NumPy

NumPy

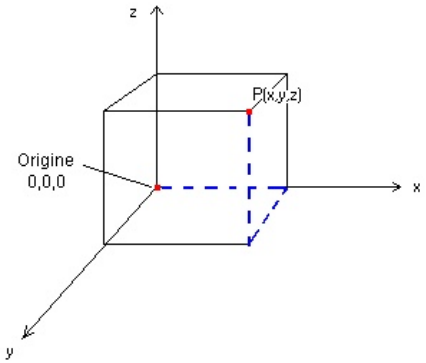
- ***libreria open source*** Python (2005 Travis Oliphant)
- aggiunge funzionalità per operare con grandi ***matrici e array multidimensionali*** con funzioni matematiche di ***alto livello***
- il ***core*** è ***ottimizzato*** (linguaggio C)
- sito di riferimento https://numpy.org/
- documentazione https://numpy.org/doc/
- installazione `pip3 install numpy`

perché NumPy

- ***funzioni*** e ***metodi*** agiscono ***globalmente*** su vettori e matrici
  - si evitano cicli espliciti (poco efficienti) sui singoli dati
- ***algoritmi testati*** e ***ottimizzati***
  - gestione della memoria più efficiente che in Python
- più ***semplice input/output*** su dati di grandi dimensioni

array multidimensionali

- oggetto principale di NumPy:
  - ***array multidimensionale omogeneo*** (elementi tutti dello ***stesso tipo***)
  - anche non omogeneo
- elementi ***indicizzati*** da una ***tupla*** di numeri interi non negativi
  - le dimensioni sono chiamate ***axes***
- esempio
- il punto P di coordinate [2,3,2] è rappresentato da un array monodimensionale: ha 1 **`axes`** con 3 **`elements`**
- una matrice bidimensionale ha 2 axes:
  il primo (axis 0) rappresenta le righe, il secondo (axis 1) le colonne
- [[2,3,2],[3,8,5]] è una **`matrice`** con 2 **`axes`** ognuno con 3 **`elements`**



500 migliori album secondo Rolling Stone

posizione	titolo	artista	anno
1	Sgt. Pepper's Lonely Hearts Club Band	The Beatles	1967
2	Pet Sounds	The Beach Boys	1966
3	Revolver	The Beatles	1966
4	Highway 61 Revisited	Bob Dylan	1965
5	Rubber Soul	The Beatles	1965

creazione NumPy array

- ***conversione*** di strutture Python
  - liste, tuple
- creazione diretta mediante ***funzioni***
  - arange, ones, zeros, …
- caricamento da file su disco

```
import numpy as np
a = np.array([0,1,2,3,4,5,6,7,8,9,10,11,12,13,14])          # a = np.arange(15)
print(a)                # [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14]
print(a.shape)          # (15,)
print(a.ndim)           # 1
print(a.dtype.name)     # 'int64' or 'int32'
print(a.itemsize)       # 4 or 8
print(a.size)           # 15
print(type(a))          # <class 'numpy.ndarray'>

b = a.reshape(3, 5)
print(b)
'''
[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]]
'''
print(b.shape)          # (3, 5)
print(b.ndim)           # 2
print(b.dtype.name)     # 'int64' or 'int32'
print(b.itemsize)       # 4 or 8
print(b.size)           # 15
print(type(b))          # <class 'numpy.ndarray'>
```

esempio

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

d = np.array( [ [1,2], [3,4] ], dtype=np.float64 )
print(d)

e = np.zeros((3, 4))
print(e)

f = np.ones((2,3))
print(f)

g = np.arange( 15, 30, 3 )
print(g)

h = np.arange( 0.5, 2, 0.4 )
print(h)
```

```
[[1.+0.j 2.+0.j]
 [3.+0.j 4.+0.j]]
[[1. 2.]
 [3. 4.]]
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]
[[1. 1. 1.]
 [1. 1. 1.]]
[15 18 21 24 27]
[0.5 0.9 1.3 1.7]
```

ndarray (attributes)

- **`ndarray.ndim`**
  - number of axes (dimensions) of the array
- **`ndarray.shape`**
  - dimensions of the array
  - a tuple of integers indicating the size of the array in each dimension
    - for a matrix with n rows and m columns, shape will be (n,m)
    - the length of the shape tuple is the number of axes, ndim.
- **`ndarray.size`**
  - total number of elements of the array
  - equal to the product of the elements of shape
- **`ndarray.dtype`**
  - type of the elements in the array