S02 T04: Práctica amb programación numérica

- Ejercicio 1

Crea una función que dado un Array de una dimensión, te haga un resumen estadístico básico de los datos. Si detecta que el array tiene más de una dimensión, debe mostrar un mensaje de error.

In [4]:
```
import numpy as np
from numpy import random

arr =random.randint(50, size=(200)) # creamos una array de una dimensión hecha de manera aletaroria para hacer
# luego creamos la función que nos hará los cálculos estadísitcos.

arr2 = np. array ([[1,0], [0,1]]) # sea una segunda matriz de prueba para comprobar la función

def estadistica ( x ):  # donde x es la array a estudiar
    if x.ndim > 1 :
        print ( " La matriz es de dimensión dos o más, no apta para hacer los cálculos  ")
    else :
        print ( " la media de valores de la matriz es :", np.mean(x))
        print ("\n")
        print ( " la mediana es :", np.median(x))
        print ("\n")
        print ( " el coeficiente de correlación :", np.corrcoef(x))
        print ("\n")
        print ( " la desviación estandar es :", np.std(x))
        print ("\n")
estadistica ( arr)
print ("\n")
estadistica ( arr2)
```

```
la media de valores de la matriz es : 27.0

la mediana es : 30.0

el coeficiente de correlación : 1.0

la desviación estandar es : 15.16212386178137



La matriz es de dimensión dos o más, no apta para hacer los cálculos
```

- Ejercicio 2

Crea una función que genere un cuadrado NxN de números aleatorios entre el 0 y el 100.

In [9]:
```
# tomamos el valor de entrada con un input
n = int( input ( " introduce un valor entero natural "))

NxN = np.zeros((n,n))

def matriz ( n, arr ):

    arr = np.zeros((n,n))

    for x in range(0,n):
        for y in range (0,n) :
            arr [x][y] = random.randint (0,100)
    return arr
NxN = matriz ( n , NxN )
print ( NxN)
```

```
introduce un valor entero natural 10
[[65. 24. 89.  5. 45. 60. 93. 35. 57. 10.]
 [43. 20. 31. 11.  1.  6. 70. 19.  9. 15.]
 [21. 83. 40. 15. 67.  1. 82. 56. 16.  8.]
 [39. 40. 24. 31. 30. 51.  8. 12. 33. 33.]
 [48. 58. 81. 12. 61. 93. 55. 62. 84. 63.]
 [ 9. 62. 47. 61. 88. 59. 21. 39. 40. 58.]
 [ 6. 91. 58. 35. 66. 89. 45. 49.  6. 68.]
 [24. 67. 21.  3. 29. 86. 40.  9. 81. 32.]
 [61. 51. 67. 56.  1. 32.  8. 73. 10. 41.]
 [21. 92. 35. 42. 24. 65. 90.  9. 92. 90.]]
```

- Ejercicio 3

Crea una función que dada una tabla de dos dimensiones,

te calcule los totales por fila y los totales por columna.

In [28]:
```
# tomamos el valor de entrada con un input para crear una matriz aleatoria
import numpy as np
k = int( input ( " introduce un valor entero natural "))
l = int( input ( " introduce otro  valor entero natural "))
tabla= np.zeros((k,l), dtype = "int64" )
for x in range(0,k):
    for y in range (0,l) :
        tabla [x][y] = int (random.randint (0,100)* ((-1)**( random.randint (1,3))))
print (tabla )


def suma ( tabla ):
    print ( " la suma a través de las columnas es : " , np. sum ( tabla , axis=0 ))
    print ( " la suma a través de las columnas es : " , np. sum ( tabla , axis=1 ))



print (suma ( tabla ))
```

```
 introduce un valor entero natural 4
 introduce otro  valor entero natural 3
[[-88 -46 -63]
 [ 83  47   2]
 [-85  80 -60]
 [ 24 -26 -46]]
 la suma a través de las columnas es :  [ -66   55 -167]
 la suma a través de las columnas es :  [-197  132  -65  -48]
None
```

- Ejercicio 4

Implementa manualmente una función que calcule el coeficiente de correlación.

Infórmate sobre sus usos e interpretación.

El coeficiente de correlación mide la ***relación*** entre dos variables.

Su valor es entre -1 y 1. Cuanto más cerca se encuentre el valor a 1 o -1

querrá decir que las variables son más dependientes una de otras, y cuando el valor

sea cero o cercano a cero, querrá decir que las variables son independientes entre sí

El hecho de que sea positivo o negativo nos indica si a la dependencia entre las

dos variables es proporcional o inversamente proporciona. Es decir, sean dos

variables x, y; si el coeficiente es positivo, cuando x crezca, y también lo hará.

Sin embargo, cuando sean inversamente proporcionales, cuando una de las variables crezca, la

decrecerá.

El coeficiente de correlación también nos sirve para para medir la bondad de un modelo

experimental o de una hipótesis, en un modelo de relación lineal, ya que entonces el cuadrado

del coeficiente de correlación coincidirá con el cuadrado del coeficiente de determinación

de la regresión lineal

Pongamos el caso que quisiera comprobarse algo tan evidente, como que existe una relación

directa entre el volumen ocupado por un líquido en una botella y el nivel del líquido en

esa botella. Un experimento tan simple como ir añadiendo cantidades de volumen de agua en

una botella y medir la altura (considerando la botella como un cilindro)

Un coeficiente cercano a 1 o a -1 nos indicaría que la hipótesis es correcta

In [47]:
```
vol = np. array ([0, 250, 500, 750, 1000, 1250, 1500, 17000, 2000, 2250 ], dtype= "int") # volumen en ml
alt = np. array ([0, 3.1, 6.2, 9.1, 12.3, 15.4, 18.0, 21.2, 24.5, 27.5 ])# altura en centímetros
# Estos resultados son de un experimento hecho en una clase de un instituto



r = np.corrcoef((vol, alt) )
print ( " el coeficiente de correlación es : ", r )
```

```
el coeficiente de correlación es :  [[1.         0.41825355]
 [0.41825355 1.        ]]
```

Este resultado nos indica que la relación entre las variables es un coeficiente de correlación de 0.42, baja, indica una una gran varianza o dispersión de los resultados respecto al ajuste lineal

In [ ]:
```

```