Graphisme

Informatique pour tous

Boucle for

Pour parcourir une liste `L` (par ex. afficher ses éléments), nous avons déjà vu qu'on peut d'énumérer ses indices:

```
for i in range(len(L)):
    print(L[i])
```

Boucle for

Pour parcourir une liste `L` (par ex. afficher ses éléments), nous avons déjà vu qu'on peut d'énumérer ses indices:

```
for i in range(len(L)):
    print(L[i])
```

Il est possible d'énumérer ses éléments directement:

```
for e in L:
    print(e)
```

Boucle for

Pour parcourir une liste `L` (par ex. afficher ses éléments), nous avons déjà vu qu'on peut d'énumérer ses indices:

```
for i in range(len(L)):
    print(L[i])
```

Il est possible d'énumérer ses éléments directement:

```
for e in L:
    print(e)
```

Attention à ne pas confondre indice et élément d'une liste!

Liste par compréhension

On a souvent besoin de créer une liste `L` de la façon suivante:

```
L = []
for ... in ...:
    L.append(...)
```

Liste par compréhension

On a souvent besoin de créer une liste `L` de la façon suivante:

```
L = []
for ... in ...:
    L.append(...)
```

Il existe un code équivalent pour faire la même chose:

```
L = [... for ... in ...]
```

Liste par compréhension

On a souvent besoin de créer une liste `L` de la façon suivante:

```
L = []
for ... in ...:
    L.append(...)
```

Il existe un code équivalent pour faire la même chose:

```
L = [... for ... in ...]
```

Exemples:

```
In [1]: [i for i in range(8)]
Out[1]: [0, 1, 2, 3, 4, 5, 6, 7]
```

Liste par compréhension

On a souvent besoin de créer une liste `L` de la façon suivante:

```
L = []
for ... in ...:
    L.append(...)
```

Il existe un code équivalent pour faire la même chose:

```
L = [... for ... in ...]
```

Exemples:

```
In [1]: [i for i in range(8)]
Out[1]: [0, 1, 2, 3, 4, 5, 6, 7]
```

```
In [2]: [3 + i**2 for i in range(5)]
Out[2]: [3, 4, 7, 12, 19]
```

Liste par compréhension

Question

Stocker dans une liste `L` les 15 premiers termes de la suite $(u_n)_n$ définie par:

$$u_n = 3n + 2$$

Graphisme

Python possède de nombreux modules possédant des fonctions que l'on peut utiliser en écrivant: `import module (as ...)`

Graphisme

Python possède de nombreux modules possédant des fonctions que l'on peut utiliser en écrivant: `import module (as ...)`

On peut afficher des courbes en utilisant le module `matplotlib.pyplot`:

```
In [11]: import matplotlib.pyplot as plt
```

Graphisme

Python possède de nombreux modules possédant des fonctions que l'on peut utiliser en écrivant: `import module (as ...)`

On peut afficher des courbes en utilisant le module `matplotlib.pyplot`:

```
In [11]: import matplotlib.pyplot as plt
```

On utilise ensuite une fonction de `matplotlib.pyplot` en la préfixant par `plt`:

```
In [12]: plt.fonction(...)
```

plt.plot

Soit `X` et `Y` deux listes de même taille.

`plt.plot(X, Y)` trace un segment du point (`X[0]`, `Y[0]`) au point (`X[1]`, `Y[1]`), puis un segment de (`X[1]`, `Y[1]`) à (`X[2]`, `Y[2]`), ...

plt.plot

Soit `X` et `Y` deux listes de même taille.

`plt.plot(X, Y)` trace un segment du point (`X[0]`, `Y[0]`) au point (`X[1]`, `Y[1]`), puis un segment de (`X[1]`, `Y[1]`) à (`X[2]`, `Y[2]`), ...

Par exemple, pour tracer un segment entre les points (0, 1) et (5, 3):

```
plt.plot([0, 5], [1, 3])
```

On écrit ensuite `plt.show()` pour afficher le résultat.

plt.plot

Code complet:

