Tömbök (numpy)

Ebben a Notebookban bemutatjuk az tömböket illetve hogyan kell a pythonban adatfájlokat beolvasni.

In [ ]:

```
%pylab inline
```

Adattömbök, matematikai eszközök, adatfájlok kezelése

A pythonban a **numpy** csomag felel az adatok beolvasásáért, matematikai operációk elvégezésért (mint a gyökvonás, szinusz ...)

Tömbök

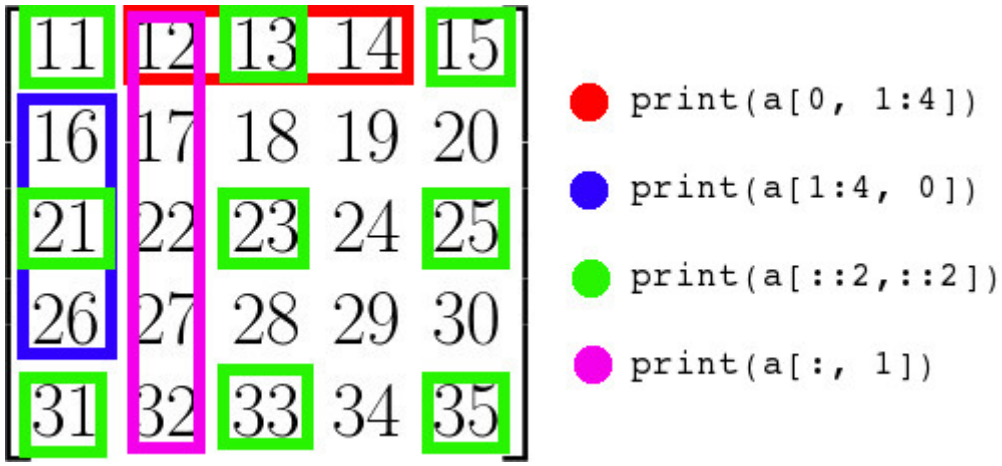
A programozásban létrehozhatunk mátrixokat, melyeket adattömböknek hívunk. Ezek bármennyi dimenziósak lehetnek.

In [ ]:

```
a = array([1, 2, 3])      # hozzunk létre egy 1 dimenziós tömböt
print(type(a))            # Írassuk ki "a" típusát: "<type 'numpy.ndarray'>"
print(a.shape)            # Nézzük meg a formáját: "(3,)"
```

A python és a legtöbb programozási nyelv nullától kezdi a tömbök indexelését. Azaz az első elem indexe 0, a másodiké 1 és így tovább. Egy 10 elemű tömbnek az utolsó eleme 9-es indexet visel (Elemszám-1)!

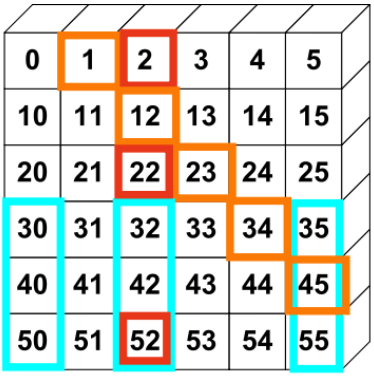
Így kell elképzelni az adattömböket:




```
>>> 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])
```



In [ ]:

```
print(a[0], a[1], a[2])   # Írassuk ki a tömb, első (0.-ik), második, ás harmadik el
emét
a[0] = 5                  # Átírjuk az első (nulladik) elem értékét
print(a)                  # Nézzük meg a megváltozott "a" vektort
```

In [ ]:

```
b = array([[1,2,3],[4,5,6]])    # Készítsünk 2 dimenziós tömböt
print(b.shape)                     # Kiíratjuk az alakját
print(b[0, 0], b[0, 1], b[1, 0])   # Írjunk ki pár elemet a tömbből
```

"Almátrixok"

A pythonban ki tudunk egy tömbből venni résztömböket. De vigyázzunk, mert a résztömbön végzett művelet a szülőtömb adott elemein is elvégződik.

In [ ]:

```
    # Készítsük els következő 2 dimenziós (3 sor, 4 oszlop) tömbböt
    # [[ 1  2  3  4]
    #  [ 5  6  7  8]
    #  [ 9 10 11 12]]
a = array([[1,2,3,4], [5,6,7,8], [9,10,11,12]])
print(a[0, 1])   # Írassuk ki a 0. sor, 1. oszlopának elem(eit): 2

    # Hozzuk létre a mátrix következő almátrixát (ami 2*2 nagyságú)
    # [[2 3]
    #  [6 7]]
    # Azaz az a mátrixból a 0-tól 2-ig levő adatokatsorok, illetve függőlegesen 1-tó
l 3-ig levő adatoszlopok
    # Vegyük észre, hogy a felső határ már nincs benne az adatsorban!
b = a[0:2, 1:3]

    # Változtassuk meg a b mátrix [0,0]-ás elemének értékét
b[0, 0] = 77

    # Ellenőrizzük az a mátrixban a "b[0,0]" helyen levő értéket (a[0,1]-as hely)
print(a[0, 1])
```

Számsorok generálása és tömbbe rendezése

Lehetőség van adott hosszú sorokat gyártani, melyekből tömb készíthető. Nézzük néhány példát erre:

In [ ]:

```
a = arange(15) # Készítsünk egy 15 elemű listát
print(a)
```

In [ ]:

```
a = arange(15)   # Készítsünk egy 15 elemű listát
a = a.reshape(3, 5) # Rendezzük a listát 3*5-s tömbbe
a                   # Nézzük meg az eredményt
```

In [ ]:

```
a.shape
```

In [ ]:

```
a.size # Megadja mekkora a tömb mérete (mennyi adat van benne)
```

In [ ]:

```
a.ndim
```

In [ ]:

```
type(a)
```

In [ ]:

```
a.dtype.name # Megnézi milyen tipusú adatok vannak a tömbben
```

In [ ]:

```
b = array([6.25, -0.25, 8.])
b
```

In [ ]:

```
type(b)
```

In [ ]:

```
b.dtype.name
```

In [ ]:

```
c = array( [ [1,2], [3,4] ], dtype=complex )
c
```

In [ ]:

```
c.dtype.name
```

In [ ]:

```
zeros( (3,4) ) #Készítsünk egy 3*4-es tömböt, mely tele van nullával
```

In [ ]:

```
ones( (2,3,4), dtype=int16 ) # Készítsünk egy 3 dimenziós tömböt, csupa egyesből
```

In [ ]:

```
a = arange(15, 100, 5) # Készítsünk egy listát 15-től, 100-ig, 5-sével
print(a)
```

In [ ]:

```
arange( 30, 10, -5 )
```

In [ ]:

```
arange( 10, 30, -5 ) #Ez üres lista, de NINCS hibaüzenet!!!
```

In [ ]:

```
arange( 0, 2, 0.3 )
```

In [ ]:

```
linspace( 0, 2, 9 ) # Készítsünk egy 9 elemű listát 0 és 2 között. Egyenlő lépésközzel.
```

Matematikai függvények

Bővebb leírás itt található: https://docs.scipy.org/doc/numpy/reference/routines.math.html (https://docs.scipy.org/doc/numpy/reference/routines.math.html)

In [ ]:

```
x = linspace( 0, 2*pi, 100 ) # A pi-nek a neve "pi"
x
```

In [ ]:

```
f = sin(x) # vegyük az előbb generált adatsornak a szinuszát
plot(f)   # majd ábárzoljuk
```

In [ ]:

```
plot(cos(x))
```

In [ ]:

```
plot(tan(x))
```

In [ ]:

```
plot(tanh(x))
```

In [ ]:

```
plot(arctan(x))
```

In [ ]:

```
B = arange(3)
B
```

In [ ]:

```
exp(B) # Emeljük "e"-adra a B elemeit
```

In [ ]:

```
sqrt(B) # Gyökvonás neve 'sqrt'
```

In [ ]:

```
C = array([2., -1., 4.])
add(B, C) # Össze is adahatunk elemenként 2 tömbböt
```

Kétváltozós függvények ábrázolása (felületi ábrák)

Ha kétváltozós függvényt szeretnénk ábrázolni, akkor ahhoz a mintavételezést a numpy csomag meshgrid() függvényével tehetjük meg az alábbi szintaxis szerint:

In [ ]:

```
xrange=linspace(-3,3,100)   # határok és pontok száma az x irányba
yrange=linspace(-3,3,100)   # határok és pontok száma az y irányba
x,y=meshgrid(xrange,yrange) # mintavételezés az x és y síkban
```