Εφαρμογές Η/Υ

6ο Μάθημα

- Μητρώα με χρήση του NumPy

Ν. Λαγαρός, Θ. Στάμος, Χ. Φραγκουδάκης

Μητρώα με χρήση του NumPy

```
import numpy as np
a = np.array([0,1,2,3])        # Δημιουργία μητρώου
print(type(a))                 # Έλεγχος του τύπου του a
print(a.dtype)                 # Έλεγχος του τύπου των στοιχείων του μητρώου
print(a.itemsize)              # Bytes που καταλαμβάνει κάθε στοιχείο
print(a.shape, np.shape(a))    # Πλειάδα, μέγεθος μητρώου σε κάθε διάσταση
print(a.size, np.size(a))      # Ο αριθμός των στοιχείων του μητρώου
print(a.nbytes)                # Πόσα bytes καταλαμβάνει το μητρώο
print(a.ndim)                  # Ο αριθμός των διαστάσεων του μητρώου
```

```
<class 'numpy.ndarray'>
int64
8
(4,) (4,)
4 4
32
1
```

Μητρώα με χρήση του NumPy

```
import numpy as np
a = np.array([0,1,2,3])
print(a[0])                 # Δεικτοδότηση όπως στις λίστες
a[0] = 10
print(a)
a.fill(0)                   # Θέτει την ίδια τιμή σε όλες τις θέσεις του μητρώου
print(a)
a[:] = 1                    # Ίδιο με το fill αλλά πιο αργό
print(a)
print(a.dtype)              # Προσοχή στον τύπο των στοιχείων του μητρώου
a[0] = 10.6                 # Θα αποκοπεί το δεκαδικό μέρος
print(a)
a.fill(-4.8)                # Το ίδιο συμβαίνει και με το fill
print(a)
```

```
0
[10  1  2  3]
[0 0 0 0]
[1 1 1 1]
int64
[10  1  1  1]
[-4 -4 -4 -4]
```

Μητρώα με χρήση του NumPy

Slicing `var[lower:upper:step]`

```
import numpy as np
a = np.array([10,11,12,13,14])
print(a[1:3])
print(a[1:-2])
print(a[-4:3])
print(a[:3])    # Τα πρώτα 3 στοιχεία του μητρώου
print(a[-2:])   # Τα 2 τελευταία στοιχεία του μητρώου
print(a[::2])   # Τα στοιχεία στις ζυγές θέσεις του μητρώου
```

```
[11 12]
[11 12]
[11 12]
[10 11 12]
[13 14]
[10 12 14]
```

Μητρώα με χρήση του NumPy

```
import numpy as np
a = np.array([[0, 1, 2, 3],    # Δισδιάστατο μητρώο
              [10,11,12,13]])
print(a)
print(a.shape)                 # Πλειάδα (γραμμές, στήλες)
print(a.size)                  # Αριθμός των στοιχείων
print(a.ndim)                  # Αριθμός των διαστάσεων
print(a[1,3])
a[1,3] = -1                    # Δεικτοδότηση, a[γραμμή, στήλη]
print(a)
print(a[1])                    # Εκτύπωσε την 2η γραμμή
```

```
[[ 0  1  2  3]
 [10 11 12 13]]
(2, 4)
8
2
13
[[ 0  1  2  3]
 [10 11 12 -1]]
[10 11 12 -1]
```

Μητρώα με χρήση του NumPy

Έστω ένα αρχείο `numpy-data00.txt` με διάφορα δεδομένα:

```
-- Beggining of file
% Day, Month, Year, Skip, Avg Power
01, 01, 2000, x876, 13 % crazy day!
% we don't have Jan 03rd
04, 01, 2000, xfred, 55
```

```
import numpy as np
arr = np.loadtxt(                   # Παράγει ένα μητρώο arr από το αρχείο txt
    'numpy-data00.txt',             # Όνομα του αρχείου
    skiprows = 1,                   # Παρέλειψε την 1η γραμμή
    dtype = int,                    # Διάβασε ακεραίους
    delimiter = ',',                # Τα δεδομένα χωρίζονται με ,
    usecols = (0, 1, 2, 4),         # Διάβασε τις συγκεκριμένες στήλες
    comments = '%'                  # Χαρακτήρας που δηλώνει σχόλιο
)
print(arr)
np.savetxt('filename', arr)
```

```
[[   1    1 2000   13]
 [   4    1 2000   55]]
```