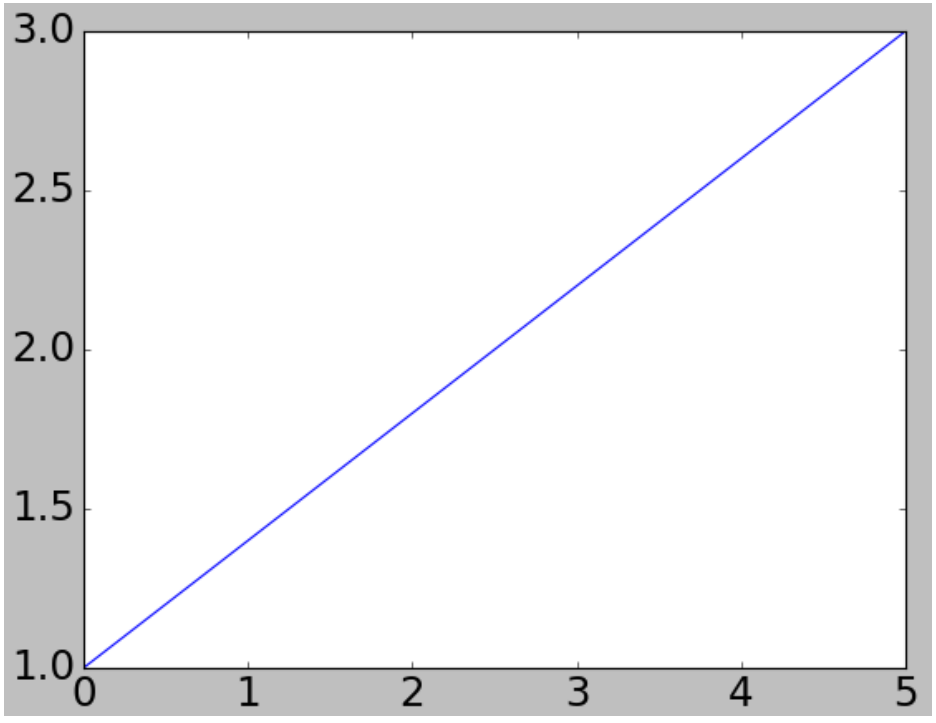
```
import matplotlib.pyplot as plt

plt.plot([0, 5], [1, 3])
plt.show()
```

plt.plot

Code complet:

```
import matplotlib.pyplot as plt

plt.plot([0, 5], [1, 3])
plt.show()
```

Résultat:



plt.plot

On peut forcer les axes à avoir la même échelle:

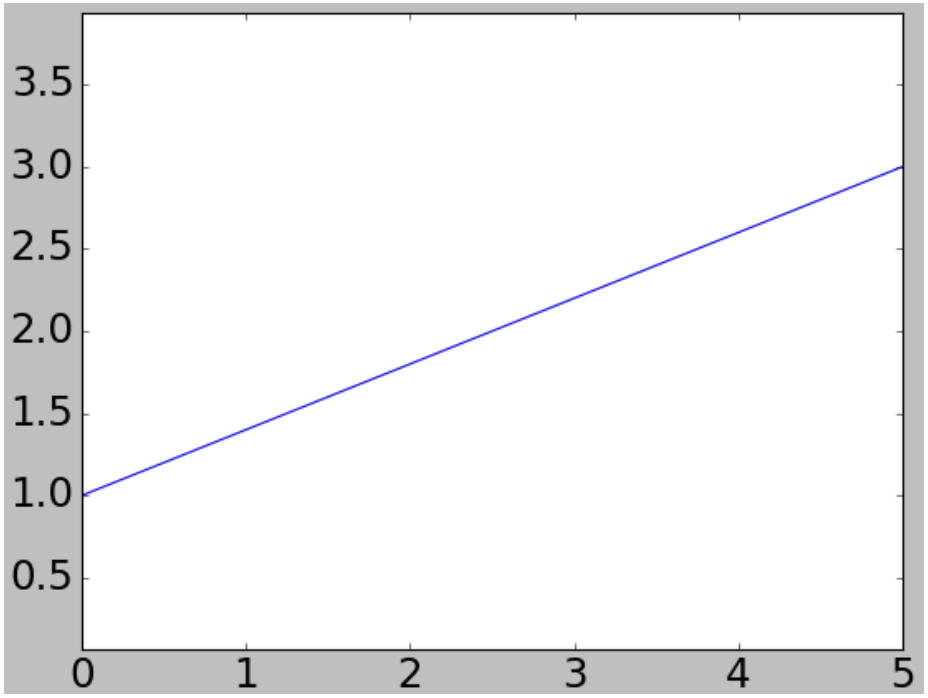
```
import matplotlib.pyplot as plt

plt.plot([0, 5], [1, 3])
plt.axis("equal")
plt.show()
```

plt.plot

On peut forcer les axes à avoir la même échelle:

```
import matplotlib.pyplot as plt

plt.plot([0, 5], [1, 3])
plt.axis("equal")
plt.show()
```

Résultat:



Courbes

Soit f une fonction.
Sa courbe représentative contient une infinité de points $(x, f(x))$: il est impossible de tous les afficher avec un ordinateur...

Courbes

Soit f une fonction.
Sa courbe représentative contient une infinité de points $(x, f(x))$: il est impossible de tous les afficher avec un ordinateur...

On peut afficher seulement certains des points $(x, f(x))$ (souvent pour des x régulièrement espacés).

Courbes

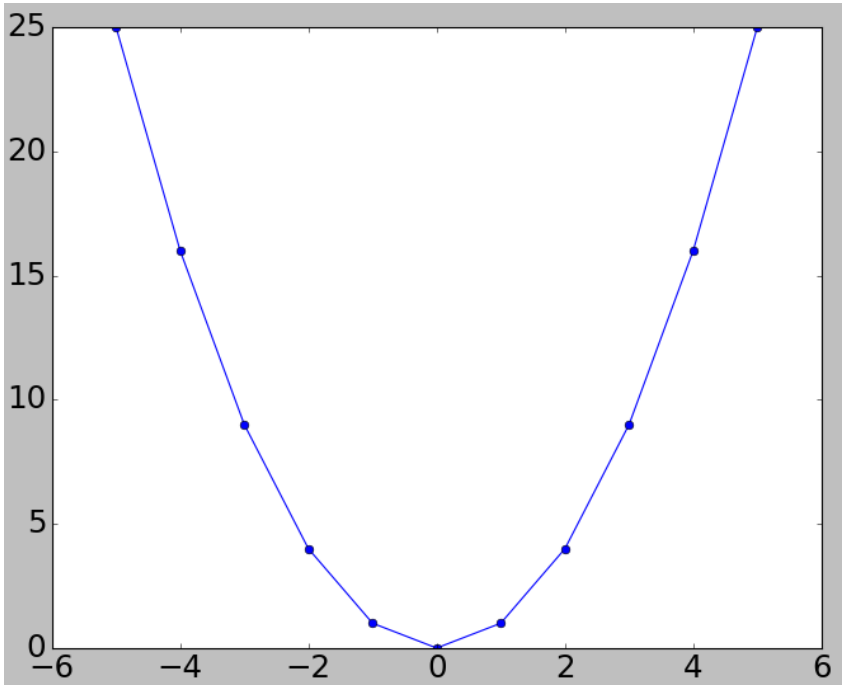
Pour afficher la fonction $x \longmapsto x^2$ sur les entiers de -5 à 5:

```
import matplotlib.pyplot as plt

X = [i for i in range(-5, 6)]
plt.plot(X, [x**2 for x in X], marker='o')
plt.show()
```

Courbes

Pour afficher la fonction $x \longmapsto x^2$ sur les entiers de -5 à 5:

```
import matplotlib.pyplot as plt

X = [i for i in range(-5, 6)]
plt.plot(X, [x**2 for x in X], marker='o')
plt.show()
```



Courbes

Pour plus de précision, on peut utiliser un pas plus petit.

Courbes

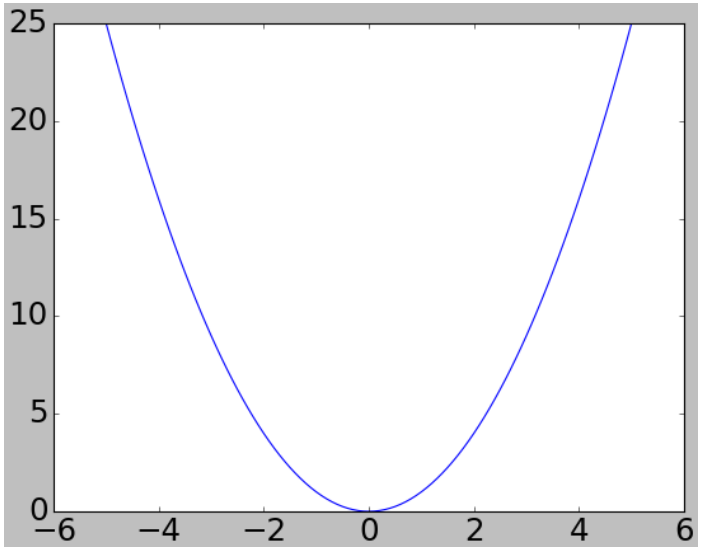
Pour plus de précision, on peut utiliser un pas plus petit.
Par exemple, avec un pas de 0.1 (c'est à dire pour les abscisses -5, -4.9, -4.8, ..., 4.9, 5):

```
X = [i*0.1 for i in range(-50, 51)]
plt.plot(X, [x**2 for x in X])
plt.show()
```

Courbes

Pour plus de précision, on peut utiliser un pas plus petit.
Par exemple, avec un pas de 0.1 (c'est à dire pour les abscisses -5, -4.9, -4.8, ..., 4.9, 5):

```
X = [i*0.1 for i in range(-50, 51)]
plt.plot(X, [x**2 for x in X])
plt.show()
```



Courbes

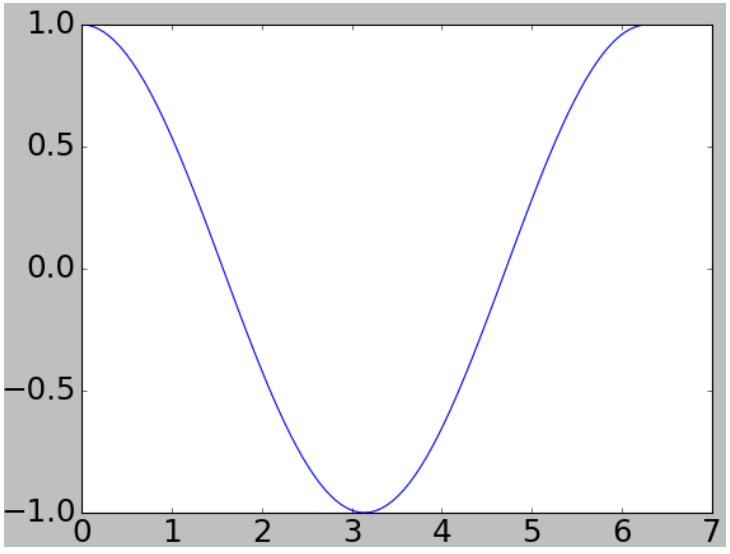
De nombreuses fonctions mathématiques existent dans le module `numpy`:

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

X = [i*0.01*2*np.pi for i in range(0, 101)]
plt.plot(X, [np.cos(x) for x in X])
plt.show()
```

Courbes

De nombreuses fonctions mathématiques existent dans le module `numpy`:

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

X = [i*0.01*2*np.pi for i in range(0, 101)]
plt.plot(X, [np.cos(x) for x in X])
plt.show()
```



Courbes

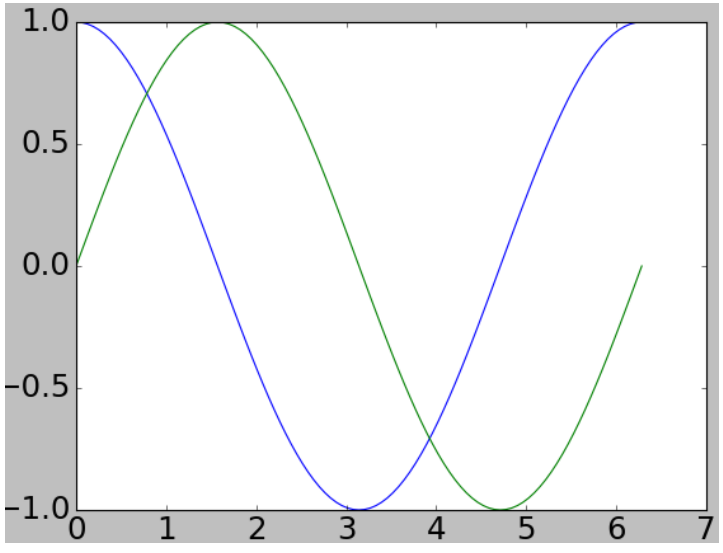
Il est possible d'afficher plusieurs fonctions sur le même dessin:

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

X = [i*0.01*2*np.pi for i in range(0, 101)]
plt.plot(X, [np.cos(x) for x in X])
plt.plot(X, [np.sin(x) for x in X])
plt.show()
```

Courbes

Il est possible d'afficher plusieurs fonctions sur le même dessin:

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

X = [i*0.01*2*np.pi for i in range(0, 101)]
plt.plot(X, [np.cos(x) for x in X])
plt.plot(X, [np.sin(x) for x in X])
plt.show()
```



Cercle

Question

Comment afficher le cercle trigonométrique (centré en 0 et de rayon 1)?

Cercle

Question

Comment afficher le cercle trigonométrique (centré en 0 et de rayon 1)?

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

T = [i*0.01*2*np.pi for i in range(0, 101)]
plt.plot([np.cos(t) for t in T], [np.sin(t) for t in T])
plt.show()
```

Cercle

Question

Écrire une fonction permettant de dessiner un cercle de centre et de rayon donné.

Cercle

Question

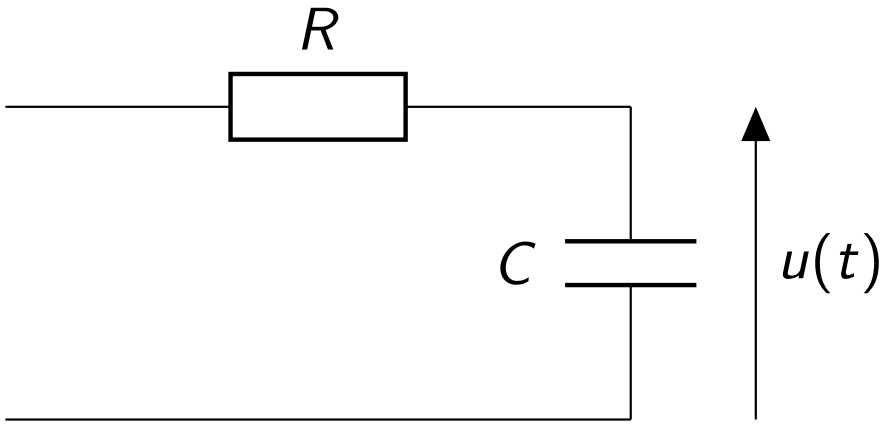
Écrire une fonction permettant de dessiner un cercle de centre et de rayon donné.

```
def cercle(c, r):
    T = [i*0.01*2*np.pi for i in range(0, 101)]
    X = [c[0] + r*np.cos(t) for t in T]
    Y = [c[1] + r*np.sin(t) for t in T]
    plt.plot(X, Y)
```

Utilisation:

```
cercle((2, 3), 2)
plt.show()
```

Exemple: décharge d'un condensateur



Question

Dessiner la tension $u(t)$ lors de la décharge d'un condensateur dans un circuit RC, vérifiant:

$$u(t) = E \times e^{-\frac{t}{RC}}$$

Exemple: décharge d'un condensateur

Question

Dessiner la tension $u(t)$ lors de la décharge d'un condensateur dans un circuit RC, vérifiant:

$$u(t) = E \times e^{-\frac{t}{RC}}$$

```
E, R, C = 1, 1, 1
T = [0.1*i for i in range(100)]
U = [E*np.exp(-t/(R*C)) for t in T]
plt.plot(T, U)
plt.show()
```

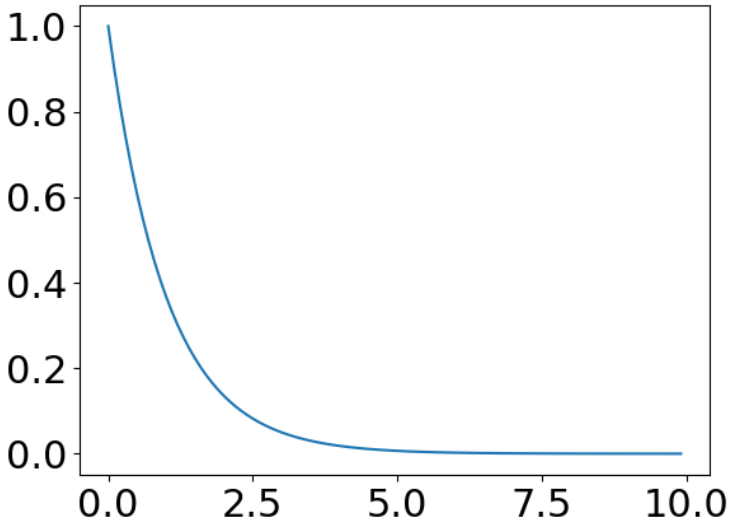
Exemple: décharge d'un condensateur

Question

Dessiner la tension $u(t)$ lors de la décharge d'un condensateur dans un circuit RC, vérifiant:

$$u(t) = E \times e^{-\frac{t}{RC}}$$

```
E, R, C = 1, 1, 1
T = [0.1*i for i in range(100)]
U = [E*np.exp(-t/(R*C)) for t in T]
plt.plot(T, U)
plt.show()
```



Un exemple un peu plus compliqué...

```
T = [i*0.1 for i in range(100)]
X = [4*(np.cos(t)**2) * np.sin(t)**3 for t in T]
Y = [(3 - 2 * np.cos(t)**2)*np.cos(t)**2 for t in T]

plt.plot(X, Y)
plt.show()
```