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.

In [4]:
```python
import numpy as np

def calcular(array):
    if array.ndim != 1:
        print("Error: El array debe de ser de una dimension")
        return None

    # calculos basicos estadisticos
    resum = {
        'màximo': np.max(array),
        'mínimo': np.min(array),
        'promedio': np.mean(array),
        'mediana': np.median(array),
        'desviacion estandar': np.std(array)
    }
    return resum

array = np.array([1, 2, 3, 4, 5])
resum = calcular(array)
print(resum)
```

```
{'màximo': 5, 'mínimo': 1, 'promedio': 3.0, 'mediana': 3.0, 'desviacion estandar': 1.414
2135623730951}
```

Crea una funció que et generi un quadrat NxN de nombres aleatoris entre el 0 i el 100.

In [5]:
```python
import numpy as np

def generar(N):
    cuadrado = np.random.randint(0, 100, size=(N, N))
    return cuadrado

cuadrado_aleatorio = generar(5)
print(cuadrado_aleatorio)
```

```
[[11 57 42 62 97]
 [17 53 43 75 58]
 [91 50 32 82  1]
 [ 4 66 51 98 15]
 [63 46 46 86 63]]
```

Crea una función que dada una tabla de dos dimensiones (NxM), te calcule los totales por fila y los totales por columna.

In [7]:
```python
import numpy as np

def calcular(matriz):
    total_fila = np.sum(matriz, axis=1)
    total_columna = np.sum(matriz, axis=0)

    return {'total_fila': total_fila, 'total_columna': total_columna}

matriz = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
totales = calcular(matriz)
print(totales)
```

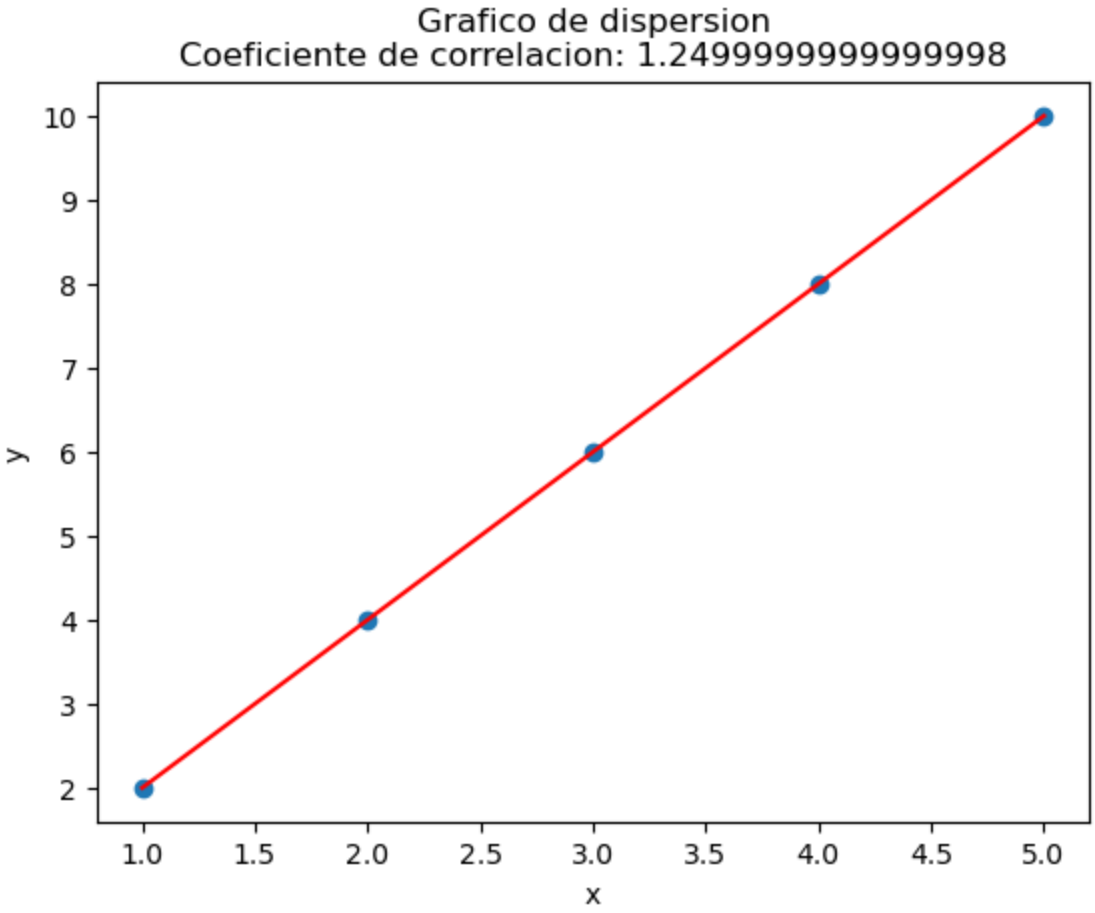
```
{'total_fila': array([ 6, 15, 24]), 'total_columna': array([12, 15, 18])}
```

Implementa manualmente una función que calcule el coeficiente de correlación. Infórmate sobre sus usos e

interpretación

In [10]:
```
import numpy as np
import matplotlib.pyplot as plt

def coeficiente_correlacion(x, y):
    # Calculos con los datos
    media_x = np.mean(x)
    media_y = np.mean(y)

    desv_x = np.std(x)
    desv_y = np.std(y)

    covarianza = np.cov(x, y)[0, 1]

    # Calcular el coeficiente de correlacion
    coef_corr = covarianza / (desv_x * desv_y)

    return coef_corr

x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]
coef_corr = coeficiente_correlacion(x, y)
print(coef_corr)

# Creando un grafico
plt.scatter(x, y)

plt.plot(np.unique(x), np.poly1d(np.polyfit(x, y, 1))(np.unique(x)), color='red')

plt.title('Grafico de dispersion\nCoeficiente de correlacion: {}'.format(coef_corr))
plt.xlabel('x')
plt.ylabel('y')


plt.show()
```

```
1.2499999999999998
```



Para entender mejor el eoficiente de correlacion se hace dentro de un grafico. Por ejemplo creo que es importante segun lo inventigado cuando estamos analizando los datos de un estudio de mercado, muchas veces nos encontramos con la necesidad de saber si entre dos variables de tipo cuantitativo existe algún tipo de relación. Por ejemplo, a la hora de evaluar un producto o un servicio de una compañía, podemos querer saber si existe alguna relación entre la puntuación que se le ha dado a ese producto y el nivel de ingresos. Hay diferentes formas de analizar estos datos, pero una de ellas es comprobar si existe correlación entre esas dos variables. Veo que el concepto es muy amplio para seguir aprendiendo

In [ ]: