Gráficos para funciones 2D

Estos gráficos representan la relación entre dos variables de la forma $z = f(x, y)$. Ejemplos de estos gráficos pueden ser gráficos de contorno, histogramas 2D o una imágen digital.

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)
```

```
Matplotlib version: 3.5.2
```

Gráficos de contornos.

Un gráficos de contorno es una herramienta para visualizar información de funciones de dos variables continuas $z = f(x, y)$.

También podemos usarlos para visualizar información tridimensional (X, Y, Z) considerando por ejemplo la variable Z como el valor de la función $f(X, Y)$ y en este caso obtendremos un mapa de elevación del terreno o mapa topográfico.

Hay dos funciones para generar gráficos de contorno: `Axes.contour()` y `Axes.contourf()`.

Ejercicio 01: Dada un función que modela la elevación de un terreno, dibujar un gráfico de contornos para la región $x \in [0, 5)$, $y \in [0, 5)$. Se requiere generar una rejilla de puntos (grid) con forma (50,50).

La salida debería ser algo parecido a lo siguiente:

```
Z:
array([[ 1.000e+00,  9.492e-01,  8.108e-01, ...,  1.634e-11,  4.572e-08,
         3.373e-06],
       [ 9.492e-01,  8.141e-01,  6.285e-01, ..., -1.916e-01, -1.791e-01,
        -1.640e-01],
       [ 8.108e-01,  6.278e-01,  4.390e-01, ..., -3.717e-01, -3.470e-01,
        -3.170e-01],
       ...,
       [ 1.634e-11,  1.518e-01,  3.304e-01, ...,  1.561e+00,  1.223e+00,
         8.485e-01],
       [ 4.572e-08,  1.550e-01,  3.369e-01, ...,  1.272e+00,  8.857e-01,
         5.177e-01],
```

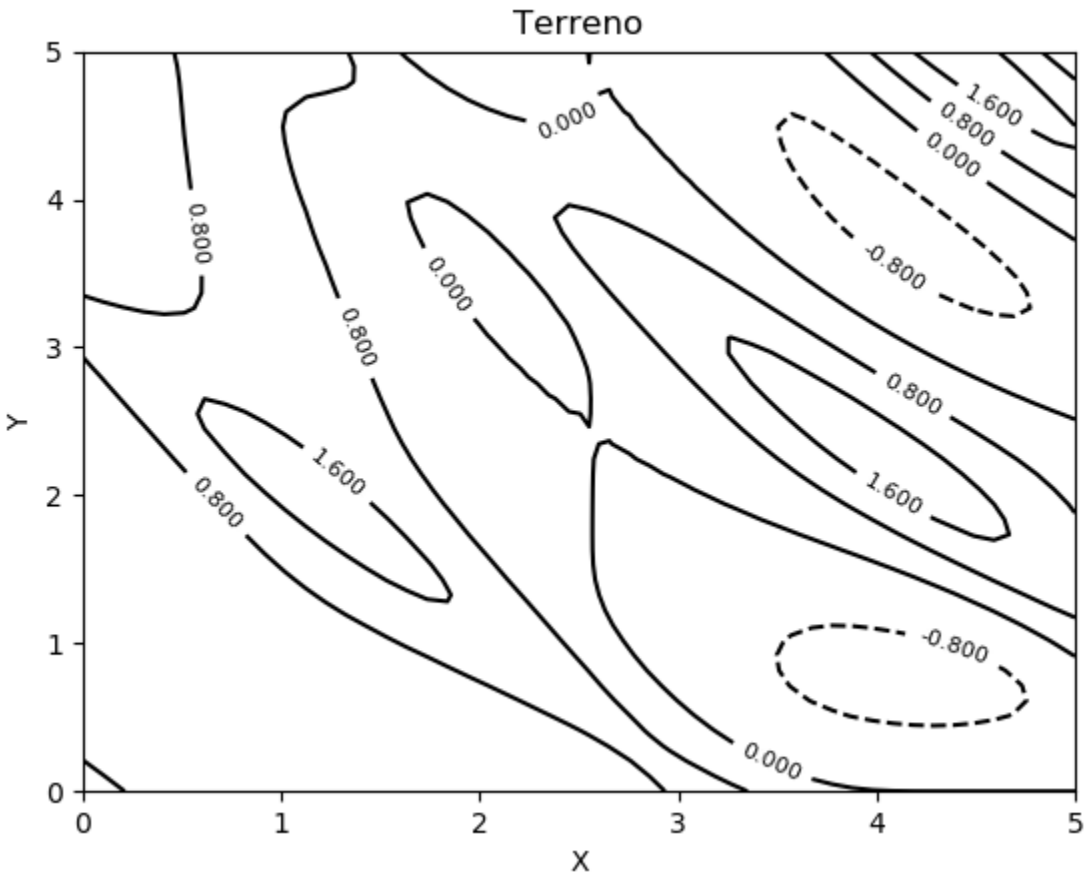
```
       [ 3.373e-06,  1.582e-01,  3.438e-01, ...,  9.251e-01,  5.411e-01,
         2.161e-01]])
```

In [2]:
```python
def terreno(x, y):
    return np.cos(x+y) ** 10 - np.sin((y * x)/2) * np.sin(x+10)

X = []
Y = []
Z = []
#Pon tu código aquí.
#Sugerencia: utiliza np.meshgrid para crear la rejilla 2D.
#utiliza np.linspace para generar las divisiones en cada eje.
#utiliza la función terreno para obtener los valores de elevación Z.

#
print('Z:')
Z
```

Out[2]:
```
Z:
[]
```

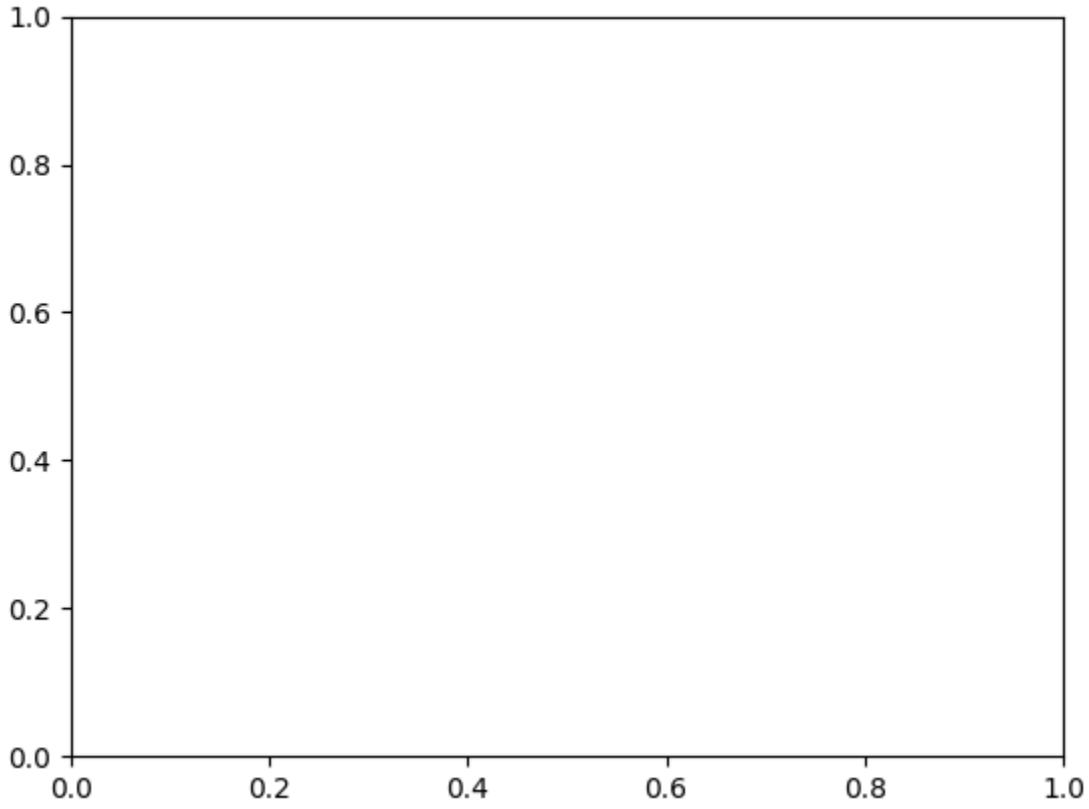
Ejercicio 02: Dados los datos de elevación generados en el ejercicio anterior X, Y, X se requiere generar un gráfico de contorno para representarlos. El gráfico usará tres niveles de contornos.

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



In [3]:
```python
fig = plt.figure()
ax = plt.axes()

#Pon tu ćodigo aquí
#Sugerencia: Añade el título y las etiquetas de los ejes.

#
```

```
contours=[]
#Pon tu código aquí.
#Sugerencia: crea el gráfico con contour()
#utiliza el parámetro levels para indicar el número de niveles.
#Recoge los contornos generados en la variable contours.

#

#Pon tu código aquí.
#Sugerencia: usa el método Axes.clabel() para añadir etiquetas
#a los contornos generados.
#Utiliza los parámetros inline y fontsize con los valores apropiados.

#
```



Como puedes ver, los contornos para iso-valores negativos se representan con líneas de puntos.

Rellenando el espacio entre contornos.

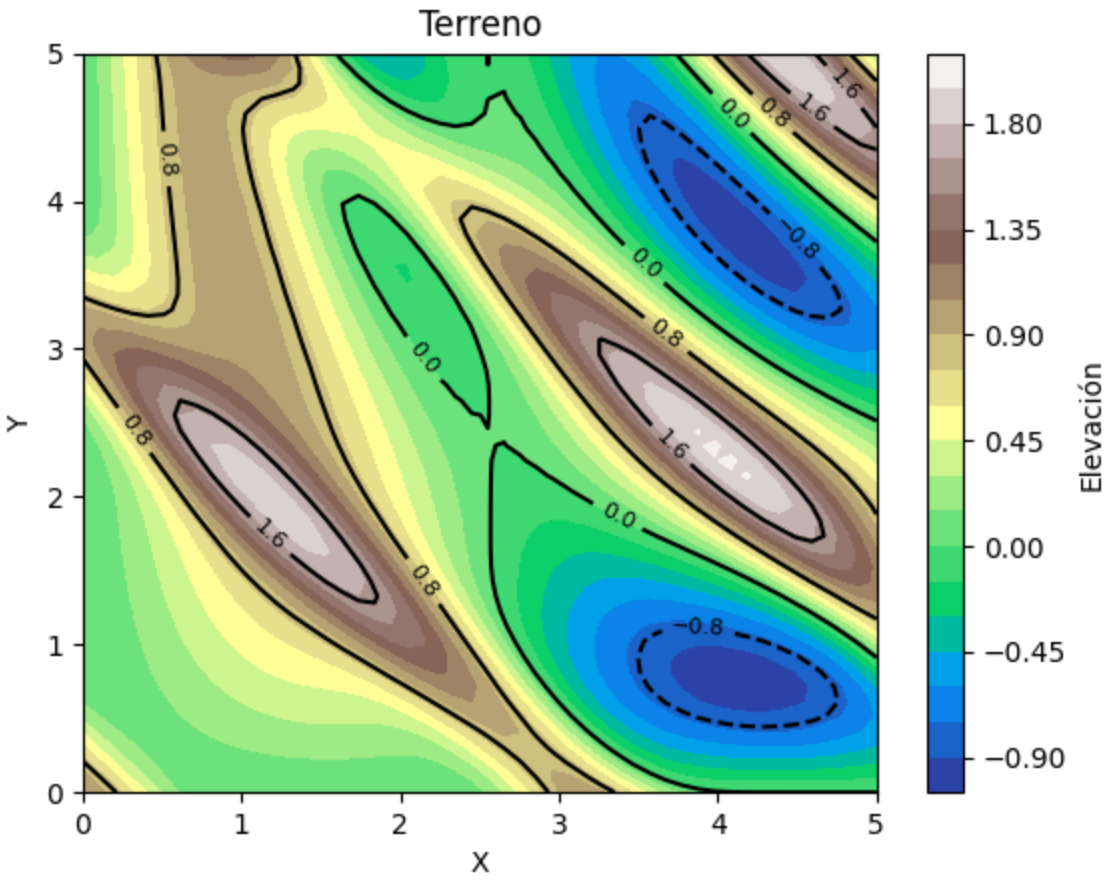
Otras veces será interesante rellenar el espacio entre contrnos con una escala de color proporcional a los valores de iso-contornos cercanos. Para ello usaremos el método `Axes.contourf()`.

Para rellenar con colores, se debe indicar un mapa de color. Como puedes ver hay muchos mapas de color que están orientados a distintos fines.

Es común, cuando se usa un mapa de color para representar el valor de una variable, añadir a la figura una barra de color asociada al mapa de color y contornos generados para que se pueda corresponder el color usado con el valor de la variable de elevación Z.

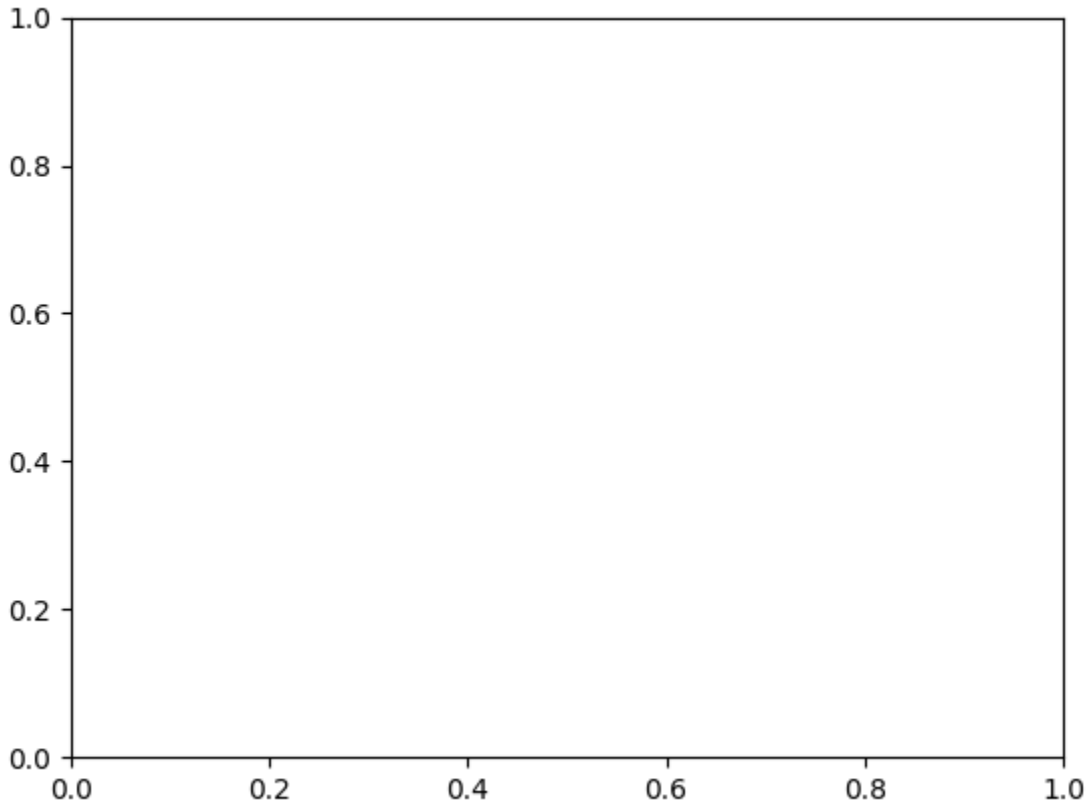
Ejercicio 03: Usando los datos de elevación de terreno (X, Y, Z), se requiere generar un mapa de contornos rellenando con un mapa color apropiado para visualizar datos topológicos y usaremos 20 niveles. Añade la correspondiente barra de color a la figura. Además vamos a añadir el gráfico anterior para que también se visualicen las líneas y sus etiquetas.

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



In [4]:
```
fig = plt.figure()
ax = plt.axes()

#Pon tu ćodigo aquí
#Sugerencia: Añade el título y las etiquetas de los ejes.

#

contours=[]
#Pon tu código aquí.
#Sugerencia: crea el gráfico con contourf()
#utiliza el argumento levels para indicar el número de niveles.
#utiliza el argumento cmap para indicar el mapa de color 'terrain'.
#Recoge los contornos generados en la variable contours.

#

cbar=None
#Pon tu código aquí.
#Sugerencia: crea la barra de color asociada a la figura con
#el método Figure.colorbar().
#Utiliza el método ColorBar.set_label() para poner la etiqueta
#   'Elevación'.

#
```

```
#Pon tu código aquí.
#Sugerencia: añade los contornos al gráfico generado
#usando ahora Axes.contour() como en el ejercicio anteriror.

#
```



Histogramas 2D.

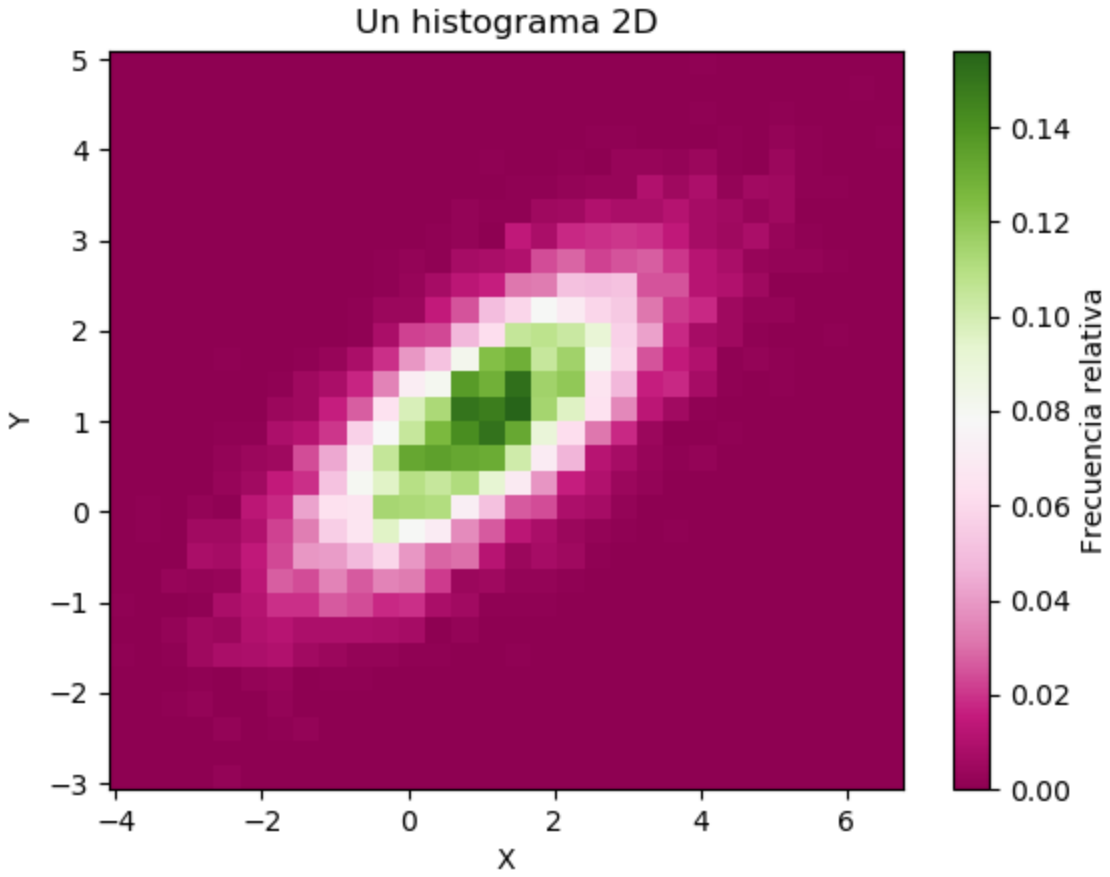
Cuando queremos estudiar una distribución de dos variables se utiliza un histograma 2D. Para generar este tipo de gráficas usamos el comando `hist2d()`.

Vamos a generar una muestra de dos variables que están relacionadas mediante una distribución gaussiana bidimensional:

In [5]:
```
gen = np.random.default_rng(0)
mean = [1, 1]
cov = [[2, 1], [1, 1]]
X, Y = gen.multivariate_normal(mean, cov, 10000).T
```

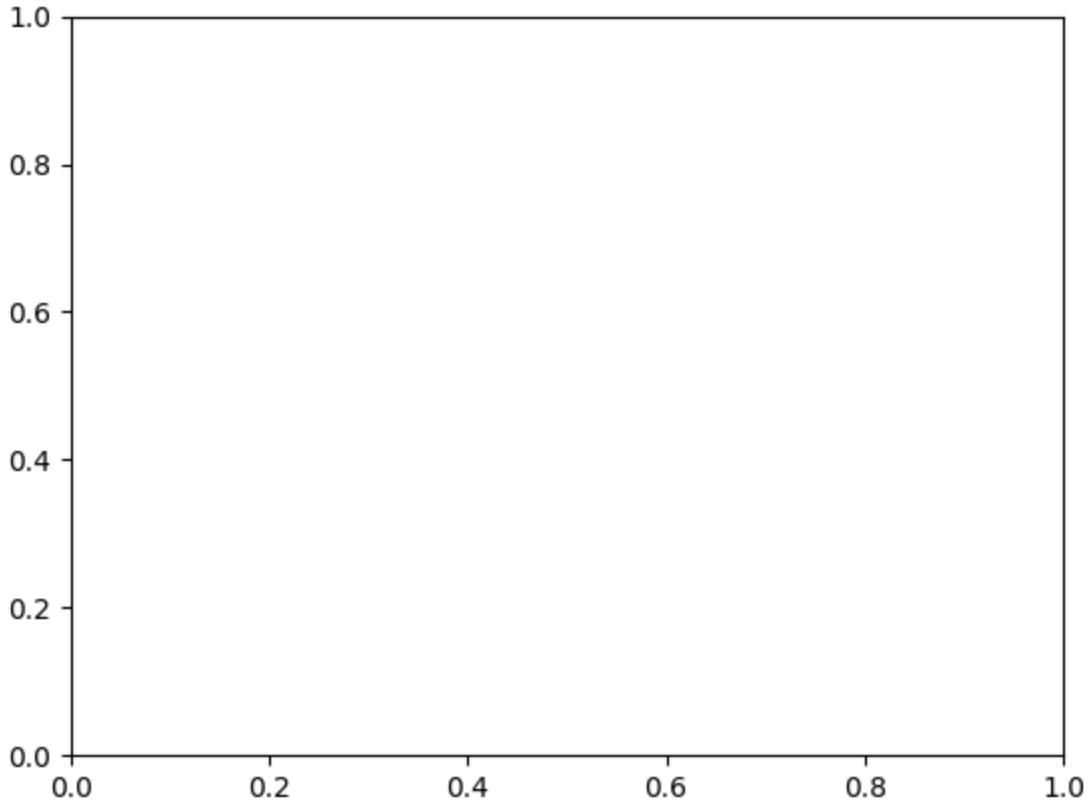
Ejercicio 04: Dadas dos variables continuas X, Y se requiere obtener un histograma 2D para analizar posibles relaciones. Se debe agrupar valores usando 30 bins por eje. Como se quiere resaltar diferencias vamos a utilizar un mapa de color 'divergente', por ejemplo, el mapa 'PiYG'. Además para facilitar la interpretación se debe añadir la correspondiene barra de color a la figura. Se requiere mostrar frecuencias relativas.

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 ćodigo aquí
#Sugerencia: Añade el título y las etiquetas de los ejes.

#

h2d = [] #el histograma.
xedges = [] #los intervalos en el eje x.
yedges = [] #los intervalos en el eje y.
img = [] #la imagen generada como gráfico.

#Pon tu código aquí.
#Sugerencia: utiliza hist2d(). Indica que se requieren 30 bins
#por eje. Queremos que el histograma represente una densidad y
#el mapa de color a utilizar será 'PiYG'.
#Recuerda recoger el resultado para poder generar la
#la barra de color.

#

cb = None #la barra de color.
#Pon tu código aquí.
#Sugerencia: crea una barra de color para la figura con el método
#Figure.colorbar() usa la imagen generada con hist2d para generarla.
#Usa el método ColorBar.set_label() para añadir la etiqueta.

#
```



Imágenes digitales.

También podemos generar una figura con una imagen digital cargada desde un archivo gráfico (`png` , `jpeg` , `tiff` , ...)

Maplotlib tiene el subpaquete `image` que ofrece funcionalidad para cargar con `image.imread()` y manipular imágenes.

La imagen cargada se devuelve como un `numpy.ndarray` y utilizaremos el método `Axes.imshow()` para mostrarla en unos ejes.

Vamos a incluir el módulo `image` en el entorno de ejecución.

In [7]:
```
from matplotlib import image
```

Ejercicio 05: Generar un gráfico con tres imágenes. Para mejorar la visualización no es necesario dibujar los ejes alrededor de las imágenes.

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

In [8]:
```
img1 = []
img2 = []
img3 = []

#Pon tu código aquí.
#Sugerencia: carga las imágenes con el método imread() del
#subpaquete matplotlib.image

#

fig = None
axs = []
#Pon tu código aquí.
#Sugerencia: Crea una figura y el array de ejes con plt.subplots().

#

#Pon tu código aquí.
#Sugerencia: añade un título superior a la figura usando el
#método Figure.suptitle()

#

#Pon tu código aquí.
#Sugerencia: muestra la imagen monocroma. Recuerda usar el
#mapa de color 'gray' y añade el título 'Lena1'
#Para desactivar los ejes usa el método Axes.set_axis_off()

#

#Pon tu código aquí.
#Sugerencia: muestra la imagen RGB y añade el título 'Lena2'
```

```
#Para desactivar los ejes usa el método Axes.set_axis_off()

#

#Pon tu código aquí.
#Sugerencia: muestra la imagen monocroma térmica. Recuerda usar el
#mapa de color 'coolwarn' y añade el título 'Laptop'
#Para desactivar los ejes usa el método Axes.set_axis_off()

#
```

In [ ]: