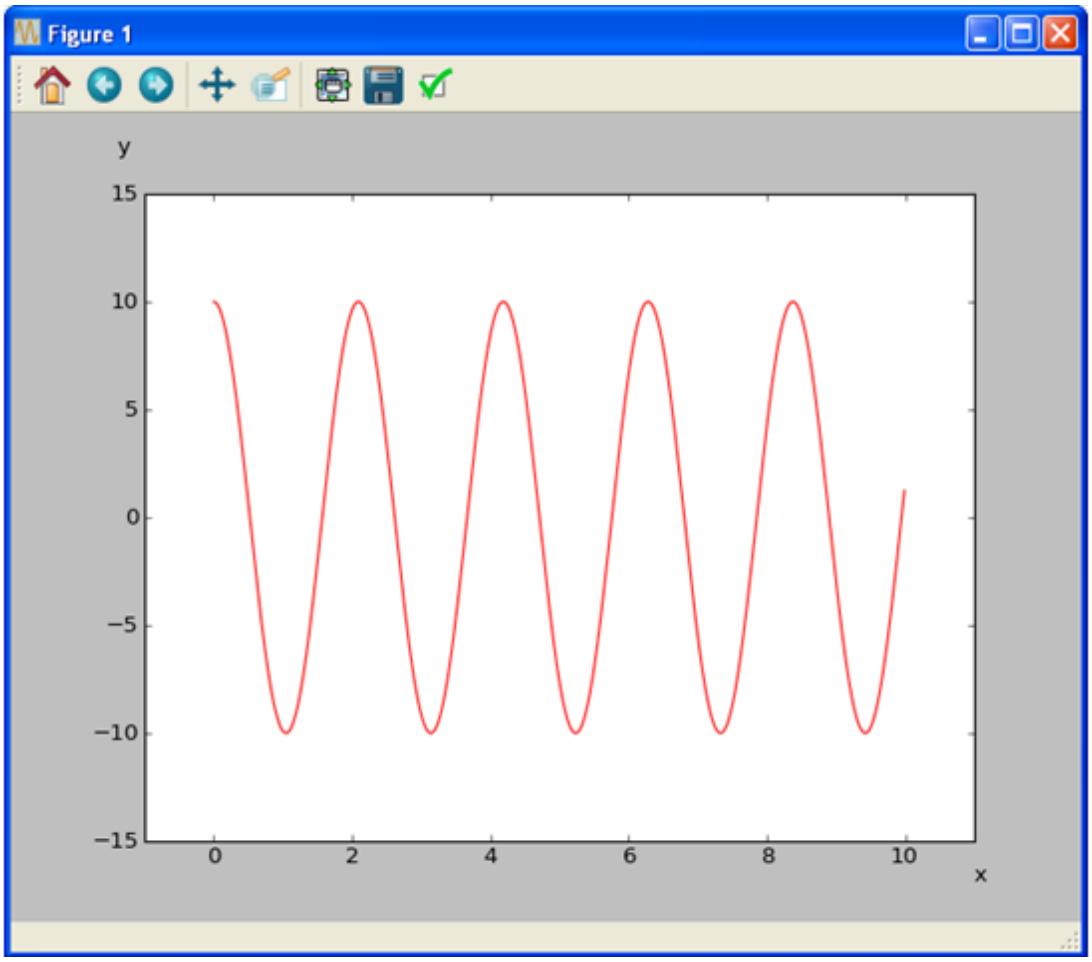
Tracer des courbes et des surfaces

1. Voici un programme Python qui permet d'afficher la **courbe d'équation cartésienne** $y = f(x)$:

```
1  from pylab import *
2
3  def f(t):
4      return 10*cos(3*t)
5
6  x = arange(0, 10, 0.01)
7  y=[]
8  for a in x:
9      y.append(f(a))
10 plot(x, y, color='red', linewidth=1)
11
12 axis([-1,11, -15, 15])
13 figtext(0.9, 0.05, 'x')
14 figtext(0.1, 0.95, 'y')
15 show()
```

Et on obtient :

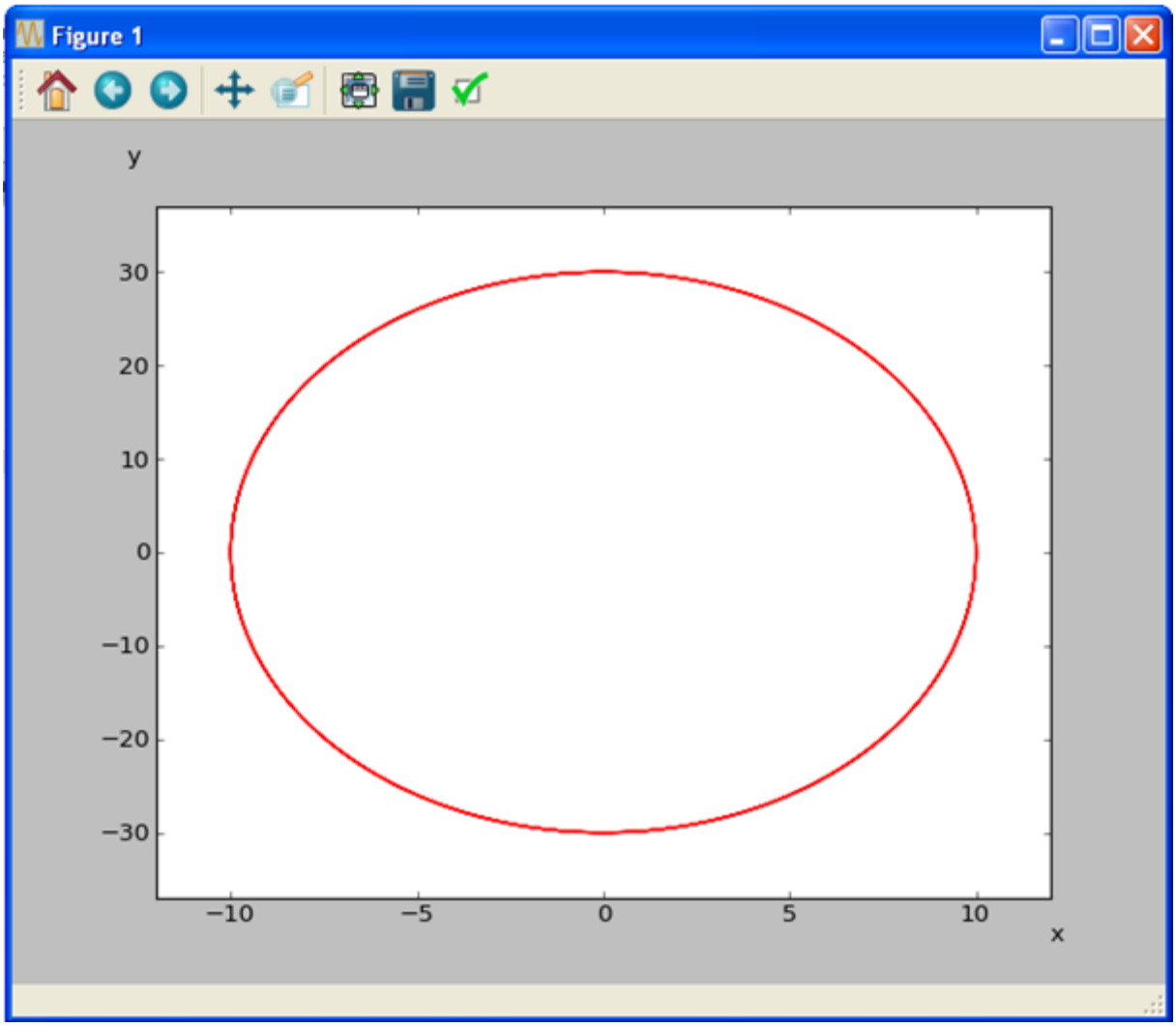


2. Voici un programme Python qui permet d'afficher une **courbe paramétrée plane** définie par :

$$\begin{cases} x = f(t) \\ y = g(t) \end{cases}$$

```
1  from pylab import *
2
3  u0=10
4  w0=3
5  def f(t):
6      return u0*cos(w0*t)
7  def g(t):
8      return -u0*w0*sin(w0*t)
9
10 t = arange(0, 10, 0.01)
11 x = f(t)
12 y = g(t)
13 plot(x, y, color='red', linewidth=1)
14
15 axis([-12,12, -37, 37])
16 figtext(0.9, 0.05, 'x')
17 figtext(0.1, 0.95, 'y')
18
19 show()
```

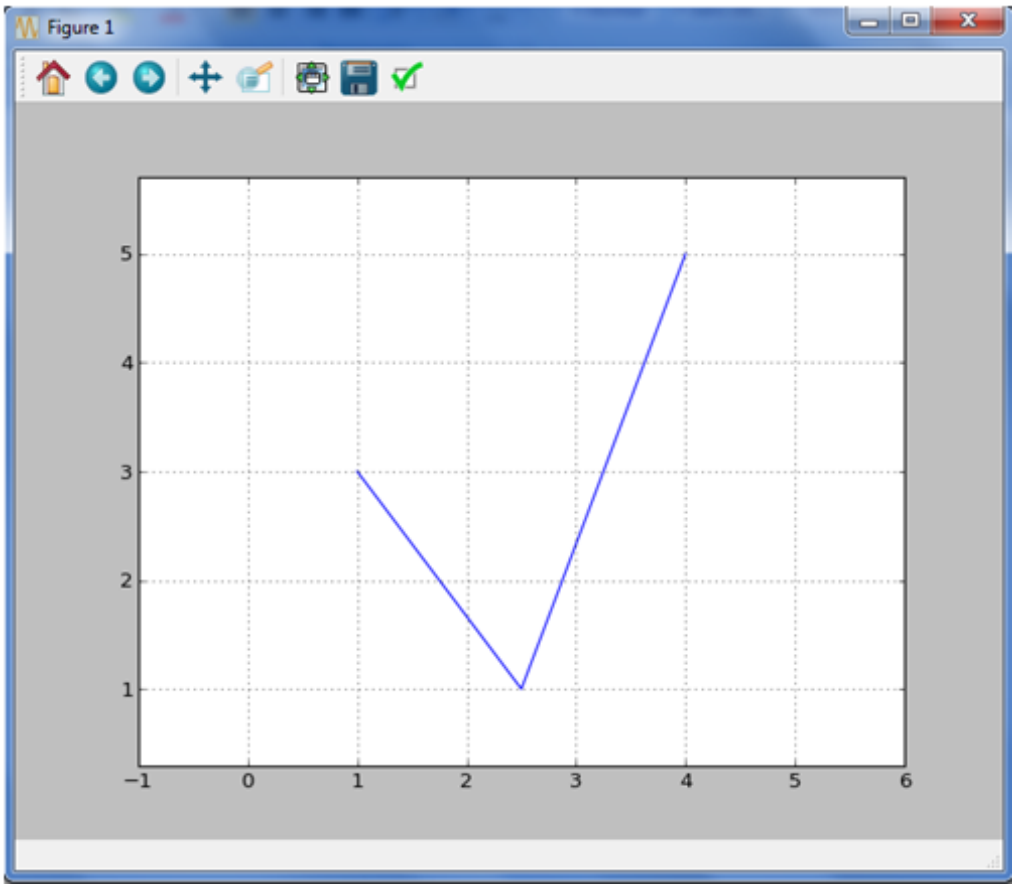
Et on obtient :



3. Voici un programme Python qui permet d'afficher une **ligne brisée** :

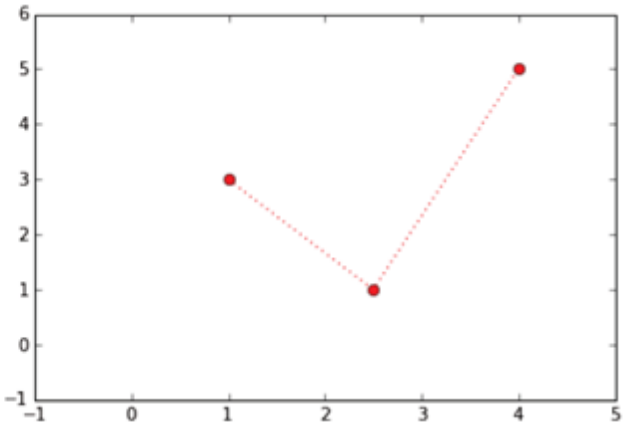
```
from pylab import *

x = [1., 2.5, 4.]
y = [3., 1., 5.]
plt.axis('equal')
plt.plot(x, y)
plt.axis([-1., 6., -1., 7.])
plt.grid()
plt.show()
```

Et on obtient :



Remarque : la fonction plot admet de nombreuses options de présentation. Le paramètre *color* permet de choisir la couleur ('g' : vert, 'r' : rouge, 'b' : bleu). Pour définir le style de la ligne, on utilise *linestyle* ('-' : ligne continue, '- -' : ligne discontinue, ' :' : ligne pointillée). Si on veut marquer les points des listes, on utilise le paramètre *marker* ('+', '.', 'o', 'v' donnent différents symboles).

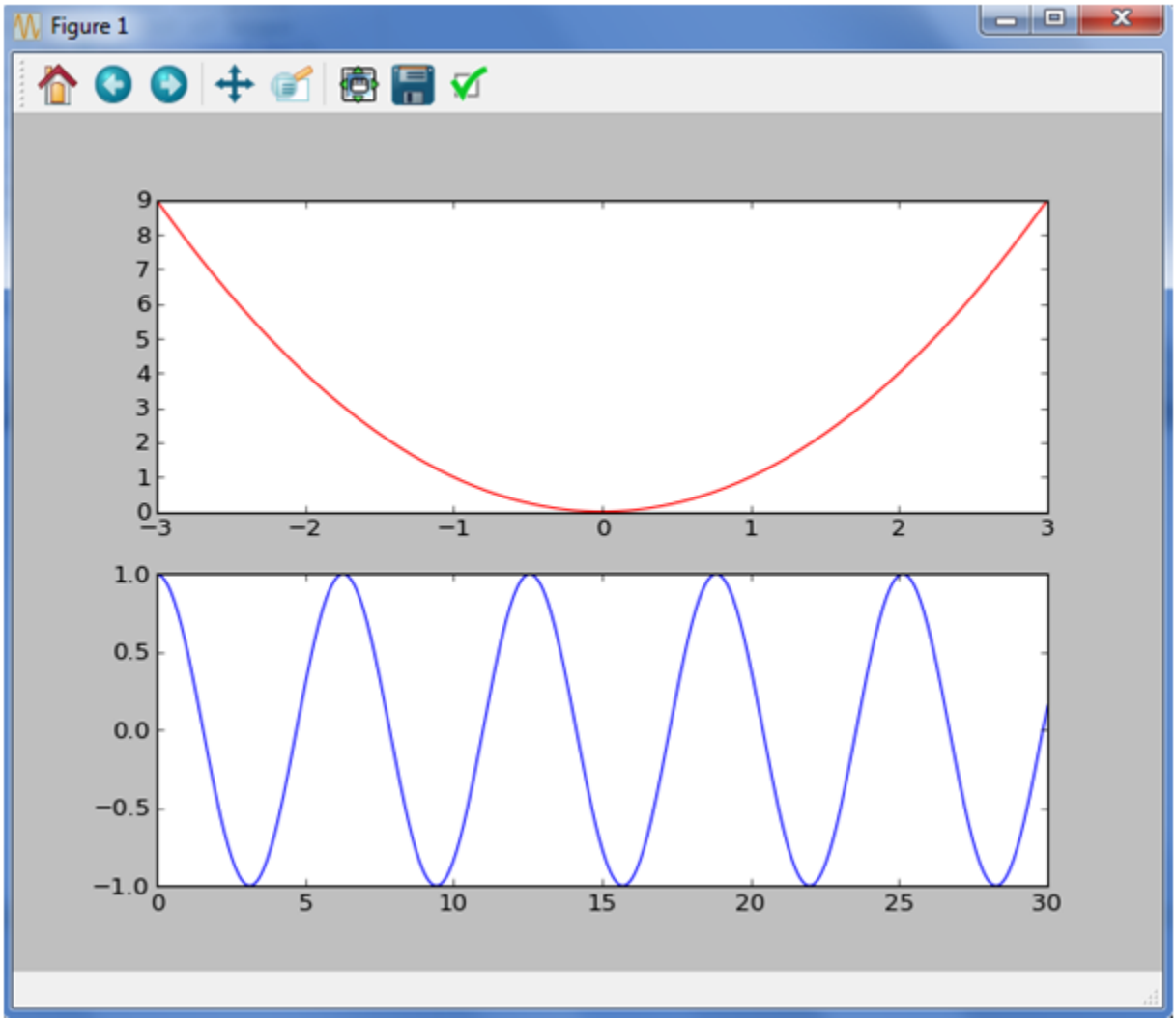
```
x = [1., 2.5, 4.]
y = [3., 1., 5.]
plt.axis([-1., 5., -1., 6.])
plt.plot(x, y, color='r', linestyle=':',
         marker='o')
plt.show()
```



4. Voici un programme Python qui permet d'afficher **deux graphiques** sur une même figure :

```
from math import *
import numpy as np
import matplotlib.pyplot as plt

def f(t):
    return t**2

def g(t):
    return cos(t)

plt.subplot(211)
x = np.linspace(-3,3,500)
y=[]
for i in range (0,500):
    y.append(f(x[i]))
plt.plot(x, y, color='red', linewidth=1)

plt.subplot(212)
x = np.linspace(0,30,500)
y=[]
for i in range (0,500):
    y.append(g(x[i]))
plt.plot(x, y, color='blue', linewidth=1)

plt.show()
```

Et on obtient :

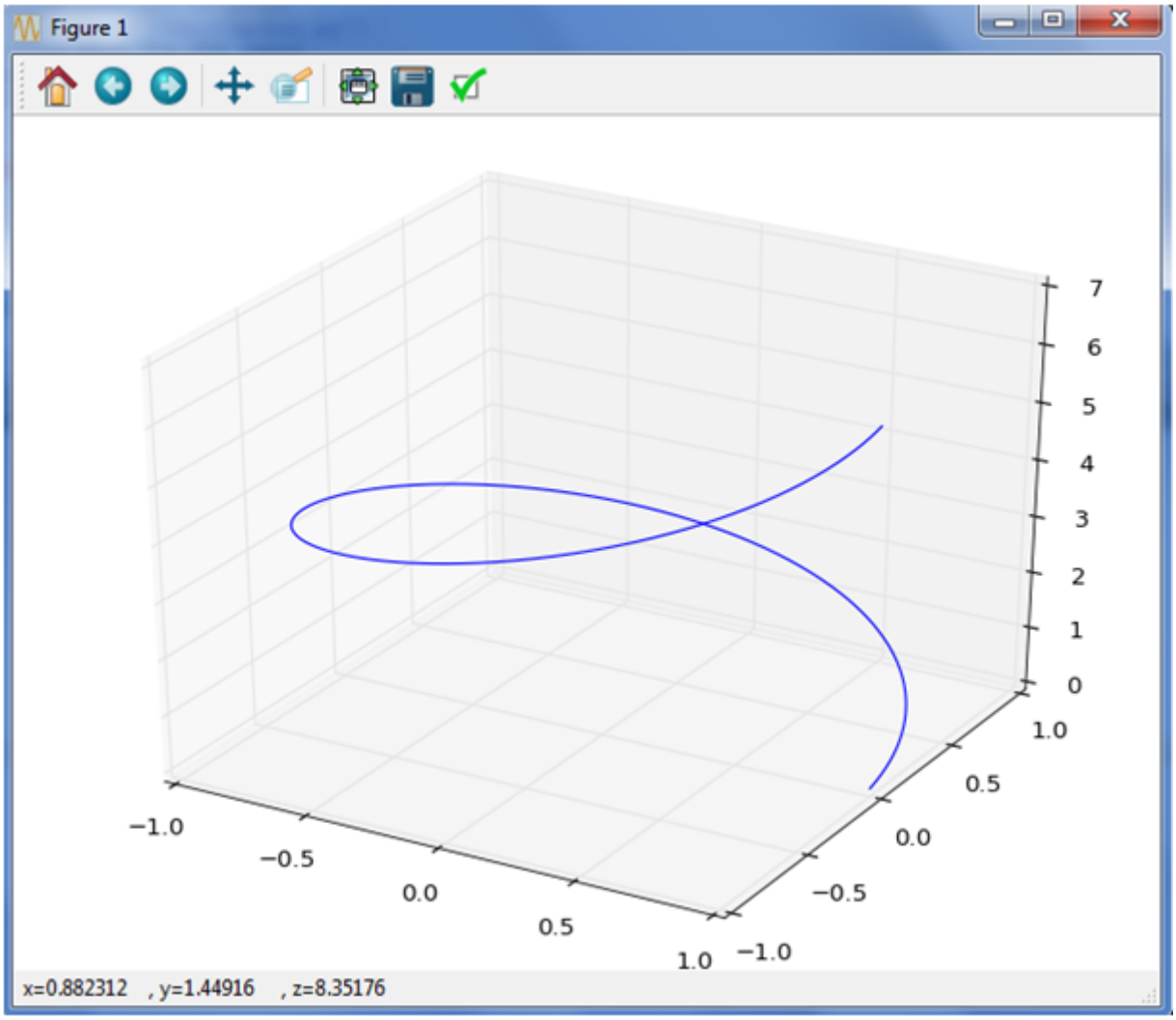


5. Voici un programme Python qui permet d'afficher une **courbe paramétrée** dans l'**espace** :

$$\begin{cases} x = f(t) \\ y = g(t) \\ z = h(t) \end{cases}$$

```
1  from math import *
2  import numpy as np
3  import matplotlib.pyplot as plt
4  from mpl_toolkits.mplot3d import Axes3D
5
6  ax = Axes3D(plt.figure())
7  T = np.arange(0, 2*np.pi, 0.01)
8  X = np.cos(T)
9  Y = np.sin(T)
10 Z = T
11 ax.plot(X, Y, T)
12 plt.show()
```

Et on obtient :

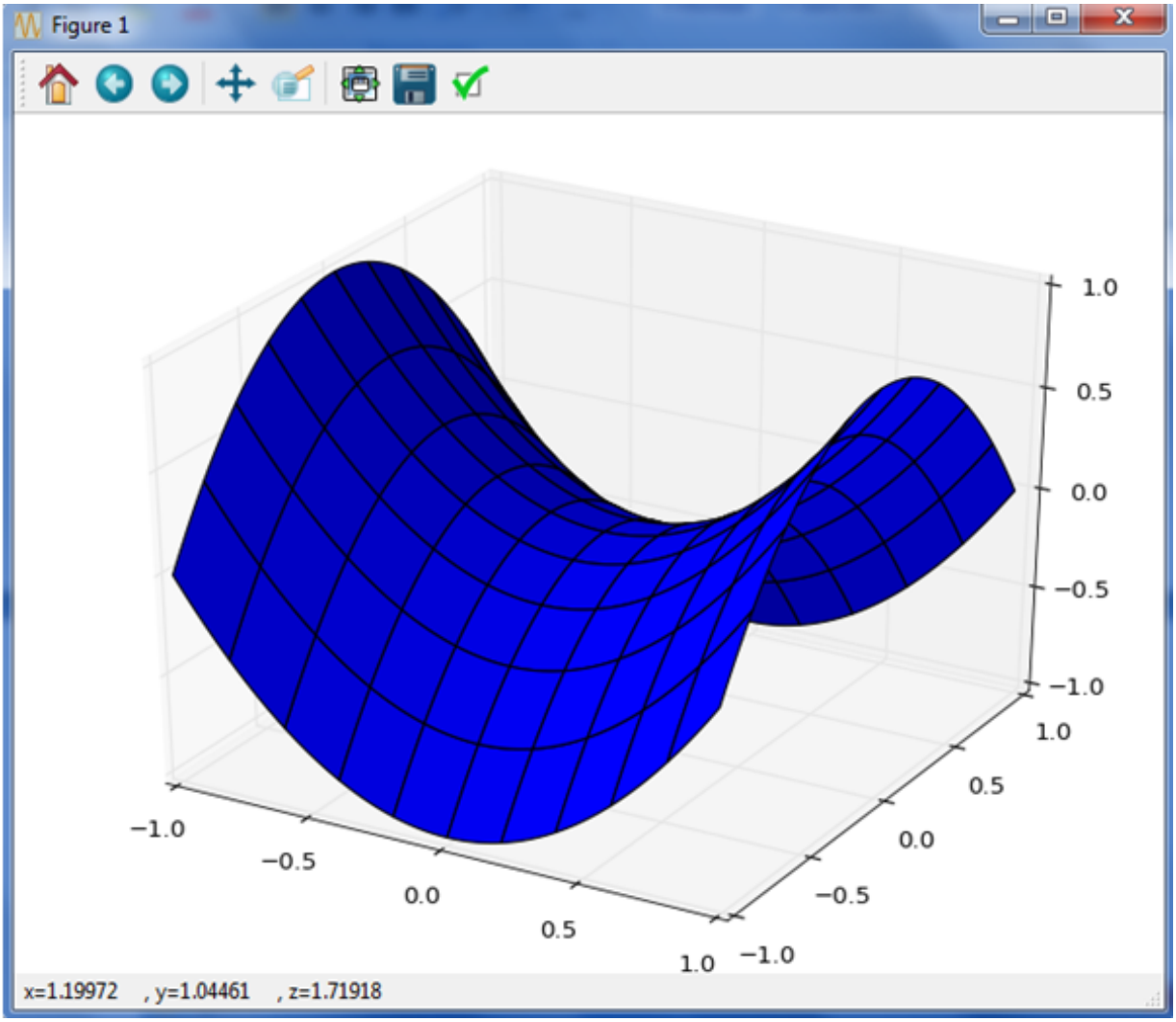


6. Voici un programme Python qui permet d'afficher une **surface** dans l'espace :

$$z = f(x, y)$$

```
from math import *
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

ax = Axes3D(plt.figure())
def f(x,y) :
    return x**2 - y**2
X = np.arange(-1, 1, 0.02)
Y = np.arange(-1, 1, 0.02)
X, Y = np.meshgrid(X, Y)
Z = f(X, Y)
ax.plot_surface(X, Y, Z)
plt.show()
```

Et on obtient :



7. Lignes de niveau :

Pour tracer des courbes d'équation $f(x, y) = k$, on fait une grille en x et en y sur laquelle on calcule les valeurs de f.

On emploie ensuite la fonction contour en mettant dans une liste les valeurs de k pour lesquelles on veut tracer la courbe d'équation $f(x, y) = k$.

Exemple :

```
def f(x,y):
    return x**2 + y**2 + x*y

X = np.arange(-1, 1, 0.01)
Y = np.arange(-1, 1, 0.01)
X, Y = np.meshgrid(X, Y)
Z = f(X, Y)
plt.axis('equal')
plt.contour(X, Y, Z, [0.1,0.4,0.5])
plt.show()
```