Gráficos de líneas

Un gráfico de líneas se utiliza para mostrar una secuencia de pares (x_i, y_i) donde $x_i < x_{i+1}$. Aunque podemos representar los puntos de forma aislada, lo común es conectarlos usando segmentos de línea recta obteniendo así un gráfico de líneas.

Este tipo de gráfico sirve para visualizar series temporales (el eje X es temporal) o en general una dependencia funcional entre dos variables $(y = f(x))$

Se recomienda consultar esta referencia para ver una descripción más completa.

Configuración del entorno.

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

In [15]:
```
%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
```

Creación de un gráfico de línea.

Para crear un gráfico de línea utilizamos la función `plt.plot()`.

Ejercicio 01: Genera un ndarray `x` que represente un muestreo del intervalo $[0, 2\pi]$ con 100 muestras.

El resultado esperado debe ser algo pareceido a lo siguiente:

```
[0.000 0.063 0.127 0.190 0.254 0.317 0.381 0.444 0.508 0.571 0.635 0.698
 0.762 0.825 0.889 0.952 1.015 1.079 1.142 1.206 1.269 1.333 1.396 1.460
 1.523 1.587 1.650 1.714 1.777 1.841 1.904 1.967 2.031 2.094 2.158 2.221
 2.285 2.348 2.412 2.475 2.539 2.602 2.666 2.729 2.793 2.856 2.919 2.983
 3.046 3.110 3.173 3.237 3.300 3.364 3.427 3.491 3.554 3.618 3.681 3.745
 3.808 3.871 3.935 3.998 4.062 4.125 4.189 4.252 4.316 4.379 4.443 4.506
 4.570 4.633 4.697 4.760 4.823 4.887 4.950 5.014 5.077 5.141 5.204 5.268
 5.331 5.395 5.458 5.522 5.585 5.649 5.712 5.775 5.839 5.902 5.966 6.029
 6.093 6.156 6.220 6.283]
```

In [16]:
```
x = []
#Pon tu código aquí
#Sugerencia: usa la función np.linspace.
```

```
#
print(x)
```

```
[]
```

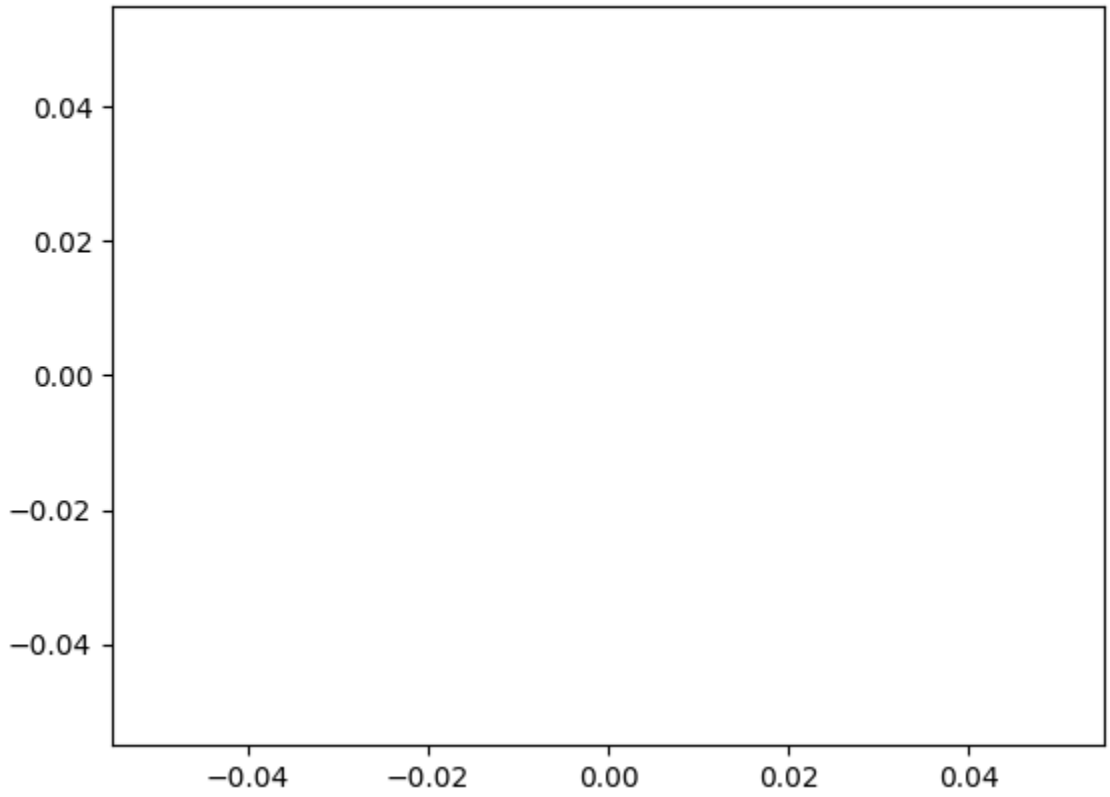
Ahora vamos a crear el objeto Figure y un objeto Axes asociado donde dibujar.

Una figura (objeto `Figure`) representa un contenedor que contiene otros objetos gráficos como por ejemplo unos ejes (un objeto `Axes`), puntos, líneas, leyendas, textos, etc... todos estos conceptos tienen su correspondiente objeto Matplotlib. En esta referencia se describen los elementos de una figura.

En la secuencia anterior se ha creado una figura y se le han añadido un objeto Axes que representan los ejes necesarios para dibujar la gráfica de línea.

Una vez que hemos creados unos ejes insertados en una figura ya podemos dibujar en estos ejes. Esta vez le indicamos por separado los valores de x e y.

In [17]:
```
fig = plt.figure() # crea una figura y la establece como figura actual.
ax = plt.axes() # añade unos ejes a la figura actual y los devuelve.
ax.plot(x, np.sin(x)) # dibujamos la función seno.
ax.plot(x, np.cos(x)) # dibujamos la función coseno.
```



Out[17]:
```
[<matplotlib.lines.Line2D at 0x7fa8863030a0>]
```

Especificar el color de la línea.

