
`x = np.deg2rad(270)`	funzione che converte gradi in radianti
`x = np.rad2deg(ni.pi/2)`	funzione che converte radianti in gradi
`x = np.round(a)`	funzione arrotondamento
`x = np.floor(a)`	funzione troncamento
`x = np.ceil(a)`	funzione arrotondamento (per eccesso)
`x = np.trunc(a)`	funzione arrotondamento (per difetto)
`x = np.prod(a)`	funzione moltiplicazione elementi array
`x = np.sum(a)`	funzione addizione elementi array

DATA MASTERS

OPERATORI LOGICI E DI CONFRONTO

`x = np.logical_and(a,b)`	AND tra array
`x = np.logical_or(a,b)`	OR tra array
`x = np.logical_not(a)`	NOT
`x = b == a` `x = np.equal(b,a)`	uguaglianza
`x = b != a` `x = np.not_equal(b,a)`	disuguaglianza
`x = b < a` `x = np.less(b,a)`	minore
`x = b > a` `x = np.greater(b,a)`	maggiore
`x = b <= a` `x = np.less_equal(b,a)`	minore uguale
`x = b >= a` `x = np.greater_equal(b,a)`	maggiore uguale
`x = np.array_equal(a,b)`	uguaglianza array
`x = np.array_equiv(a,b)`	equivalenza array

INDICI E SLICING

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

`x = a[2]`	3
`x = a[:2]`	[1 2]
`x = a[2:]`	[3]
`x = a[-1]`	3
`x = b[0][1]`	2
`x = c[0][1][2]`	6
`x = a[0:2]`	[1 2]
`x = b[0:2,2]`	[3 6]
`x = b[:,2]`	[3 6]
`x = c[:,1,:]`	[[4 5 6][10 11 12]]
`x = c[1,1,:]`	[10 11 12]
`x = c[1,1,1]`	11

COPIA, ORDINAMENTO E FILTRAGGIO

```
a = np.array([1,6,4,2,5,3,9,8,7])
b = np.array([[2,1,10],[7,6,3],[11,8,9]])
c = np.array([12,66,42,22,54,3,95,84,75])
```

`x = a.view()`	Crea un nuovo oggetto array che punta agli stessi elementi
`x = a.copy()`	Copia l'array a in una nuova locazione di memoria
`np.copyto(c,a)`	Copia l'array a in una nuova locazione di memoria
`a = a.sort()`	[1 2 3 4 5 6 7 8 9]
`b = b.sort()`	[[1 2 10] [3 6 7] [8 9 11]]
`filter_a = a > 4` `new_a = a[filter_a]`	[6 5 9 8 7]

PROPRIETÀ ARRAY

`b.size`	Numero di elementi nell'array b
`b.shape`	Dimensioni (righe, colonne)
`b.dtype`	Tipo di elementi nell'array b
`b.astype(dtype)`	Converte gli elementi di b secondo il tipo dtype

NUMPY CHEATSHEET

ITERARE UN NDARRAY

```
a = np.array([[1, 2], [3, 4]])
for x in a:
  print(x)
```
```
[1 2]
[3 4]
```

```
a = np.array([[1, 2], [3, 4]])
for x in a:
  for y in x:
    print(y)
```
```
1
2
3
4
```

```
a = np.array([[1, 2], [3, 4]])
for x in np.nditer(a):
  print(x)
```
```
1
2
3
4
```

```
# Ordinamento row-major(C)
a = np.array([[1, 2], [3, 4]])
for x in np.nditer(a, order="c"):
    print(x)
```
```
1
2
3
4
```

```
# Ordinamento column-major(Fortran)
a = np.array([[1, 2], [3, 4]])
for x in np.nditer(a, order="f"):
    print(x)
```
```
1
2
3
4
```

```
# Lettura e modifica Ndarray
a = np.array([[1, 2], [3, 4]])
with np.nditer(a, op_flags=['readwrite']) as it:
    for x in it:
        x[...] = 2*x
print(a)
```
```
[[2 4]
 [6 8]]
```

```
# Loop per blocchi column-major
a = np.array([[1, 2], [3, 4]])
for x in np.nditer(a,
        flags=["external_loop"],
        order="f"):
    print(x)
```
```
[1 3]
[2 4]
```

```
# Indicizzazione elementi
a = np.array([[1, 2], [3, 4]])
it = np.nditer(a,
        flags=["multi_index"])
for x in it:
    print(x, it.multi_index)
```
```
1 (0, 0)
2 (0, 1)
3 (1, 0)
4 (1, 1)
```

```
# Indicizzazione elementi row-major
a = np.array([[1, 2], [3, 4]])
it = np.nditer(a, flags=["c_index"])
while not it.finished:
    print("%d - %d" % (it[0], it.index))
    is_not_finished = it.iternext()
```
```
1 - 0
2 - 1
3 - 2
4 - 3
```

```
# Indicizzazione elementi row-major
a = np.array([[1, 2], [3, 4]])
it = np.nditer(a, flags=["c_index"])
while not it.finished:
    print("%d - %d" % (it[0], it.index))
    is_not_finished = it.iternext()
```
```
1 - 0
2 - 2
3 - 1
4 - 3
```

```
# Indicizzazione elementi column-major
a = np.array([[1, 2], [3, 4]])
it = np.nditer(a, flags=["f_index"])
while not it.finished:
    print("%d - %d" % (it[0], it.index))
    is_not_finished = it.iternext()
```
```
1 - 0
2 - 2
3 - 1
4 - 3
```

```
# Iteratore su array broadcastable
a = np.array([1,2,3])
b = np.array([(1,2,3),(4,5,6)])
for x, y in np.nditer([a, b]):
    print(x, y)
```
```
1 1
2 2
3 3
1 4
2 5
3 6
```

UNIONE DI NDARRAY

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

`np.concatenate([a,b], axis=0)`	[[1 2 3] [4 5 6] [7 8 9] [10 11 12]]
`np.concatenate([a,b], axis=1)`	[[1 2 3 4 5 6] [7 8 9 10 11 12]]
`np.hstack((a, c))`	array([[1, 2, 3, 7, 8, 9], [4, 5, 6, 10, 11, 6]])
`np.vstack((a, c))`	array([[1, 2, 3], [4, 5, 6], [7, 8, 9], [10, 11, 6]])
`np.stack((a, c), axis=1)`	array([[[1, 2, 3], [7, 8, 9]], [[4, 5, 6], [10, 11, 6]]])
`np.column_stack((a, c))`	array([[1, 2, 3, 7, 8, 9], [4, 5, 6, 10, 11, 6]])
`np.intersect1d(a, c)`	array([6])
`np.union1d(a, b)`	array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11])

SPLIT DI ARRAY

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

`np.split(a,3)`	[array([1, 2]), array([3, 4]), array([5, 6])]
`np.array_split(a, 4)`	[array([1, 2]), array([3, 4]), array([5]), array([6])]
`np.hsplit(b,3)`	[array([[1],[4],[7]]), array([[2],[5],[8]]), array([[3],[6],[9]])]
`np.vsplit(b,3)`	[array([[1, 2, 3]]), array([[4, 5, 6]]), array([[7, 8, 9]])]

LAVORARE CON LE SHAPE

`b = np.array([(1,2,3),(4,5,6)])` `b.shape`	(2, 3)
`b = np.array([(1,2,3),(4,5,6)])` `b.reshape(3,2)`	[[1 2] [3 4] [5 6]]
`b = np.array([(1,2,3),(4,5,6)])` `b.flatten()`	[1 2 3 4 5 6]
`b = np.array([(1,2,3),(4,5,6)])` `b.T`	[[1 4] [2 5] [3 6]]

ALL, ANY, WHERE

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

`np.all(a)`	**False**
`np.any(a)`	**True**
`np.where(a<4, a, a*4)`	[[1 2 3] [16 20 0] [28 32 36]]

STATISTICA

`np.mean(a)` `np.mean(a,axis=0)`	Media aritmetica
`np.median(a)`	Valore mediano
`a.amin()` `a.amin(axis=0)`	Valore minimo
`a.amax()` `a.amax(axis=0)`	Valore massimo
`np.var(a)`	Varianza
`np.std(a)` `np.std(a,axis=1)`	Deviazione standard
`np.percentile(a,75)`	Percentile 75 %