NumPy_Ricard_Perpinyà

January 18, 2021

1 Resum de llibreria NumPy ~ Ricard Perpinyà Alòs

1.1 1. Creació ndarrays

Exemple inicial:

[2]:
```
import numpy as np
```

[4]:
```
# Creem una llista 1D amb nou elements i la convertim en un ndarray 1D
llista_1D = range(9)
print(llista_1D)
print(type(llista_1D))

array_1D = np.array(llista_1D)
print(array_1D) #Print
print("\n","Forma: ", array_1D.shape) #Shape
print("Mida: ", len(array_1D),",",array_1D.size) #Mida
print(type(array_1D)) #Tipus dades
```

```
range(0, 9)
<class 'range'>
[0 1 2 3 4 5 6 7 8]

 Forma:  (9,)
Mida:  9 , 9
<class 'numpy.ndarray'>
```

1.2 Crear array a partir d'un rang amb `np.arange()` i refer la seva forma amb `.reshape()`:

[43]:
```
#Crear a partir d'un rang i refer la seva forma
a = np.arange(1,16).reshape(3,5)
print(a)
print("Dimensió: ",np.ndim(a)," \n ","Forma: ", np.shape(a))
```

```
[[ 1  2  3  4  5]
 [ 6  7  8  9 10]
 [11 12 13 14 15]]
```

```
Dimensió:  2
  Forma:  (3, 5)
```

1.3 Crear array a partir d'una llista:

[14]:
```
#Crear a partir de llista

a = []
for i in range(1,16):
    a.append(i)

print(a, type(a))

b = np.array(a)
print(b, type(b))

b.reshape(5,3)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15] <class 'list'>
[ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15] <class 'numpy.ndarray'>
```

[14]:
```
array([[ 1,  2,  3],
       [ 4,  5,  6],
       [ 7,  8,  9],
       [10, 11, 12],
       [13, 14, 15]])
```

1.4 Crear array a partir d'una llista 2D (concatenació de llistes):

[16]:
```
#Crear a partir de llista 2D

A = [[1,2,3],[4,5,6],[7,8,9]]

B = np.array(A)
print(B,type(B))
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]] <class 'numpy.ndarray'>
```

1.5 Definir un tipus de dada (`float, int, complex...`).

Els arrays són homogenis, és a dir, que tots els seus elements són del mateix tipus. Podem definir quin tipus de dades volem amb `dtype =`, i s'aplicarà a tots els elements.

[21]:
```
#Definir un tipus de dada:

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

```
print(a)
```

```
[[1.+0.j 2.+0.j 3.+0.j]
 [3.+0.j 4.+0.j 5.+0.j]]
```

1.6 Crear una matriu de zeros amb `np.zeros()`:

[54]:
```
#Crear una matriu de 0s

m = np.zeros((3,3))

print(m)
```

```
[[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
```

1.7 Crear una matriu d'uns amb `np.ones()`:

[6]:
```
#Crear un array d'1s
import numpy as no

array1 = np.ones((5,6))

print(array1)
```

```
[[1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1. 1.]]
```

1.8 Funció empty()

La funció `empty(shape, dtype=float, order='C')` retorna un ndarray amb la forma indicada. Els valors de les seves components no estan inicialitzats, i poden tenir qualsevol valor aleatori.

[168]:
```
#Funció empty

empty_floats = np.empty([3,2]) #FLOATS
empty_int = np.empty([3,3], dtype=int)


print(empty_floats)
print(empty_int)
```

```
[[ 69.21818182  95.30181818]
 [221.4        169.42909091]
 [373.58181818 243.55636364]]
[[4602678819172646912 4607182418800017408 4609434218613702656]
 [4611686018427387904 4612811918334230528 4613937818241073152]
 [4615063718147915776 4616189618054758400 4617315517961601024]]
```

1.9 Funció `eye()`

La funció `eye()` retorna una matriu identitat.

[7]:
```
#Funció eye

I = np.eye(2,2)

print(I)
```

```
[[1. 0.]
 [0. 1.]]
```

1.10 Matriu de valors aleatoris

Per crear matrius de valors aleatoris utilitzarem les següents funcions:

- `np.random.rand()` per valors float
- `np.random.randint()` per valors integer

[10]:
```
#Matriu random
import numpy as np
a = np.random.rand(3,6) #Floats
b = np.random.randint(1,200, size = (3,3) ) #Integers

print(a,"\n")

print(b)
```

```
[[0.75517981 0.69389895 0.13543066 0.2848111  0.75144873 0.29365062]
 [0.82667393 0.37604696 0.76918582 0.89932937 0.79318784 0.13026701]
 [0.03628289 0.38396672 0.78573416 0.15658112 0.65274561 0.42085763]]

[[ 95  12   9]
 [134 145 101]
 [183 123 189]]
```

1.11 Indexació

La indexació és igual que amb les llistes:

Si tenim `array = [1 2 3 4 5 6]`, `array[0]` retornarà el valor 1

si tenim `array2 = [[1 2 3], [4 5 6], [7 8 9]]`, `array2[2][2]` ens retornarà el valor 9.

[93]:
```
#Indexació

a = np.arange(1,15)

print(a,"\n"+"Nombre assenyalat: ",a[3])

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

print(b,"\n"+"Nombre assenyalat: ",b[2][1])

#Un altre exemple

c = np.random.rand(3,3)

print(c, "\n"+"Nombre assenyalat: ", c[2][2])
```

```
[ 1  2  3  4  5  6  7  8  9 10 11 12 13 14]
Nombre assenyalat:  4
[[1 2 3]
 [4 5 6]
 [7 8 9]]
Nombre assenyalat:  8
[[0.91458828 0.62028838 0.73172856]
 [0.33980238 0.08140407 0.50807665]
 [0.05341082 0.88726039 0.81148099]]
Nombre assenyalat:  0.8114809878319887
```

1.12 Transformar matriu a vector:

Utilitzarem `np.ravel()`

[102]:
```
#Vector retornat de qualsevol matriu (vector fila)

a = np.arange(1,26).reshape(5,5)
print(a)

rav = np.ravel(a)

print(rav)
```

```
[[ 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]]
[ 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]
```

1.13 Funcions `np.arange()` i `np.linspace()`

`np.arange()`: > Para crear secuencias de números enteros, *numpy* procura una función análoga a *range* que devuelve arrays en lugar de listas. ~(Sofyan)

[110]:
```
# arange

a = np.arange(1,25,2)

print(a)

#Podem utilitzar reshape per canviar-ne la forma:

a2 = a.reshape(3,4)

print("Reshape: ",a2)
```

```
[ 1  3  5  7  9 11 13 15 17 19 21 23]
Reshape:  [[ 1  3  5  7]
 [ 9 11 13 15]
 [17 19 21 23]]
```

`np.linspace()`: La funció **arange()** té el problema que, atesa la precisió finita dels càlculs amb float, a vegades no es pot predir exactament el nombre d'elements que generarà. Quan això sigui important, es pot usar la funció linspace() que permet especificar exactament el nombre d'elements:

linspace(min, max, num_elements)

[119]:
```
# linspace

espai = np.linspace(1,10,4)
print(espai)
```

```
[ 1.  4.  7. 10.]
```

1.14 Creació ndarray amb funcions

[128]:
```
#crear ndarray amb funcions:

def funcio(i,j):
    return i+j

A = np.fromfunction(funcio,(3,3))

print(A)
```

```
[[0. 1. 2.]
 [1. 2. 3.]
 [2. 3. 4.]]
```

1.15 Funció array.reshape()

[131]:
```
import numpy as np

llista = []

for i in range(1,16):
    llista.append(i)

print(llista)

array2 = np.array(llista)
print(array2)

array_reshape = array2.reshape(3,5)

print("Reshaped array: ","\n",array_reshape)


#És més ràpid això:

a = np.arange(1,16).reshape(3,5)

print("Manera ràpida: "+"\n",a)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]
[ 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15]
Reshaped array:
 [[ 1  2  3  4  5]
 [ 6  7  8  9 10]
 [11 12 13 14 15]]
Manera ràpida:
 [[ 1  2  3  4  5]
 [ 6  7  8  9 10]
 [11 12 13 14 15]]
```

1.16 Operacions de matrius amb un nombre i operacions de dues matrius element a element

[136]:
```
#Algunes operacions:

A = np.linspace(1,9,9).reshape(3,3)

print("Array A: "+"\n",A)
```

```
Array A:
 [[1. 2. 3.]
 [4. 5. 6.]
```

```
 [7. 8. 9.]]
```

[139]:
```
# Suma i resta d'un valor A TOTS ELS ELEMENTS:

A = np.linspace(1,9,9).reshape(3,3)

print("Array A: "+"\n",A)

C = A+2
print("Array C = A+2 :"+"\n",C)

D = A-2
print("Array D = A-2 :"+"\n",D)
```

```
Array A:
 [[1. 2. 3.]
 [4. 5. 6.]
 [7. 8. 9.]]
Array C = A+2 :
 [[ 3.  4.  5.]
 [ 6.  7.  8.]
 [ 9. 10. 11.]]
Array D = A-2 :
 [[-1.  0.  1.]
 [ 2.  3.  4.]
 [ 5.  6.  7.]]
```

[143]:
```
#Producte DE TOTS ELS ELEMENTS per un valor:
E = A*2

#Divisió DE TOTS ELS ELEMENTS per un valor:
F = A/2

#Potència DE TOTS ELS ELEMENTS per un valor:
G = A**2

#Mòdul DE TOTS ELS ELEMENTS per un valor:
H = A%2

#Operadors booleans DE TOTS ELS ELEMENTS:
I = A > 5


print("Array E = A*2 :"+"\n",E)
print("Array F = A/2 :"+"\n",F)
print("Array G = A**2 :"+"\n",G)
print("Array H = A%2 :"+"\n",H)
```

```
print("Array I = A > 5 :"+"\n",I)
```

```
Array E = A*2 :
 [[ 2.  4.  6.]
 [ 8. 10. 12.]
 [14. 16. 18.]]
Array F = A/2 :
 [[0.5 1.  1.5]
 [2.  2.5 3. ]
 [3.5 4.  4.5]]
Array G = A**2 :
 [[ 1.  4.  9.]
 [16. 25. 36.]
 [49. 64. 81.]]
Array H = A%2 :
 [[1. 0. 1.]
 [0. 1. 0.]
 [1. 0. 1.]]
Array I = A > 5 :
 [[False False False]
 [False False  True]
 [ True  True  True]]
```

[162]:
```
#Operacions ELEMENT A ELEMENT:
#Si utilitzem dos arrays, les operacions anteriors s'apliquen per parells␣
 ↪d'elements

A = np.linspace(1,9,9).reshape((3,3))
B = np.linspace(10,18,9).reshape((3,3))

print("Array A: ")
print(A)
print("Array B: ")
print(B)


#Suma i resta de matrius:

C = A + B
print("Array C = A+B")
print(C) #(La resta seria igual)

#Multiplicació i divisió:

D = A/B

print("Array D = A/B: ")
```

```
print(D) #(La divisió seria igual)

#Potència i mòdul

E = A**B

print("Array E = A**B: ")
print(E) #(El mòdul seria igual)

#Booleans

F = A > B

print("Array F = A > B")
print(F)
```

```
Array A:
[[1. 2. 3.]
 [4. 5. 6.]
 [7. 8. 9.]]
Array B:
[[10. 11. 12.]
 [13. 14. 15.]
 [16. 17. 18.]]
Array C = A+B
[[11. 13. 15.]
 [17. 19. 21.]
 [23. 25. 27.]]
Array D = A/B:
[[0.1        0.18181818 0.25      ]
 [0.30769231 0.35714286 0.4       ]
 [0.4375     0.47058824 0.5       ]]
Array E = A**B:
[[1.00000000e+00 2.04800000e+03 5.31441000e+05]
 [6.71088640e+07 6.10351562e+09 4.70184985e+11]
 [3.32329306e+13 2.25179981e+15 1.50094635e+17]]
Array F = A > B
[[False False False]
 [False False False]
 [False False False]]
```

1.17 Operacions unitàries:

- array.min()
- array.max()
- array.sum()

1.18 Eines d'àlgebra lineal:

1.18.1 1. Transposició

Es pot transposar una matriu utilitzant el mètode `array.transpose()`

[16]:
```
#Transposar:
a = np.random.randint(1,5, size=(3,3))
print(a,"\n")

b = a.transpose()
print(b)
```

```
[[3 2 1]
 [4 2 2]
 [4 3 3]]

[[3 4 4]
 [2 2 3]
 [1 2 3]]
```

1.18.2 2. Determinants i matriu inversa

- Per calcular determinants ho podem fer amb la funció `np.linalg.det()`.
- Per calcular la matriu inversa ho podem fer amb la funció `np.linalg.inv()`.

[29]:
```
#Determinant

a = np.array([[1,1,2],[2,1,1],[1,1,1]])
det = np.linalg.det(a)
print("A: "+"\n",a,"\n")
print("Determinant d'A: ",det)

inv = np.linalg.inv(a)
print("Inversa d'A: "+"\n",inv)
```

```
A:
 [[1 1 2]
 [2 1 1]
 [1 1 1]]

Determinant d'A:  1.0
Inversa d'A:
 [[ 0.  1. -1.]
 [-1. -1.  3.]
 [ 1. -0. -1.]]
```

1.18.3 3. Rang i traça

- Rang: funció `np.linalg.matrix_rank(array)`

- Traça: funció `np.trace(array)`

[32]:
```
#Rang

a = np.array([[1,1,2],[2,1,1],[1,1,1]])
rang = np.linalg.matrix_rank(a)

print("Rang: ", rang)

#Traça

traça = np.trace(a)

print("Traça: ",traça)
```

```
Rang:  3
Traça:  3
```

1.18.4 4. Producte de matrius

Per multiplicar dues matrius, utilitzarem `np.dot(array1,array2)`

[40]:
```
a = np.random.randint(1,10, size = (3,3))
b = np.random.randint(1,10, size = (3,3))

producte = np.dot(a,b)
print("A: ")
print(a)
print("B: ")
print(b)
print("Producte: "+"\n",producte)
```

```
A:
[[1 7 7]
 [5 4 5]
 [6 1 9]]
B:
[[6 7 4]
 [7 3 4]
 [5 7 6]]
Producte:
 [[ 90  77  74]
 [ 83  82  66]
 [ 88 108  82]]
```

2 Exercicis d'exemple

2.0.1 Exercici 1

Efectua: 1. Donades les llistes x = [1,2,3] i y = [2,4,3], converteix-les en ndarrays i verifica, element a element, si x és menor que y, si x és més gran que y o si són diferents. 2. Donades les llistes n = [1,3,5] i m = [0,2,7], converteix-les en ndarrays i comprova, element a element, si n és més gran que m, si n és més gran o igual que m, si n és igual a m, si n és menor que m, o si n és menor o igual a m.

[18]:
```
#1
import numpy as np

x = [1,2,3]
y = [2,3,4]

array_x = np.array(x)
array_y = np.array(y)

verificar1 = array_x < array_y
verificar2 = array_x > array_y
verificar3 = array_x != array_y

print("x i y convertits en ndarrays: "+"\n"+"x =",array_x,"\n"+"y =",array_y)

print("Verificacions: "+"\n",verificar1,"\n", verificar2,"\n", verificar3)
```

```
x i y convertits en ndarrays:
x = [1 2 3]
y = [2 3 4]
Verificacions:
 [ True  True  True]
 [False False False]
 [ True  True  True]
```

[20]:
```
#2
n = [1,3,5]
m = [0,2,7]

array_n = np.array(n)
array_m = np.array(m)


verificar1 = array_n > array_m
verificar2 = array_n >= array_m
verificar3 = array_n == array_m
verificar4 = array_n < array_m
verificar5 = array_n <= array_m
```

```
print(verificar1)
print(verificar2)
print(verificar3)
print(verificar4)
print(verificar5)
```

```
[ True  True False]
[ True  True False]
[False False False]
[False False  True]
[False False  True]
```

2.0.2 Exercici 2

Donada la llista x= [1,2,3,4,5,6,7,8,9,4,6]: 1. Converteix-la en un ndarray 1D i calcula el producte dels seus elements. 2. Els valors màxim i mínim i els seus índexs. 3. El valor mitjà

[23]:
```
#1

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

producte = np.prod(array_x)
print(producte)
```

```
8709120
```

[31]:
```
#2
maxim = array_x.max(axis=0)
minim = array_x.min()
print("Màxim: ",maxim,"\n"+"Mínim: ",minim)

i_max = np.argmax(array_x)
i_min = np.argmin(array_x)

print("Màxim: ",i_max,"\n"+"Mínim: ",i_min)
```

```
Màxim:  9
Mínim:  1
Màxim:  8
Mínim:  0
```

2.0.3 Referències

1. 'Programació en Python', José M. Gómez, Ricardo Graciani, Manuel López, Xavier Luri, Sònia Estradé, Angela Rossell, Universitat de Barcelona (2016).
2. 'Syllabus INFO-H-100 Informatique', T. Massart, Université Libre de Bruxelles (2016).

3. http://www.numpy.org

[ ]: