ENSIAS

جامعة محمد الخامس بالرباط
Université Mohammed V de Rabat

Langage Python

Lecture7: Matplotlib

Abderrahim MESBAH
a.mesbah@um5r.ac.ma

Plan

- Tracé des courbes
- Nuage de points
- Figures multiple
- Histogramme

Tracé des courbes

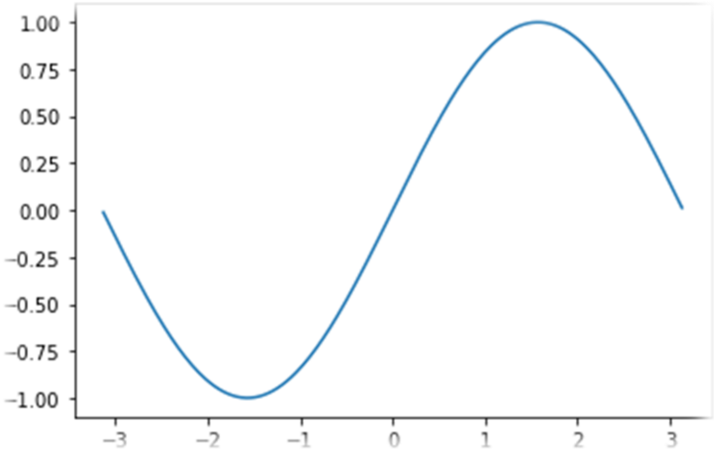
Tracé des courbes

- **Les fonctions plot/show**

```
import math
import matplotlib.pyplot as plt

nbSamples = 256
xRange =[-math.pi, math.pi]
x , y = [ ] , [ ]
for n in range (nbSamples):
      k = n/nbSamples
      x.append(xRange[0 ] + ( xRange[1] - xRange[0] )*k )
      y.append(math.sin(x[-1]))


plt.plot( x , y )
plt.show( )
```



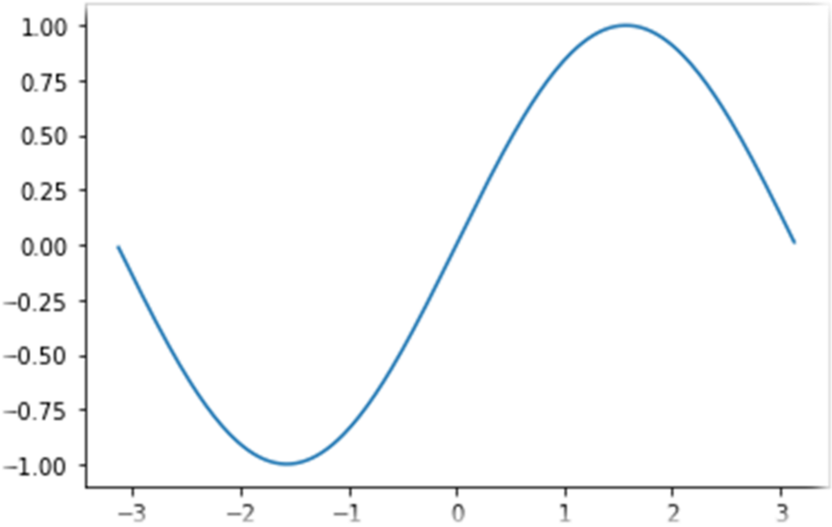
Tracé des courbes

- matplotlib peut fonctionner avec des tableaux numpy

```
import math
import matplotlib.pyplot as plt
import numpy as np
nbSamples = 256
xRange =[-math.pi, math.pi]
x , y = np.zeros (nbSamples) , np.zeros (nbSamples)

for i in range (nbSamples):
     k = i/nbSamples
     x[i]=(xRange[0 ] + ( xRange[1] - xRange[0] )*k )
     y[i]=(math.sin(x[i]))


plt.plot( x , y )
plt.show( )
```



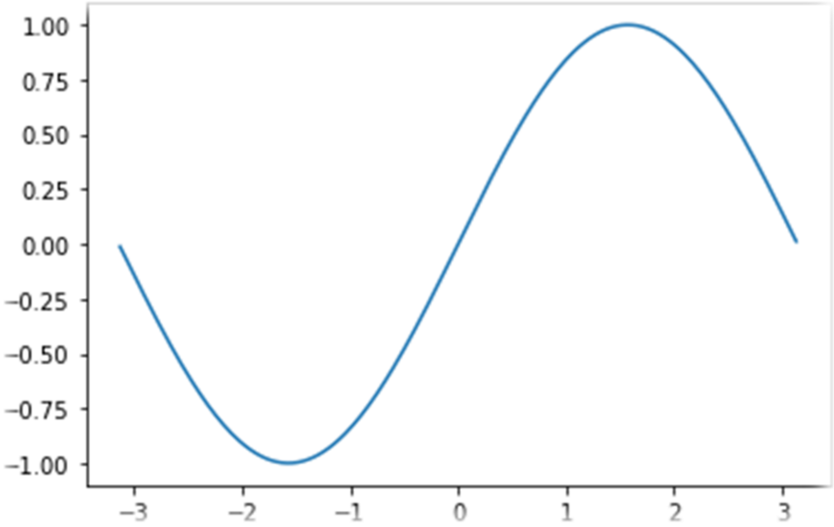
Tracé des courbes

- matplotlib peut fonctionner avec des tableaux numpy

```
import matplotlib.pyplot as plt
import numpy as np
nbSamples = 256

x=np.linspace (-math.pi, math.pi, nbSamples)
y=np.sin(x)

plt.plot( x , y )
plt.show( )
```



Tracé des courbes

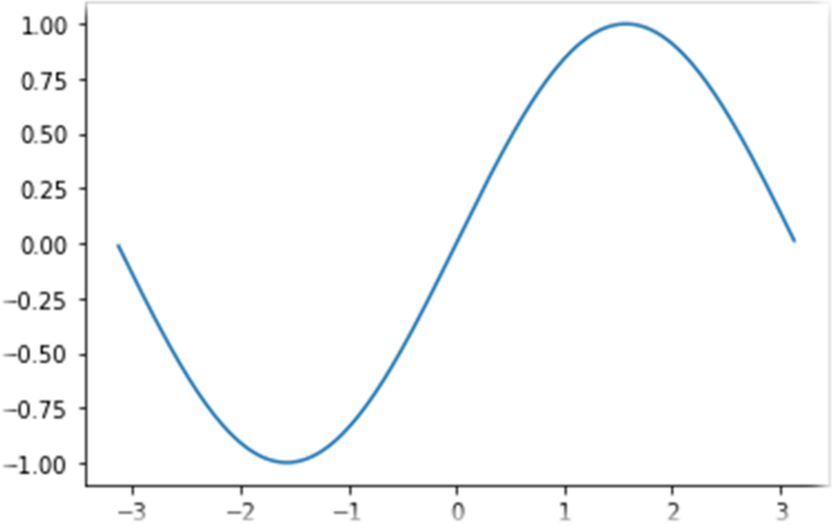
- L'exportation vers un fichier PDF: **savefig**

```
import matplotlib.pyplot as plt
import numpy as np
nbSamples = 256

x=np.linspace (-math.pi, math.pi, nbSamples)
y=np.sin(x)

plt.plot( x , y )
plt.show( )

plt.savefig('sin−plot.png’)
```



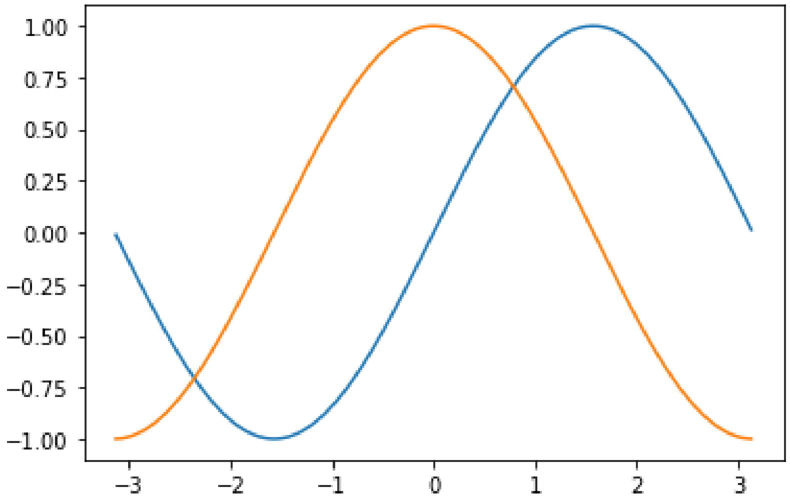
Tracé des courbes

- Courbes multiple

```
import matplotlib.pyplot as plt
import numpy as np
nbSamples = 256

x=np.linspace (-math.pi, math.pi, nbSamples)
y=np.sin(x)
z=np.cos(x)

plt.plot(x , y  )
plt.plot(x , z  )
plt.show()
```



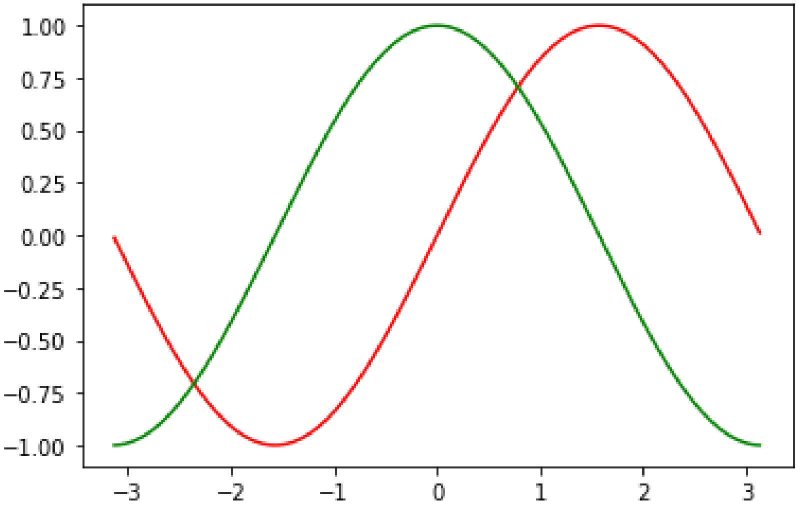
Tracé des courbes

- Courbes multiple: **Couleurs personnalisées**

```
import matplotlib.pyplot as plt
import numpy as np
nbSamples = 256

x=np.linspace (-math.pi, math.pi, nbSamples)
y=np.sin(x)
z=np.cos(x)

plt.plot(x , y, c='red’) #c=‘red’ ou bien color=‘red’
plt.plot(x , z, c='green' )
plt.show()
```



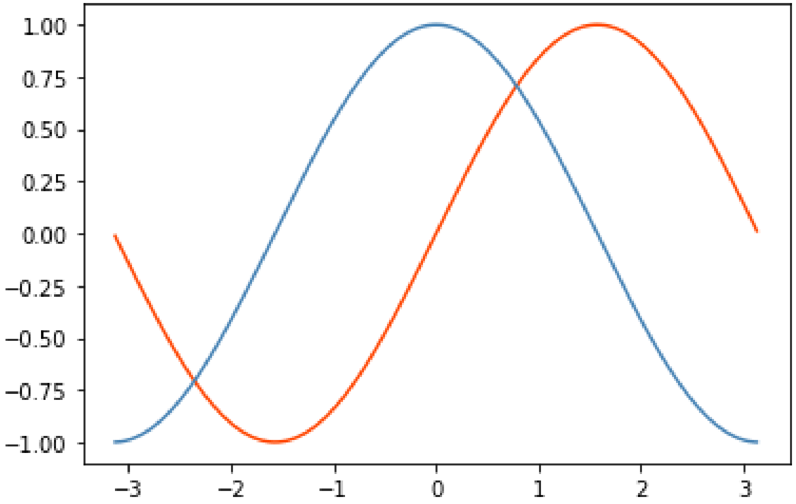
Tracé des courbes

- Courbes multiple: **Couleurs personnalisées(notation hexadécimale)**

```
import matplotlib.pyplot as plt
import numpy as np
nbSamples = 256

x=np.linspace (-math.pi, math.pi, nbSamples)
y=np.sin(x)
z=np.cos(x)

plt.plot(x , y, c='#FF4500')
plt.plot(x , z, c='#4682B4' )
plt.show()
```



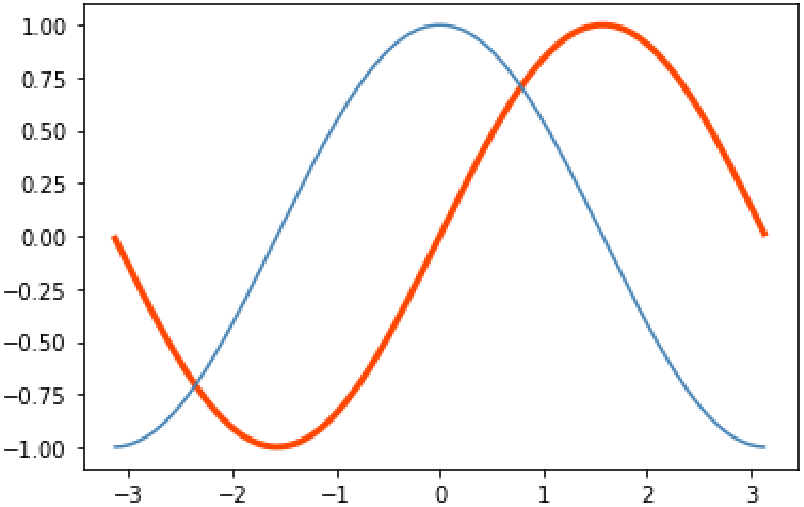
Tracé des courbes

- Courbes multiple: **Épaisseur des lignes**

```
import matplotlib.pyplot as plt
import numpy as np
nbSamples = 256

x=np.linspace (-math.pi, math.pi, nbSamples)
y=np.sin(x)
z=np.cos(x)

plt.plot(x , y, linewidth=3 ,c=‘red’)
plt.plot(x , z, c=‘green’ )
plt.show()
```



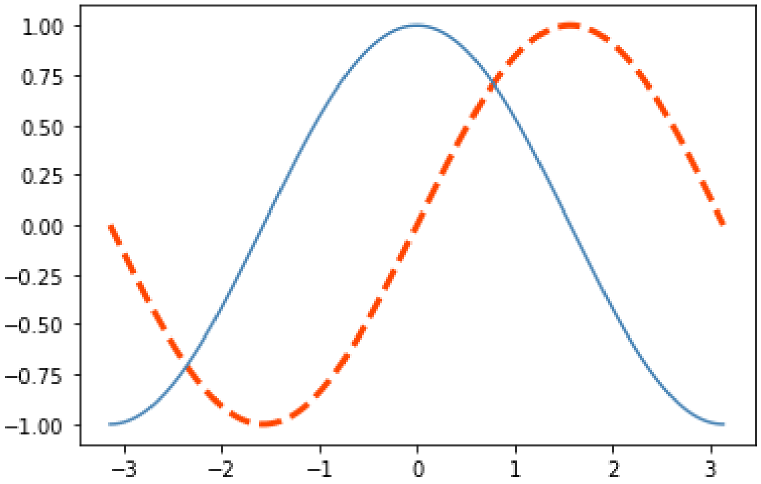
Tracé des courbes

- Courbes multiple: **Motifs de lignes**

```
import matplotlib.pyplot as plt
import numpy as np
nbSamples = 256

x=np.linspace (-math.pi, math.pi, nbSamples)
y=np.sin(x)
z=np.cos(x)

plt.plot(x , y, linestyle='--' ,linewidth=3, c='#FF4500')
plt.plot(x , z, c='#4682B4' )
plt.show()
```



- **'-', '--', '-.', ':', 'None', ' ', '', 'solid', 'dashed', 'dashdot', 'dotted'**

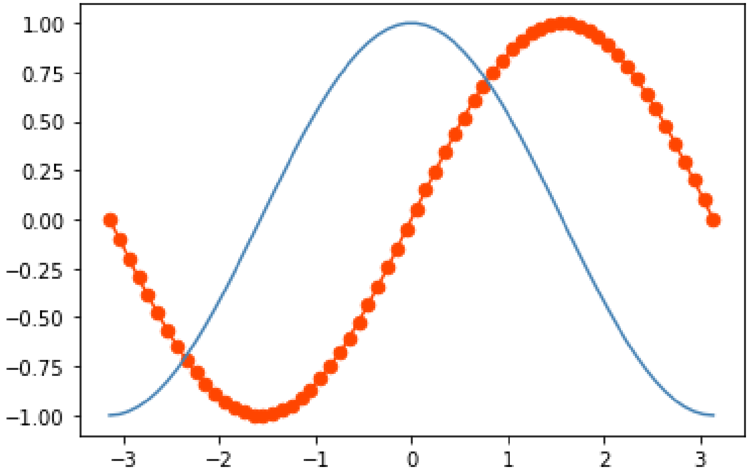
Tracé des courbes

- Courbes multiple: **Il est parfois pertinent d'afficher les points de données**

```
import matplotlib.pyplot as plt
import numpy as np
nbSamples = 64

x=np.linspace (-math.pi, math.pi, nbSamples)
y=np.sin(x)
z=np.cos(x)

plt.plot(x , y, marker='.' , markersize =13, c='#FF4500')
plt.plot(x , z, c='#4682B4' )
plt.show()
```



- **’ . ’ , ’ , ’ , ’ o ’ , ’ 1 ’ et autres**

Tracé des courbes

- Courbes multiple: **légende**

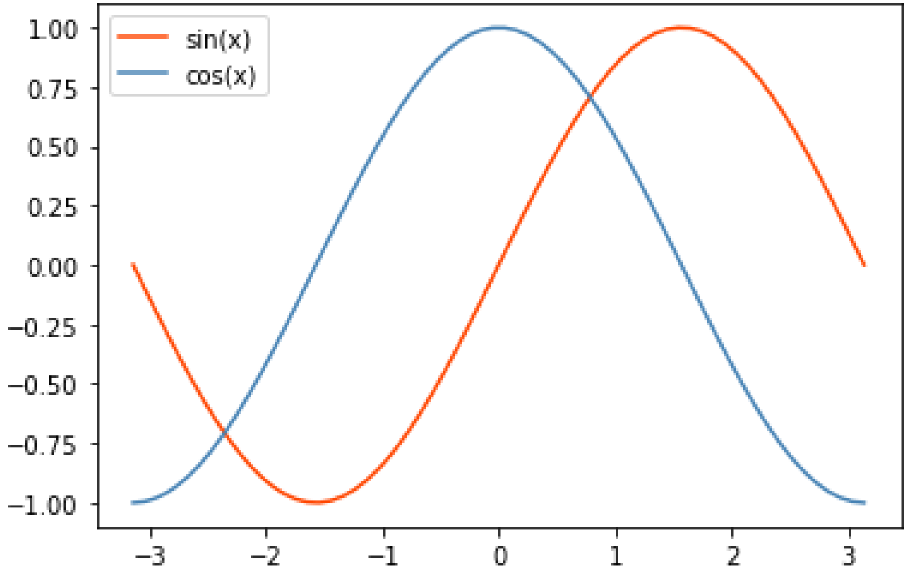
```
import matplotlib.pyplot as plt
import numpy as np
nbSamples = 64

x=np.linspace (-math.pi, math.pi, nbSamples)
y=np.sin(x)
z=np.cos(x)

plt.plot(x , y, c='#FF4500',label='sin(x)')
plt.plot(x , z, c='#4682B4',label='cos(x)’)

plt.legend(loc=‘upper left’)

plt.show()
```



- **best, upper right, upper left, lower left, lower right, right, center left, center right, lower center, upper center, center**

Tracé des courbes

- Courbes multiple: **les étiquettes des axes**

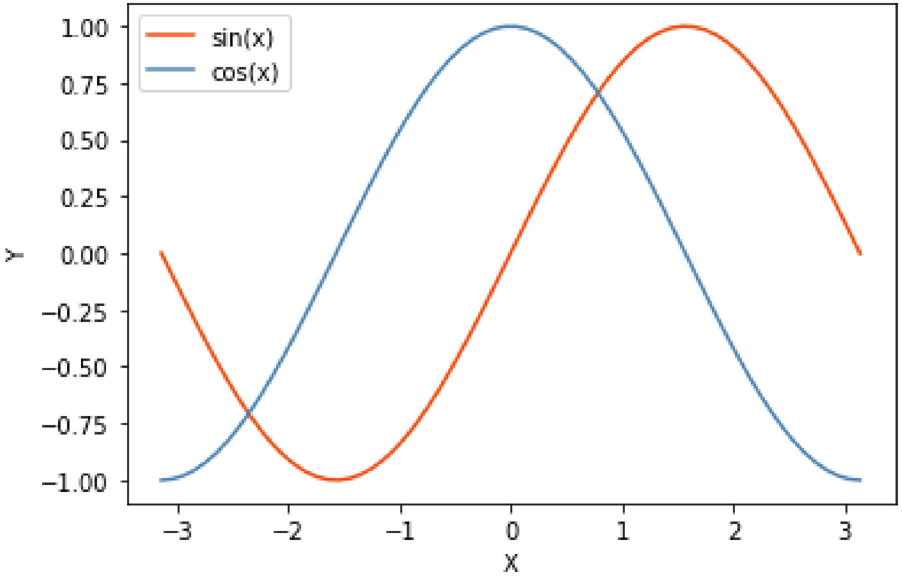
```
import matplotlib.pyplot as plt
import numpy as np
nbSamples = 64

x=np.linspace (-math.pi, math.pi, nbSamples)
y=np.sin(x)
z=np.cos(x)

plt.plot(x , y, c='#FF4500',label='sin(x)')
plt.plot(x , z, c='#4682B4',label='cos(x)')
plt.legend(loc=‘upper left’)

plt.xlabel('X')
plt.ylabel('Y')

plt.show()
```



Tracé des courbes

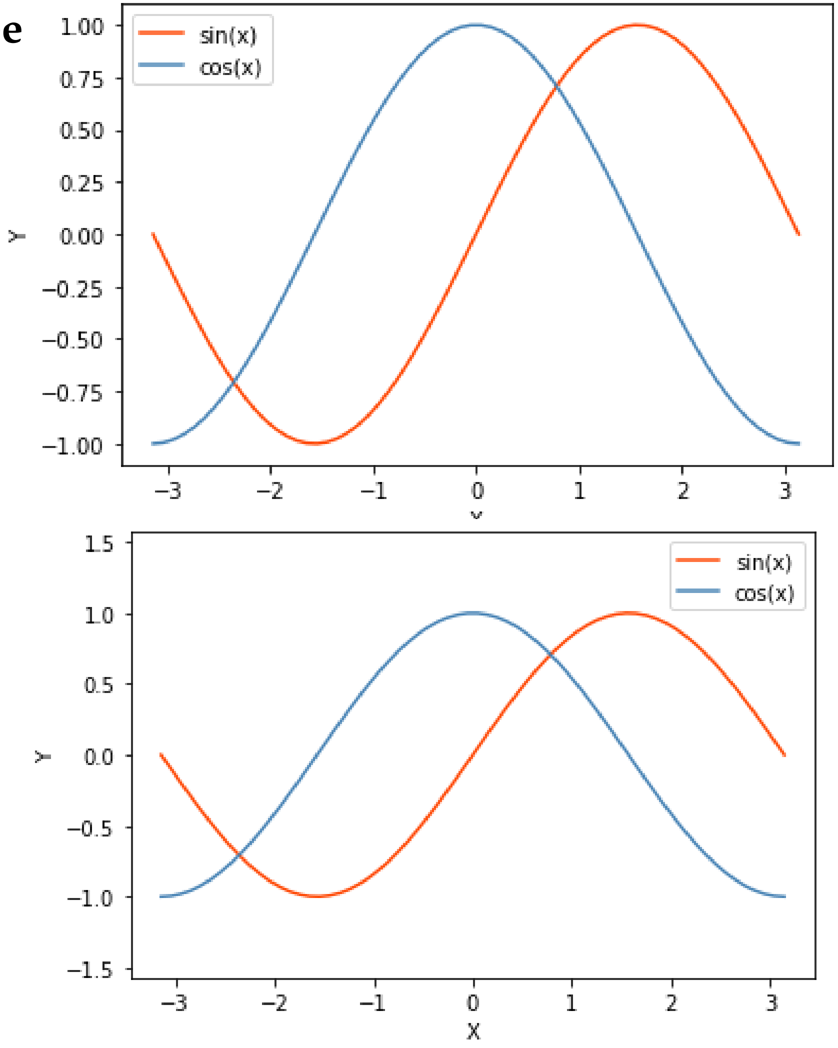
- Courbes multiple: **Échelle d'axe personnalisée**

```
import matplotlib.pyplot as plt
import numpy as np
import math
nbSamples = 64

x=np.linspace (-math.pi, math.pi, nbSamples)
y=np.sin(x)
z=np.cos(x)

fig = plt.figure()
axis = fig.add_subplot(111)
axis.set_ylim(-0.5*math.pi , 0.5*math.pi)

plt.plot(x , y, c='#FF4500',label='sin(x)')
plt.plot(x , z, c='#4682B4',label='cos(x)')
plt.xlabel('X')
plt.ylabel('Y')
plt.legend(loc=‘upper right')
plt.show()
```



Tracé des courbes

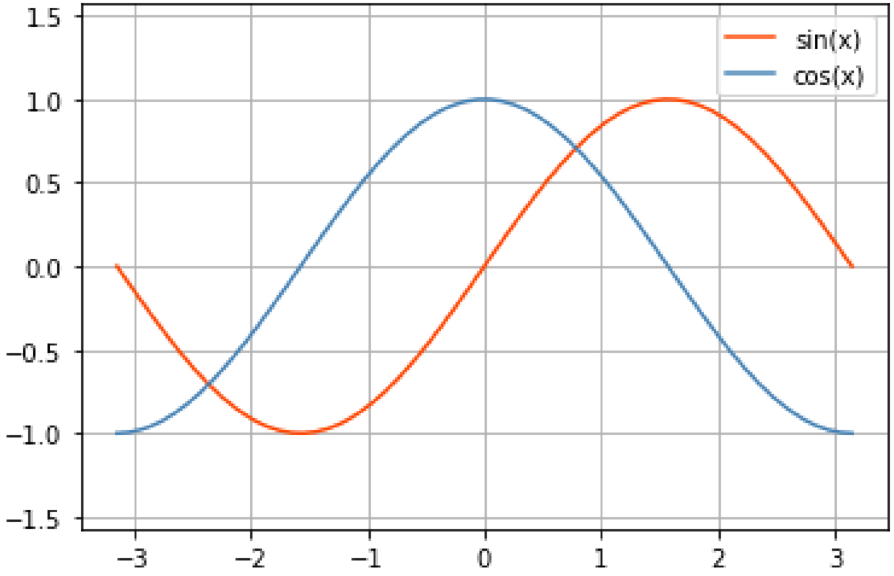
- Courbes multiple: **Grille**

```
import matplotlib.pyplot as plt
import numpy as np
import math
nbSamples = 64

x=np.linspace (-math.pi, math.pi, nbSamples)
y=np.sin(x)
z=np.cos(x)

fig = plt.figure()
axis = fig.add_subplot(111)
axis.set_ylim(-0.5*math.pi , 0.5*math.pi)
axis.grid(True)

plt.plot(x , y, c='#FF4500',label='sin(x)')
plt.plot(x , z, c='#4682B4',label='cos(x)')
plt.legend()
plt.show()
```



Figures multiple

Figures multiple

- **Cycle de vie d’une figure**

```
import matplotlib.pyplot as plt
import numpy as np

t = np.arange(0.0, 2.0, 0.01)
s1 = np.sin(2*np.pi*t)
s2 = np.sin(4*np.pi*t)

plt.figure()
```

Début de la figure

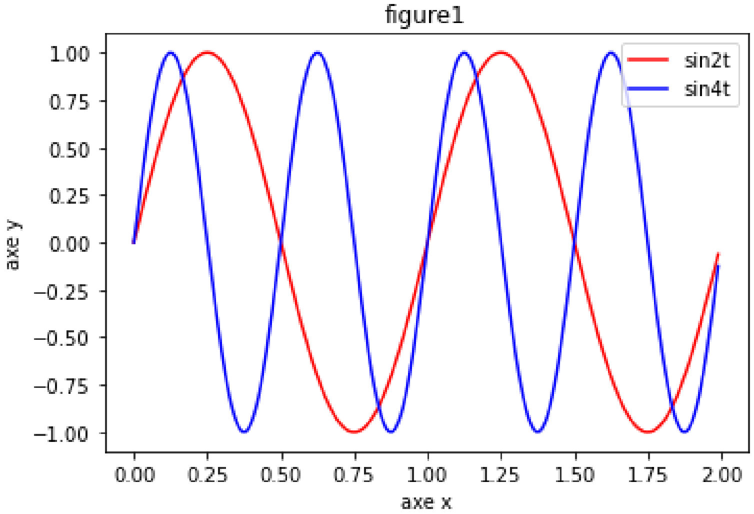
```
plt.plot(t, s1, c='red', label='sin2t’)
plt.plot(t, s2, c='blue', label='sin4t’)
plt.title(‘figure1’)
plt.xlabel(‘axe x’)
plt.ylabel(‘axe y’)
plt.legend()
plt.legend(loc='upper right’)
```

Contenu

```
plt.show()
```

Affichage de la figure

```
plt.savefig(‘text.png’)
```



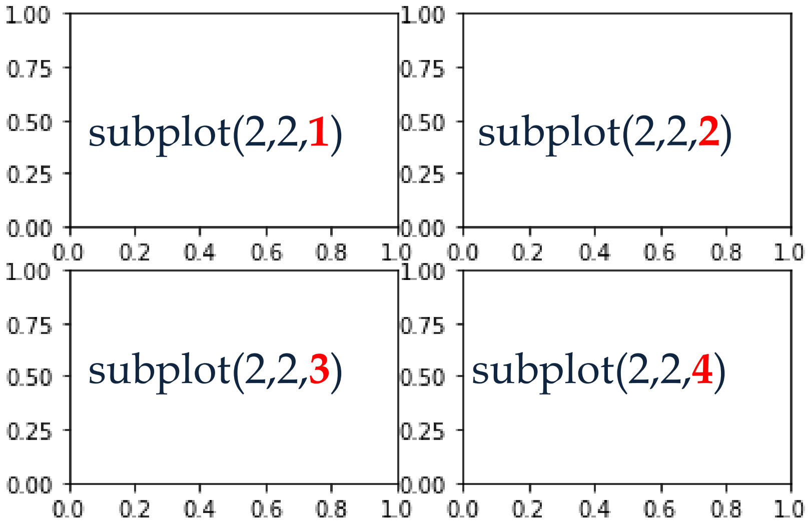
Figures multiple

- **Figure multiples : plt.subplot(lignes, colonnes, position)**

```
import matplotlib.pyplot as plt

plt.subplot(2,2,1)
plt.subplot(2,2,2)
plt.subplot(2,2,3)
plt.subplot(2,2,4)

plt.show
```



Figures multiple

- **Figure multiples : plt.subplot(lignes, colonnes, position)**

```
import matplotlib.pyplot as plt
import numpy as np
t = np.arange(0.0, 2.0, 0.01)
s1 = np.sin(2*np.pi*t)
s2 = np.sin(4*np.pi*t)

plt.figure()
```

Début de la figure

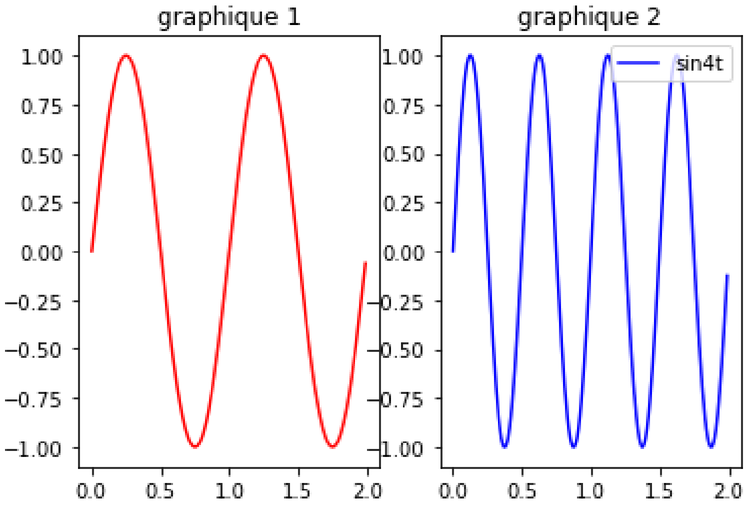
```
plt.subplot(1,2,1)
plt.plot(t, s1, c='red', label='sin2t')
plt.title('graphique 1')
```

Graphique 1

```
plt.subplot(1,2,2)
plt.plot(t, s2, c='blue', label='sin4t')
plt.title('graphique 2')
plt.legend(loc='upper right')

plt.show()
```

Affichage de la figure



Figures multiple

- **Matplotlib Fonction vs OOP**

Fonction : plus simple et permet de tracer 99% des graphiques

```
plt.plot(x, y)
plt.show()
```

OOP : plus techniques

```
fig, ax = plt.subplots()
ax. plot(x, y)
plt.show()
```

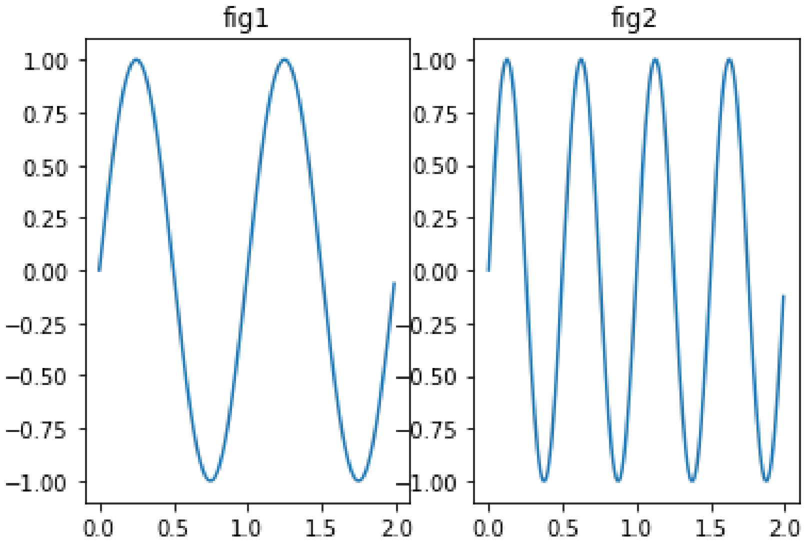
Figures multiple

- **Horizontale**

```
import matplotlib.pyplot as plt
import numpy as np

t = np.arange(0.0, 2.0, 0.01)
s1 = np.sin(2*np.pi*t)
s2 = np.sin(4*np.pi*t)

fig, ax = plt.subplots(1,2, constrained_layout=True)
ax[0].plot(t,s1)
ax[0].set_title('fig1')
ax[1].plot(t,s2)
ax[1].set_title('fig2')
plt.show()
```



Figures multiple

- **Labels**

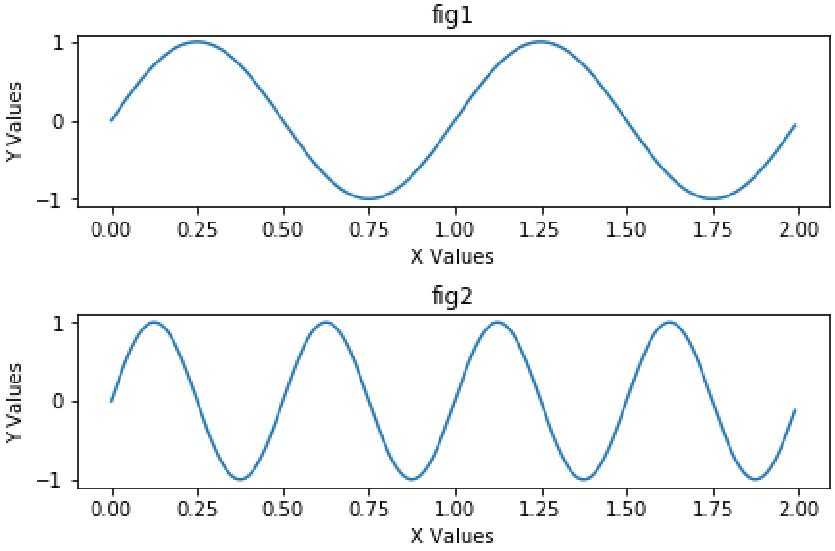
```
import matplotlib.pyplot as plt
import numpy as np

t = np.arange(0.0, 2.0, 0.01)
s1 = np.sin(2*np.pi*t)
s2 = np.sin(4*np.pi*t)

fig, ax = plt.subplots(2, constrained_layout=True)
ax[0].plot(t,s1)

ax[0].set(xlabel='X Values', ylabel='Y Values',
title='fig1')


ax[1].plot(t,s2)
ax[1].set(xlabel='X Values', ylabel='Y Values',
title='fig2')
plt.show()
```



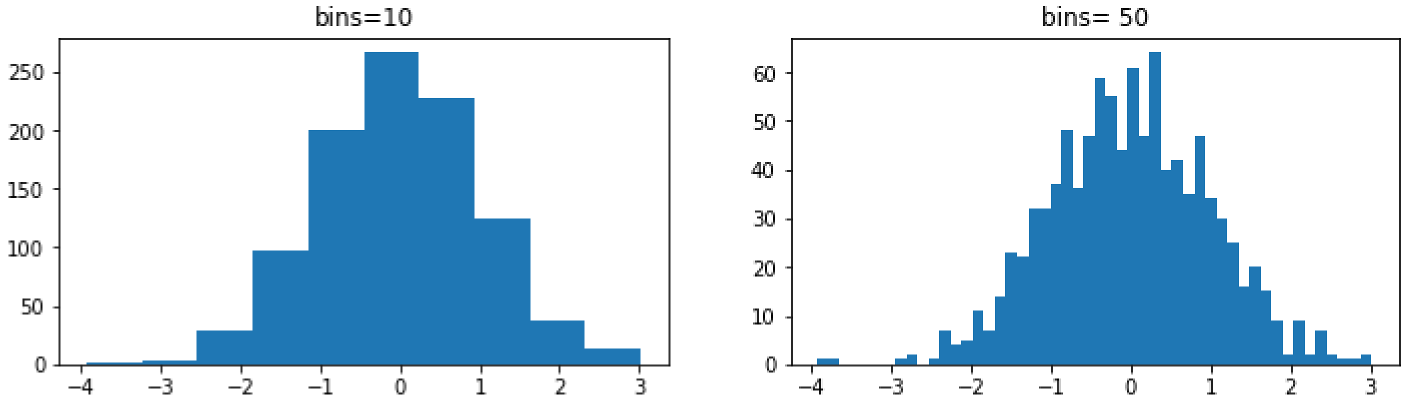
Histogramme

Histogramme

- **Tracé un histogramme**

```
x = np.random.randn(1000)

plt.figure(figsize=(12, 3))
plt.subplot(121)
plt.hist(x, bins=10)
plt.title('bins=10')
plt.subplot(122)
plt.hist(x, bins=50)
plt.title('bins= 50')
plt.show()
```

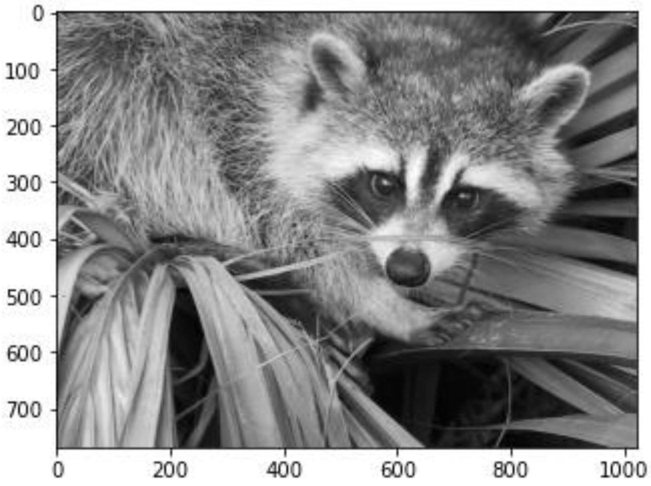


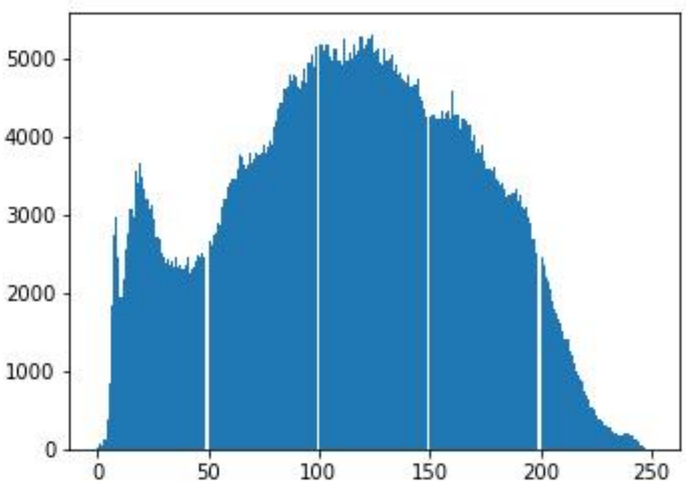
Histogramme

- **histogramme d'une image**

```
from scipy import misc
face = misc.face(gray=True)

plt.figure(figsize=(12, 4))
plt.subplot(121)
plt.imshow(face, cmap='gray')
plt.subplot(122)
plt.hist(face.ravel(), bins=255)
plt.show()
```





Nuage de points

Nuage de points

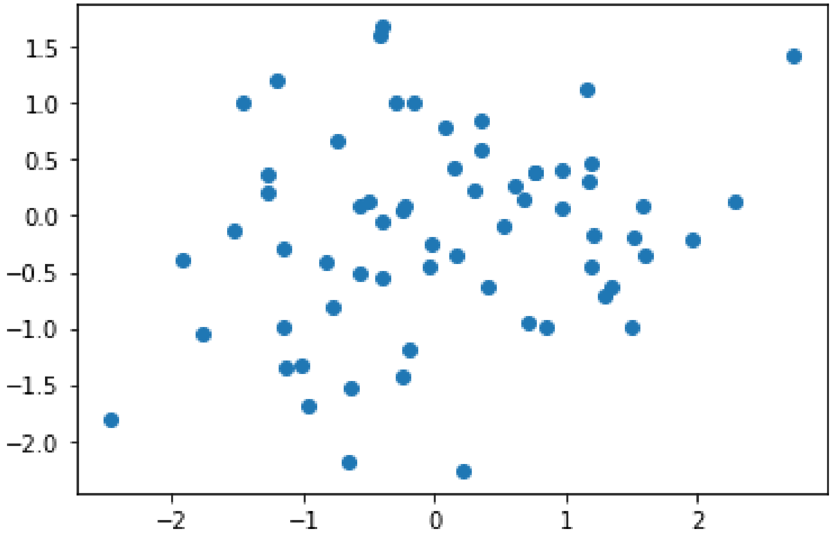
- **Scatter()**

```
import matplotlib.pyplot as plt
import numpy as np

nbPoint = 64

x = np.random.standard_normal(nbPoint)
y = np.random.standard_normal (nbPoint )

plt.scatter(x , y)
plt.show()
```



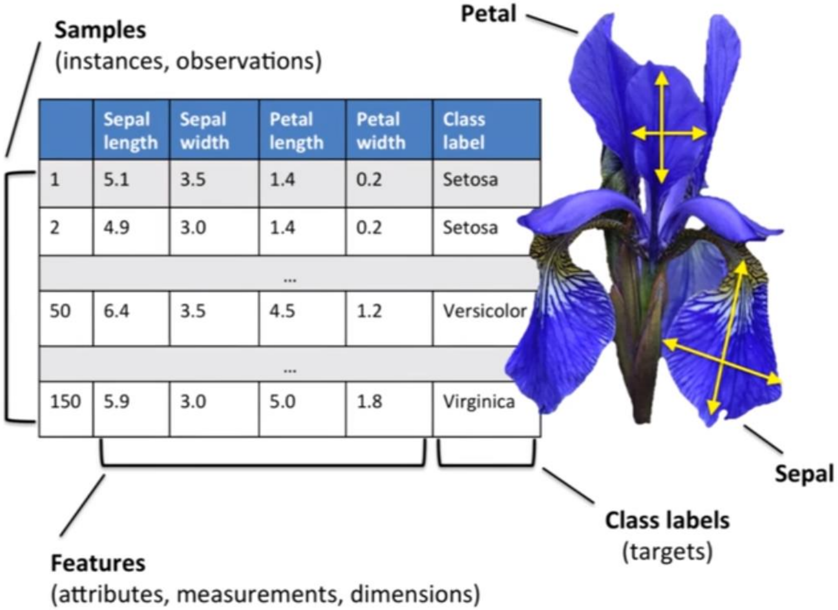
Nuage de points

- **Graphique de Classification avec Scatter()**

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn.datasets import load_iris

iris = load_iris()
x = iris.data
y = iris.target

print(f'x contient {x.shape[0]} exmples et {x.shape[1]} variables')
print(f'il y a {np.unique(y).size} classes')
```



	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
			...		
50	6.4	3.5	4.5	1.2	Versicolor
			...		
150	5.9	3.0	5.0	1.8	Virginica

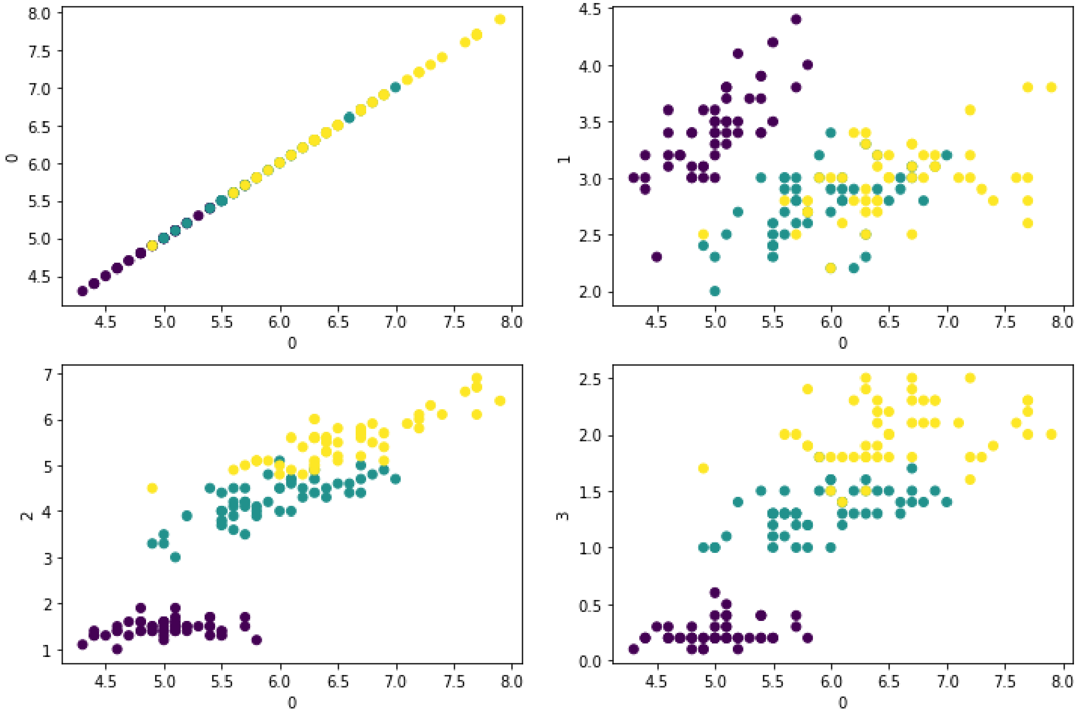
Samples
(instances, observations)
Petal
Sepal
Class labels
(targets)
Features
(attributes, measurements, dimensions)

150 exemples de fleurs d'iris répartis en trois classes

Nuage de points

- **Graphique de Classification avec Scatter()**

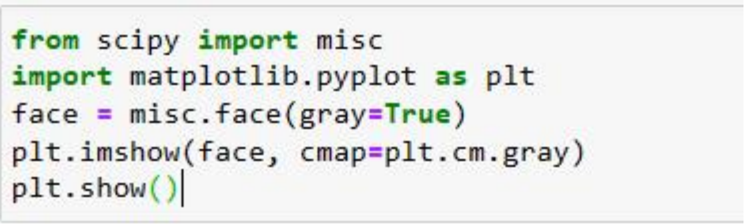
```
n = x.shape[1]
plt.figure(figsize=(12, 8))
for i in range(n):
    plt.subplot(n//2, n//2, i+1)
    plt.scatter(x[:, 0], x[:, i], c=y)
    plt.xlabel('0')
    plt.ylabel(i)
plt.show()
```



Exercice

Le programme suivant permet d’afficher une image à partir de scipy.

- Donnez le type et la taille du tableau image.
- Zoomez de ¼ vers le milieu de l’image (utiliser les dimensions du tableau avec le tuple shape et des techniques de slicing).
- Augmentez la luminosité des pixels qui ont des valeurs supérieurs à 150.



```
from scipy import misc
import matplotlib.pyplot as plt
face = misc.face(gray=True)
plt.imshow(face, cmap=plt.cm.gray)
plt.show()
```