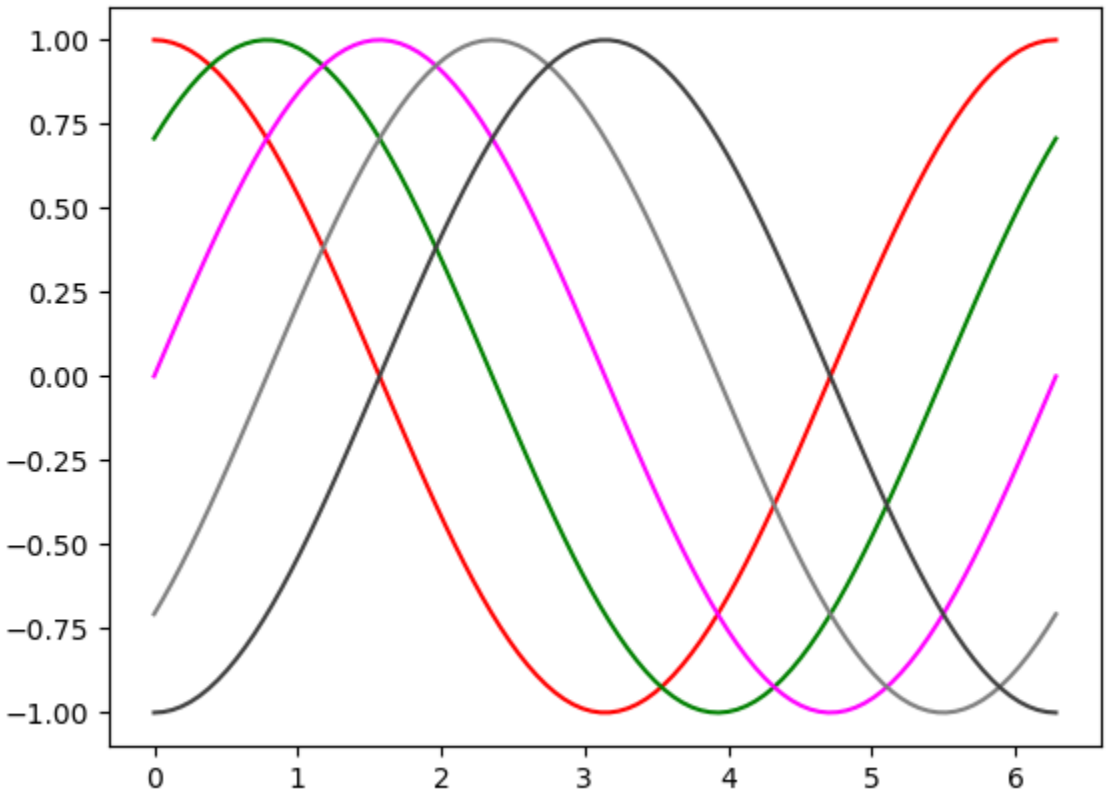
Para indicar el color de la línea usaremos el parámetro `color`.

Hay varias formas de indicar el color:

- Como una letra, p.e., `color='r'` , para los colores `rgbcmyk`
- Como un nombre, p.e., `color='red'`
- Como un valor hexadecimal `color='#FF0000'` , con el formato `#RRGGBB`
- Como un tripleta `color=(1.0, 0.0, 0.0)` , con el formato (r, g, b) con valores $[0, 1]$

Ejercicio 02: Dibujar gráficos de líneas con diferentes colores para las funciones $\cos(x)$, $\cos(x - 0.25\pi)$, $\cos(x - 0.5\pi)$, $\cos(x - 0.75\pi)$ y $\cos(x - \pi)$.

Intenta ajustar los valores de colores para obtener una figura lo más parecida posible a la siguiente:



In [18]:
```
fig = None
#Pon tu código aquí.
#Sugerencia: crea la figura "actual"

#

ax = None
#Pon tu código aquí.
#Sugerencia: Crea y añade unos ejes a la figura "actual".

#

#Pon tu código aquí.
#Sugerencia: crea un gráfico de línea con plot para las funciones
#indicadas. Usa el parámetro color para indicar el color.

#
```

Como especificar el estilo de línea.

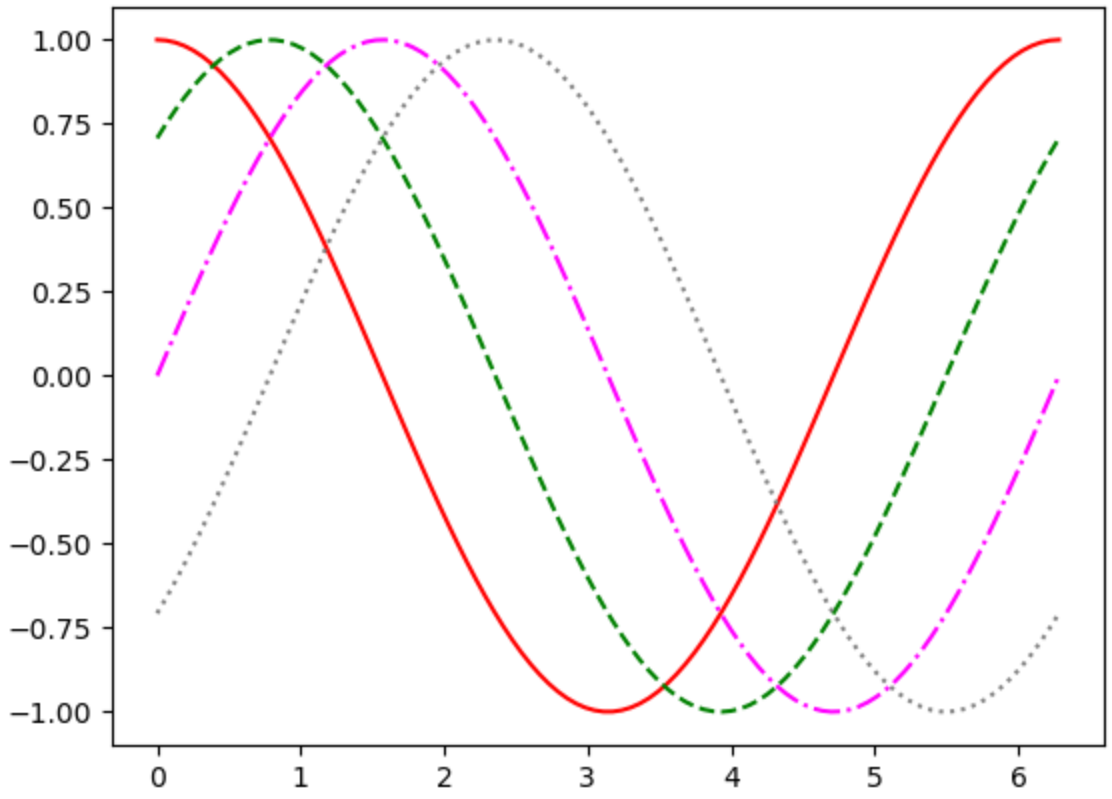
Por defecto se utiliza línea sólida, pero esto también podemos cambiarlo. Para ello se utiliza el parámetro `linestyle`.

Podemos utilizar nombres o de forma codificada. Algunos ejemplos son:

- Línea contínua: `linestyle='solid'` o `linestyle='-'`
- Línea discontínua: `linestyle='dashed'` o `linestyle='--'`
- Línea de puntos y rayas: `linestyle='dashdot'` o `linestyle='-.'`
- Línea de puntos: `linestyle='dotted'` o `linestyle=':'`

Ejercicio 03: Dibujar gráficos de líneas con diferentes colores para las funciones $\cos(x)$, $\cos(x - 0.25\pi)$, $\cos(x - 0.5\pi)$ y $\cos(x - 0.75\pi)$.

Intenta ajustar los valores de colores para obtener una figura lo más parecida posible a la siguiente:



In [19]:
```
fig = None
#Pon tu código aquí.
#Sugerencia: crea la figura "actual"

#

ax = None
#Pon tu código aquí.
#Sugerencia: Crea y añade unos ejes a la figura "actual".

#

#Pon tu código aquí.
#Sugerencia: crea un gráfico de línea con plot para las funciones
#indicadas. Usa el parámetro color para indicar el color.
```

```
#
```

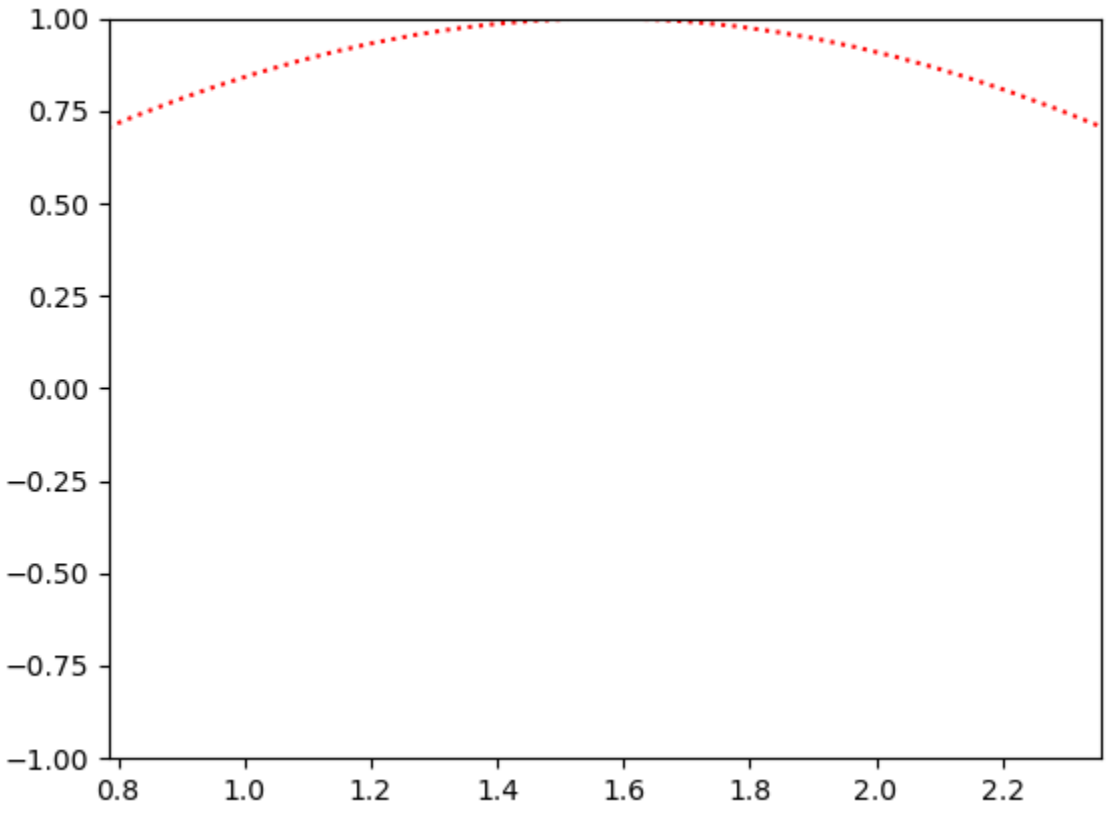
Otros ajustes de la gráfica.

Configurando los límites de la gráfica.

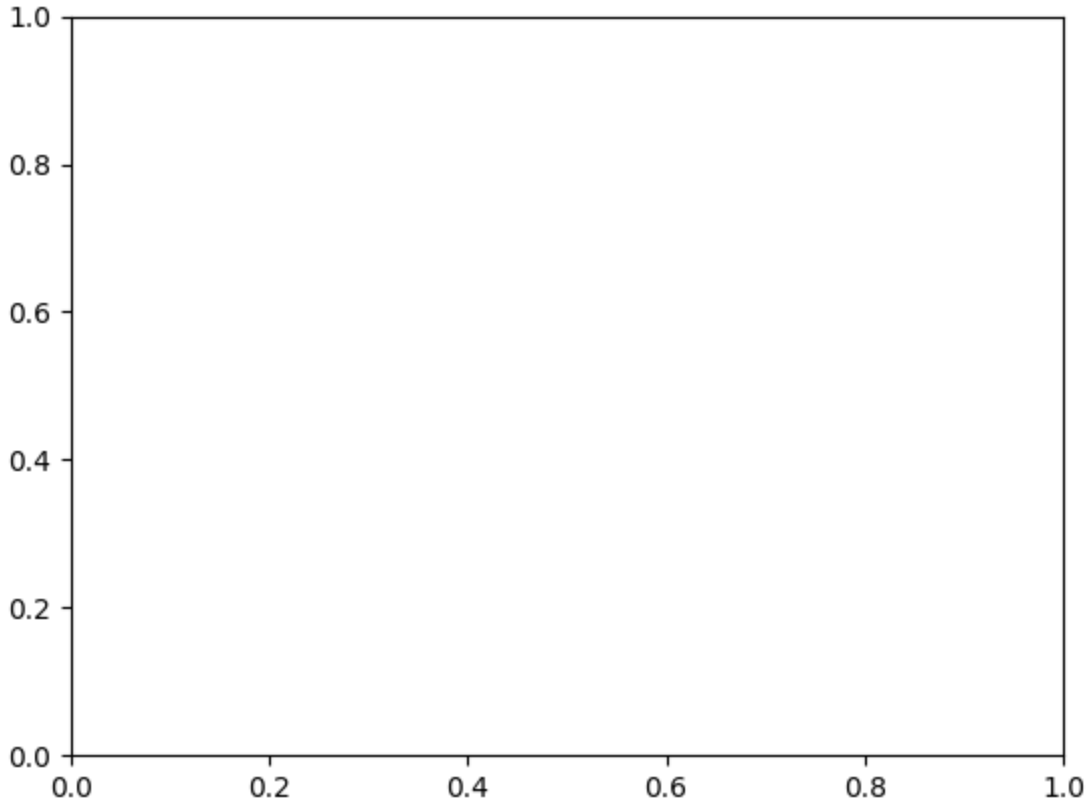
Por defecto los límites para los ejes se reajustan de forma automática para que se pueda dibujar todos los datos usados. Pero nosotros podemos especificar los límites que queremos dibujar con el atributos `Axes.xlim` y `Axes.ylim` del objeto Axes usando el métodos set_xlim(), set_ylim().

Ejercicio 04: Dibujar la función $\sin(x)$ usando el intervalo `x` ya creado con los límites $x \in [0.25\pi, 0.75\pi]$, $y \in [-1, 1]$.

Intenta ajustar los parámetros para obtener una figura lo más parecida posible a la siguiente:



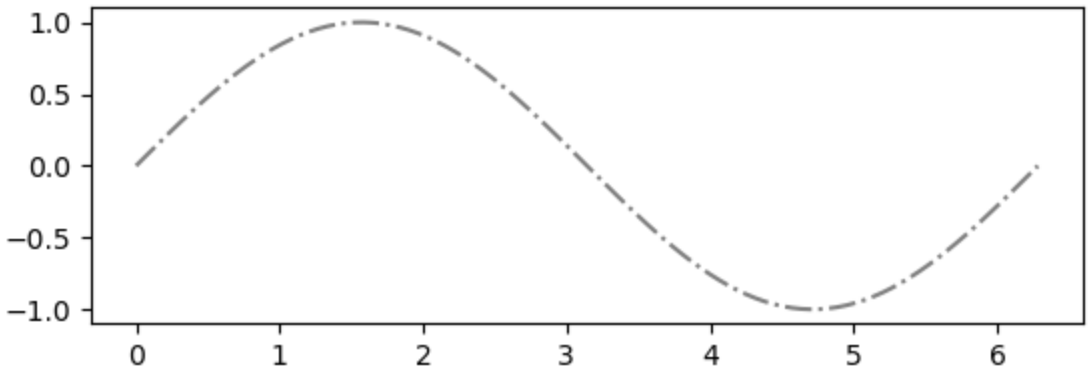
In [20]:
```
fig = plt.figure()
ax = plt.axes()
#Pon tu código aquí.
#Sugerencia: actualiza los límites indicados.

#

#Pon tu código aquí.
#Sugerencia: usa x, np.sin(x) para dibujar.

#
```



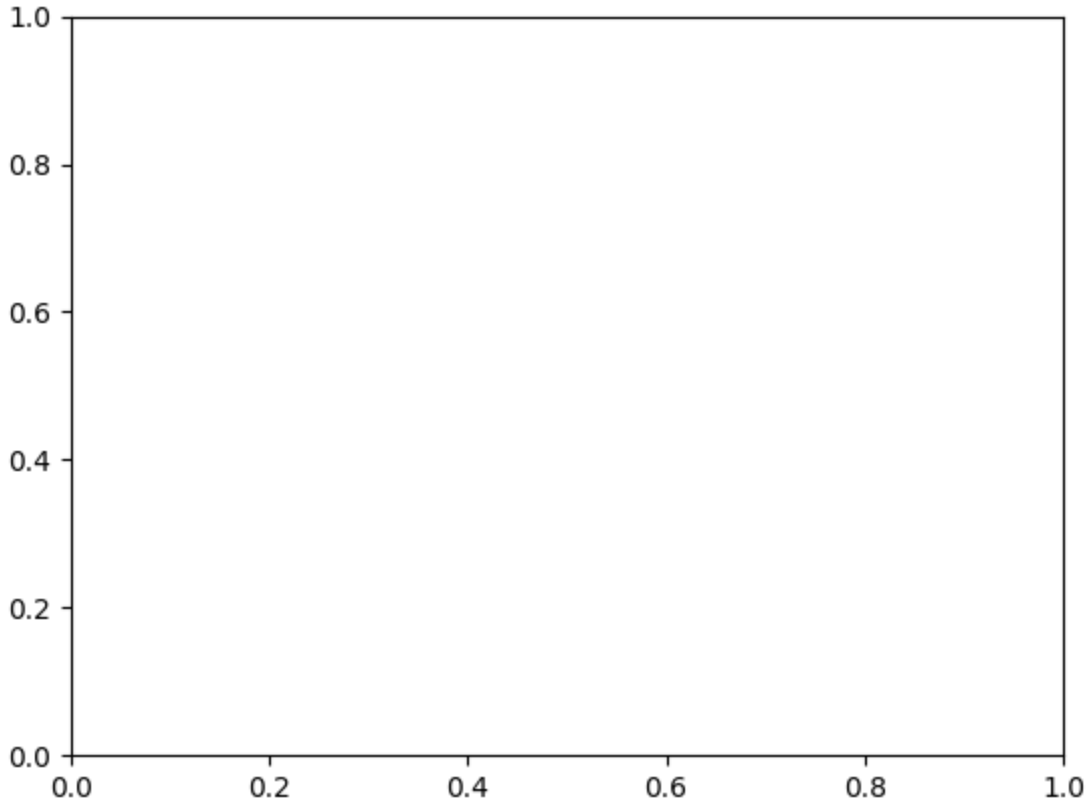
Por defecto la relación de aspecto no es la misma para los ejes X e Y por efecto estético. Sin embargo, algunas veces será interesante que el ratio de aspecto entre eje X e Y sea igual. Esto se puede especificar con el atributo `Axes.aspect` y modificable con `set_aspect()` .dotted

Ejercicio 05: Dibujar la función $\sin(x)$ usando el intervalo `x` ya creado con los límites $[0, 2\pi]$. Se requiere un relación de aspecto 1:1.

Intenta ajustar los parámetros para obtener una figura lo más parecida posible a la siguiente:



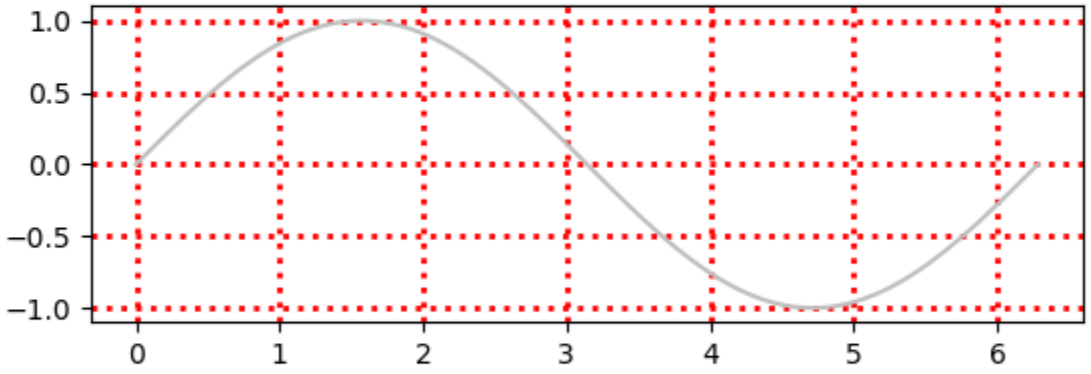
In [21]:
```
fig = plt.figure()
ax = plt.axes()
#Pon tu código aquí.
#Sugerencia: actualiza los límites indicados. Además

#

#Pon tu código aquí.
#Sugerencia: usa x, np.sin(x) para dibujar.

#
```



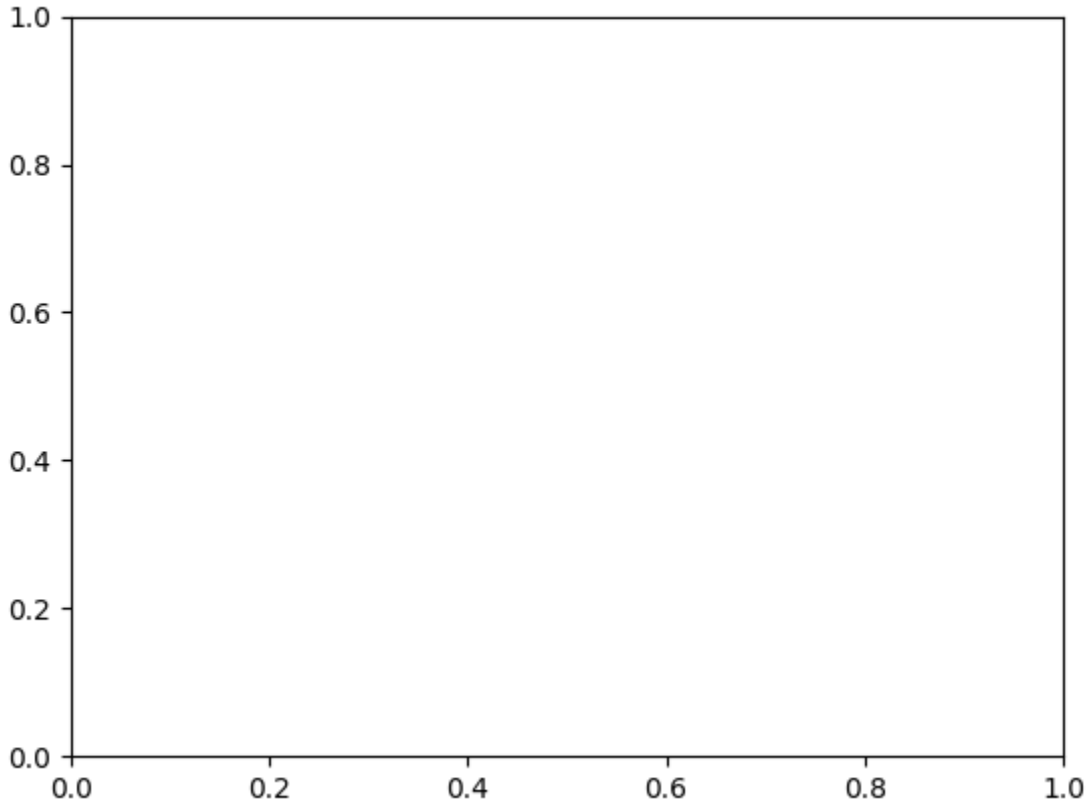
Suele ser interesante utilizar una rejilla en los ejes. Por defecto está desactivada pero podemos activarla con el método `Axes.grid()`.

Ejercicio 06: Dibujar la función $\sin(x)$ usando el intervalo `x` ya creado con los límites $[0, 2\pi]$. Se requiere un relación de aspecto 1:1 y utilizar una rejilla.

Intenta ajustar los parámetros para obtener una figura lo más parecida posible a la siguiente:



In [22]:
```
fig = plt.figure()
ax = plt.axes()
#Pon tu código aquí.
#Sugerencia: actualiza los límites indicados. Además

#

#Pon tu código aquí.
#Sugerencia: usa x, np.sin(x) para dibujar.

#
```



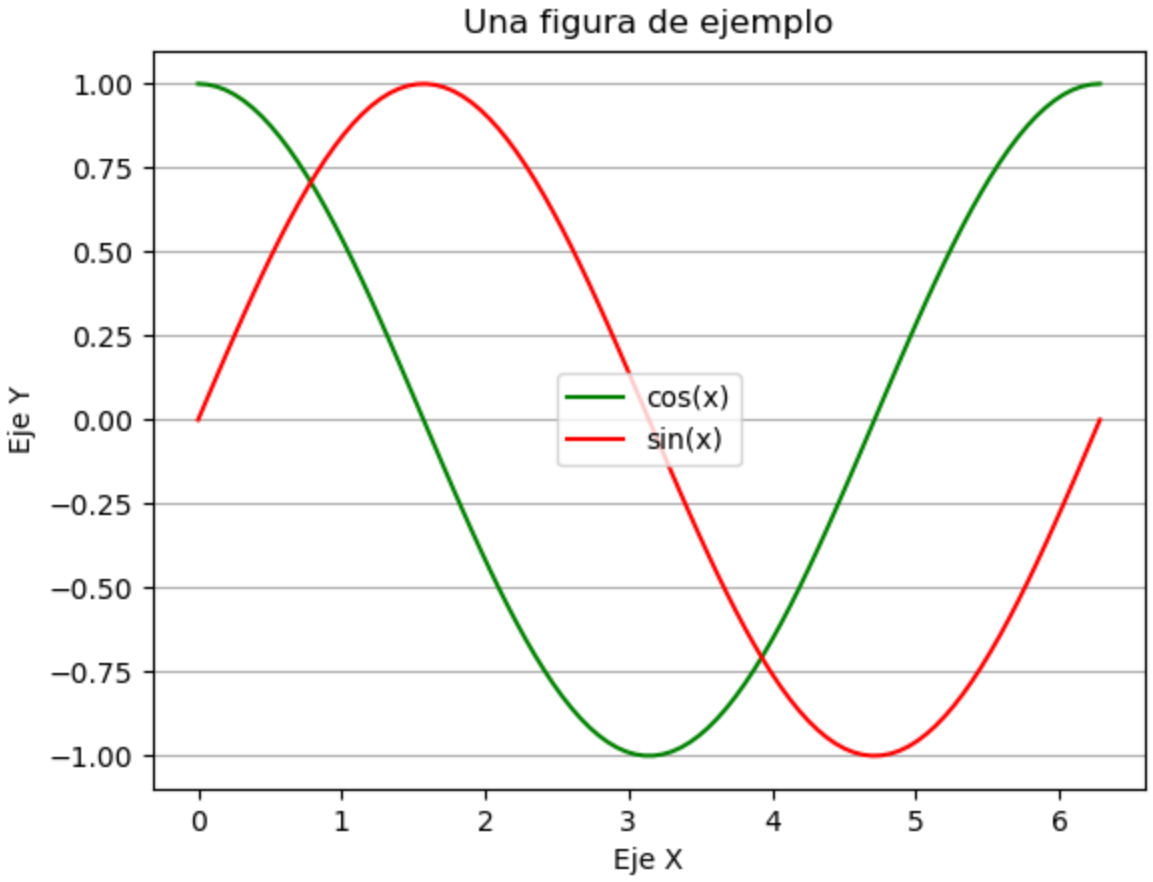
Añadiendo texto a la gráfica.

Podemos añadir:

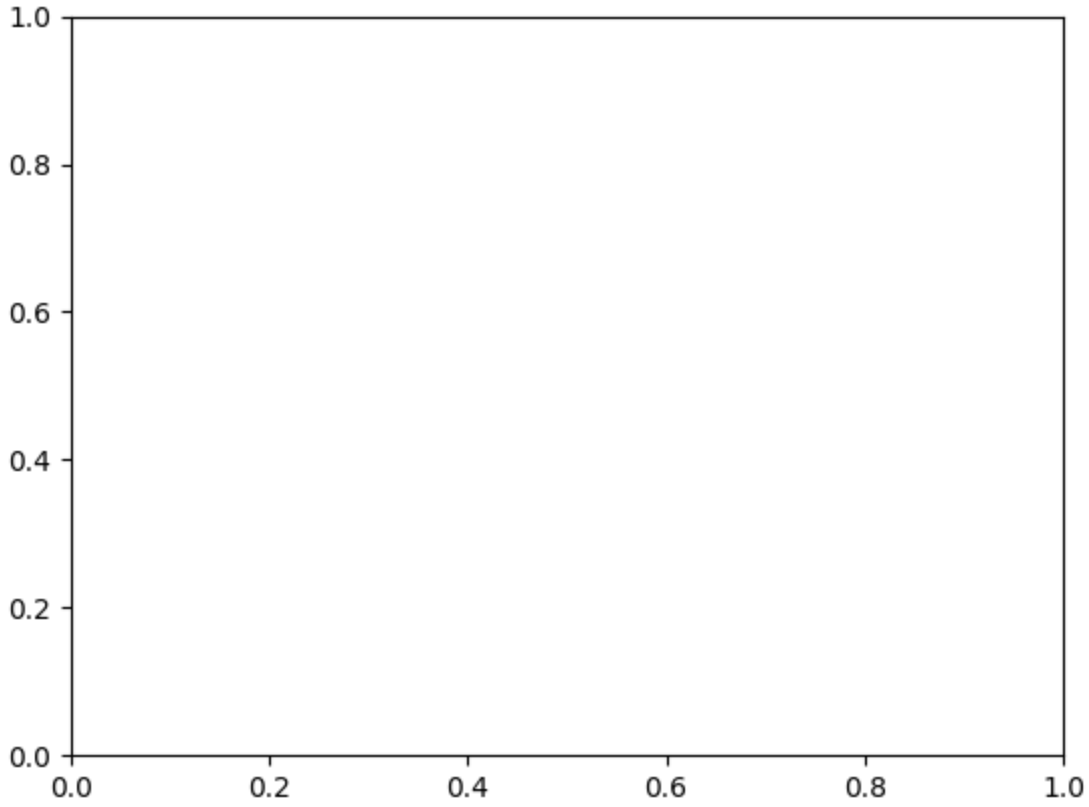
- Un título a la figura con atributo `Axes.title`.
- Nombres de los ejes con los atributos `Axes.xlabel` y `Axes.ylabel`.
- Un caja con las leyendas de las gráficas dibujadas. Para ello usaremos el método `Axes.legend()`. Las leyendas mostradas serán aquellas que han sido dibujadas (en nuestro caso con el método `plot()`) usando el parámetro `label` con el texto de la leyenda.

Ejercicio 07: Dibujar las función $\cos(x)$ y $\sin(x)$ usando el intervalo x ya creado. La figura tendrá por título: `'Una figura de ejemplo'`. Los ejes X e Y tendrán como etiquetas `'Eje X'`, `'Eje Y'` respectivamente. Añadir las leyendas `cos(x)` y `sin(x)` para identificar cada gráfica.

Intenta ajustar los parámetros para obtener una figura lo más parecida posible a la siguiente:



In [23]:
```
fig = plt.figure()
ax = plt.axes()

#Pon tu código aquí.
#Sugerencia: usa x, np.cos(x) y np.sin(x) para dibujar.
#Usa el parámetro label para indicar la leyenda de cada función.

#

#Pon tu código aquí.
#Sugerencia: usa el método legend() para añadir una
#caja con las leyendas.

#
```



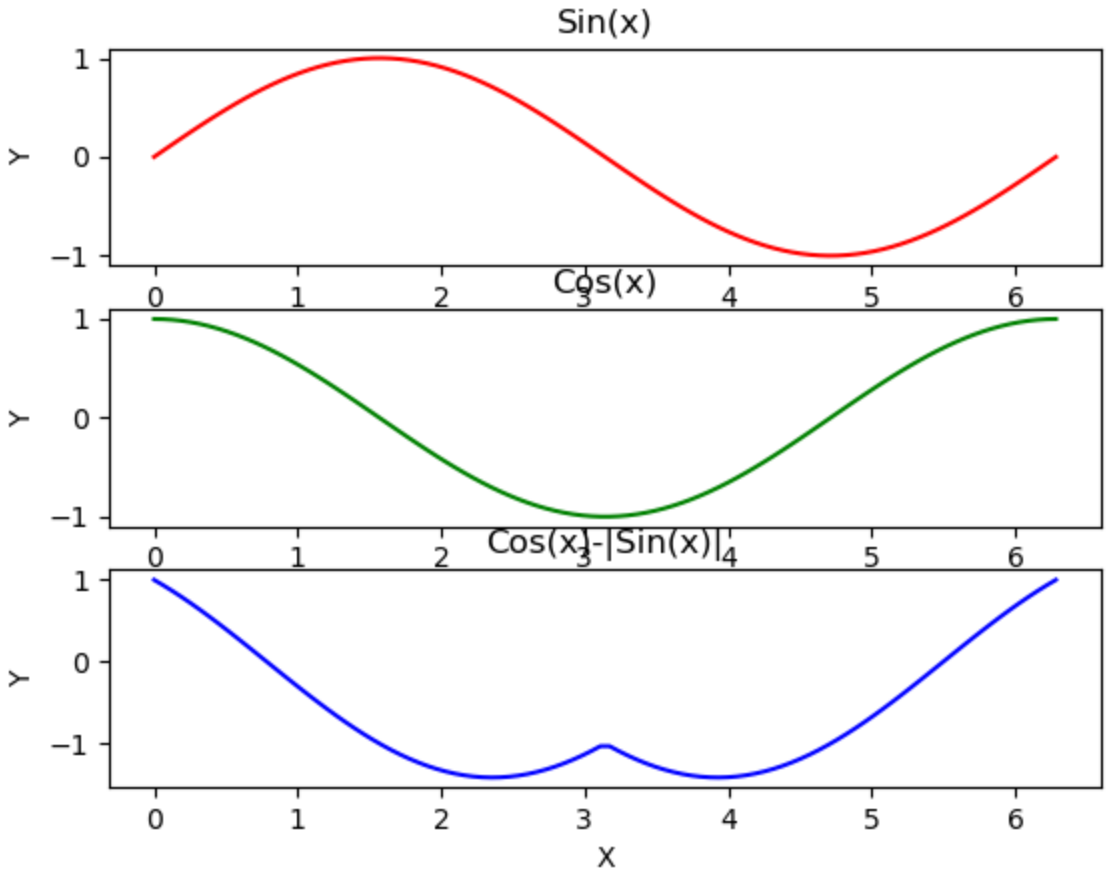
Subplots.

Podemos generar una figura con varios ejes para dibujar gráficas distintas en cada uno de ellos en vez de usar el mismo eje. Para ello usaremos el comando `plt.subplot()`.

Este comando devuelve una figura nueva y un array de objetos Axes según el número de subplots indicados con los parámetros (si sólo se indica un subplot, se devuelve un objeto Axes).

Ejercicio 08: Dibujar las función $\cos(x)$, $\sin(x)$ y $|\cos(x) - \sin(x)|$ en el intervalo x ya creado. Cada gráfica se dibuja en un eje separado, que tendrá por título la función dibujada. Los ejes X e Y de todos los ejes tendrán como etiquetas "X", "Y" respectivamente.

Intenta ajustar los parámetros para obtener una figura lo más parecida posible a la siguiente:

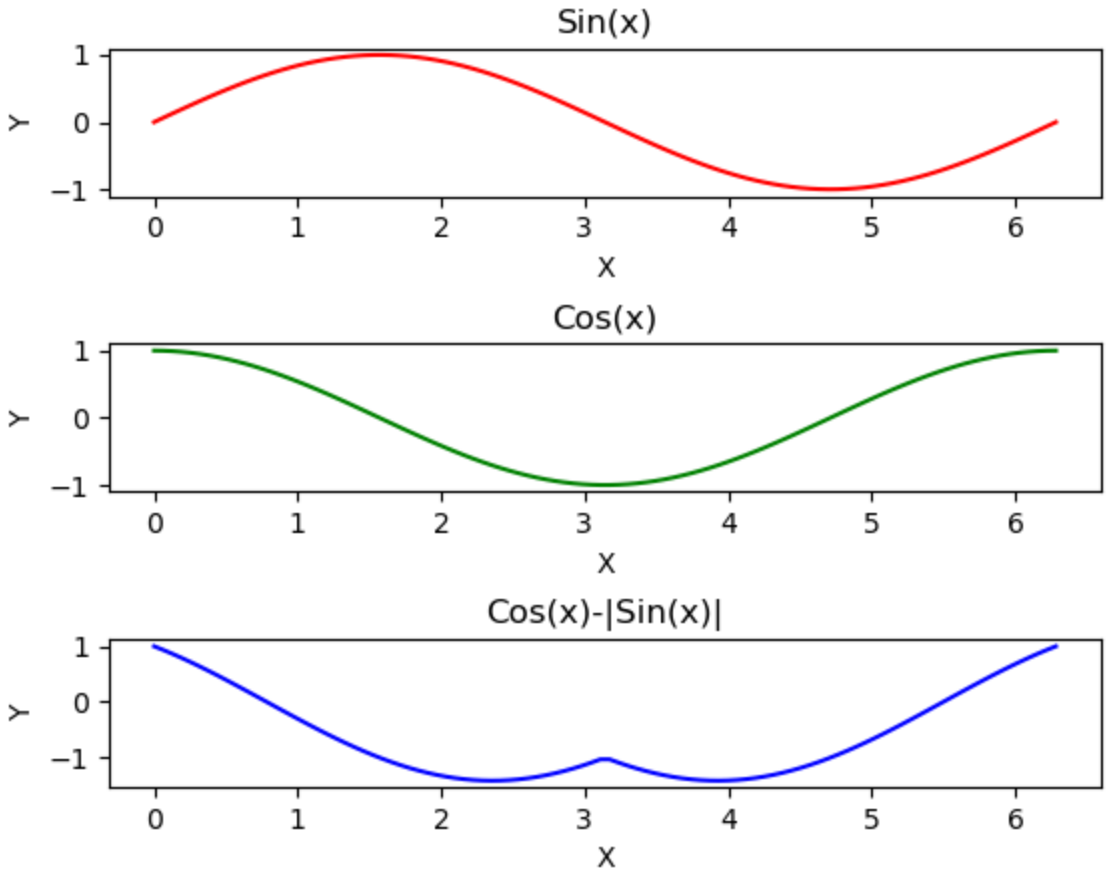


In [24]:
```
fig = None
axes = None
#Pon tu código aquí.
#Sugerencia: usa la función subplots() para crear una figura con tres subplots.

#

#Pon tu código aquí.
#Sugerencia usa axes[] para acceder al i-ésimo eje para dibujar y poner el título.

#
```

En el ejemplo anterior se ha creado una figura con tres ejes (subplots). Como ves los ejes se apilan en vertical por defecto. Hay un problema con los títulos que están sobre escritos con la gráfica superior. Para solucionar esto podemos indicar el porcentaje de espacio en blanco en vertical (atributo `hspace`) que se deja entre subplots usando el parámetro diccionario `gridspec_kw`. El valor de este parámetro es un diccionario de la forma `gridspec_kw = {'atributo':valor}`. Para indicar el expacio a usar en sentido vertical se utiliza el atributo `hspace`. Así pondríamos el parámetro `gridspec_kw={'hspace':1}`.

Ejercicio 09: Dibujar las función $\cos(x)$, $\sin(x)$ y $|\cos(x) - \sin(x)|$ en el intervalo x ya creado. Cada gráfica se dibuja en un eje separado, que tendrá por título la función dibujada. Los ejes X e Y de todos los ejes tendrán como etiquetas "X", "Y" respectivamente. Dejar espacio entre los subplots para que se muestren bien los títulos y leyendas.

Intenta ajustar los parámetros para obtener una figura lo más parecida posible a la siguiente:

