In [1]:
```
# COMPUTACIÓN BLANDA - Sistemas y Computación

#
----------------------------------------
# Introducción a numpy
#
----------------------------------------
# Lección 03
#
# ** FILE I/O
# ** División de arrays
# ** Propiedades de arrays
#
# ----------------------------------------
```

In [2]:
```
# Se importa la librería numpy
import numpy as np

# Grabar una matriz

# Matriz identidad
i2 = np.eye(2)
print(i2)

# Se almacena el array en un arcnivo de
texto np.savetxt("eye.txt", i2)
```

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

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

In [3]:
```
# Archivos CSV. Archivos separados por comas.

# Estructura de los datos

# El siguiente ejemplo hace referencia a datos históricos de la compañía Apple. # Los datos están en el formato CSV. La primera columna contiene un símbolo que # identifica el inventario. En nuestro caso, es AAPL.
# El segundo es la fecha en el formato dd-mm-yyyy. La tercera columna está vací a.
# A continuación tenemos los datos: precio de apertura, precio alto, precio baj o,
# precio de cierre. El último dato, y no menos importante, es el volumen del dí a.
# ESte es un ejemplo de la lína:

# AAPL,28-01-2011, ,344.17,344.4,333.53,336.1,21144800

# Por ahora, estamos interesados solamenete en el precio de cierre y en el volu men (c, v)
# En el ejemplo precedente: 336.1 y 21144800.
# Para extraer estos dos datos se tiene:

c,v=np.loadtxt('data.csv', delimiter=',', usecols=(6,7), unpack=True)

print(c, '\n')
print(v)
```

```
[336.1  339.32  345.03  344.32  343.44  346.5  351.88  355.2  358.16  354.54
```

```
356.85  359.18  359.9  363.13  358.3  350.56  338.61  342.62  342.88  348.16
353.21 349.31 352.12 359.56 360. 355.36 355.76 352.47 346.67 351.99]

[21144800. 13473000. 15236800. 9242600. 14064100. 11494200. 17322100.
13608500. 17240800. 33162400. 13127500. 11086200. 10149000. 17184100.
18949000. 29144500. 31162200. 23994700. 17853500. 13572000. 14395400.
16290300. 21521000. 17885200. 16188000. 19504300. 12718000. 16192700.
18138800. 16824200.]
```

In [4]:
```
# El precio promedio ponderado por volumen (VWAP) es una cantidad muy important
e en finanzas.
# Representa un precio "medio" de un activo financiero. Cuanto mayor sea el
vol umen, más significativo suele ser
# un movimiento de precio. VWAP se utiliza a menudo en el comercio
 algorítmico y
# se calcula utilizando valores de volumen como pesos.

# Las siguientes son las acciones que tomaremos:
# 1. Leer los datos en matrices.
# 2. Calcular VWAP.

c,v=np.loadtxt('data.csv', delimiter=',',
usecols=(6,7),unpack=True) vwap = np.average(c, weights=v)
print("VWAP =", vwap)
```

```
VWAP = 350.5895493532009
```

In [5]:
```
# En finanzas, TWAP es otra medida de precio "promedio". Ahora que estamos en e
so, calculemos
```

```
        # también el precio medio ponderado en el tiempo. En realidad, es solo una
        vari ación de un tema.
        # La idea es que Las cotizaciones de precios recientes son más importantes,
        por lo que deberíamos
        # dar a los precios recientes ponderaciones más altas. los La forma más
        sencill a es crear una matriz
        # con la función arange de aumentar los valores de cero a el número de
        elemento s en la matriz de
        # precios de cierre. El siguiente es el código TWAP:

        t = np.arange(len(c))
        print("twap =", np.average(c, weights=t))

        twap = 352.4283218390804

In [7]: # Vectores h (alto) y l (bajo). Se obtienen de las columnas 4 y 5 de la matriz.

        h,l=np.loadtxt('data.csv', delimiter=',', usecols=(4,5),
        unpack=True) print(h)
        print(l)

        [344.4 340.04 345.65 345.25 344.24 346.7 353.25 355.52 359. 360.
        357.8 359.48 359.97 364.9 360.27 359.5 345.4 344.64 345.15 348.43
        355.05 355.72 354.35 359.79 360.29 361.67 357.4 354.76 349.77 352.32]
        [333.53 334.3 340.98 343.55 338.55 343.51 347.64 352.15 354.87 348.
        353.54 356.71 357.55 360.5 356.52 349.52 337.72 338.61 338.37 344.8
        351.12 347.68 348.4 355.92 357.75 351.31 352.25 350.6 344.9 345. ]
In [9]: # Cálculo de los valores máximo y mínimo
```

```
print("El más alto =", np.max(h))
print("El más bajo =", np.min(l))
```

```
highest = 364.9
lowest = 333.53
```

In [10]:
```
# NumPy nos permite calcular la extensión de una matriz con una función llamada
ptp.
# La función ptp devuelve la diferencia entre los valores máximo y mínimo de #
una matriz. En otras palabras, es igual a max (matriz) - min (matriz). Llaman
do a la función ptp.

print("Extensión de precio más alto: ", np.ptp(h))
print("Extensión de precio más bajo: ", np.ptp(l))
```

```
Extensión de precio más alto: 24.859999999999957
Extensión de precio más bajo: 26.970000000000027
```