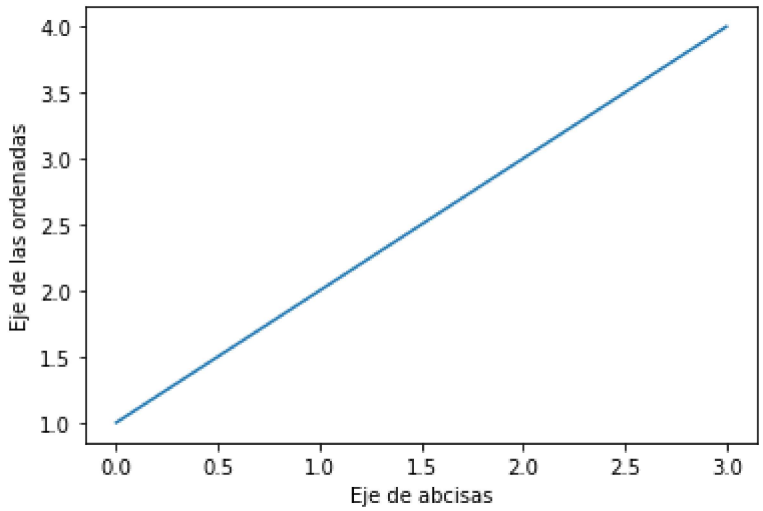
MATPLOTLIB

In [4]:
```
%matplotlib inline
```

Matplotlib tiene una sub-libreria que hace que matplot lib que es una coleccionde funciones de estilizado, la libreria es pyplot

In [5]:
```
import matplotlib.pyplot as plt
```

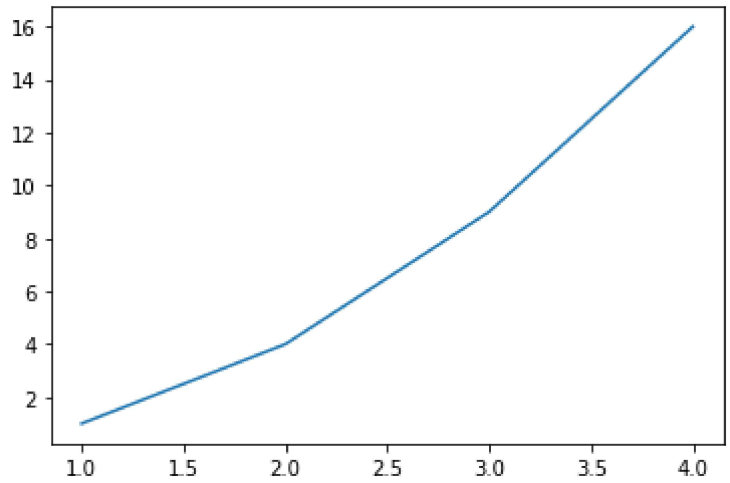
In [6]:
```
x=[1,2,3,4]
plt.plot(x)
plt.xlabel("Eje de abcisas")
plt.ylabel("Eje de las ordenadas")
plt.show()
```



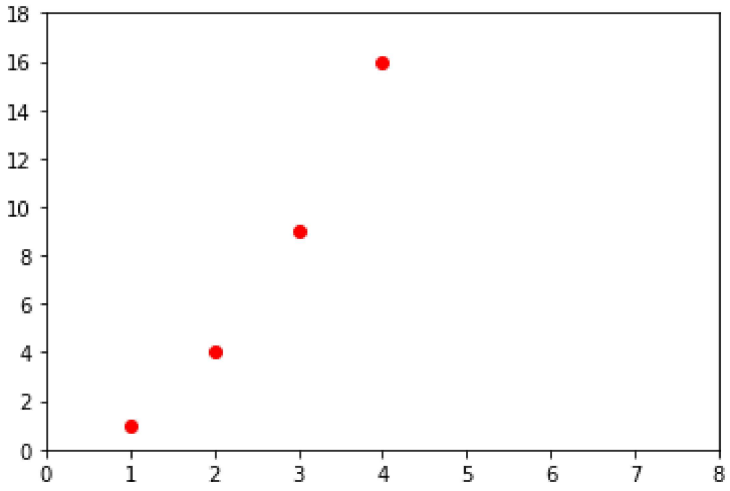
Éste plot difiere del de R por que este te crea un grafico continuo, a diferencia del de R que son puntos

In [7]:
```
x=[1,2,3,4]
y=[1,4,9,16]
plt.plot(x,y)
```

Out[7]:
```
[<matplotlib.lines.Line2D at 0x236ac86d040>]
```



In [9]:
```
plt.plot(x,y,"ro")
plt.axis([0,8,0,18])
plt.show()
```



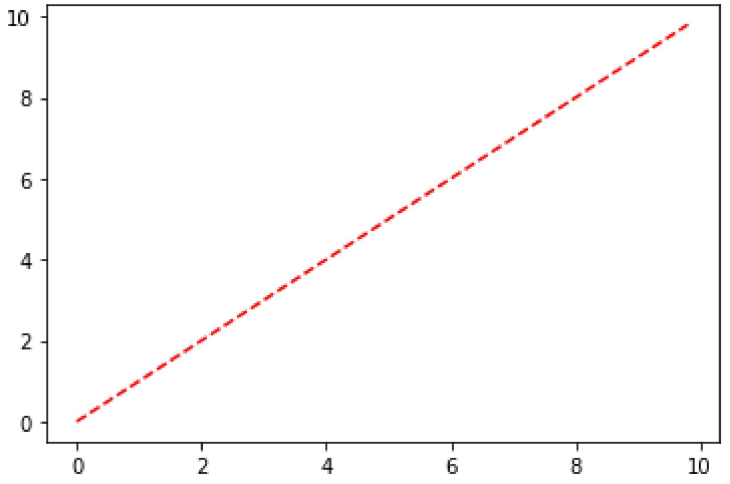
In [10]:
```
import numpy as np
```

In [12]:
```
data=np.arange(0.0,10.0,0.2)
data
```

Out[12]:
```
array([0. , 0.2, 0.4, 0.6, 0.8, 1. , 1.2, 1.4, 1.6, 1.8, 2. , 2.2, 2.4,
       2.6, 2.8, 3. , 3.2, 3.4, 3.6, 3.8, 4. , 4.2, 4.4, 4.6, 4.8, 5. ,
       5.2, 5.4, 5.6, 5.8, 6. , 6.2, 6.4, 6.6, 6.8, 7. , 7.2, 7.4, 7.6,
       7.8, 8. , 8.2, 8.4, 8.6, 8.8, 9. , 9.2, 9.4, 9.6, 9.8])
```

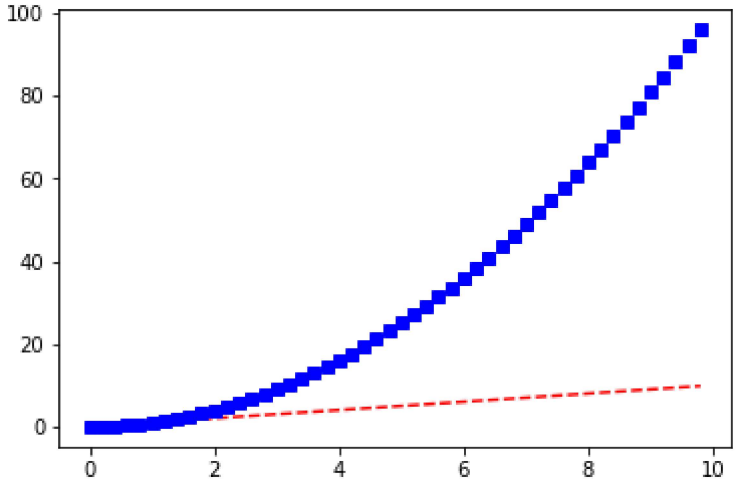
In [14]:
```
plt.plot(data,data,"r--")
```

Out[14]:
```
[<matplotlib.lines.Line2D at 0x236ae254940>]
```

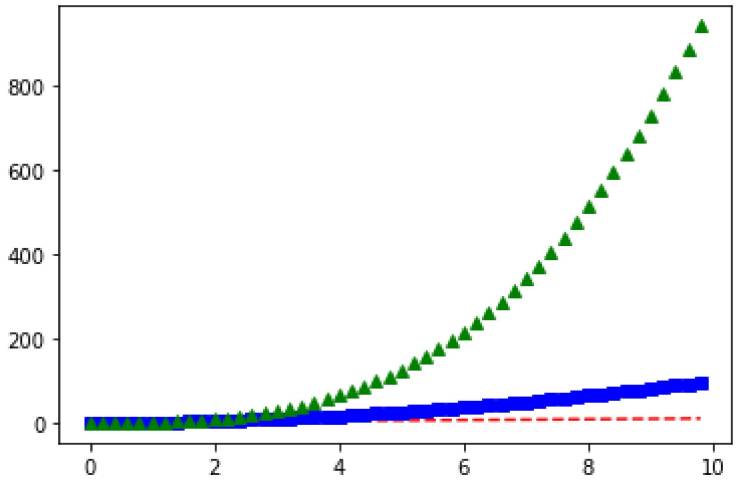


In [15]:
```
plt.plot(data,data,"r--",data,data**2,"bs")
```

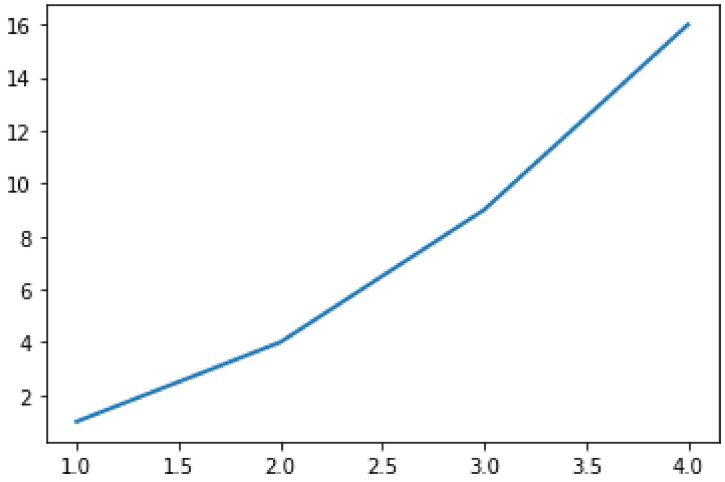
Out[15]:
```
[<matplotlib.lines.Line2D at 0x236ae5b4400>,
 <matplotlib.lines.Line2D at 0x236ae5b4250>]
```



In [17]:
```
plt.plot(data,data,"r--",data,data**2,"bs",data,data**3,"g^")
plt.show()
```

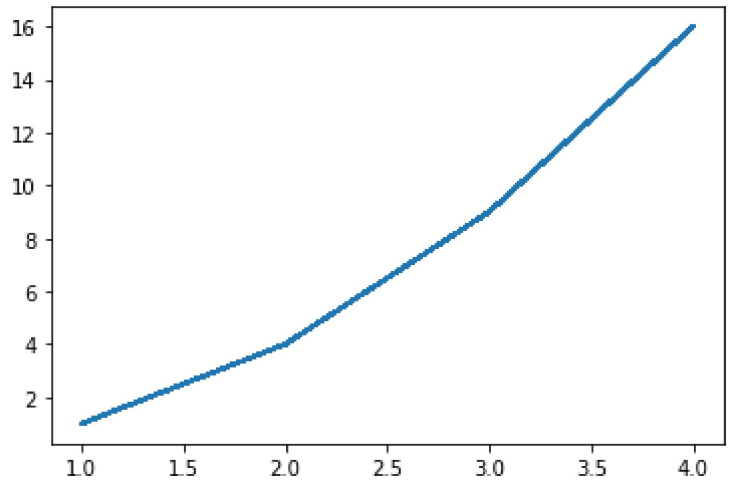


In [18]:
```
plt.plot(x,y,linewidth=2.0)
plt.show()
```



Hay diferentes parametros, primero coloca la inicial del color g,b,r, etc y despues el signo, para ver que signo quieres ver documentación de matplotlib

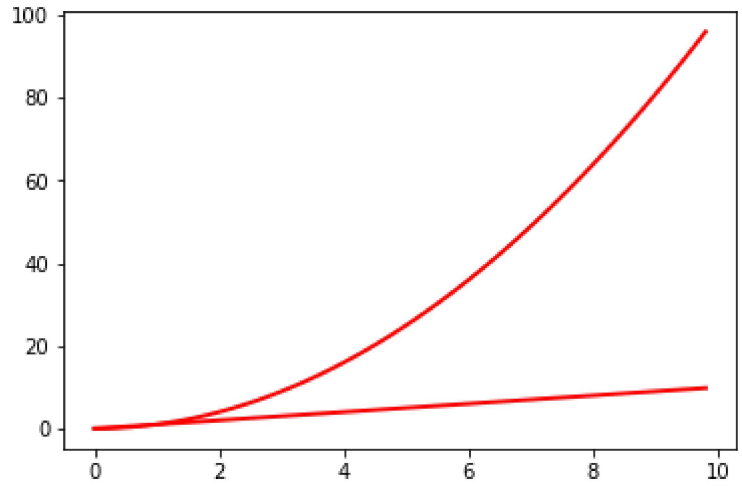
In [21]:
```
line,=plt.plot(x,y,"-")
line.set_antialiased(False)
```



Siempre es necesario colocar como tal despues de la variable donde se vaya a guardar el grafico una coma debido a que sino, el parámetro de set_antialiased no será valido, el siguiente parámetro es set parameters para ver diferentes parámetros:

In [22]:
```
lines=plt.plot(data,data,data,data**2)
plt.setp(lines,color="r",linewidth=2.0)
```

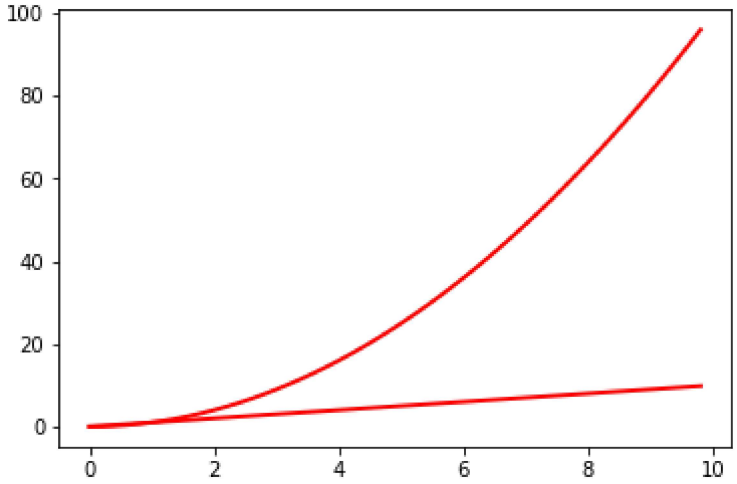
Out[22]: [None, None, None, None]



Como pudimos observar el uso de setp que se traduciría a set_propertys es la equivalencia a hacer cambios sobre todos los elementos del gráfico

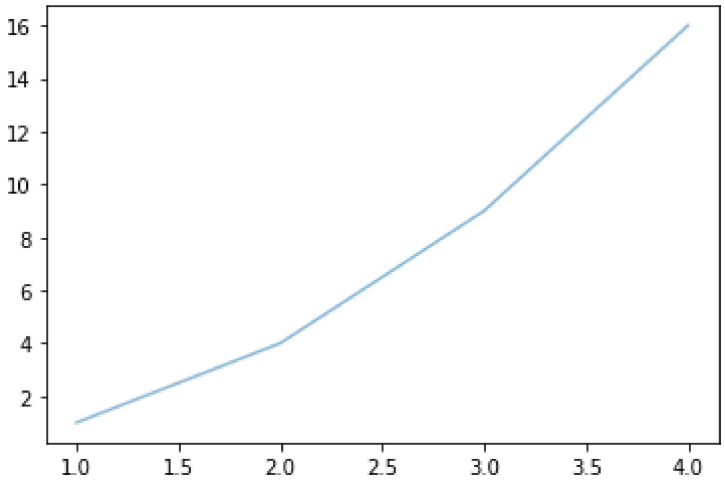
In [23]:
```
lines=plt.plot(data,data,data,data**2)
plt.setp(lines,"color","r","linewidth",2.0)
```

Out[23]: [None, None, None, None]



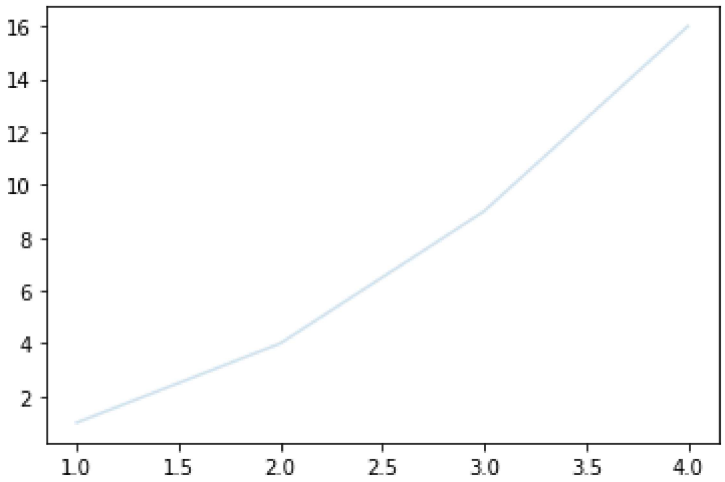
In [24]:
```
plt.plot(x,y,alpha=0.5)
```

Out[24]:
```
[<matplotlib.lines.Line2D at 0x236adfa3be0>]
```



In [26]:
```
plt.plot(x,y,alpha=0.2)
```

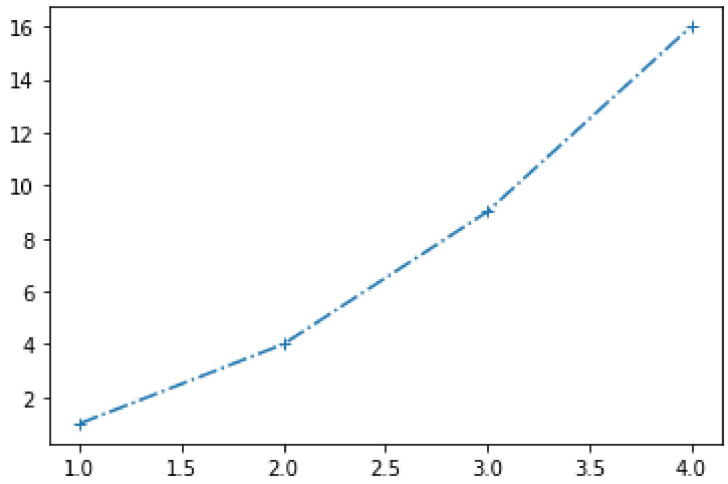
Out[26]:
```
[<matplotlib.lines.Line2D at 0x236ae3ba3d0>]
```



Alpha es de opacidad, parámetro de opacidad

In [27]:
```
plt.plot(x,y,marker="+",linestyle="-.")
```

Out[27]: [<matplotlib.lines.Line2D at 0x236ae29abe0>]



In [28]:
```
plt.setp(lines)
```

```
  agg_filter: a filter function, which takes a (m, n, 3) float array and a dpi value, an
d returns a (m, n, 3) array
  alpha: float or None
  animated: bool
  antialiased or aa: bool
  clip_box: `.Bbox`
  clip_on: bool
  clip_path: Patch or (Path, Transform) or None
  color or c: color
  contains: unknown
  dash_capstyle: {'butt', 'round', 'projecting'}
  dash_joinstyle: {'miter', 'round', 'bevel'}
  dashes: sequence of floats (on/off ink in points) or (None, None)
  data: (2, N) array or two 1D arrays
  drawstyle or ds: {'default', 'steps', 'steps-pre', 'steps-mid', 'steps-post'}, defaul
t: 'default'
  figure: `.Figure`
  fillstyle: {'full', 'left', 'right', 'bottom', 'top', 'none'}
  gid: str
  in_layout: bool
  label: object
  linestyle or ls: {'-', '--', '-.', ':', '', (offset, on-off-seq), ...}
  linewidth or lw: float
  marker: marker style string, `~.path.Path` or `~.markers.MarkerStyle`
  markeredgecolor or mec: color
  markeredgewidth or mew: float
  markerfacecolor or mfc: color
  markerfacecoloralt or mfcalt: color
  markersize or ms: float
  markevery: None or int or (int, int) or slice or List[int] or float or (float, float)
or List[bool]
  path_effects: `.AbstractPathEffect`
  picker: unknown
  pickradius: float
  rasterized: bool or None
  sketch_params: (scale: float, length: float, randomness: float)
  snap: bool or None
  solid_capstyle: {'butt', 'round', 'projecting'}
  solid_joinstyle: {'miter', 'round', 'bevel'}
  transform: `matplotlib.transforms.Transform`
  url: str
  visible: bool
  xdata: 1D array
```

```
ydata: 1D array
zorder: float
```

Se le tiene que pasar como parámetro el grafico que querramos agregarle estilos o modificar algun parámetro

MÚLTIPLES GRÁFICOS EN UNA MISMA FIGURA

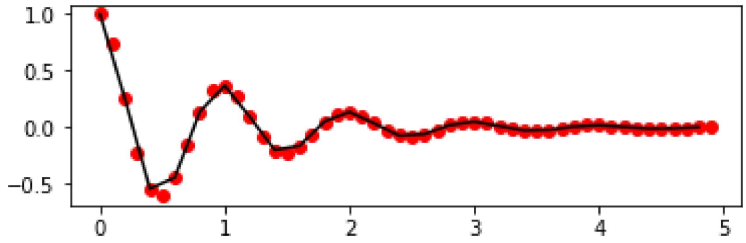
Una librería complementaria para graficar es la siguiente:

In [29]:
```
def f(x):
    return np.exp(-x)*np.cos(2*np.pi*x)
```

In [30]:
```
x1=np.arange(0,5,0.1)
x2=np.arange(0,5,0.2)
```

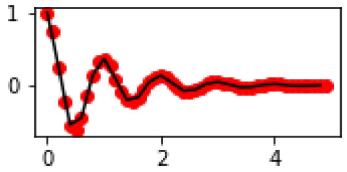
In [36]:
```
plt.figure(1)#para pintar sobre la misma figura
plt.subplot(211)#doble de ancho para el primer grafico
plt.plot(x1,f(x1),"ro",x2,f(x2),"k")
```

Out[36]:
```
[<matplotlib.lines.Line2D at 0x236ae8f3550>,
 <matplotlib.lines.Line2D at 0x236ae8f3580>]
```



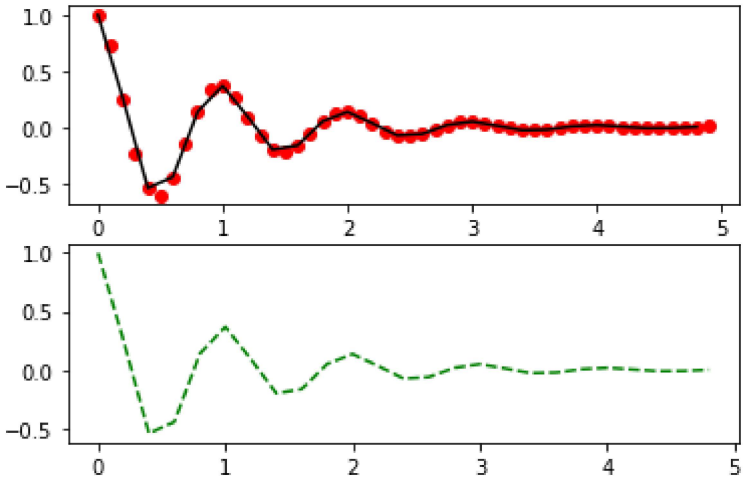
In [34]:
```
plt.figure(1)#para pintar sobre la misma figura
plt.subplot(321)#doble de ancho para el primer grafico
plt.plot(x1,f(x1),"ro",x2,f(x2),"k")
```

Out[34]:
```
[<matplotlib.lines.Line2D at 0x236ae805970>,
 <matplotlib.lines.Line2D at 0x236ae8059a0>]
```



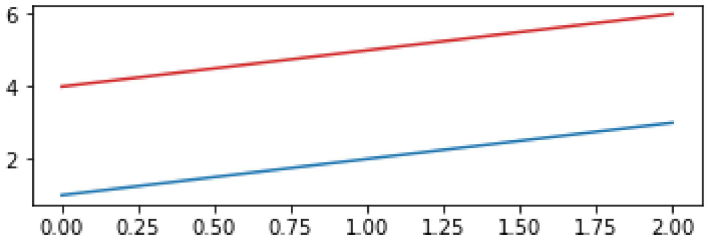
In [35]:
```
plt.figure(1)#para pintar sobre la misma figura
plt.subplot(211)#doble de ancho para el primer grafico
plt.plot(x1,f(x1),"ro",x2,f(x2),"k")
plt.subplot(212)#2 a x por cada altura y
plt.plot(x2,f(x2),"g--")
```

Out[35]:
```
[<matplotlib.lines.Line2D at 0x236ae884f40>]
```



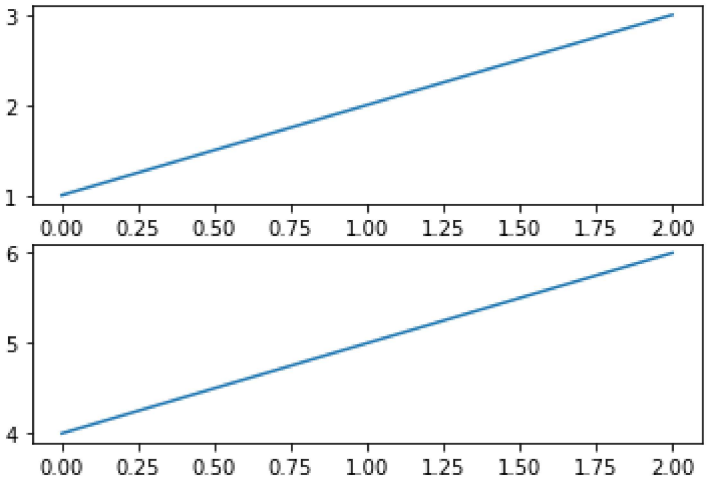
In [37]:
```
plt.figure(1)
plt.subplot(2,1,1)
plt.plot([1,2,3])
plt.plot(2,1,2)
plt.plot([4,5,6])
```

Out[37]: [<matplotlib.lines.Line2D at 0x236afa2ca60>]



In [38]:
```
plt.figure(1)
plt.subplot(2,1,1)
plt.plot([1,2,3])
plt.subplot(2,1,2)
plt.plot([4,5,6])
```

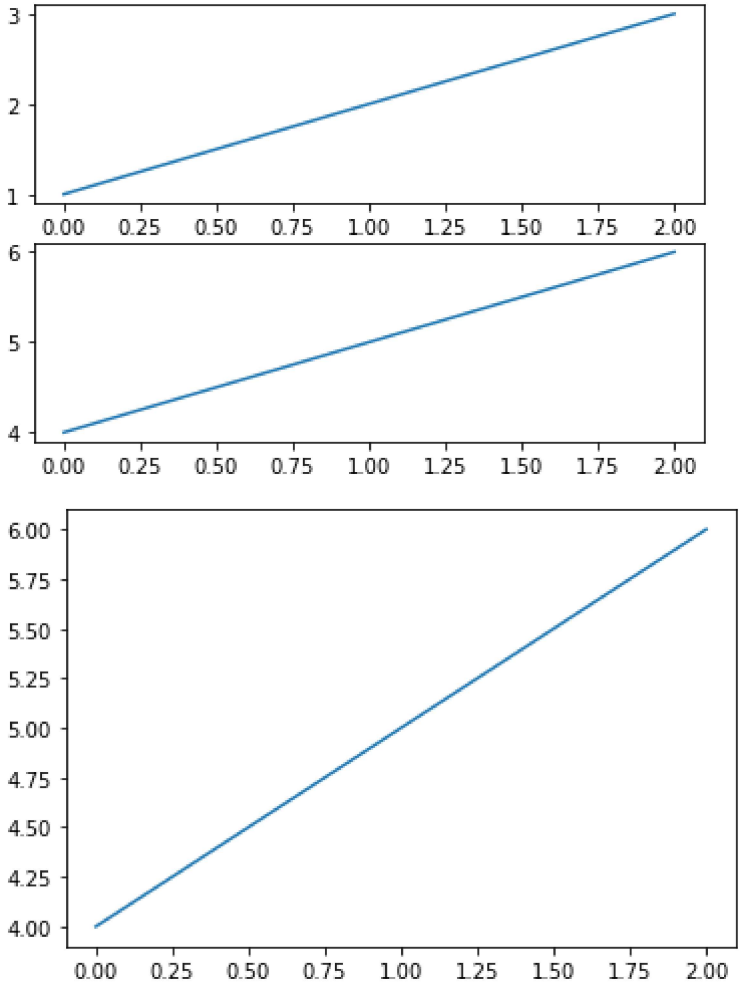
Out[38]: [<matplotlib.lines.Line2D at 0x236afabc1c0>]



In [39]:
```
plt.figure(1)
plt.subplot(2,1,1)
plt.plot([1,2,3])
```

```
plt.subplot(2,1,2)
plt.plot([4,5,6])

plt.figure(2)
plt.plot([4,5,6])
```

Out[39]: [<matplotlib.lines.Line2D at 0x236afb93820>]

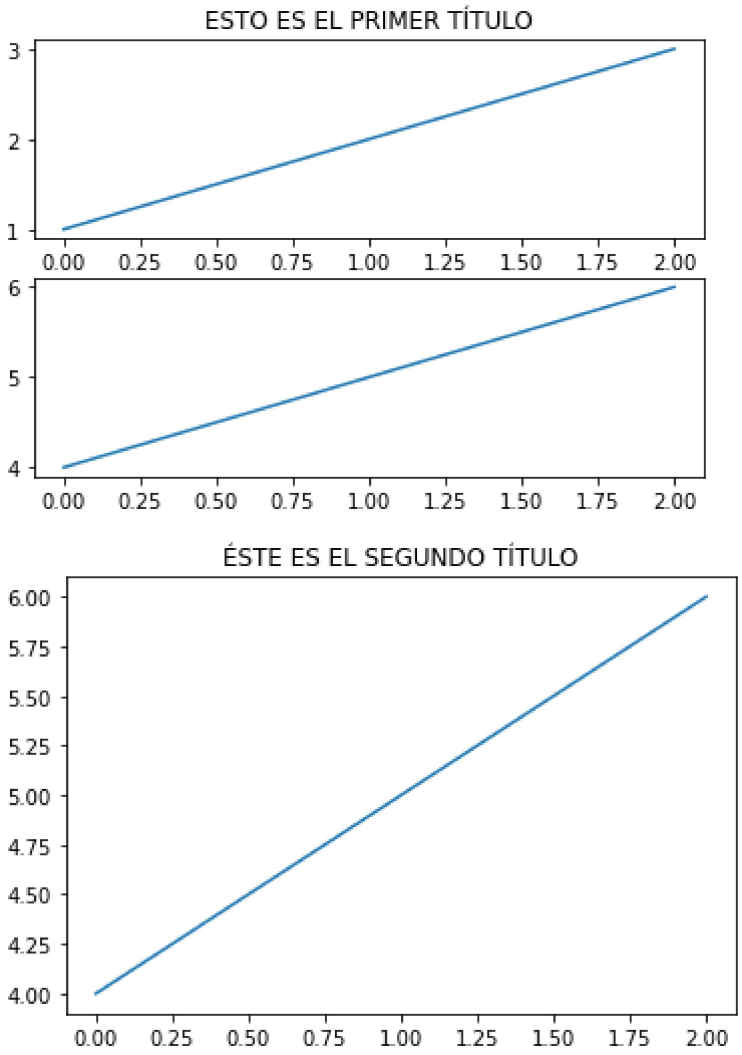


In [46]:
```
plt.figure(1)
plt.subplot(2,1,1)
plt.plot([1,2,3])
plt.subplot(2,1,2)
plt.plot([4,5,6])

plt.figure(2)
plt.plot([4,5,6])

plt.figure(1)#regresa a la primera figura
plt.subplot(2,1,1)
plt.title("ESTO ES EL PRIMER TÍTULO")

plt.figure(2)
plt.title("ÉSTE ES EL SEGUNDO TÍTULO")
```

```
<ipython-input-46-f269048de19e>:11: MatplotlibDeprecationWarning: Adding an axes using t
he same arguments as a previous axes currently reuses the earlier instance.  In a future
version, a new instance will always be created and returned.  Meanwhile, this warning ca
n be suppressed, and the future behavior ensured, by passing a unique label to each axes
```

```
instance.
  plt.subplot(2,1,1)
```

Out[46]: `Text(0.5, 1.0, 'ÉSTE ES EL SEGUNDO TÍTULO')`



FIGURAS Y EJES EN MATPLOTLIB

Que diferencia una figura de un eje, la figura es la forma basica de representación, cuando hablamos de figura te puedes imaginar un cuadrado, en donde se posibilita la aparicion de una figura. Por otra parte eje se asume como la parte interna donde se incluyen el grafico, la escala, etc. Sino quieres que se sobreponga un grafico sobre otro basta con usar plt.hold() que hace que cada grafico borre el anterior y no se sobre ponga del otro

RANGOS DE DIBUJO

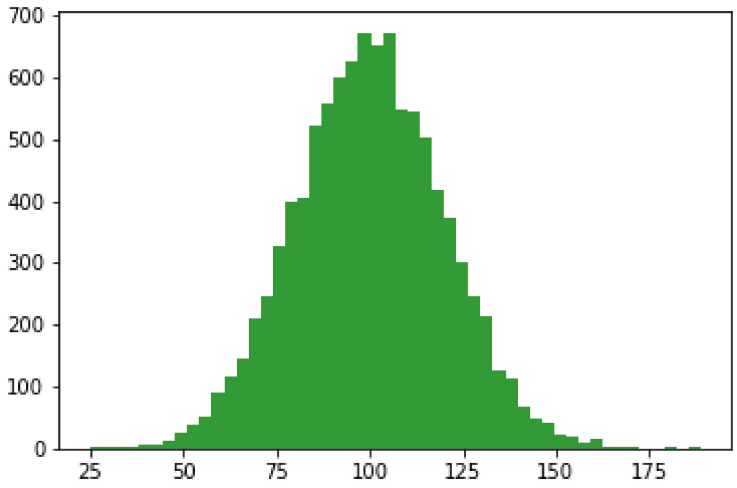
subplot(2,3) 2 filas 3 columnas

TEXTO EN MATPLOTLIB

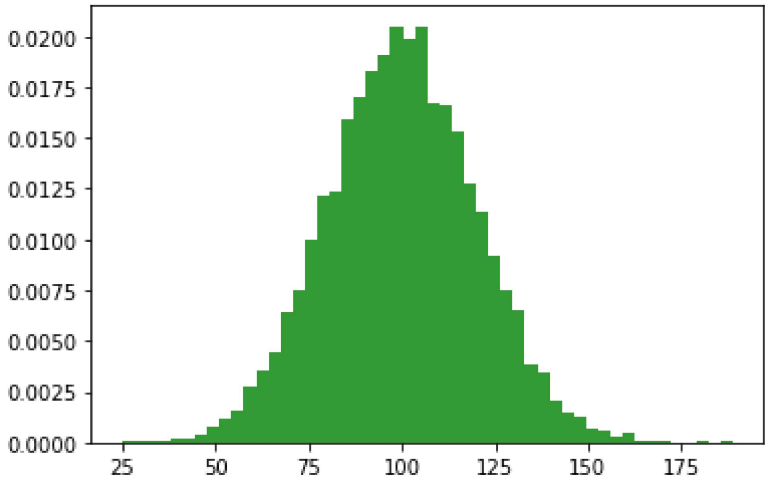
In [47]:
```
mu=100
sigma=20
x=mu+sigma*np.random.randn(10000)
```

n es el numero de elementos del histogramaa,bins la cantidad de divisiones que hace d elos datos,patchhes conjuntos de datos respectivamente

In [49]:
```
n,bins,patches=plt.hist(x,50,facecolor="g",alpha=0.8)
```

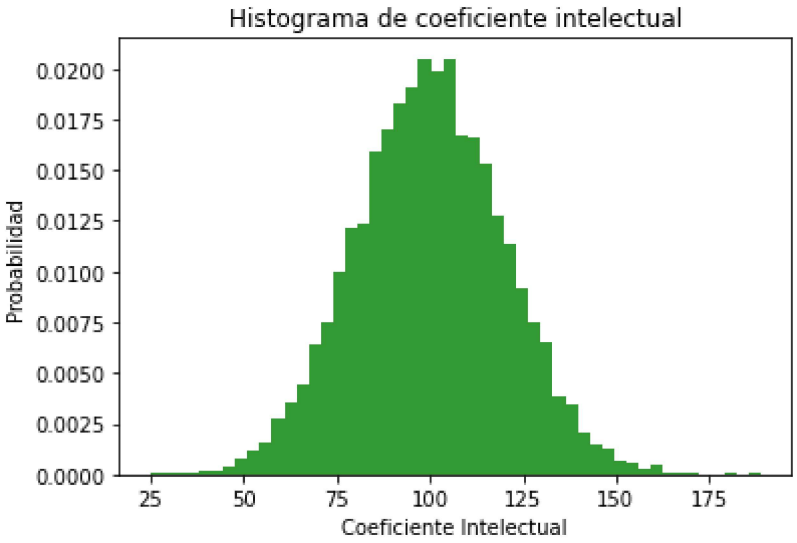


In [53]:
```
n,bins,patches=plt.hist(x,50,facecolor="g",alpha=0.8,density=True)
```

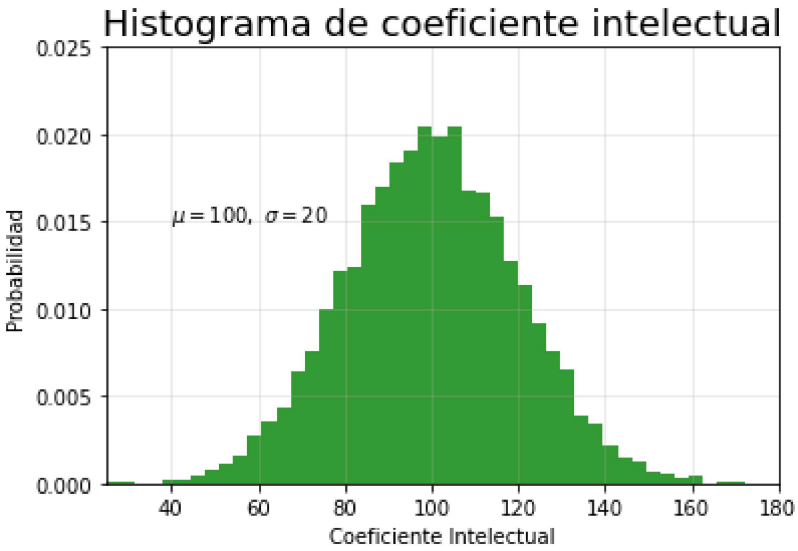


In [54]:
```
n,bins,patches=plt.hist(x,50,facecolor="g",alpha=0.8,density=True)
plt.xlabel("Coeficiente Intelectual")
plt.ylabel("Probabilidad")
plt.title("Histograma de coeficiente intelectual")
```

Out[54]:
```
Text(0.5, 1.0, 'Histograma de coeficiente intelectual')
```

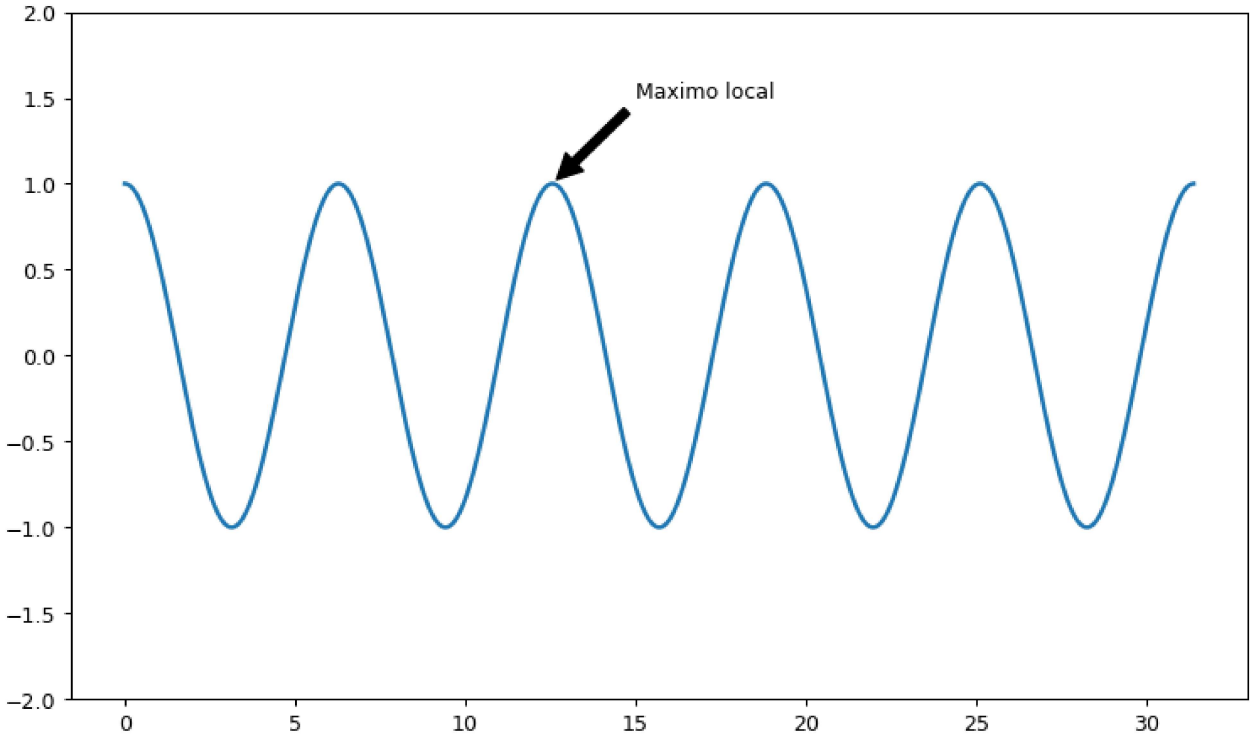


In [62]:
```
n,bins,patches=plt.hist(x,50,facecolor="g",alpha=0.8,density=True)
plt.xlabel("Coeficiente Intelectual")
plt.ylabel("Probabilidad")
plt.title("Histograma de coeficiente intelectual",fontsize=18)
plt.text(40,0.015,r'$\mu=100,\ \sigma=20$')#Coordenada 40,0.015
#para colocar texto de LaTeX es necesario colocar una r antes de la cadena de texto
plt.axis([25,180, 0,0.025])#x de 25 a 180 y de 0 a 0.025
plt.grid(True,alpha=0.3)

plt.show()
```



In [67]:
```
#como colocar una flechita con texto anotado
plt.figure(figsize=(10,6),dpi=90)#se define el tamaño del dibujo en pulgadas
plt.subplot(1,1,1)#1 fila 1 columna 1 er gráfico
x=np.arange(0,10*np.pi,0.01)
y=np.cos(x)
plt.plot(x,y,lw=2.0)
plt.annotate("Maximo local",xy=(4*np.pi,1),xytext=(15,1.5),arrowprops=dict(facecolor="b
plt.ylim(-2,2)
```

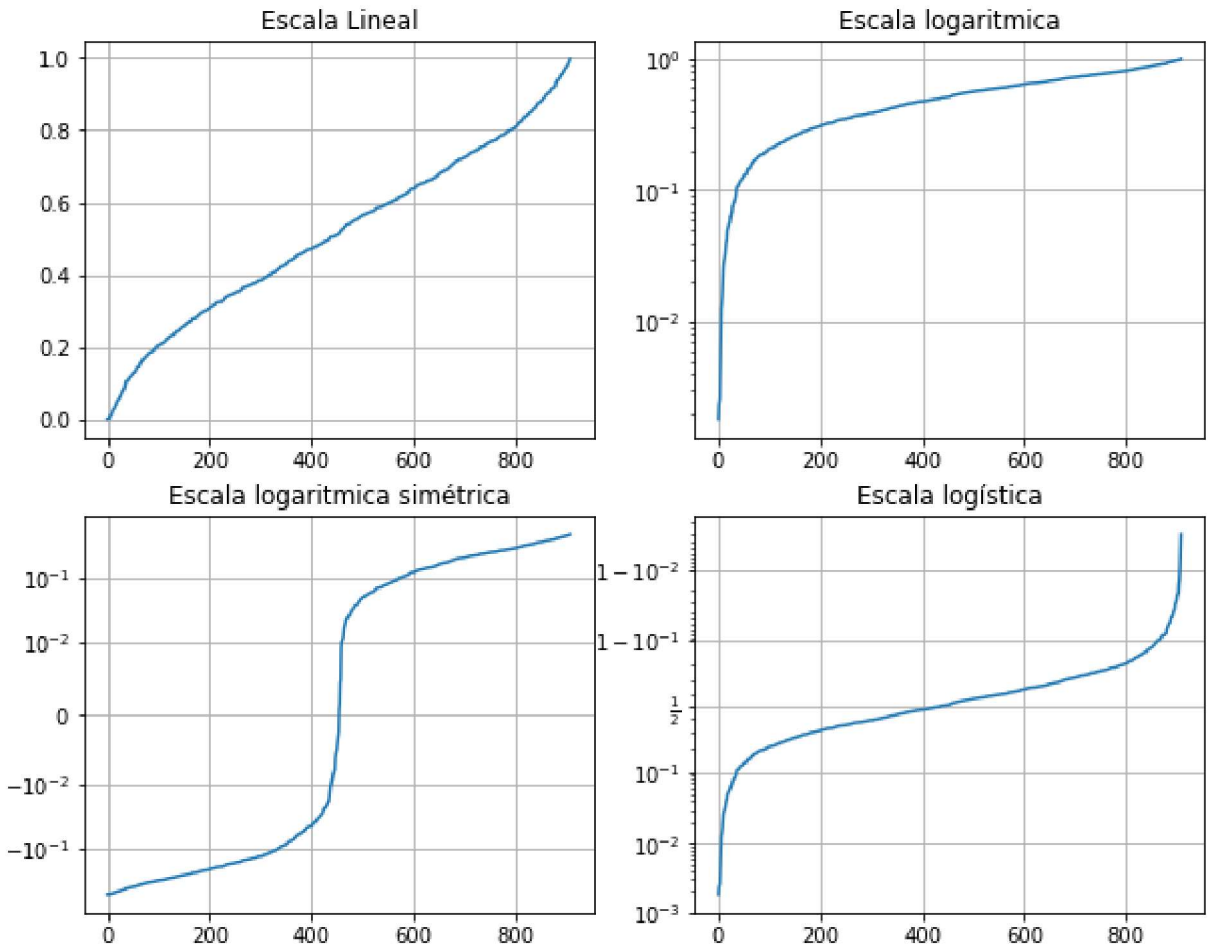
Out[67]: (-2.0, 2.0)



In [70]:
```
mu=0.5
sd=0.3
y=mu+sd*np.random.randn(1000)
y=y[(y>0)&(y<1)]
y.sort()
x=np.arange(len(y))
```

In [83]:
```
plt.figure(figsize=(10,8))

plt.subplot(2,2,1)
plt.plot(x,y)
plt.yscale("linear")
plt.xscale("linear")
plt.title("Escala Lineal")
plt.grid(True)

plt.subplot(2,2,2)
plt.plot(x,y)
plt.title("Escala logaritmica")
plt.yscale("log")
plt.grid(True)

plt.subplot(2,2,3)
plt.plot(x,y-y.mean())#eje logaritmico simetrico tiene sentido cuando se pinta x contra
#para obtener el desplazzamiento vertical
plt.yscale("symlog",linthresh=0.01)
plt.title("Escala logaritmica simétrica")
plt.grid(True)



plt.subplot(2,2,4)
plt.plot(x,y)
plt.yscale("logit")
plt.title("Escala logística")
```

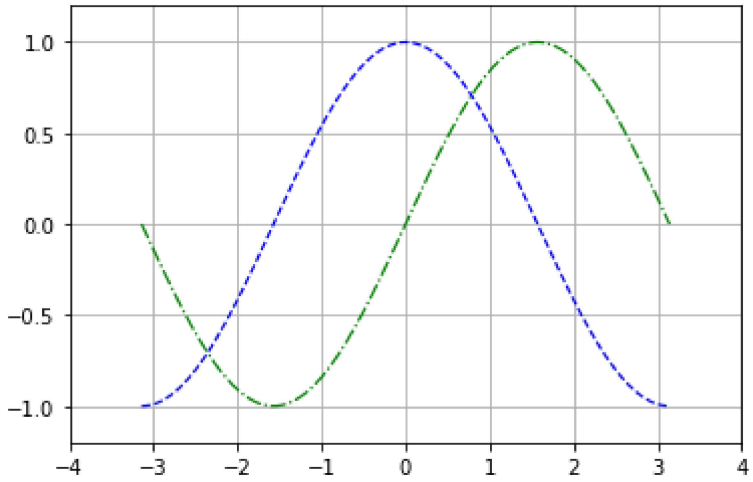
```
plt.grid(True)
plt.show()
```



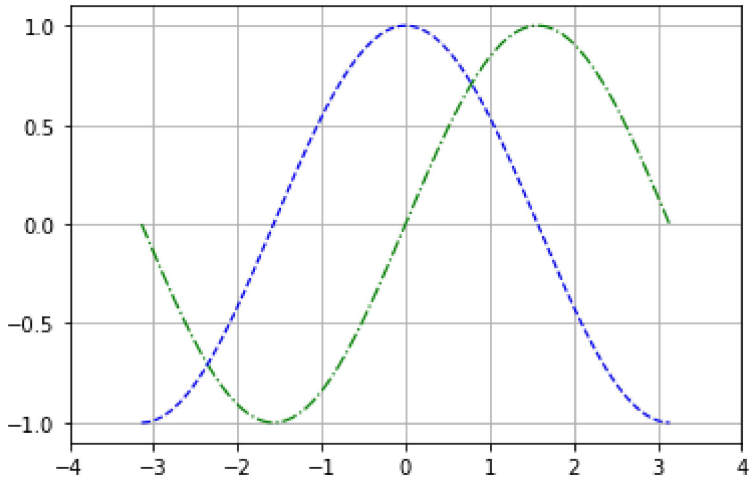
CAMBIOS EN LOS EJES DE DIBUJO

Sirve para colocar los ejes en la parte central

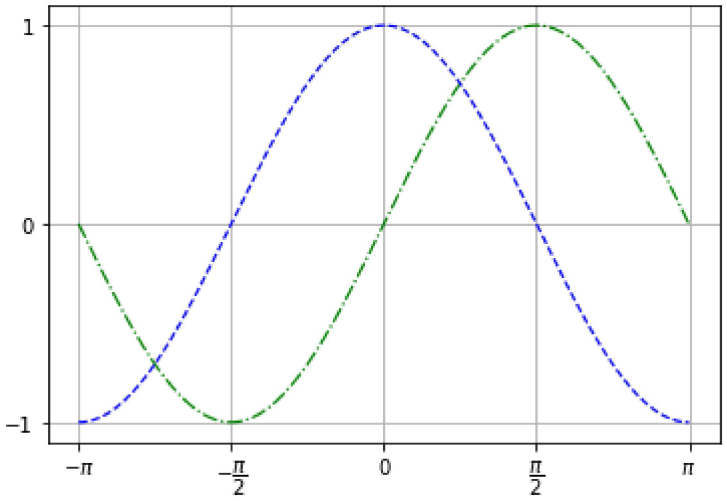
In [92]:
```
x=np.linspace(-np.pi,np.pi,256,endpoint=True)#rango de -pi a pi,256 valores del seno
S,C=np.sin(x),np.cos(x)#Se asignan a las dos variabbles las funciones, S la primera y C
plt.plot(x,S,color="green",linewidth=1.2,linestyle="-.")
plt.plot(x,C,color="blue",linewidth=1.2,linestyle="--")
plt.xlim(-4,4)
plt.ylim(-1.2,1.2)
plt.xticks(np.linspace(-4,4,9,endpoint=True))#de -a a 4, has 9 divisiones y marca los p
plt.yticks(np.linspace(-1,1,5,endpoint=True))
plt.grid(True)
plt.show()
```



In [93]:
```
x=np.linspace(-np.pi,np.pi,256,endpoint=True)#rango de -pi a pi,256 valores del seno
S,C=np.sin(x),np.cos(x)#Se asignan a las dos variabbles las funciones, S la primera y C
plt.plot(x,S,color="green",linewidth=1.2,linestyle="-.")
plt.plot(x,C,color="blue",linewidth=1.2,linestyle="--")
plt.xlim(-4,4)
plt.ylim(S.min()*1.1,S.max()*1.1)
plt.xticks(np.linspace(-4,4,9,endpoint=True))#de -a a 4, has 9 divisiones y marca los p
plt.yticks(np.linspace(-1,1,5,endpoint=True))
plt.grid(True)
plt.show()
```

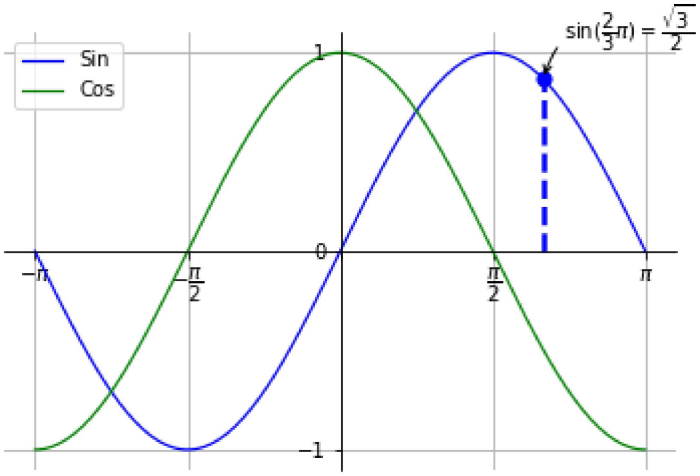


In [101…
```
x=np.linspace(-np.pi,np.pi,256,endpoint=True)#rango de -pi a pi,256 valores del seno
S,C=np.sin(x),np.cos(x)#Se asignan a las dos variabbles las funciones, S la primera y C
plt.plot(x,S,color="green",linewidth=1.2,linestyle="-.")
plt.plot(x,C,color="blue",linewidth=1.2,linestyle="--")
plt.xlim(x.min()*1.1,x.max()*1.1)
plt.ylim(S.min()*1.1,S.max()*1.1)
plt.xticks([-np.pi,-np.pi/2,0,np.pi/2,np.pi],[r'$-\pi$',r'$-\dfrac{\pi}{2}$',r'0',r'$\d
plt.yticks(np.linspace(-1,1,3,endpoint=True))
plt.grid(True)
plt.show()
```



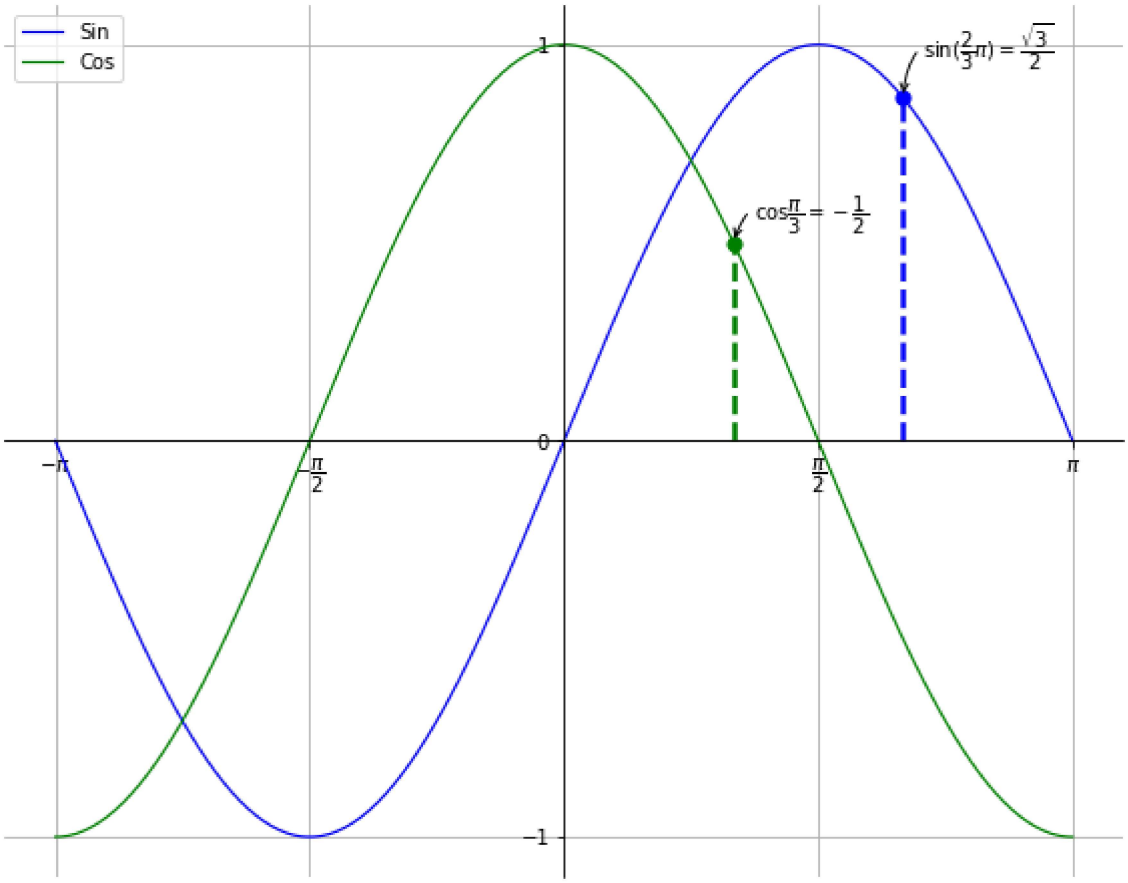
In [121…

```
x=np.linspace(-np.pi,np.pi,256,endpoint=True)#rango de -pi a pi,256 valores del seno
S,C=np.sin(x),np.cos(x)#Se asignan a las dos variabbles las funciones, S la primera y C
plt.plot(x,S,color="blue",linewidth=1.2,linestyle="-",label="Sin")
plt.plot(x,C,color="green",linewidth=1.2,linestyle="-",label="Cos")
plt.xlim(x.min()*1.1,x.max()*1.1)
plt.ylim(S.min()*1.1,S.max()*1.1)
plt.xticks([-np.pi,-np.pi/2,0,np.pi/2,np.pi],[r'$-\pi$',r'$-\dfrac{\pi}{2}$',' ',r'$\df
plt.yticks(np.linspace(-1,1,3,endpoint=True))
plt.grid(True)

ax=plt.gca()#Da acceso a los ejes del grafico y darnos acceso
ax.spines['right'].set_color('none')
ax.spines['top'].set_color('none')
ax.xaxis.set_ticks_position('bottom')
ax.spines['bottom'].set_position(('data',0))
ax.yaxis.set_ticks_position('left')
ax.spines['left'].set_position(('data',0))

plt.legend(loc='upper left')
x0=2*np.pi/3
plt.plot([x0,x0],[0,np.sin(x0)],color="blue",linewidth=2.5,linestyle="--")
plt.scatter([x0,],[np.sin(x0),],50,color="blue")
plt.annotate(r'$\sin(\dfrac{2}{3}\pi)=\dfrac{\sqrt{3}}{2}$',xy=(x0,np.sin(x0)),xycoords
             xytext=(+10,+20),textcoords="offset points",
            arrowprops=dict(arrowstyle="->",connectionstyle="arc3,rad=0.5"))#10 puntos

plt.show()
```



In [141…

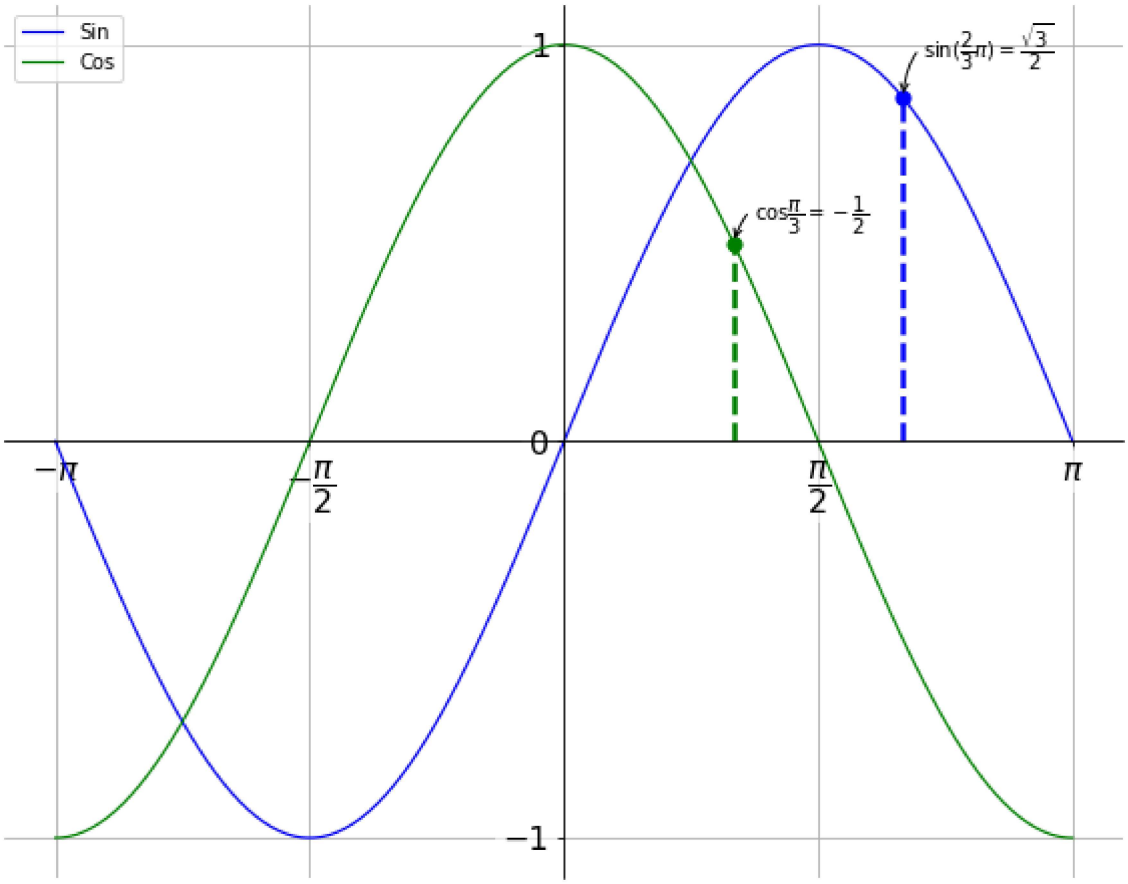
```
plt.figure(figsize=(10,8))
x=np.linspace(-np.pi,np.pi,256,endpoint=True)#rango de -pi a pi,256 valores del seno
S,C=np.sin(x),np.cos(x)#Se asignan a las dos variabbles las funciones, S la primera y C
plt.plot(x,S,color="blue",linewidth=1.2,linestyle="-",label="Sin")
plt.plot(x,C,color="green",linewidth=1.2,linestyle="-",label="Cos")
plt.xlim(x.min()*1.1,x.max()*1.1)
plt.ylim(S.min()*1.1,S.max()*1.1)
plt.xticks([-np.pi,-np.pi/2,0,np.pi/2,np.pi],[r'$-\pi$',r'$-\dfrac{\pi}{2}$',' ',r'$\df
plt.yticks(np.linspace(-1,1,3,endpoint=True))
plt.grid(True)

ax=plt.gca()#Da acceso a los ejes del grafico y darnos acceso
ax.spines['right'].set_color('none')
ax.spines['top'].set_color('none')
ax.xaxis.set_ticks_position('bottom')
ax.spines['bottom'].set_position(('data',0))
ax.yaxis.set_ticks_position('left')
ax.spines['left'].set_position(('data',0))
x1=np.pi/3
plt.plot([x1,x1],[0,np.cos(x1)],color="green",linewidth=2.5,linestyle="--")#genera la l
plt.scatter([x1,],[np.cos(x1)],50,color="green")#genera el punto
plt.annotate(r'$\cos{\dfrac{\pi}{3}}=-\dfrac{1}{2}$',
             xy=(x1,np.cos(x1)),
            xycoords="data",
            xytext=(+10,+10),
             textcoords="offset points",
            arrowprops=dict(arrowstyle="->",connectionstyle="arc3,rad=0.7"))

plt.legend(loc='upper left')
x0=2*np.pi/3
plt.plot([x0,x0],[0,np.sin(x0)],color="blue",linewidth=2.5,linestyle="--")
plt.scatter([x0,],[np.sin(x0),],50,color="blue")
plt.annotate(r'$\sin(\dfrac{2}{3}\pi)=\dfrac{\sqrt{3}}{2}$',xy=(x0,np.sin(x0)),xycoords
             xytext=(+10,+20),textcoords="offset points",
            arrowprops=dict(arrowstyle="->",connectionstyle="arc3,rad=0.7"))#10 puntos

plt.show()
```



In [146…

```
plt.figure(figsize=(10,8))
x=np.linspace(-np.pi,np.pi,256,endpoint=True)#rango de -pi a pi,256 valores del seno
S,C=np.sin(x),np.cos(x)#Se asignan a las dos variabbles las funciones, S la primera y C
plt.plot(x,S,color="blue",linewidth=1.2,linestyle="-",label="Sin")
plt.plot(x,C,color="green",linewidth=1.2,linestyle="-",label="Cos")
plt.xlim(x.min()*1.1,x.max()*1.1)
plt.ylim(S.min()*1.1,S.max()*1.1)
plt.xticks([-np.pi,-np.pi/2,0,np.pi/2,np.pi],[r'$-\pi$',r'$-\dfrac{\pi}{2}$',' ',r'$\df
plt.yticks(np.linspace(-1,1,3,endpoint=True))
plt.grid(True)

ax=plt.gca()#Da acceso a los ejes del grafico y darnos acceso
ax.spines['right'].set_color('none')
ax.spines['top'].set_color('none')
ax.xaxis.set_ticks_position('bottom')
ax.spines['bottom'].set_position(('data',0))
ax.yaxis.set_ticks_position('left')
ax.spines['left'].set_position(('data',0))
x1=np.pi/3
plt.plot([x1,x1],[0,np.cos(x1)],color="green",linewidth=2.5,linestyle="--")#genera la l
plt.scatter([x1,],[np.cos(x1)],50,color="green")#genera el punto
plt.annotate(r'$\cos{\dfrac{\pi}{3}}=-\dfrac{1}{2}$',
             xy=(x1,np.cos(x1)),
            xycoords="data",
            xytext=(+10,+10),
             textcoords="offset points",
            arrowprops=dict(arrowstyle="->",connectionstyle="arc3,rad=0.7"))

plt.legend(loc='upper left')
x0=2*np.pi/3
plt.plot([x0,x0],[0,np.sin(x0)],color="blue",linewidth=2.5,linestyle="--")
```

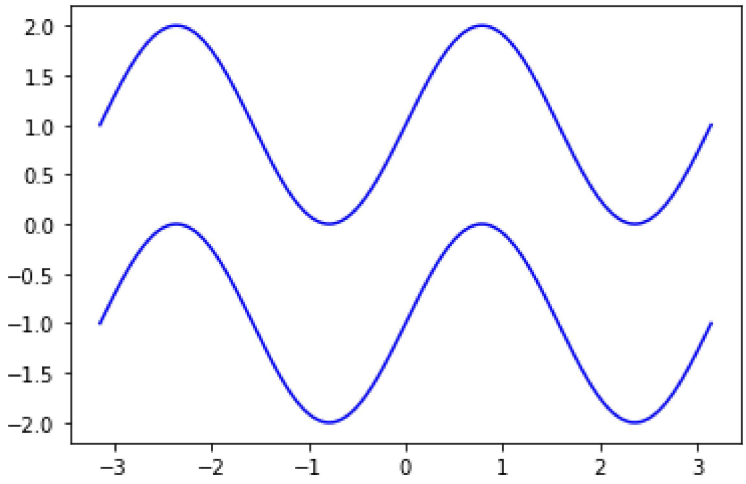
```
plt.scatter([x0,],[np.sin(x0),],50,color="blue")
plt.annotate(r'$\sin(\dfrac{2}{3}\pi)=\dfrac{\sqrt{3}}{2}$',xy=(x0,np.sin(x0)),xycoords
             xytext=(+10,+20),textcoords="offset points",
            arrowprops=dict(arrowstyle="->",connectionstyle="arc3,rad=0.7"))#10 puntos

for label in ax.get_xticklabels()+ax.get_yticklabels():
    label.set_fontsize(16)
    label.set_bbox(dict(facecolor="white",edgecolor="None",alpha=0.6))

plt.show()
```



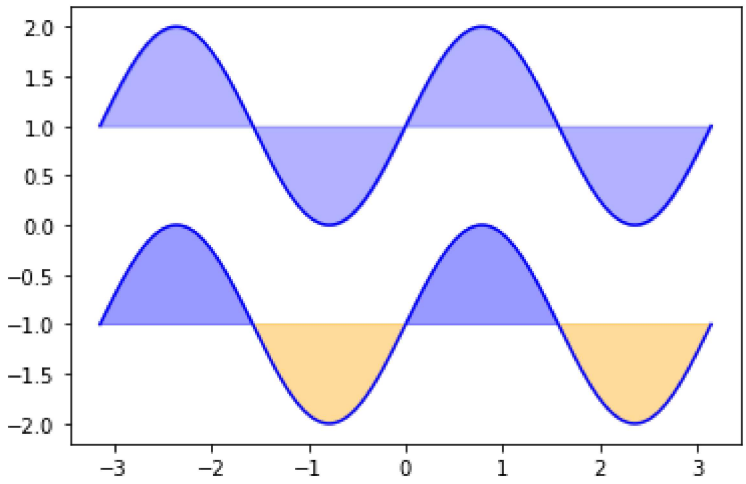
EJERCICIO 1-ORIGINAL

In [221…

```
n = 256
X = np.linspace(-np.pi, np.pi, n, endpoint=True)
Y = np.sin(2 * X)
plt.plot(X, Y + 1, color='blue', alpha=1.00)
plt.plot(X, Y - 1, color='blue', alpha=1.00)

plt.show()
```

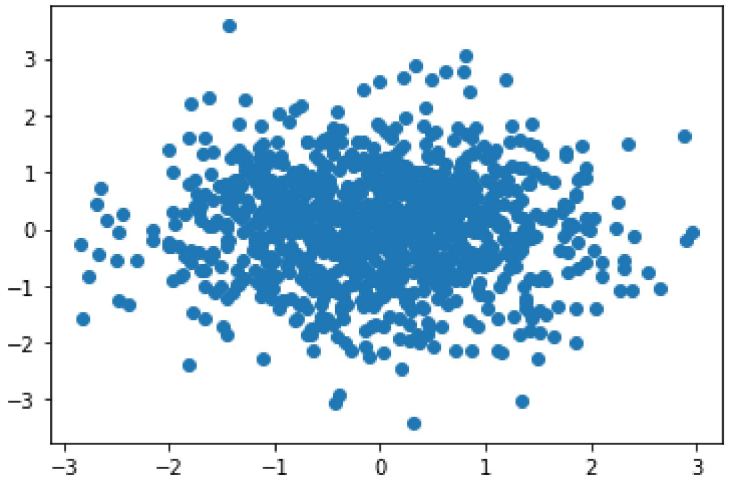


In [230…
```
#RESUELTO
n = 256
X = np.linspace(-np.pi, np.pi, n, endpoint=True)
Y = np.sin(2 * X)
plt.plot(X, Y + 1, color='blue', alpha=1.00)
ax=plt.gca()
ax.fill_between(X,Y+1,1,color="blue",alpha=0.3)
plt.plot(X, Y - 1, color='blue', alpha=1.00)
plt.fill_between(X,Y-1,-1,(Y-1)<-1,color='orange',alpha=0.4)
plt.fill_between(X,Y-1,-1,(Y-1)>-1,color='blue',alpha=0.4)
plt.show()
```

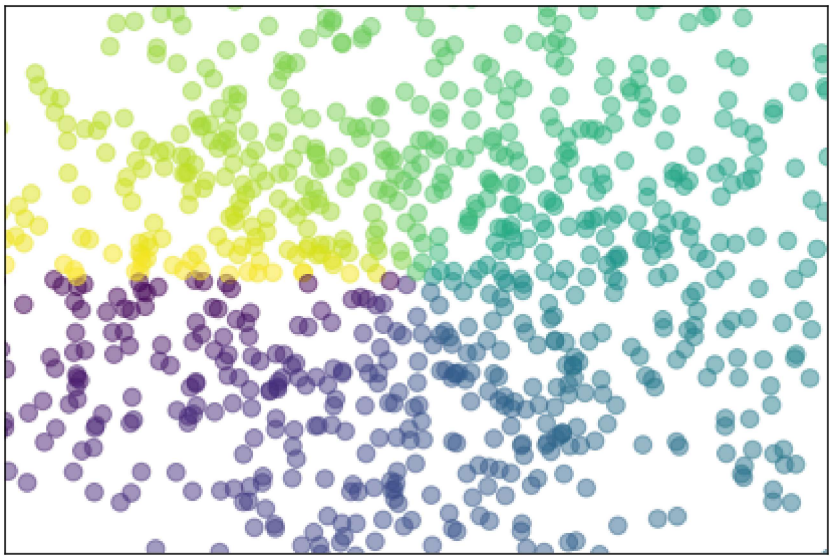


EJERCICIO 2-ORIGINAL

In [222…
```
n = 1024
X = np.random.normal(0,1,n)
Y = np.random.normal(0,1,n)
plt.scatter(X,Y)
#EJERCICIO 2
plt.show()
```



In [243…
```
#RESUELTO
n = 1024
X = np.random.normal(0,1,n)
Y = np.random.normal(0,1,n)
T=np.arctan2(Y,X)

plt.axes([0.025, 0.025, 0.95, 0.95])
plt.scatter(X, Y, c=T,s=75, alpha=.5)#S ES PARA HACER MAS GORDOS LOS PUNTOS,C es de col

plt.xlim(-1.5, 1.5)
plt.xticks(())#QUITA LOS EJES DE X PARA NO MOSTRAR ESCALA
plt.ylim(-1.5, 1.5)
plt.yticks(())#QUITA LOS EJES DE Y PARA NO MOSTRAR ESCALA

#EJERCICIO 2
plt.show()
```

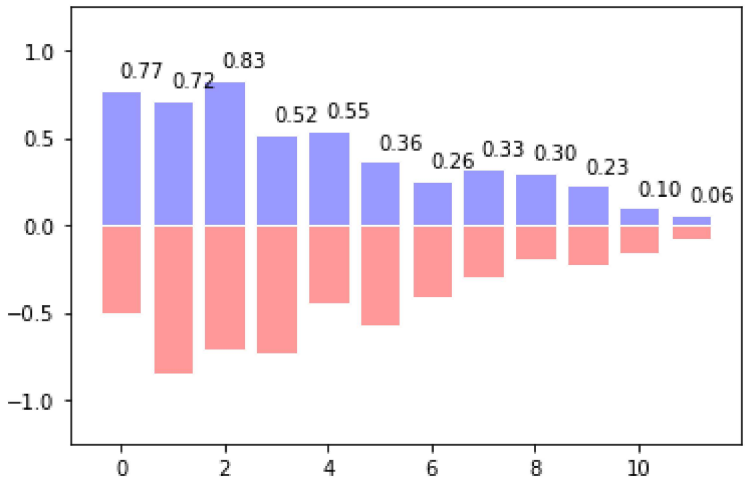


EJERCICIO 3-ORIGINAL

In [217…
```
n = 12
X = np.arange(n)
Y1 = (1 - X / float(n)) * np.random.uniform(0.5, 1.0, n)
Y2 = (1 - X / float(n)) * np.random.uniform(0.5, 1.0, n)
plt.bar(X, +Y1, facecolor='#9999ff', edgecolor='white')
```

```
plt.bar(X, -Y2, facecolor='#ff9999', edgecolor='white')
for x, y in zip(X, Y1):
        plt.text(x + 0.4, y + 0.05, '%.2f' % y, ha='center', va='bottom')
plt.ylim(-1.25, +1.25)
```

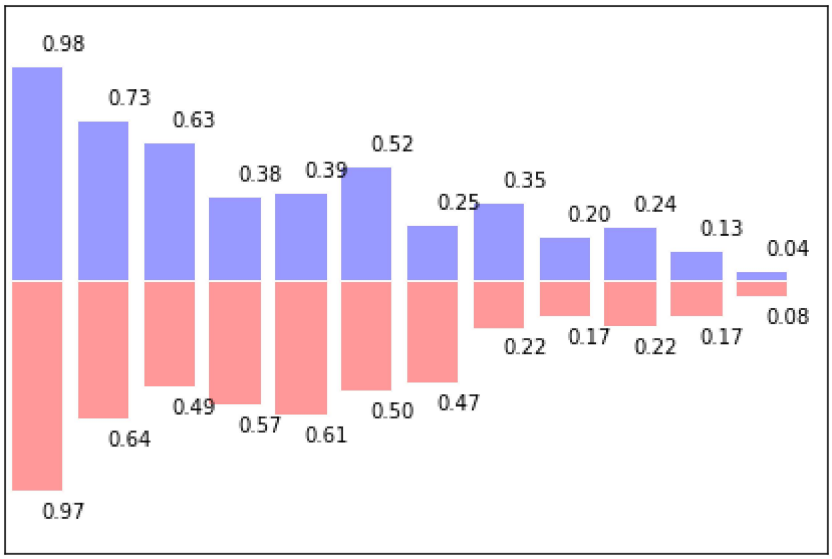
Out[217… (-1.25, 1.25)



In [250…

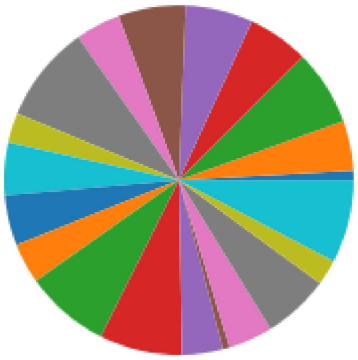
```
n = 12
X = np.arange(n)
Y1 = (1 - X / float(n)) * np.random.uniform(0.5, 1.0, n)
Y2 = (1 - X / float(n)) * np.random.uniform(0.5, 1.0, n)

plt.axes([0.025,0.025,0.95,0.95])
plt.bar(X, +Y1, facecolor='#9999ff', edgecolor='white')
plt.bar(X, -Y2, facecolor='#ff9999', edgecolor='white')
for x, y in zip(X, Y1):
        plt.text(x + 0.4, y + 0.05, '%.2f' % y, ha='center', va='bottom')
for x,y in zip(X,Y2):
    plt.text(x+0.4,-y-0.05,'%.2f'%y,ha='center',va='top')

plt.xlim(-.5,n)
plt.xticks(())
plt.ylim(-1.25, +1.25)
plt.yticks(())
```

Out[250… ([], [])

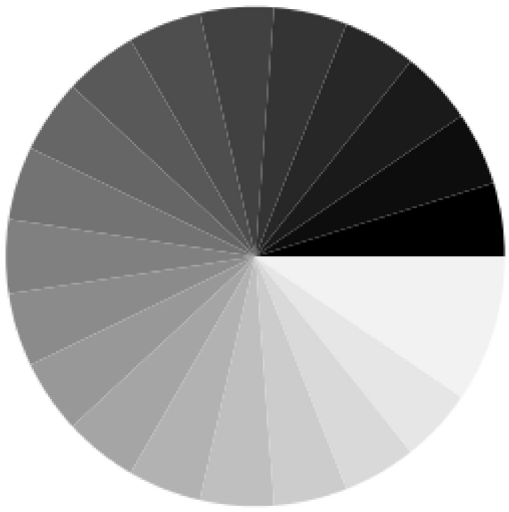


EJERCICIO 4-ORIGINAL

In [253…
```
Z = np.random.uniform(0, 2, 20)
plt.pie(Z)
plt.show()
```



In [252…
```
#RESUELTO
n = 20
Z = np.ones(n)
Z[-1] *= 2

plt.axes([0.025, 0.025, 0.95, 0.95])
plt.pie(Z, colors = ['%f' % (i/float(n)) for i in range(n)])

plt.axis('equal')
plt.xticks()
plt.yticks()

plt.show()
```



In [ ]: