In [44]:

```python
#Crea una funció que donat un Array d’una dimensió, et faci un resum estadístic bàsic de les dades. Si detecta que
#l’array té més d’una dimensió, ha de mostrar un missatge d’error.

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

def listaAleatorios(n):
    import random
    lista = [0]  * n
    for i in range(n):
          lista[i] = random.randint(0, 10)
    return lista

print("Ingrese cuantos numeros aleatorios desea obtener")
n=int(input())

aleatorios=listaAleatorios(n)
print(aleatorios)
```

```
Ingrese cuantos numeros aleatorios desea obtener
10
[10, 4, 5, 9, 10, 0, 4, 2, 10, 2]
```

In [3]:

```python
def func():
    arr = np.array(aleatorios)
    print(arr)

func()

arr = np.array(aleatorios)
print(arr)
```

```
[6 0 9 9 7 3 2 3 5 9]
[6 0 9 9 7 3 2 3 5 9]
```

In [4]:

```python
if arr.ndim>1:
    print("No pot tenir més d'una dimensió")
```

In [5]:

```python
dimArray= (arr.ndim)

print("El teu array té "+ str(dimArray)+ " dimensió")
```

```
El teu array té 1 dimensió
```

In [6]:

```python
#Crides de diferents maneres de l'array analitzant la informació que ens podria interessar.
print(arr[2] + arr[3])
print(arr[1:5])
print(arr[4:])
print(arr[-3:-1])
print(arr[1:5:2])

print(arr.dtype)
```

```
18
[0 9 9 7]
[7 3 2 3 5 9]
[3 5]
[0 9]
int32
```

In [7]:

```python
#Copy, no canvia l'original
x = arr.copy()
arr[0] = 42

print(arr)
print(x)
```

```
[42  0  9  9  7  3  2  3  5  9]
[6 0 9 9 7 3 2 3 5 9]
```

In [8]:

```python
#view, canvia l'original
y = arr.view()
arr[1] = 33

print(arr)
print(x)


print(x.base)
print(y.base)
```

```
[42 33  9  9  7  3  2  3  5  9]
[6 0 9 9 7 3 2 3 5 9]
None
[42 33  9  9  7  3  2  3  5  9]
```

In [9]:

```python
aa = np.mean(arr)
print(aa)

bb=np.median(arr)
print(bb)

cc = np.std(arr)
print(cc)

dd= np.min(arr)
print(dd)

ee = np.max(arr)
print(ee)

ff= np.count_nonzero(arr)
print(ff)

gg= np.abs(arr)
print(gg)

hh= np.sin(arr)
print(hh)

ii= np.cos(arr)
print(ii)

jj= np.tan(arr)
print(jj)

kk = np.sin(np.pi/2)
print(kk)

ll= np.deg2rad(arr)
print(ll)

nn = np.hypot(arr[0], arr[1])
print(nn)

oo = np.cosh(arr)
print(oo)

pp  = np.unique(arr)
print(pp)

qq= np.lcm(arr[4], arr[3])
print(qq)

rr = np.gcd(arr[0], arr[6])
print(rr)

newarr = np.diff(arr)

print(newarr)
```

```
12.2
8.0
13.052202879207785
2
42
10
[42 33  9  9  7  3  2  3  5  9]
[-0.91652155  0.99991186  0.41211849  0.41211849  0.6569866   0.14112001
  0.90929743  0.14112001 -0.95892427  0.41211849]
[-0.39998531 -0.01327675 -0.91113026 -0.91113026  0.75390225 -0.9899925
 -0.41614684 -0.9899925   0.28366219 -0.91113026]
[  2.29138799 -75.3130148   -0.45231566  -0.45231566   0.87144798
  -0.14254654  -2.18503986  -0.14254654  -3.38051501  -0.45231566]
1.0
[0.73303829 0.57595865 0.15707963 0.15707963 0.12217305 0.05235988
 0.03490659 0.05235988 0.08726646 0.15707963]
53.41348144429457
[8.69637471e+17 1.07321790e+14 4.05154203e+03 4.05154203e+03
 5.48317035e+02 1.00676620e+01 3.76219569e+00 1.00676620e+01
 7.42099485e+01 4.05154203e+03]
[ 2  3  5  7  9 33 42]
63
2
[ -9 -24   0  -2  -4  -1   1   2   4]
```

In [10]:

```python
for x in np.nditer(arr, flags=['buffered'], op_dtypes=['S']):
    print(x)
```

```
b'42'
b'33'
b'9'
b'9'
b'7'
b'3'
b'2'
b'3'
b'5'
b'9'
```

In [17]:

```python
for idx, x in np.ndenumerate(arr):
    print(idx, x)
```

```
(0,) 42
(1,) 33
(2,) 10
(3,) 1
(4,) 9
(5,) 8
(6,) 1
(7,) 6
(8,) 8
(9,) 7
```

In [11]:

```python
for x in np.nditer(arr):
    print(x)
```

```
42
33
9
9
7
3
2
3
5
9
```

In [12]:

```python
print(np.sort(arr))

a = np.where(arr == 4)
b = np.where(arr%2 == 0)
c = np.where(arr%2 == 1)
d = np.searchsorted(arr, 7)
e = np.searchsorted(arr, 1, side='right')
f= x = np.searchsorted(arr, [3, 5, 8])
g= np.searchsorted(arr, 10)


print(a)
print(b)
print(c)
print(d)
print(e)
print(f)
print(g)
```

```
[ 2  3  3  5  7  9  9  9 33 42]
(array([], dtype=int64),)
(array([0, 6], dtype=int64),)
(array([1, 2, 3, 4, 5, 7, 8, 9], dtype=int64),)
9
0
[0 8 9]
10
```

In [13]:

```python
np.sort(arr)
h = [True, False, True, False, False, True, True, False, True,False]

newArr = arr[h]

print(newArr)
```

```
[42  9  3  2  5]
```

In [14]:

```python
newList= []

for i in arr:
    if i > 5:
        newList.append(True)
    else:
        newList.append(False)

newArray = arr[newList]
print(newArray)
print(newList)
```

```
[42 33  9  9  7  9]
[True, True, True, True, True, False, False, False, False, True]
```

In [15]:

```python
filtering = []

for x in arr:
    if x % 2 == 0:
        filtering.append(True)
    else:
        filtering.append(False)

newFilter = arr[filtering]
print(newFilter)
print(filtering)
```

```
[42  2]
[True, False, False, False, False, False, True, False, False, False]
```

In [16]:

```python
#Crea un quadrat de XperX amb nums aleatoris de 0 a 100

from numpy import random
import pandas as pd

quadrat = random.randint(100, size=(5, 5))

print(quadrat)
df = pd.DataFrame(quadrat)
print(df)
```

```
[[92 38 37 33 24]
 [66 82  3 95 12]
 [31 38 42 39 10]
 [42 24  1 53 78]
 [32 14 97 59 53]]
    0   1   2   3   4
0  92  38  37  33  24
1  66  82   3  95  12
2  31  38  42  39  10
3  42  24   1  53  78
4  32  14  97  59  53
```

In [17]:

```python
#Crea una funció que donada una taula de dues dimensions, et calculi els totals per fila i els totals per columna.

from numpy import random

nouArr = random.randint(100, size=(2, 5))

print(nouArr)
```

```
[[19 74 51 51 65]
 [ 6 22 46 75 65]]
```

In [18]:

```python
dff= pd.DataFrame((nouArr),
index= ['x', 'y'])
print(dff)

#Columnes
Total = dff[0].sum()
print ("Column 1 sum:",Total)

total2 = dff[1].sum()
print("Colum 2 sum:", total2)

total3 = dff[2].sum()
print("Colum 3 sum:", total3)

total4 = dff[3].sum()
print("Colum 4 sum:", total4)

total5= dff[4].sum()
print("Colum 5 sum:", total5)

#Files

totalFila = np.sum(nouArr, axis=1)
print(totalFila)
```

```
    0   1   2   3   4
x  19  74  51  51  65
y   6  22  46  75  65
Column 1 sum: 25
Colum 2 sum: 96
Colum 3 sum: 97
Colum 4 sum: 126
Colum 5 sum: 130
[260 214]
```

In [48]:

```python
#Implementa manualment una funció que calculi el coeficient de correlació.
#Informa’t-en sobre els seus usos i interpretació.

nouArr1 = nouArr.transpose()
nouArr1 = pd.DataFrame(nouArr1)
print(nouArr1)
nouArr1.columns = ['x', 'y']
print(nouArr1)

def corr_coeff(dataFrame, var1, var2):
    dataFrame["corrn"] = (dataFrame[var1] - np.mean(dataFrame[var1])) * (dataFrame[var2] - np.mean(dataFrame[var2]))
    dataFrame["corr1"] = (dataFrame[var1] - np.mean(dataFrame[var1])) ** 2
    dataFrame["corr2"] = (dataFrame[var2] - np.mean(dataFrame[var2])) ** 2
    coeficienteCorrelacionPearson = sum(dataFrame["corrn"]) / np.sqrt(sum(dataFrame["corr1"]) * sum(dataFrame["corr2"]))
    return coeficienteCorrelacionPearson

corr_coeff(nouArr1, "x", "y")

cols = nouArr1.columns.values
for x in cols:
    for y in cols:
        print(x + ", " + y + " : " + str(corr_coeff(nouArr1, x, y)))



plt.plot(nouArr1["x"], nouArr1["y"], "ro")
plt.title("Correlació entre x i y")

nouArr1.corr()

plt.matshow(nouArr1.corr())

#El coeficiente de correlación de Pearson mide la asociación lineal entre variables. Su valor se puede interpretar así:

#+1 – Correlación positiva completa
#+0,8 – Fuerte correlación positiva
#+0,6 – Correlación positiva moderada
#0 – sin correlación alguna
#-0,6 – Correlación negativa moderada
#-0,8 – Fuerte correlación negativa
#-1 – Correlación negativa completa



#L'entrada per aquesta funció sol ser una matriu d'un tamany mxn on:
#Cada columna representa els valors d'una variable aleatoria
#Cada fila representa una sola mostra de n variables aleatories
#n representa el número total de diferents variables aleatories
#m representa el número total de mostres per cada variable

#Totes les entrades diagonals són 1 perquè perque el coeficient de correlació d'una variable amb ella mateixa sempre
#donarà aquest valor. Les diagonals solen ser iguals entre elles.

#Primer tenim una correlació completament positiva entre dues variables (+1)
#Després l'altre coeficient de correlació hauria de ser proper a 0.

#El primer rand.uniform() La trucada genera una distribucio uniforme aleatoria.
# I vstack() apila verticalment altres matrius en ell. Així podrem accedir seqüencialment.

#el segon te una relació positiva completa amb el primer, el tercer té una correlació negativa
#completa amb el primer i el quart és completament aleatori. Hauria de tenir una correlació de ~ 0.
```

```
    0   1
0  19   6
1  74  22
2  51  46
3  51  75
4  65  65
    x   y
0  19   6
1  74  22
2  51  46
3  51  75
4  65  65
x, x : 1.0
x, y : 0.4193095821245791
x, corrn : -0.03703409175535179
x, corr1 : -0.6032181834082021
x, corr2 : -0.9624622936598533
y, x : 0.4193095821245791
y, y : 1.0
y, corrn : -0.008949488920798441
y, corr1 : -0.23653672850418583
y, corr2 : -0.49657622250809147
corrn, x : 0.0363930062914001
corrn, y : -0.04754657923169269
corrn, corrn : 1.6263676734041405e-19
corrn, corr1 : 0.0
corrn, corr2 : 0.0
corr1, x : nan
corr1, y : nan
corr1, corrn : nan
corr1, corr1 : nan
corr1, corr2 : nan
corr2, x : nan
corr2, y : -0.8766648948633549
corr2, corrn : -0.0019812140398738423
corr2, corr1 : 1.2420026740645948e-22
corr2, corr2 : 1.0
```

Out[48]: <matplotlib.image.AxesImage at 0x1e31a11b0a0>