Μητρώα με χρήση του NumPy

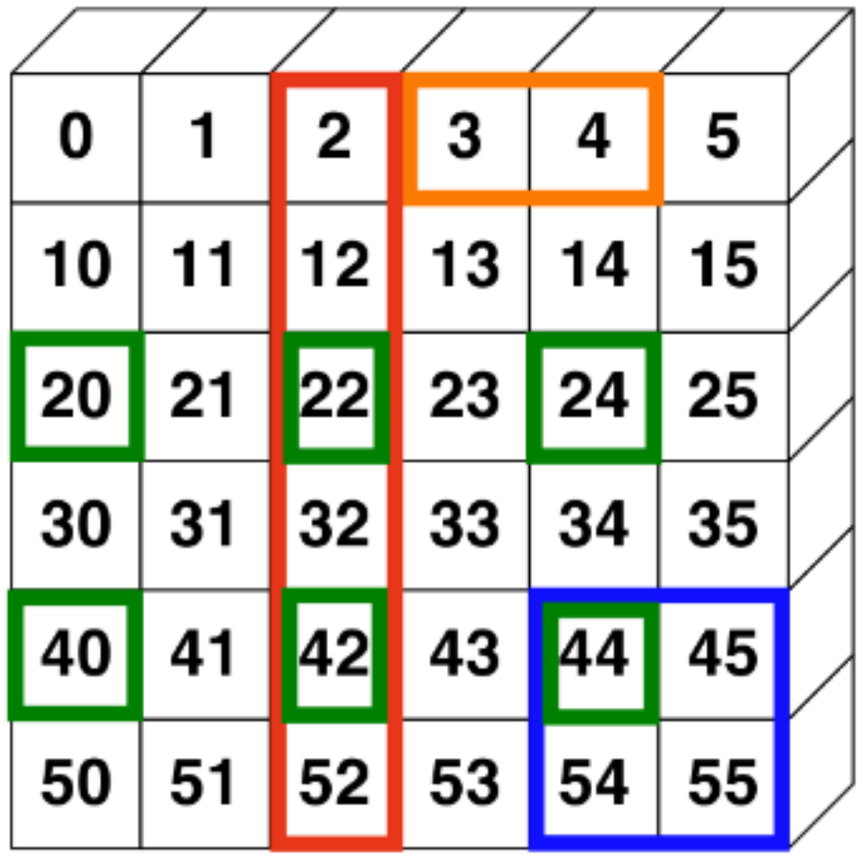
Slices σε πολλές διαστάσεις (όπως στις λίστες)

```
>>> a[0,3:5]
array([3, 4])

>>> a[4:,4:]
array([[44, 45],
       [54, 55]])

>>> a[:,2]
array([2,12,22,32,42,52])

>>> a[2::2,::2]
array([[20, 22, 24],
       [40, 42, 44]])
```



Μητρώα με χρήση του NumPy

Slices σε πολλές διαστάσεις (όπως στις λίστες)

```
import numpy as np
a = np.arange(36).reshape(6,6)
print(a)
print(a[0, 3:5])      # Από την 1η γραμμή το slice 3:5
print(a[4:, 4:])      # Το κομμάτι μετά την 4η γραμμή και 4η στήλη
print(a[:,2])         # Η 3η στήλη
print(a[2::2, ::2])   # Το slice από τη 2η γραμμή με βήμα 2 με όλες τις στήλες
                      # με βήμα 2
```

```
[[ 0  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 26 27 28 29]
 [30 31 32 33 34 35]]
[3 4]
[[28 29]
 [34 35]]
[ 2  8 14 20 26 32]
[[12 14 16]
 [24 26 28]]
```

Μητρώα με χρήση του NumPy

```
import numpy as np
a = np.array((0, 1, 2, 3, 4))
print(a)
b = a[2:4]  # Το slice b είναι αναφορά στις θέσεις μνήμης του a
print(b)
b[0] = 10   # Αλλάζοντας το b αλλάζουμε πρακτικά το a
print(a)
```

```
[0 1 2 3 4]
[2 3]
[ 0  1 10  3  4]
```

Μητρώα με χρήση του NumPy

Fancy Indexing

```
import numpy as np
a = np.arange(0, 80, 10)
print(a)
indices = [1, 2, -3]
print(a[indices])  # Εκτυπώνει τις θέσεις του μητρώου με δείκτες από το indices
mask = np.array([0,1,1,0,0,1,0,0], dtype=bool)
print(mask)
print(a[mask]) # Εκτυπώνει τις θέσεις του μητρώου που συντισοιχούν στο True
mask2 = a < 30
print(a[mask2])
```

```
[ 0 10 20 30 40 50 60 70]
[10 20 50]
[False  True  True False False  True False False]
[10 20 50]
[ 0 10 20]
```

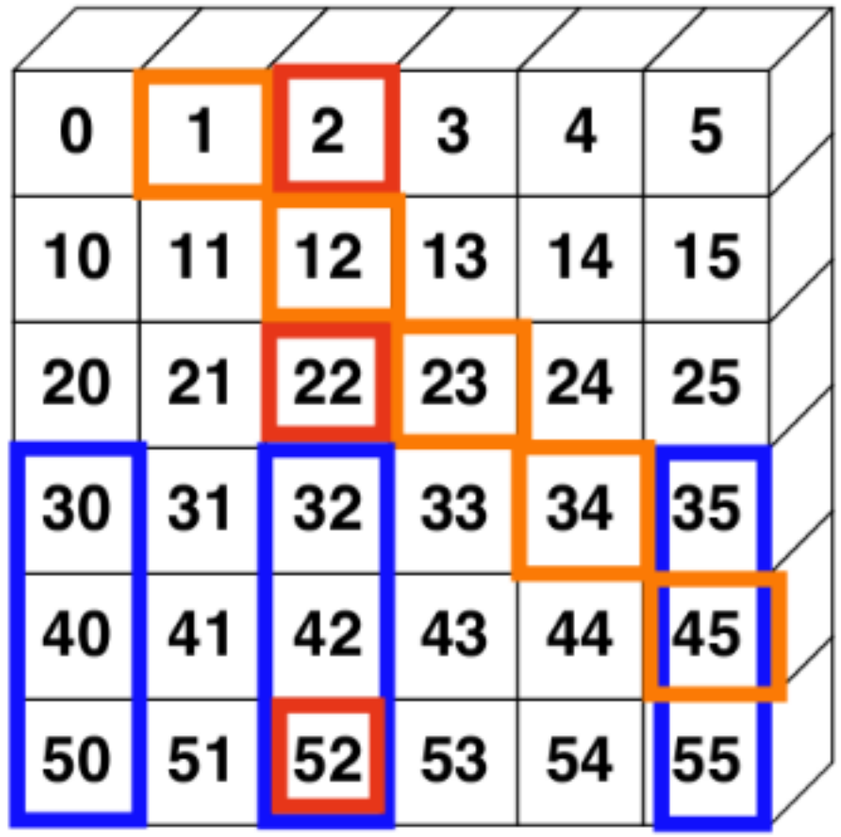
Μητρώα με χρήση του NumPy

Fancy Indexing

```
>>> a[(0,1,2,3,4),(1,2,3,4,5)]
array([ 1, 12, 23, 34, 45])

>>> a[3:,[0, 2, 5]]
array([[30, 32, 35],
       [40, 42, 45],
       [50, 52, 55]])

>>> mask = array([1,0,1,0,0,1],
                 dtype=bool)
>>> a[mask,2]
array([2,22,52])
```



Αντίθετα από το slicing το fancy indexing δημιουργεί αντίγραφα από τις θέσεις μνήμης του αρχικού μητρώου.

Μητρώα με χρήση του NumPy

Fancy Indexing

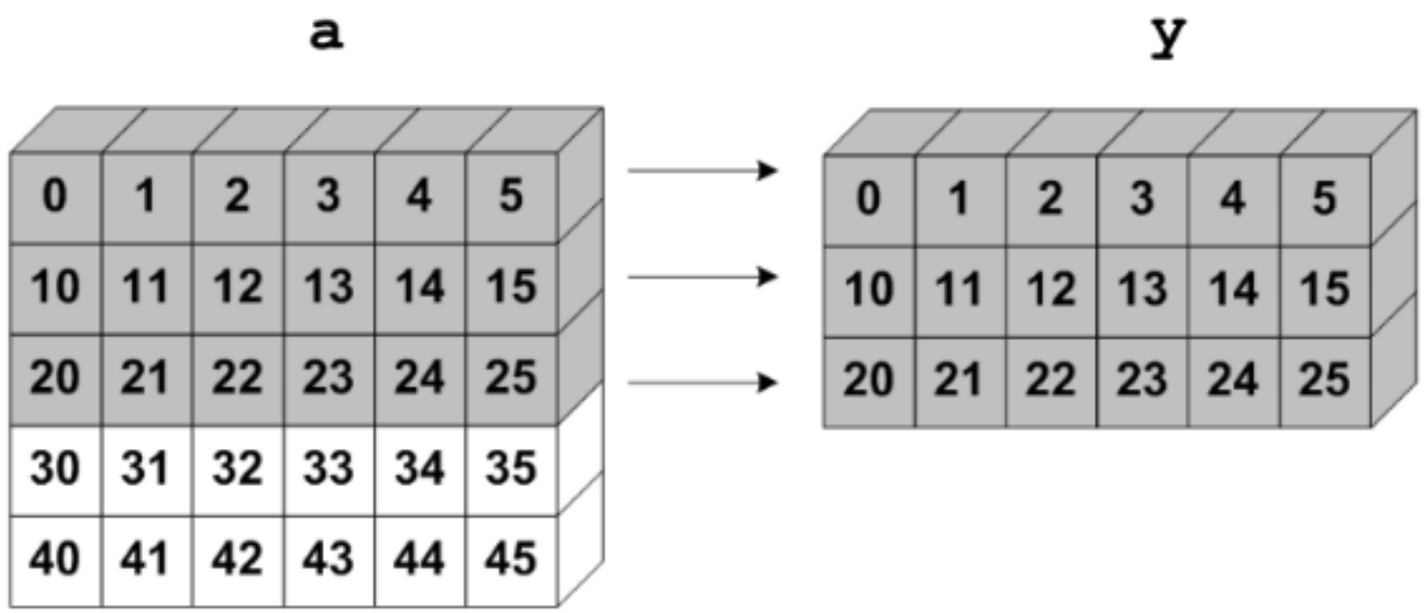
```
import numpy as np
a = np.arange(36).reshape(6,6) # Δημιουργία δισδιάστατου μητρώου 6x6
print(a)
print(a[(0, 1, 2, 3, 4), (1, 2, 3, 4, 5)]) # Δείκτες στην αντίστοιχη διάσταση
print(a[3:, [0, 2, 5]]) # Από την 3η γραμμή τις συγκεκριμένες στήλες
mask = np.array([1,0,1,0,0,1], dtype=bool)
print(a[mask, 2])       # Τις συγκεκριμένες γραμμές από την 3η στήλη
```

```
[[ 0  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 26 27 28 29]
 [30 31 32 33 34 35]]
[ 1  8 15 22 29]
[[18 20 23]
 [24 26 29]
 [30 32 35]]
[ 2 14 32]
```

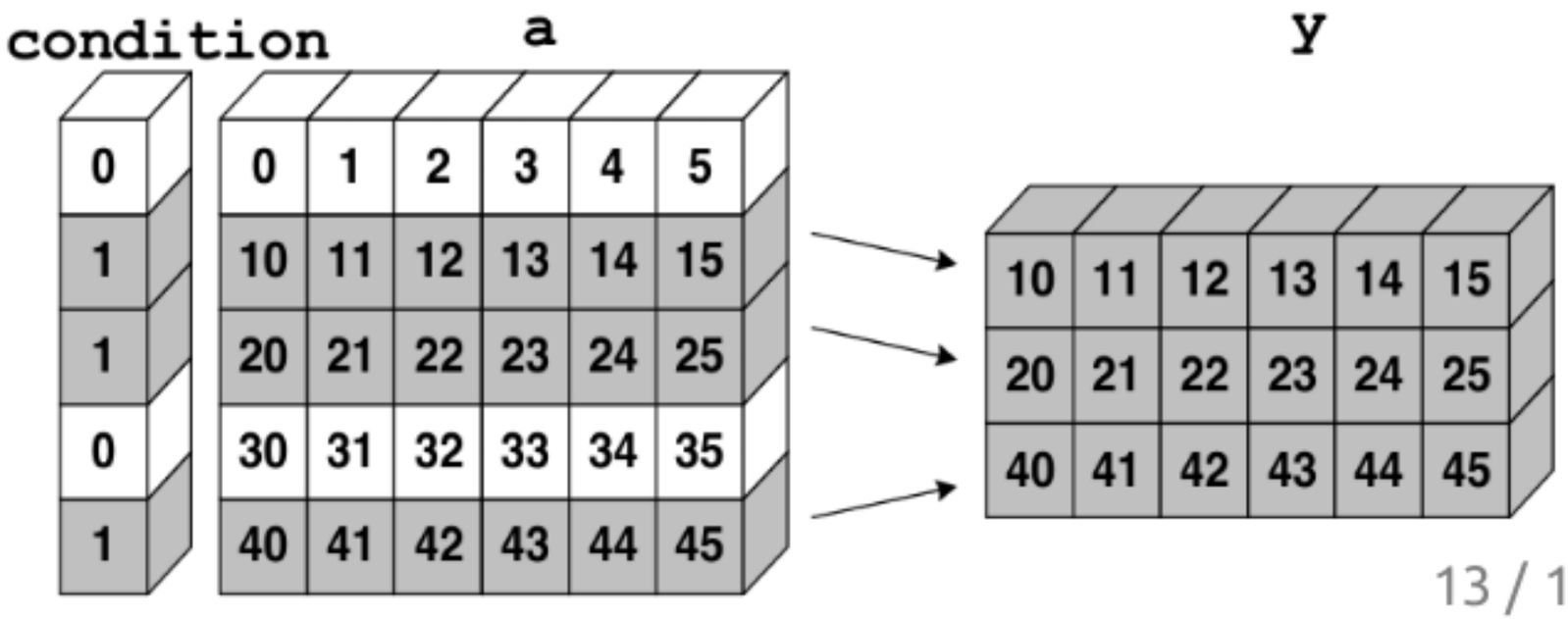
Μητρώα με χρήση του NumPy

Incomplete Indexing

```
>>> y = a[:3]
```



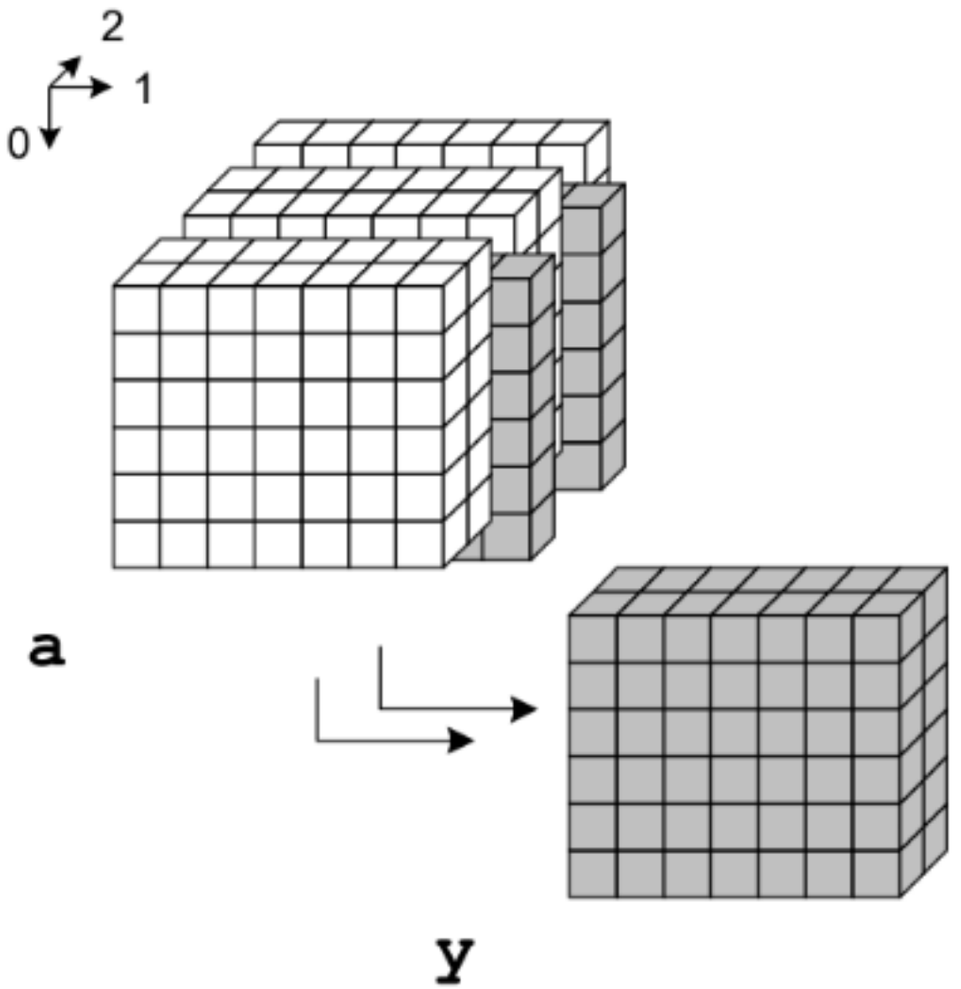

```
>>> y = a[condition]
```



Μητρώα με χρήση του NumPy

Incomplete Indexing σε 3 διαστάσεις

```
>>> y = a[:,:,[2,-2]]
```



Μητρώα με χρήση του NumPy

```
import numpy as np
a = np.array([0, 12, 5, 20])
print(a)
print(a > 10)           # Τύπωσε ένα μητρώο με True εκεί που αληθεύει η συνθήκη
print(np.where(a>10))  # Επιστρέφει μια πλειάδα μητρώων
a = np.array([[0, 12, 5, 20],
              [1, 2, 11, 15]])
print(a)
loc = np.where(a > 10) # Επιστρέφει μια πλειάδα μητρώων
print(loc)
print(a[loc])
```

```
[ 0 12  5 20]
[False  True False  True]
(array([1, 3]),)
[[ 0 12  5 20]
 [ 1  2 11 15]]
(array([0, 0, 1, 1]), array([1, 3, 2, 3]))
[12 20 11 15]
```

Μητρώα με χρήση του NumPy

```
import numpy as np
a = np.arange(6)
print(a)
print(a.shape)
a.shape = (2,3)    # Άλλαξε το σχήμα του μητώου σε 2x3 (ίδιες θέσεις μνήμης)
print(a)
b = a.reshape(3,2) # Επιστρέφει νέο μητρώο με σχήμα 3x2
print(a)
c = a.reshape(4,2) # Το reshape δεν μπορεί να αλλάξει το αριθμό των στοιχείων
```

```
[0 1 2 3 4 5]
(6,)
[[0 1 2]
 [3 4 5]]
[[0 1]
 [2 3]
 [4 5]]
Traceback (most recent call last):
  File "/home/christodoulos/python civil/numpy/numpy10.py", line 9, in <module>
    a.reshape(4,2)
ValueError: total size of new array must be unchanged
```

Μητρώα με χρήση του NumPy

```
import numpy as np
a = np.arange(6).reshape(2,3)
print(a)
print(a.transpose())  # Ανάστροφο μητρώο
print(a.T)            # Ενναλακτικός τρόπος δημιουργίας ανάστροφου μητρώου
b = a.T
b[0,1] = 30           # Αλλαγές στο ανάστροφο επηρεάζουν το αρχικό μητρώο
print(a)
```

```
[[0 1 2]
 [3 4 5]]
[[0 3]
 [1 4]
 [2 5]]
[[0 3]
 [1 4]
 [2 5]]
[[ 0  1  2]
 [30  4  5]]
```

Μητρώα με χρήση του NumPy

```
import numpy as np
a = np.array([[11,21,31],
              [12,22,32],
              [13,22,33]])
print(a)
print(a.diagonal())          # Εκτυπώνει τη διαγώνιο
print(a.diagonal(offset=1)) # offset>0 μετακινεί πάνω από την κύρια διαγωνιο
i = [0,1,2]
print(a[i, i])
a[i, i] = 2                  # Χρήση fancy indexing
i2 = np.array([0,1])
a[i2, i2+1] = 1              # Πάνω διαγώνιος
a[i2+1, i2] = -1             # Κάτω διαγώνιος
print(a)
```

```
[[11 21 31]
 [12 22 32]
 [13 22 33]]
[11 22 33]
[21 32]
[11 22 33]
[[ 2  1 31]
 [-1  2  1]
 [13 -1  2]]
```

Ερωτήσεις;