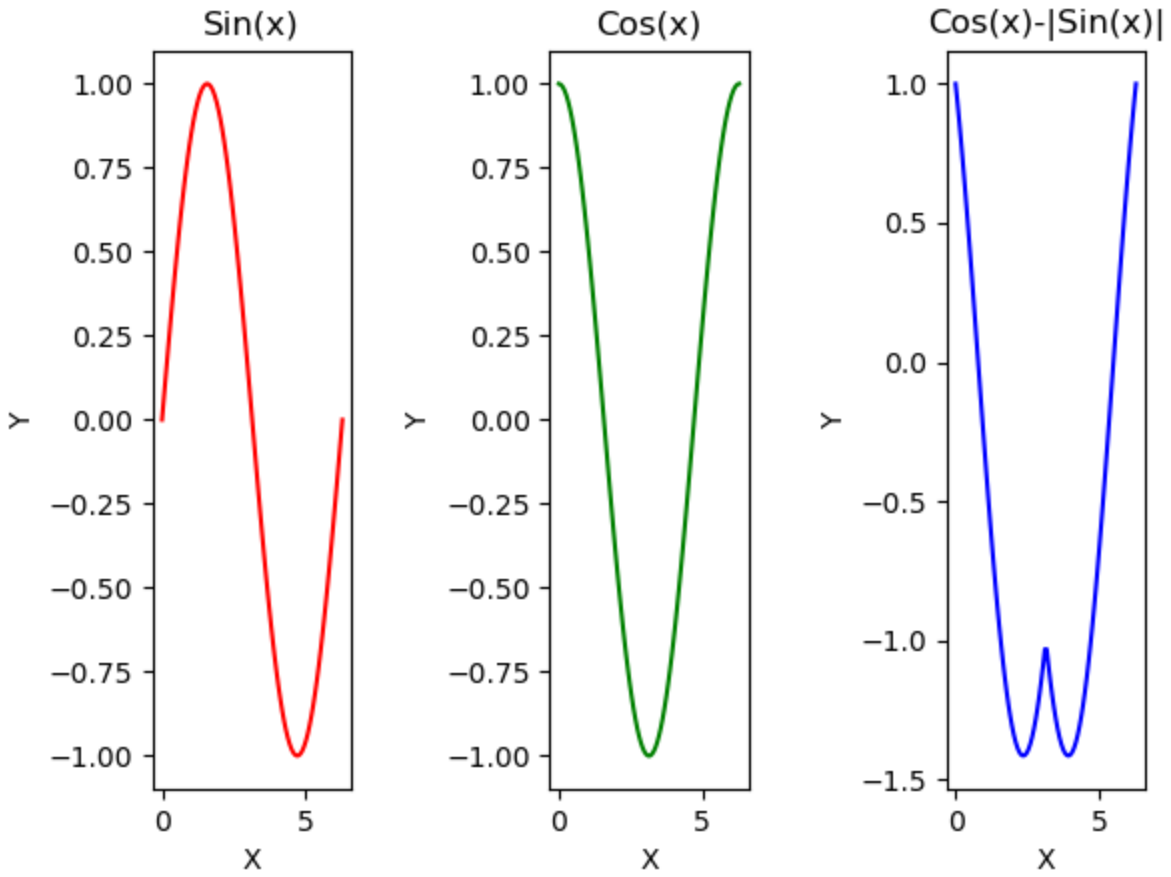


In [25]:
```
fig = None
axes = None
#Pon tu código aquí.
#Sugerencia: usa la función subplots() para crear una figura con tres subplots.
#Utitliza el diccionario gridspec_kw para indicar el valor apropiado para el atributo hs

#

#Pon tu código aquí.
#Sugerencia usa axes[] para acceder al i-ésimo eje para dibujar y poner el título.

#
```

También podemos poner las gráficas en horizontal. En este caso para aumentar el espacio en horizontal usaríamos el parámetro `gridspec_kw={'wspace':1.0}` ajustando el atributo `wspace`.

Ejercicio 10: Dibujar las función $\cos(x)$, $\sin(x)$ y $\cos(x) - |\sin(x)|$ en el intervalo x ya creado. Cada gráfica se dibuja en un eje separado, en sentido horizontal, y tendrá por título la función dibujada. Los ejes X e Y de todos los ejes tendrán como etiquetas "X", "Y" respectivamente. Dejar espacio entre los subplots para que se muestren bien los títulos y leyendas.

Intenta ajustar los parámetros para obtener una figura lo más parecida posible a la siguiente:

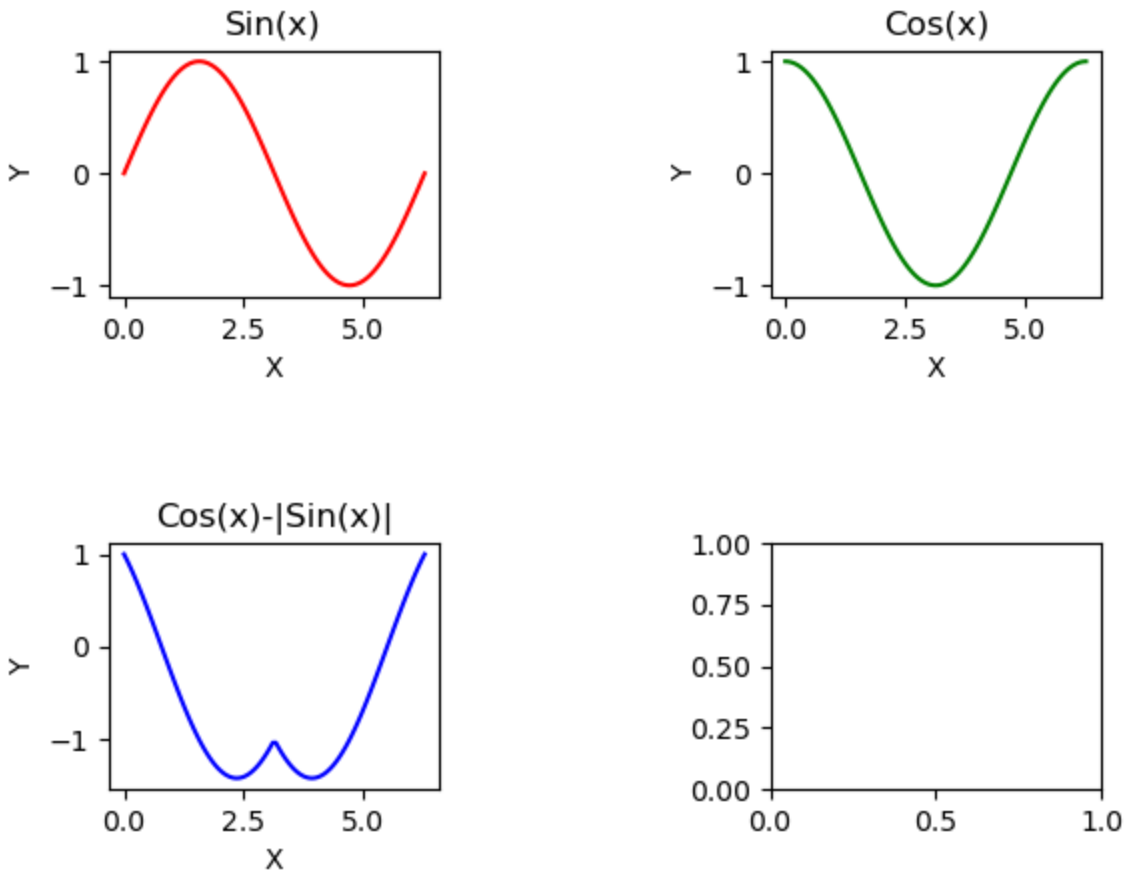


In [26]:
```
fig = None
axes = None
#Pon tu código aquí.
#Sugerencia: usa la función subplots() para crear una figura con tres subplots.
#Utitliza el diccionario gridspec_kw para indicar el valor apropiado para el atributo ws

#

#Pon tu código aquí.
#Sugerencia usa axes[] para acceder al i-ésimo eje para dibujar y poner el título.

#
```

Combinando las dos ideas anteriores también podemos formar una rejilla.

Ejercicio 11: Dibujar las función $\cos(x)$, $\sin(x)$ y $|\cos(x) - \sin(x)|$ en el intervalo x ya creado. Cada gráfica se dibuja en un eje separado, rellenando una rejilla 2x2, y tendrá por título la función dibujada. Los ejes X e Y de todos los ejes tendrán como etiquetas "X", "Y" respectivamente. Dejar espacio entre los subplots para que se muestren bien los títulos y leyendas.

Intenta ajustar los parámetros para obtener una figura lo más parecida posible a la siguiente:



In [27]:
```
fig = None
axes = None
#Pon tu código aquí.
#Sugerencia: usa la función subplots() para crear una figura con tres subplots.
#Utitliza el diccionario gridspec_kw para indicar el valor apropiado para el
#atributos hspace y wspace.

#

#Pon tu código aquí.
#Sugerencia usa axes[] para acceder al i-ésimo eje para dibujar y poner el título.

#
```

In [ ]:

In [ ]: