1. Import du package :

In [1]:
```
import numpy as np
```

2-Creer des donnees

2.1-Creer un numpy array a partir d'une liste/d'une liste de liste(matrice)

In [25]:
```
tab1=np.array([1,2,3,4,5,0,-5,-7,-8])
tab2=np.array([[1,2],[2,3]])
tab3=np.array([[[1,2],[3,4],],[[0,2],[0,5]],[[0,1],[2,9]]])
```

In [10]:
```
tab2
```

Out[10]:
```
array([[1, 2],
       [2, 3]])
```

2.2 Creer un numpy array a partir des fonction predefinies

In [13]:
```
zero=np.zeros((3,3)) # matrice de zero
one=np.ones((3,3))     #matrice de un
eye=np.eye(6)          # matrice identite
```

In [14]:
```
eye
```

Out[14]:
```
array([[1., 0., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0., 0.],
       [0., 0., 1., 0., 0., 0.],
       [0., 0., 0., 1., 0., 0.],
       [0., 0., 0., 0., 1., 0.],
       [0., 0., 0., 0., 0., 1.]])
```

In [17]:
```
# creer une liste avec np.arange(debut , fin -1, pas )
array_arrange=np.arange(1,11,1)
```

```
array_arrange
```

Out[17]:
```
array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
```

In [18]:
```
# creer une liste avec np.linspace(debut , fin, nombre d'element)
np.linspace(0,2,5) # tres utile pour les graphiques
```

Out[18]:
```
array([0. , 0.5, 1. , 1.5, 2. ])
```

In [24]:
```
# creer un numpy array de nombres aleatoires np.random.random(shape)
np.random.random((4,2)) # utile pour initialiser les poids dans les reseaux de neurones
```

Out[24]:
```
array([[0.44559828, 0.96737936],
       [0.90123875, 0.09899042],
       [0.911054  , 0.66042759],
       [0.86678829, 0.13974047]])
```

3-Fonctions mathematiques et statistiques avec numpy

In [38]:
```
# fonctions mathematiques basiques
tab_sum=np.sum(tab1)
tab_mean=np.mean(tab1)
tab_min=np.min(tab1)
tab_max=np.max(tab1)
tab_std=np.std(tab1)
tab_sqrt=np.sqrt(tab1)
tab_abs=np.abs(tab1)
tab_power=np.power(tab1,3)
tab_exp=np.exp(tab1)
tab_log=np.log(tab1)
tab_median=np.median(tab1)
tab_median
```

```
C:\Users\Lemofouet valdini\AppData\Local\Temp\ipykernel_10064\3042533463.py:7: RuntimeWarning: invalid value encountered in sqrt
  tab_sqrt=np.sqrt(tab1)
C:\Users\Lemofouet valdini\AppData\Local\Temp\ipykernel_10064\3042533463.py:11: RuntimeWarning: divide by zero encountered in lo
g
  tab_log=np.log(tab1)
C:\Users\Lemofouet valdini\AppData\Local\Temp\ipykernel_10064\3042533463.py:11: RuntimeWarning: invalid value encountered in log
  tab_log=np.log(tab1)
```

Out[38]:
```
1.0
```

4-manipulation des matrice avec numpy

In [39]:
```
# Produit matriciel
matrice=[[1,0,3],[2,2,8]]
matrice_p= np.dot(tab2,matrice)
matrice_p
```

Out[39]:
```
array([[ 5,  4, 19],
       [ 8,  6, 30]])
```

In [42]:
```
# transpose
np.transpose(matrice_p)
```

Out[42]:
```
array([[ 5,  8],
       [ 4,  6],
       [19, 30]])
```

In [45]:
```
# inverse matrice (pour les matrice carrees non singulieres)
np.linalg.inv(tab2)
```

Out[45]:
```
array([[-3.,  2.],
       [ 2., -1.]])
```

In [48]:
```
 # determinant
np.linalg.det(eye)
```

Out[48]:
```
1.0
```

5- Acceder aux elements,modifier,selectionner et mettre a jour un numpy array

In [49]:
```
# acceder
tab1[2]
```

Out[49]:
```
3
```

In [51]:
```
tab1[2]=7
tab1[2]
```

Out[51]:
```
7
```

In [52]:
```
# selectionner un element qui respecte une condition
tab1[tab1>2]
```

Out[52]:
```
array([7, 4, 5])
```

In [53]:
```
#indice des elements respectant la condition
np.where(tab1>2)
```

Out[53]:
```
(array([2, 3, 4], dtype=int64),)
```

In [54]:
```
# slicing et indexation avancee
#slicing
tab2[0,:]
```

Out[54]:
```
array([1, 2])
```

In [55]:
```
tab2[:,0]
```

Out[55]:
```
array([1, 2])
```

In [61]:
```
#indexation avancee
tab2[[0,1,1],[1,1,0]]
```

Out[61]:
```
array([2, 3, 2])
```

In [63]:
```
# addition
ar1=np.array([1,2,3])
ar2=np.array([4,5,3])
ar1+ar2
```

Out[63]:
```
array([False, False,  True])
```

In [64]:
```
# multiplication
ar1*ar2
```

Out[64]:
```
array([ 4, 10,  9])
```

In [75]:
```
ar3=np.array([[1,2,3],[4,2,7]])
ar3+np.array([1,1,1])
```

Out[75]:
```
array([[2, 3, 4],
       [5, 3, 8]])
```

In [72]:
```
ar4=np.arange(9)
ar4
ar4.reshape((3,3))
```

Out[72]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8])
```

In [73]:
```
#ajouter un element dans une matrice
np.append(tab1,[0,6,9])
```

Out[73]:
```
array([ 1,  2,  7,  4,  5,  0, -5, -7, -8,  0,  6,  9])
```

In [78]:
```
#connaitre le type d'un objet
ar3.dtype
```

Out[78]:
```
dtype('int32')
```

In [ ]: