Numpy

Βιβλιοθήκη Python για μαθηματικούς υπολογισμούς - Πράξεις

Στη Numpy η δομή `array` επιτρέπει τη συνηθισμένες πράξεις μεταξύ διανυσμάτων που έχουμε στη γραμμική άλγεβρα, το οποίο δεν ισχύει στις λίστες. Έτσι αν προσθέσουμε δύο `array` παίρνουμε το αναμενόμενο στα μαθηματικά αποτέλεσμα για τα διανυσμάτα.

In [1]:
```
import numpy as np
x = np.array([1,2,3])
y = np.array([4,5,6])
print(x+y)
```

```
[5 7 9]
```

Η ίδια πράξη για λίστες έχει διαφορετικό αποτέλεσμα.

In [2]:
```
a=[1,2,3]
b=[4,5,6]
print(a+b)
```

```
[1, 2, 3, 4, 5, 6]
```

Επίσης αν αφαιρέσουμε δύο `array` παίρνουμε το αναμενόμενο στα μαθηματικά αποτέλεσμα για τα διανυσμάτα. Ενώ για λίστες η πράξη αφαίρεση δεν ορίζεται.

In [3]:
```
print(x-y)
print(a-b)
```

```
[-3 -3 -3]

---------------------------------------------------------------------
---------
TypeError                                 Traceback (most recent c
all last)
<ipython-input-3-5695965f13eb> in <module>()
      1 print(x-y)
----> 2 print(a-b)

TypeError: unsupported operand type(s) for -: 'list' and 'list'
```

Η πράξη πολλαπλασισμός (`*`) στη γραμμική άλγεβρα δεν ορίζεται για διανύσματα, παρά μόνο για πίνακες. Στην numpy ο πολλαπλασιασμός, (`*`), καθώς και η διαίρεση, (`/`) και η ύψωση σε δύναμη, (`**`), ορίζονται για `array` , με τον ακόλουθο τρόπο. Κάθε μια από τις παραπάνω πράξεις εκτελείται σε δύο αντικείμενα `array` εφαρμόζοντας την πράξη μεταξύ κάθε δύο αντίστοιχων στοιχείων των αντικειμένων `array` . Στο παραπάνω παράδειγμα για τα x και y, έχουμε:

In [4]:
```
print(x*y)
```

```
[ 4 10 18]
```

Αντίστοιχα η διαίρεση και η ύψωση σε δύναμη μεταξύ των x και y γίνεται

In [5]:
```
print(x/y)
print(x**y)
```

```
[ 0.25  0.4   0.5 ]
[  1  32 729]
```

Με τον ίδιο τρόπο, δηλαδή εφαρμόζοντας την πράξη σε κάθε στοιχείο του `array` γίνεται και όλες οι πράξεις, `+,-, * ,/,**` , μεταξύ αριθμών και `array` .

In [6]:
```
print(2+x)
print(2-x)
print(2*x)
print(x/2)
print(x**2)
```

```
[3 4 5]
[ 1  0 -1]
[2 4 6]
[ 0.5  1.   1.5]
[1 4 9]
```

Εσωτερικό γινόμενο

Αν θελήσουμε να ορίσουμε το συνηθισμένο εσωτερικό γινόμενο μεταξύ διανυσμάτων τότε πρέπει να χρησιμοποιήσουμε την εντολή `dot` της Numpy.

In [7]:
```
x=np.array([1,2,3])
y=np.array([4,5,6])
a=np.dot(x,y) # 1ος τρόπος
print(a)
a=x.dot(y) # 2ος τρόπος
print(a)
```

```
32
32
```

Άσκηση 1:

Φτιάξτε μια συνάρτηση η οποία να δέχεται ως όρισμα 2 διανύσματα να ελέγχει αν αυτά είναι ορθογώνια και να τυπώνει ένα σχετικό μήνυμα.

Με τη χρήση της `dot` τρόπο ορίζεται και η πράξη πολλαπλασιασμός πίνακα με διάνυσμα και πίνακα με πίνακα.

In [8]:
```
A=np.array([[1,2],[3,4]])
x=np.array([1,0])
print(np.dot(A,x)) # Πολλαπλασιασμός πίνακα 2x2 με διάνυσμα δίνει διάνυσμα
x=np.array([[1],[0]])
print(np.dot(A,x)) # Πολλαπλασιασμός πίνακα 2x2 με πίνακα 2x1 δίνει πίνακα 2x1
```

```
[1 3]
[[1]
 [3]]
```

Φυσικά η πράξη πολλαπλασιασμός πρέπει να είναι επιτρεπτή και οι διαστάσεις των πινάκων πρέπει να είναι συμβατές.

In [9]:
```
x=np.eye(2)
y=np.array([[1,2],[3,4]])
print(np.dot(x,y)) # Πολλαπλασιασμός πινάκων 2x2
x=np.array([[1,2],[2,3],[3,4]])
y=np.array([1,2]) # Πολλαπλασιασμός πίνακα 3x2 με πίνακα 2x1
print(np.dot(x,y))
print(np.dot(y,x)) # Πολλαπλασιασμός πίνακα 2x1 με πίνακα 3x2 δεν μπορεί να γίνει
```

```
[[ 1.  2.]
 [ 3.  4.]]
[ 5  8 11]

---------------------------------------------------------------------
---------
ValueError                                Traceback (most recent c
all last)
<ipython-input-9-f026157b8619> in <module>()
      5 y=np.array([1,2]) # Πολλαπλασιασμός πίνακα 3x2 με πίνακα 2
x1
      6 print(np.dot(x,y))
----> 7 print(np.dot(y,x)) # Πολλαπλασιασμός πίνακα 2x1 με πίνακα
3x2 δεν μπορεί να γίνει

ValueError: shapes (2,) and (3,2) not aligned: 2 (dim 0) != 3 (dim
0)
```

Άσκηση 2:

Φτιάξτε μια συνάρτηση η οποία να δέχεται ως όρισμα 2 πίνακες A και B και να ελέγχει αν ο A ειναι ο αντίστροφος του B