TP - Visualisation

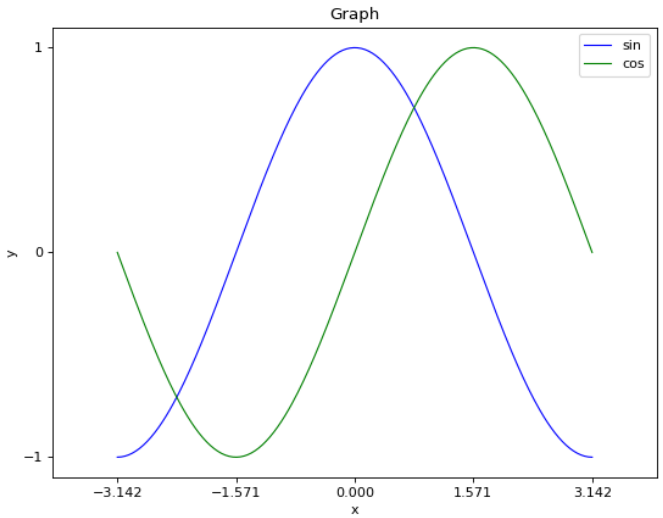
Dominique Benielli , Marina Kreme, Denis Arrivault

02 et 03 juillet 2019

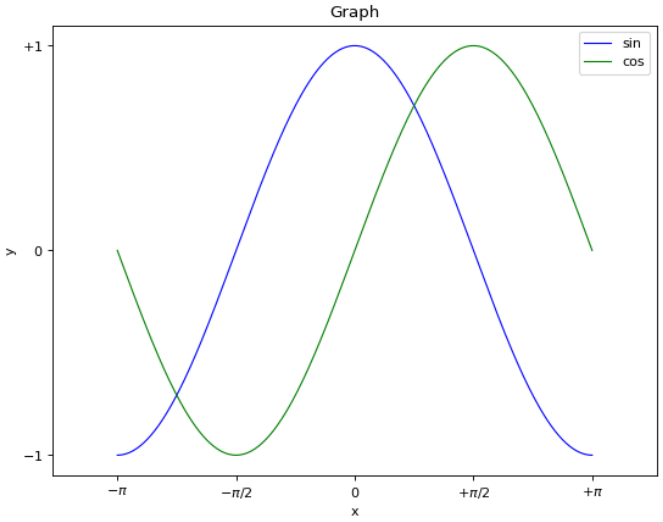
1 Plots

Fonction Sinus, cosinus

Tracer les fonctions $\sin(x)$ et $\cos(x)$. Essayez d'obtenir les rendus suivants :



et



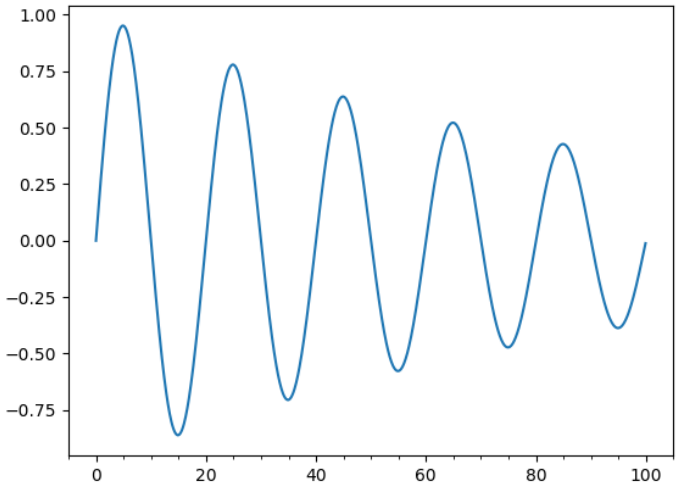
Axis

Il est possible de controler les ticks des axes de la manière suivante :

```
from matplotlib.ticker import MultipleLocator, FormatStrFormatter

majorLocator = MultipleLocator(20)
majorFormatter = FormatStrFormatter('%d')
minorLocator = MultipleLocator(5)
ax.xaxis.set_major_locator(majorLocator)
ax.xaxis.set_major_formatter(majorFormatter)
ax.xaxis.set_minor_locator(minorLocator)
```

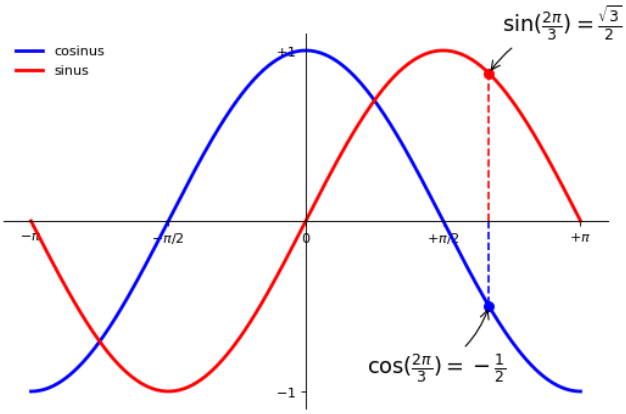
Vous permet de réaliser la figure suivante :



Annotations

En utilisant les annotations et scatter suivants obtenez la figure qui suit :

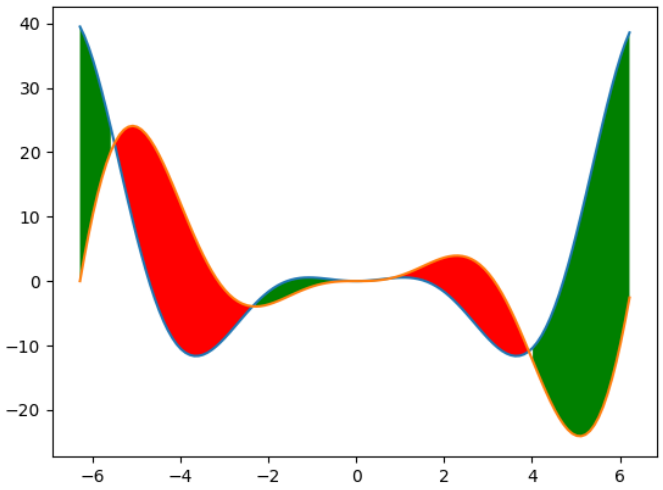
```
plt.scatter([t,],[np.cos(t),], 50, color ='blue')
plt.annotate(r'$\sin(\frac{2\pi}{3})=\frac{\sqrt{3}}{2}$',
             xy=(t, np.sin(t)),  xycoords='data',
             xytext=(+10, +30), textcoords='offset points',
                fontsize=16,
             arrowprops=dict(arrowstyle="->", connectionstyle="
                arc3,rad=.2"))
plt.scatter([t,],[np.sin(t),], 50, color ='red')
plt.annotate(r'$\cos(\frac{2\pi}{3})=-\frac{1}{2}$',
             xy=(t, np.cos(t)),  xycoords='data',
             xytext=(-90, -50), textcoords='offset points',
                fontsize=16,
             arrowprops=dict(arrowstyle="->", connectionstyle="
                arc3,rad=.2"))
```



Fill between

Définissez les fonctions $f_1 = x^2 cos(x)$ et $f_2 = x^2 sin(x)$ A l'aide des methode python ci dessous realisez le graphique suivant :

```
ax.fill_between(b, y1, y2, where= ... , facecolor='green')
```



Subplot

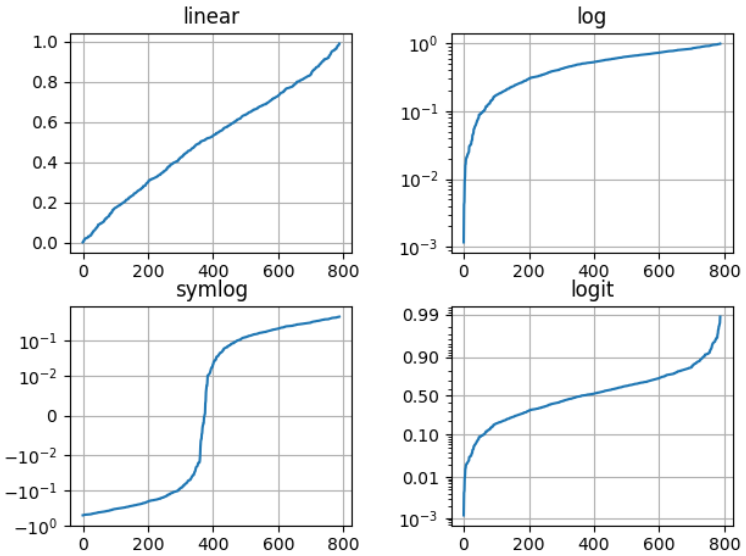
Afin de se familiariser avec les subplot, tracer une même courbe 4 fois en utilisant des échelles differentes, par exemple on pourra prendre

- courbe 1 : *yscale('linear')*
- courbe 2 : *yscale('log')*
- courbe 3 : *yscale('symlog')*
- courbe 4 : *yscale('logit')*

avec quelques aides comme

```
from matplotlib.ticker import NullFormatter  # useful for `logit`
    scale
y = 0.5 + 0.4 * np.random.randn(1000)
y = y[(y > 0) & (y < 1)]
y.sort()
x = np.arange(len(y))
...
plt.plot(x, y - y.mean())
plt.yscale('symlog', linthreshy=0.01)
plt.title('symlog')
plt.grid(True)
...
plt.gca().yaxis.set_minor_formatter(NullFormatter())
plt.subplots_adjust(top=0.92, bottom=0.08, left=0.10, right=0.95,
    hspace=0.25, wspace=0.35)
```

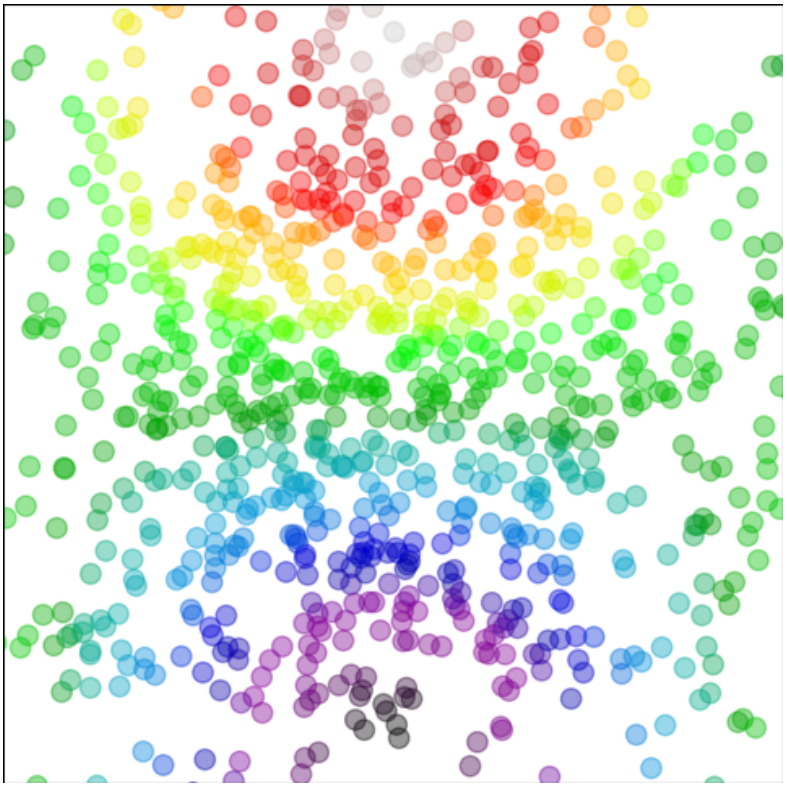
Il est possible d'obtenir le graphique suivant :



2 2D ... 3D

Scatter

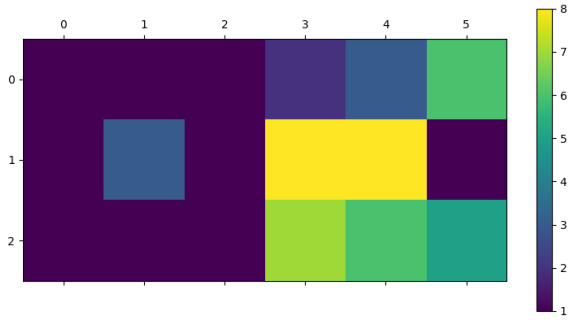
```
...
color = np.cos(x)* np.sin(y)
...
plt.scatter(x,y, s=80, c=color,cmap=plt.cm.spectral, alpha=.6)
```



Matrice show

Définissez une matrice numpy, en utilisant la methode matshow, qui dessine une matrice comme une image réalisez une figure approchante de la suivante :

```
im1 = plt.matshow(matrice)
```



Histogramme 2d

Une methode utile l'histogramme 2d : A la manière de la methode *hist* histogramme 1D tracer un histogramme 2D avec la methode hist2d. Grace à des fonctions random on pourra obtenir une figure suivante :

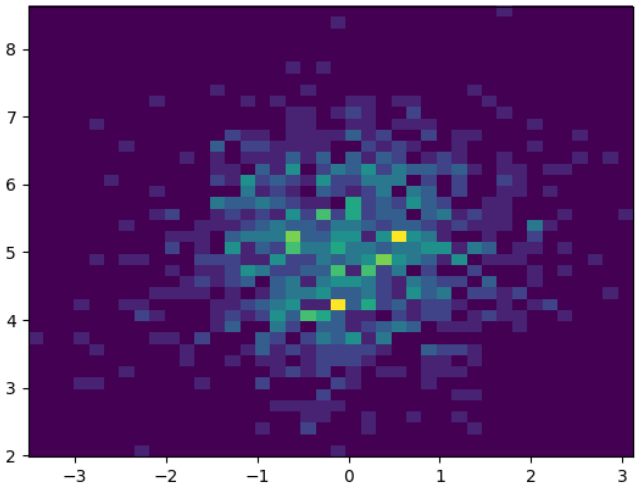


Image show et interpolation

A l'aide du keyword *interpolation* de la methode imshow réaliser la figure suivate

```
A = np.random.rand(5, 5)
plt.imshow(A, interpolation='nearest')
```

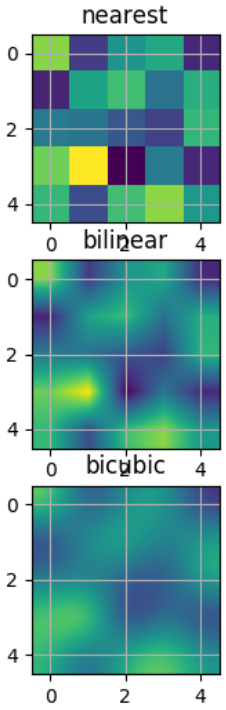


Image read

Ouvrez l'image de votre choix et affichez la avec la methode *imshow*

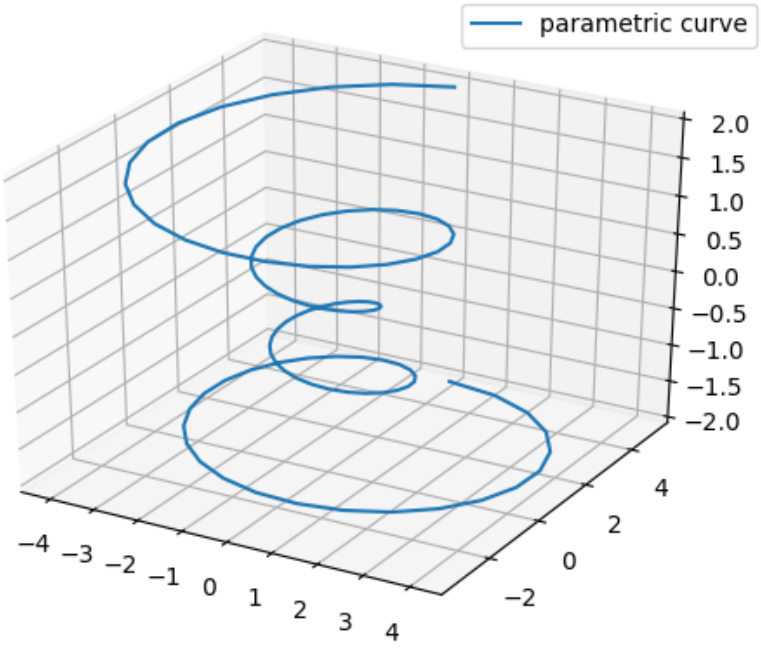
```
import os
import matplotlib.image as mpimg
cwd = os.getcwd()
path = cwd + ...
img=mpimg.imread(path+'terre.jpeg')
imgplot = plt.imshow(img)
```

line plot 3d

Grâce à la méthode *gca(projection='3d')*, realisez un line plot 3d de votre choix

indications :

```
from mpl_toolkits.mplot3d import Axes3D
ax = fig.gca(projection='3d')
ax.plot(x, y, z, label='parametric curve')
```



Surface plot

A l'ade du code, réalisez le plot de la sphère

```
u = np.linspace(0, 2 * np.pi, 100)
```

```
v = np.linspace(0, np.pi, 100)
x = 10 * np.outer(np.cos(u), np.sin(v))
y = 10 * np.outer(np.sin(u), np.sin(v))
z = 10 * np.outer(np.ones(np.size(u)), np.cos(v))
surf = ax.plot_surface(x, y, z, cmap=cm.gist_earth,
                       linewidth=0, antialiased=True)
```