Gráficos de puntos

En un gráfico de puntos (*scatter plot*) permiten representar los datos como puntos dispersos en unos ejes coordenados. Un gráfico de puntos se utiliza para mostrar relaciones entre dos variables, especialmente cuando esta dependencia no es funcional, y también sirven para mostrar distribuciones de datos discretas.

Configuración del entorno.

Lo primero es configurar el entorno de ejecución. Véase el cuaderno "Primeros pasos con Matplotlib" para más detalles.

In [1]:
```
%matplotlib notebook
import matplotlib as mpl
import matplotlib.pyplot as plt
print('Matplotlib version: {}'.format(mpl.__version__))
import numpy as np
np.set_printoptions(floatmode='fixed', precision=3)
import pandas as pd
```

```
Matplotlib version: 3.5.2
```

Graficos de puntos con usando el comando plot.

La primera alternativa para generar un grafo de puntos es usando el mismo comando usado para generar gráficos de líneas: plot().

Como vamos a mostrar puntos aislados, sin conectar con segmentos de línea recta, no especificaremos un estilo de línea y en su lugar especificaremos un estilo de marcador de punto con el argumento `marker`.

Ejercicio 01: Para realizar los ejercicios vamos a simular cuatro distribuciones de 10 puntos (x, y) aleatorios.

Para ello usaremos una distribución aleatoria Gausiana con desviaciones $\sigma_x = 0.25, \sigma_y = 0.25)$ y medias (μ_x^i, μ_y^i): $(0.0, 0.0), (0.0, 1.0), (1.0, 1.0), (1.0, 0.0)$.

El resultado mostrado debería ser parecido a lo siguiente:

```
Media dist1: [-0.087 -0.004]
Media dist2: [-0.048  1.079]
Media dist3: [1.048 1.128]
Media dist4: [ 1.153 -0.028]
```

In [2]:
```
gen = np.random.default_rng(0)

dist1=[]
#Pon tu código aquí.
#Sugerencia: usa standard_normal() para generar una distribución Gaussiana
```

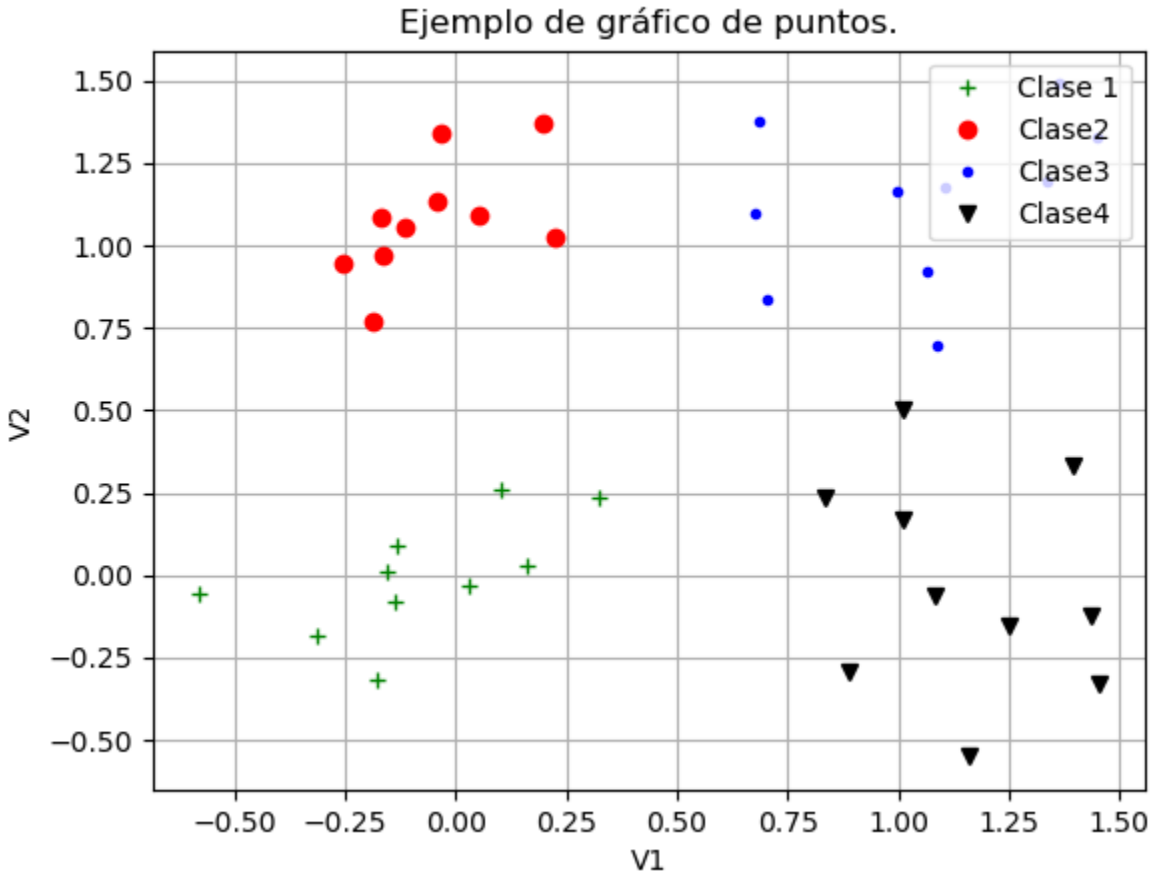
```
#con media (0,0) y desviación (0.25, 0.25)

#

dist2=[]
#Pon tu código aquí.
#Sugerencia: usa standard_normal() para generar una distribución Gaussiana
#con media (0,1) y desviación (0.25, 0.25)

#

dist3=[]
#Pon tu código aquí.
#Sugerencia: usa standard_normal() para generar una distribución Gaussiana
#con media (1,1) y desviación (0.25, 0.25)

#

dist4=[]
#Pon tu código aquí.
#Sugerencia: usa standard_normal() para generar una distribución Gaussiana
#con media (1,0) y desviación (0.25, 0.25)

#

print('Media dist1:', np.mean(dist1, axis=0))
print('Media dist2:', np.mean(dist2, axis=0))
print('Media dist3:', np.mean(dist3, axis=0))
print('Media dist4:', np.mean(dist4, axis=0))
```

```
Media dist1: nan
Media dist2: nan
Media dist3: nan
Media dist4: nan
```

```
/home/ma1macuf/env-master-ccdd/lib/python3.9/site-packages/numpy/core/fromnumeric.py:347
4: RuntimeWarning: Mean of empty slice.
  return _methods._mean(a, axis=axis, dtype=dtype,
/home/ma1macuf/env-master-ccdd/lib/python3.9/site-packages/numpy/core/_methods.py:189: R
untimeWarning: invalid value encountered in double_scalars
  ret = ret.dtype.type(ret / rcount)
```

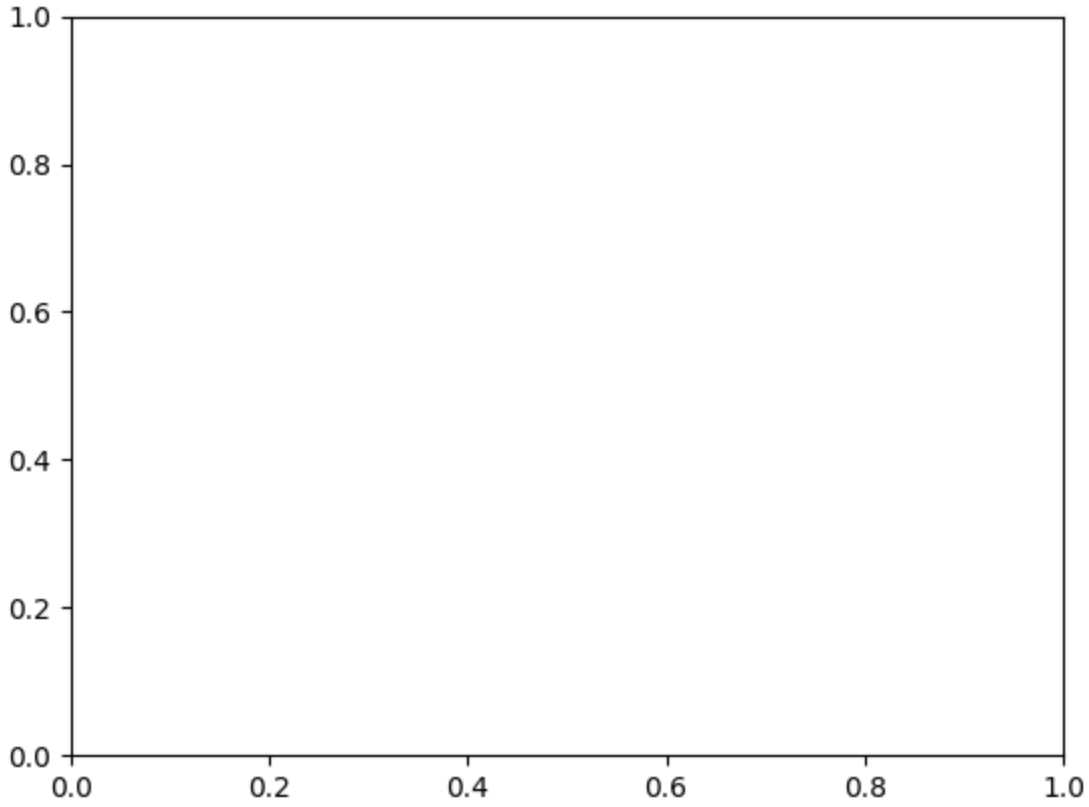
Ejercicio 02: Dadas las cuatro distribuciones anteriores, generar un gráfico de puntos para visualizar las distribuciones.

Intenta ajustar los parámetros para que el resultado sea lo más parecido posible a la siguiente figura.



In [3]:

```
fig = plt.figure()
ax = plt.axes()

#Pon tu código aquí
#Sugerencia: utiliza el método plot para dibujar los puntos (x,y)
#uno para cada distribución.
#Utiliza '' como estilo de línea.
#Utiliza el argumento marker para indicar el tipo de punto.
#Utiliza el argumento label='Clase X' para añadir una legenda a
#cada distribución.

#

#Pon tu código aquí.
#Visualiza etiquetas 'V1' y 'V2' para los ejes X,Y respectivamente.
#Activa la rejilla en los ejes.
#Activa la visualización de las leyendas situándola en
#la esquina superior derecha.

#
```

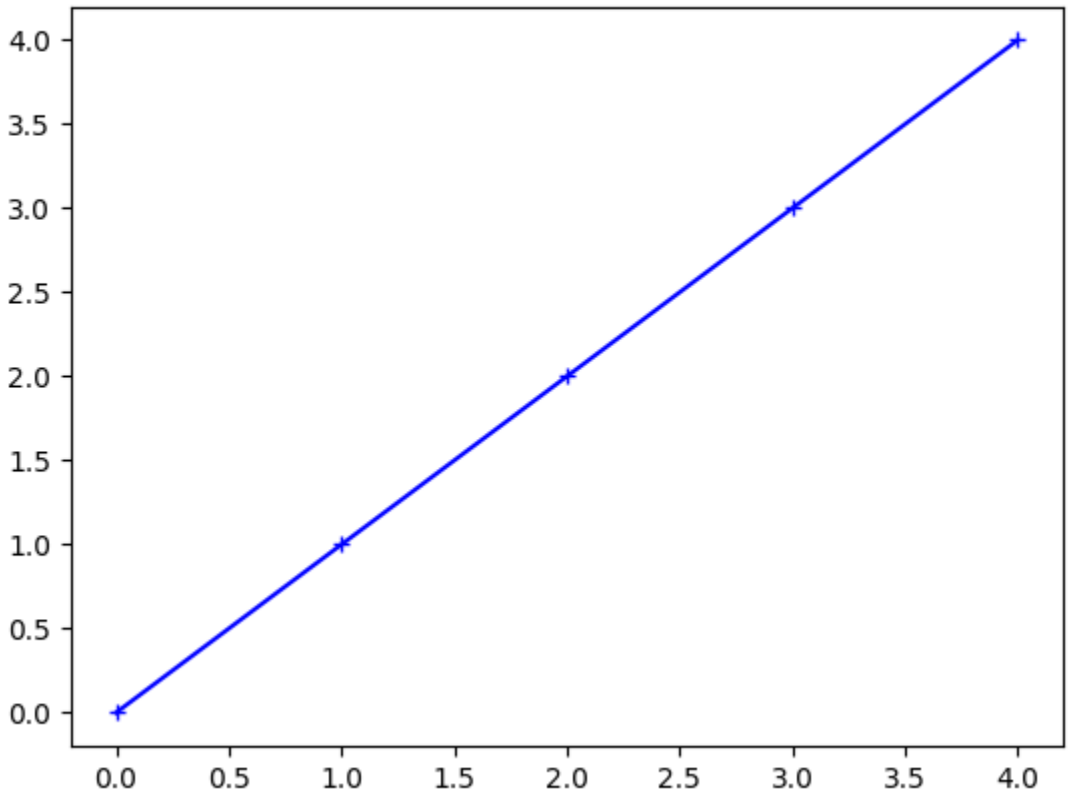


Especificando el estilo de línea, marcador y color a la vez.

Como hemos visto, el comando plot permite dibujar líneas y puntos aislados. Para especificar el estilo de línea hemos usado el parámetro 'linestyle', para el tipo de marcador 'marker' y el parámetro 'color' para el indicar el color.

alternativamente, podemos especificar estos tres parámetros a la vez con el parámetro `fmt` usando el formato `'[marker][line][color]'`. Por ejemplo el parámetro `fmt='+-b'` signifca usar cruces para marcar los puntos, línea contínua para unir estos puntos y color azul. Veamos un ejemplo:

In [4]:
```
fig = plt.figure()
ax = plt.axes()
ax.plot(np.arange(0, 5), np.arange(0,5), '+-b')
```



Out[4]: `[<matplotlib.lines.Line2D at 0x7f6e6f9716a0>]`

Gráficos de puntos usando el comando scatter().

El comando scatter() está diseñado para dibujar gráficos de puntos dispersos en exclusiva.

Como ventaja respecto a usar `plot()` es el mayor control para indicar propiedades de cada punto aislado. Como desventaja es su lentitud que es evidente cuando se dibuja una distribución de puntos muy grande.

Usando la función `scatter()` podemos fijar propiedades distintas a cada uno de los puntos a dibujar, como por ejemplo, el rádio del punto, el color, su estilo, ...

Ejercicio 03: Supongamos que cada punto de las distribuciones que hemos generado anteriormente tiene asociado un peso que indica por ejemplo la importancia de ese punto en la distribución. Para simular estos pesos vamos a utilizar una distribución uniforme en el rango $[0, 1)$, donde el valor 1.0 indicaría máxima importancia.

El resultado mostrado debería parecerse a lo siguiente:

```
Media pesos1: 0.551
Media pesos2: 0.474
Media pesos3: 0.578
Media pesos4: 0.544
```

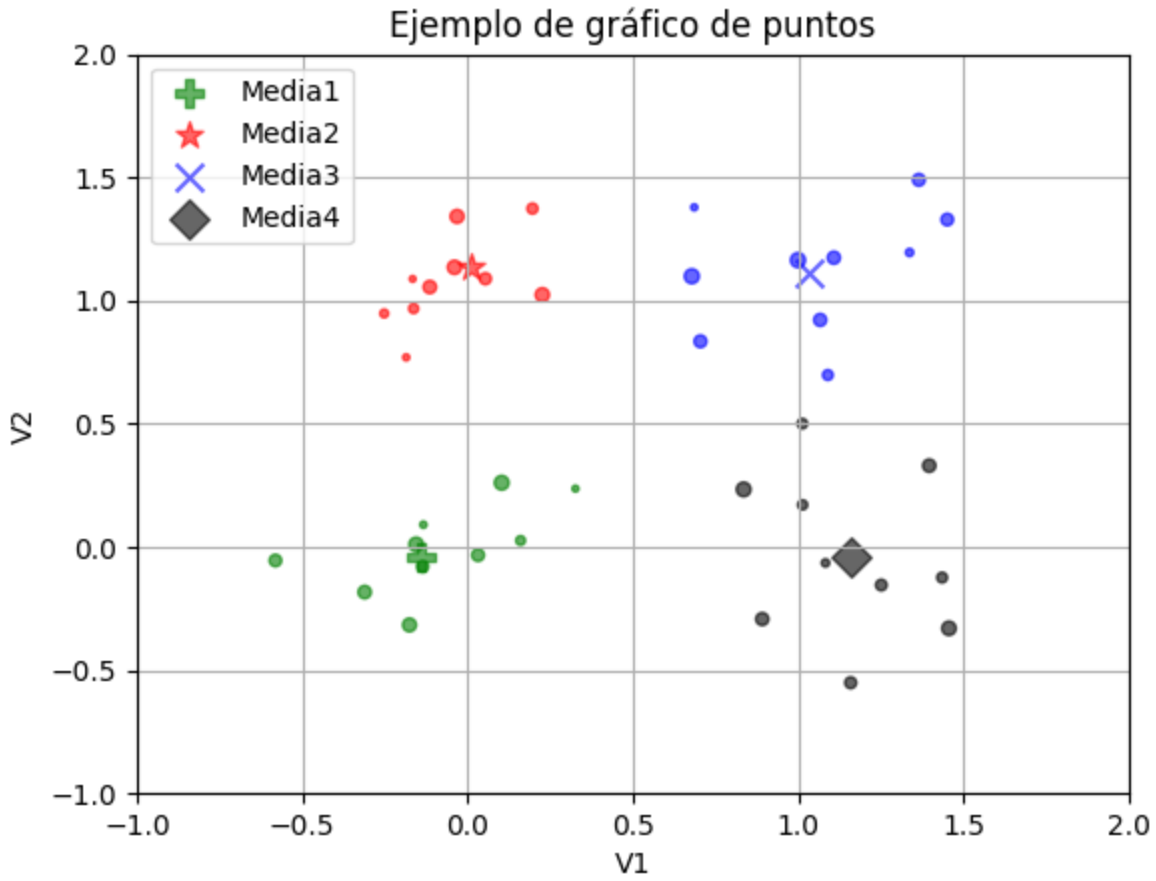
In [5]:
```
gen=np.random.default_rng(0)

pesos1 = []
```

```
#Pon tu código aquí:
#Sugerencia: usa el método gen.random() para generar una
#distribución aleatoria uniforme de 10 valores.

#

pesos2 = []
#Pon tu código aquí:
#Sugerencia: usa el método gen.random().

#

pesos3 = []
#Pon tu código aquí:
#Sugerencia: usa el método gen.random().

#

pesos4 = []
#Pon tu código aquí:
#Sugerencia: usa el método gen.random().

#

media1 = 0
media2 = 0
media3 = 0
media4 = 0
#Pon tu código aquí.
#Sugerencia: calcula el media ponderada de cada distribución.
#utiliza la ufunc average.

#

print("Media1 ponderada: ", media1)
print("Media2 ponderada: ", media2)
print("Media3 ponderada: ", media3)
print("Media4 ponderada: ", media4)
```

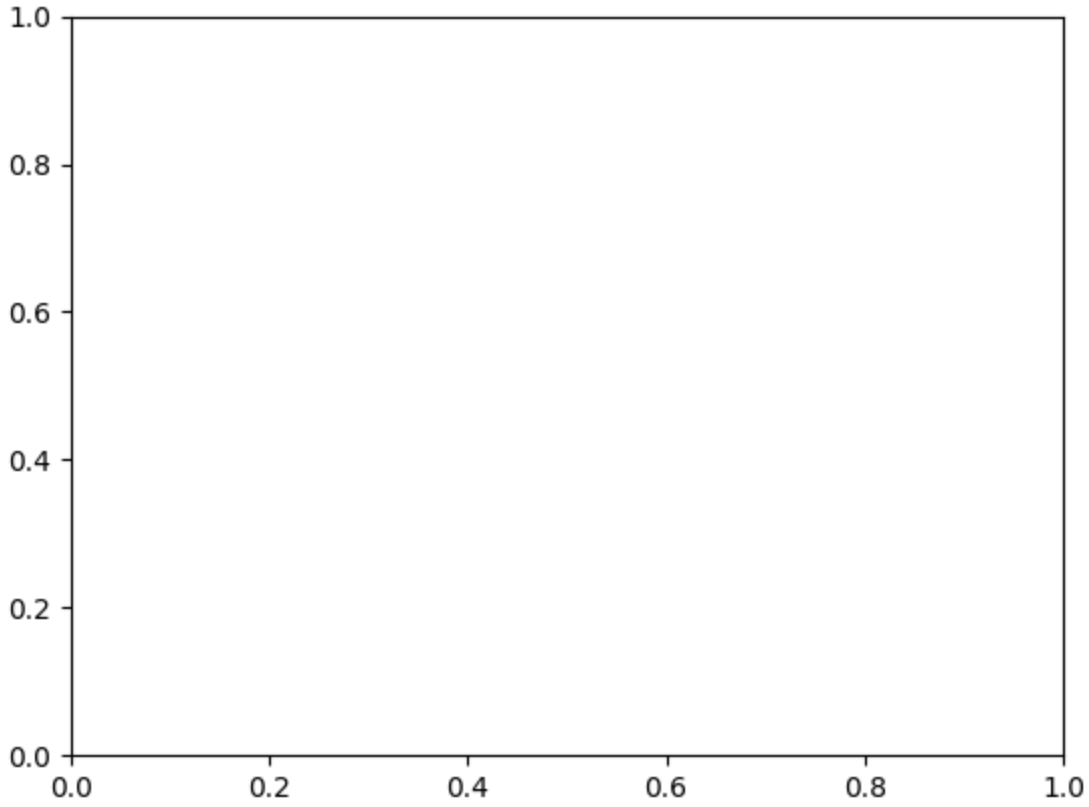
```
Media1 ponderada:  0
Media2 ponderada:  0
Media3 ponderada:  0
Media4 ponderada:  0
```

Ejercicio 04: Dibuja las distribuciones de puntos usando los pesos asociados a cada punto para dibujar un círculo de radio proporcional. Además se requiere calcular las medias ponderadas de cada distribución.

Intenta ajustar los parámetros para que el resultado sea lo más parecido posible a la siguiente figura.



In [6]:
```
fig = plt.figure()
ax = plt.axes()

#Pon tu código aquí.
#Sugerencia: usa el comando scater para cada distribución.
#Utiliza el parámetro 's' para indicar la escala. Este parámetro
#mide el tamaño del marcador en "puntos²". Queremos que un punto
#con peso 0 tenga un tamaño de 5 puntos² y un punto con peso 1.0
#tenga un tamaño de 25 puntos².

#

#Pon aquí tu código.
#Sugerencia: Dibuja un marcador para cada punto que representa
#la media ponderada.

#

#Pont tu código aquí.
#Sugerencia: añade leyendas, título, limites, grid, ...

#
```



In [ ]: