np_tuto1.py

```
######################################################################
#################### simulation- Numpu tuto1#########################
######################################################################

import numpy as np
""" # or
# import numpy
# or
from numpy import array, ones
# or
from numpy import *
 """

# #### ---------- Dimensions 0 ----------
# %%
l0=120
a0 = np.array(l0)
print(l0,type(l0), a0,type(a0),sep='\n*****\n')
print('l0+l0=',l0+l0)
print('3*l0=',3*l0)
print('a0+a0=',a0+a0)
print('3*a0=',3*a0)
print(a0,'a0 est un array de dimension0',)

# %%
s0='bjr'
as0=np.array(s0)
s0=120
as0 = np.array(s0)
print(s0,type(s0), as0,type(as0),sep='\n*****\n')
print('s0+s0=',s0+s0)
print('3*s0=',3*s0)
print('as0+as0=',as0+as0)
print('3*as0=',3*as0)
print(as0,'as0 est un array de dimension0',)

# #### ---------- Dimensions 1 ----------

# %%
# 1-D array
l1=[3, 1, 7, 5]
a1 = np.array(l1)
print(l1,type(l1), a1,type(a1),sep='\n*****\n')
print('l1+l1=',l1+l1)
print('10*l1=',10*l1)
print(a1,'\n','a1+a1=',a1+a1)
print(a1,'\n','10*a1=',10*a1)
print(a1,'a1 est un array de dimension1',)

# %%
l1_0=[1,2,3,'dddd',{1,2},True,3.14,1+2j,{1:2,3:4}]
a1_0=np.array(l1_0)
```

```
print(l1_0,a1_0)
# ===>les elements d'un array peuvent etre de type
int,str,float,complex,boolean,set,dict
l1_1=[1,2,3,[5,6]]
a1_1=np.array(l1_1)
# ===>les elements d'une matrice ne peuvent pas etre des listes (sont reserves aux
array(dimensions superieures)))))

# ### D1: acceder aux elements d'un array vs d'une liste

# %%

print(l1[0],a1[0],l1[1],a1[1],l1[-1],a1[-1])
print(type(l1[0]),type(a1[0]))
print(len(l1),len(a1))
l1.append(7)
# a1.append(7) ===>error
print(l1.index(3))
# print(a1.index(3))===>error
# ====> .append,.extend,.pop,.inset,.index,.remove sont des methode reservees aux listes
et donc ne s'appliquent pas aux array

# ### D2: Itérer un array vs une liste
# %%
# iteration de la liste
#--- element par element-----
for e in l1:
    print('{} est un eleme de la liste {}'.format(e,l1))
#--- par indice -----
for i in range(len(l1)):
    print('{} est le {} eme eleme de la liste {}'.format(l1[i],i,l1))

# %%
# iteration de l array
#--- element par element-----
for e in a1:
    print('{} est un eleme de la liste {}'.format(e,a1))
#--- par indice -----
for i in range(len(a1)):
    print('{} est le {} eme eleme de la liste {}'.format(a1[i],i,a1))

# #### ---------- Dimensions 2 ----------
# %%
# 2-D array
l2=[[1,2,3,4,5],[6,7,8,9,10],[11,12,13,14,15]]
a2 = np.array(l2)
print(l2,type(l2), a2,type(a2),sep='\n*****\n')
print('l2+l2=',l2+l2)
print('10*l2=',10*l2)
print(a2,'\n','a2+a2=',a2+a2)
print(a2,'\n','10*a2=',10*a2)
print(a2,'a2 est un array de dimension 2',)
print('cosA+sinA+exp(10A)=',np.cos(a2)+np.sin(a2)+np.exp(10*a2))

# %%
# 2-D array
# l2_0=[[1,2,],[6,7,8,9,10],[11,12,13,14,15]]
```

```
# a2_0 = np.array(l2_0)
l2_1=[['a','b','c','d','e'],[6,7,8,9,10],[11,12,13,14,15]]
a2_1 = np.array(l2_1)
print(a2_1)
# ===> tout les elements d'un array deviennent de meme type (st)
l2_2=[['a','b','c',{1,'aa'},'e'],[{"a":1},False,8.5,9,10],[11,12,13,14,15]]
a2_2 = np.array(l2_2)
print('a2_2=',a2_2)
# ===>les elements peuvent etre de type int,str,float,complex,boolean,set,dict
l2_3=[['a','b','c',{1,'aa'},[]],[6,7,8,9,10],[11,12,13,14,15]]
a2_3 = np.array(l2_3)
# ===>les elements ne peuvent pas etre des listes (sont reserves aux array(dimensions superieures)))))

# %%
print('l2=',l2,'\na2=',a2)
print(l2[0],a2[0],l2[1],a2[1],l2[-1],a2[-1])
print(type(l2[0]),type(a2[0]))
print(len(l2),len(a2))
print(l2[0][0],a2[0][0],l2[1][0],a2[1][0],l2[-1][0],a2[-1][0],l2[-1][-1],a2[-1][-1])
print(type(l2[0][0]),type(a2[0][0]))

# N.B.: Pour la simension 2, pour acceder a un element, a2[i][j]. Pour simplifier on peut utiliser a2[i,j]. On utilise l'indice de la ligne et l'indice de la colonne.
print(a2[0,0],a2[-1,-1])

# ### D2: Itérer un array vs une liste
# %%
# iteration de la liste a2
#--- element par element-----
for e in l2:
    for x in e:
        print('{} est un eleme de la liste {}'.format(x,l2))
# --- par indices -----
for i in range(len(l2)):
    for j in range(len(l2[i])):
        print('{} est le {} eme eleme de la liste {}'.format(l2[i][j],i,l2))


# %%
# iteration de la matrice a2
#--- element par element-----
for e in a2:
    for x in e:
        print('{} est un eleme de la liste {}'.format(x,a2))
# --- par indices -----
for i in range(len(a2)):
    for j in range(len(a2[i])):
        print('{} est le {} eme eleme de la liste {}'.format(a2[i,j],i,a2))

# %%
print('\n---------- Dimensions ----------')
# Dimensions
# 0-D array
a = np.array(120)
print('0 d array:', a,type(a))
```

```
# %%
# 1-D array
one_d = np.array([3, 1, 7, 5])
print('1 d array:', one_d,type(one_d))

# %%
# 2-D array
two_d = np.array([[1, 3, 2], [5, 6, 4]])
print('2 d array:\n', two_d,type(two_d))

# %%
# 3-D array
three_d = np.array([[[1, 2], [2, 3]], [[3, 5], [6, 7]]])
print('3 d array:\n', three_d,type(three_d))

# %%
print('\n---------- Array attributes ----------')
print('Array dimensions:',one_d, one_d.ndim,sep='\n',)
################################################
print('\n---------- Array attributes ----------')
print('Array dimensions:',two_d, two_d.ndim,sep='\n',)
################################################
print('\n---------- Array attributes ----------')
print('Array dimensions:',three_d, three_d.ndim,sep='\n',)

# %%
print('\n---------- Accessing/Indexing ----------')
print('first item in one d array:',one_d,'=> one_d[0]=', one_d[0], sep='\n')
print('first item in two d array:',two_d,'=>two_d[0, 0]=', two_d[0, 0], sep='\n')
print('first item in three d array:',three_d,'=>three_d[0, 0, 0]=', three_d[0, 0, 0],
sep='\n')

# %%

d=np.array([[1,2,3],[10,20,30],[100,200,300],[1000,2000,3000]])
print(d[0,0],d[1,1],d[1,-1],d[1],d[:,-1],d[-1][0])

# %%
print('\n---------- Iterating ----------')
print(one_d,'\n',10*'*','\n')
# 1-d array iterating
for x in one_d:
    print(x)

# %%
print('\n---------- Iterating ----------')
print(two_d,'\n',10*'*','\n')
# 2-d array iterating
for x in two_d:
    print(x)

# 2-d array iterating
print('Iterating over 2 d array')
for x in two_d:
    for y in x:
        print(y)
```

```
# %%
# 3-d array iterating
print(three_d)
print('Iterating over 3 d array')
for x in three_d:
    for y in x:
        for f in y:
            print(f)

# %%
print('Iterating over all items in array using nditer:')
for x in np.nditer(two_d):
 print(x)

# %%

print(two_d)
print('\n---------- Search ----------')
# where function
print(np.where(two_d == 4))  # will find all indices match the condition
print(np.where(two_d % 2 == 0))  # will find all indices match the condition (all even
numbers)
print(np.where(two_d % 2 != 0))  # will find all indices match the condition (all odd
numbers)
print(np.where(two_d % 2 == 11))  # will find all indices match the condition (all
numbers divisible by11)

# %%
print('\n---------- Sort ----------')
print(two_d)
print('Sorting array:\n', np.sort(one_d))
print('Sorting each row in array: \n', np.sort(two_d))

# %%
dd=np.array([[1,2,3,5],[1,2,3,5],[1,2,3,'bjr']])
print(dd)

# %%
np.diag([20,30,40])
np.ones(10)
np.ones((3,5))
10*np.ones((3,5))
np.zeros(10)
np.zeros((3,5))
print(a2)
a2+a2
a2*a2
l2_3=[[1,2,3,4,5],[ 6,7,8,9,10], [11,12,13,14,15],[16,17,18,19,20],
[21,22,23,24,25]]
a2_3=np.array(l2_3)
print(a2_3)
a2_3.dot(a2_3)
```