Feuille de TP n° 10 - Graphiques

Le sous-module `pyplot` du module `matplotlib` permet de créer des graphiques. Il contient des instructions telles que `plot`, `show`, ...

```
import matplotlib.pyplot as plt
plt.plot([5,0],[3,2]) # un segment
plt.plot([1,7,5,1],[2,2,4,1]) # une ligne polygonale
plt.show()
```

Exercice 1. Tester ces instructions.

La commande `show` permet d'afficher le graphique tracé.

Pour tracer une ligne polygonale reliant les points de coordonnées $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, on utilise les instructions suivantes :

```
x=[x1,x2, ..., xn] # liste des abscisses
y=[y1,y2, ..., yn] # liste des ordonnées
plt.plot(x,y)
plt.show()
```

La commande `plot` possède beaucoup d'options, dont par exemple :

```
plt.plot(x,y,'b:',linewidth=4,label='Ma courbe')
# couleur bleue (\texttt{'b'}), pointillés (\texttt{':'}), épaisseur 4, légende.
```

Autres abréviations pour les couleurs : `'r'`, `'g'`, `'k'`, etc (pour rouge, vert, noir, etc). Par défaut les couleurs sont automatiquement modifiées pour chaque courbe ajoutée. Autres options de tracé : `'.'`, `'o-'`, `'-'` (points, gros points reliés, tirets).

Pour obtenir la liste complète des options, on utilise l'aide de la fonction `plot` :

```
> help(plt.plot)
```

Il est également possible de configurer les axes, d'ajouter un titre, d'afficher la légende en choisissant son positionnement, etc. Ceci grâce aux commandes suivantes, à placer entre les instructions `plot` et `show` (et précédées de `plt.`).

```
axis("equal") # repère orthonormé
axis([a,b,c,d]) # bornes du dessin : a<x<b et c<y<d
xlim(a,b) # Bornes des abscisses : a<x<b
ylim(c,d) # Bornes des ordonnées : c<y<d
grid() # affichage d'une grille
axhline(color="black") # affichage des axes
axvline(color="black")
xlabel("x") # noms des axes
ylabel("y")
legend(loc="upper left") # affichage et position de la légende
title("Un joli dessin") # titre de la figure
```

Exercice 2. On rappelle que les racines n-èmes de l'unité ont pour coordonnées $\left(\cos\left(k\frac{2\pi}{n}\right), \sin\left(k\frac{2\pi}{n}\right)\right)$ pour k allant de 0 à $n-1$. Les fonctions et constantes mathématiques (dont cos, sin et π) font partie du module `math`.

1. Tracer le polygone régulier à 10 côtés formé par les racines 10-èmes de l'unité.
2. Créer un graphique contenant les polygones réguliers formés par les racines $2p$-èmes de l'unité, pour p allant de 2 à 10. Ajouter les noms des polygones (`label="p="+str(p)`) et afficher la légende en haut à droite (ce qui peut nécessiter d'agrandir la fenêtre d'affichage).
3. À partir de combien de points peut-on considérer qu'on a une bonne représentation du cercle trigonométrique ? (Il est possible d'agrandir la fenêtre)
4. Définir $n = 24$ et $m = 7$. Tracer ensuite le polygone reliant les points de coordonnées $\left(\cos\left(k\frac{2m\pi}{n}\right), \sin\left(k\frac{2m\pi}{n}\right)\right)$ pour k allant de 0 à n. Ajouter le titre « un joli dessin ». On pourra modifier les valeurs de m et n.

Module Numpy

Le module `numpy` permet d'utiliser des matrices. Il définit un nouveau type : `array`, pour représenter des tableaux multidimensionnels comme des vecteurs ou des matrices. En général, on abrège `numpy` en `np` via `import numpy as np`.

Pour l'instant nous utiliserons juste la fonction `linspace` et des fonctions vectorielles. La fonction `linspace(a,b,p)` renvoie un tableau contenant p nombres réels uniformément répartis entre a et b.
Visuellement, un tableau (array) se présente comme une liste (ou liste de listes selon la taille). Mais les opérations sur les tableaux sont radicalement différentes.

Exercice 3. Tester les instructions suivantes.

```
> import numpy as np
> LA=[0,2,4]
> LB=[1,1,5]
> type(LA) # quel est le type de LA ?
> TA=np.array([0,2,4])
> TB=np.array([1,1,5])
> type(TA) # quel est le type de LA ?
> print(LA+LB)
> print(5*LA)
> print(TA+TB)
> print(5*TA)
> print(TA[1])
```

Sur les tableaux, les opérations arithmétiques usuelles s'appliquent aux coefficients du tableau, et ne réalisent pas des opérations comme la concaténation pour des listes.

Plus généralement, il existe sur Python des fonctions dites vectorielles. Ce sont des fonctions f que l'on peut utiliser aussi bien sur des nombres que sur des tableaux de nombres (par exemple `+` ou `*`).
Si `L=[x1,...,xn]` est une liste en entrée, la fonction f renvoie alors la liste `f(L)=[f(x1),...,f(xn)]`. Le module `numpy` apporte avec lui beaucoup de fonctions usuelles définies comme fonctions vectorielles.

Exercice 4. Exécuter les instructions suivantes.

```
> import math as m
> import numpy as np
> m.sin(pi/6) # fonction sinus du module math
> np.sin(pi/6) # fonction sinus du module numpy
# une différence ?
> L=np.linspace(0,10,11)
> m.sin(L)
> np.sin(L) # et là ?
> L*pi/2 # on constate qu'on peut multiplier
# un tableau numpy par un scalaire
> np.sin(L*pi/2) # Quelles valeurs devrait-on obtenir ?
```

Il faut alors également veiller aux conflits de notations possibles possibles si deux modules contiennent des fonctions de même nom. Par exemple la fonction `sin` est présente à la fois dans le module `math` et dans le module `numpy`. C'est pour cela qu'habituellement on utilise les commandes `import module as surnom` afin d'éviter les conflits de notation.

L'avantage des tableaux pour tracer des graphiques est la simplicité de leur utilisation. L'instruction `linspace` fournit rapidement un tableau de nombres uniformément répartis, et une fonction vectorielle peut facilement modifier ce tableau de nombres.

Exercice 5. Exécuter les instructions suivantes :

```
> L=np.linspace(0,6,7)
> type(L)
> def f(x) :
>     return x**2-1
> print(L)
> print(f(L))
> plt.plot(L,f(L))
```

Si l'on veut ajouter des points, il suffit de remplacer le 7 dans la définition de `L` par 200 ou un plus grand nombre.

Exercice 6. Soit $f : x \mapsto x\sqrt{1-x^2}$.

1. Tracer sur un graphique le cercle trigonométrique. On utilisera un repère orthonormé, une grille, les axes du repère, et une légende.
   On s'aidera d'un paramétrage du cercle trigonométrique pour tracer celui-ci.
2. Tracer sur un même graphique la courbe représentative de la fonction f et le cercle trigonométrique. Afficher la zone $[-1,2;1,2] \times [-1,2;1,2]$.
3. Supprimer les axes, la grille et toute autre fioriture, et afficher les deux courbes en noir, avec une épaisseur de 16 points.

Exercice 7. Représenter graphiquement la courbe d'équation

$$\begin{cases} x = \sin(t)(1-\cos t) \\ y = \cos(t)(1-\sin t) \end{cases} \quad \text{pour } t \in [0, 2\pi].$$

Exercice 8. Représenter sur un même graphe les fonctions suivantes, sur un voisinage de 0 :

$$\begin{cases} f_0(x) = \sin x \\ f_1(x) = x \\ f_2(x) = x - \frac{x^3}{6} \\ f_3(x) = x - \frac{x^3}{6} + \frac{x^5}{120}. \end{cases}$$

Qu'observe-t-on ?

Exercice 9. Un joueur joue à pile ou face : s'il gagne, il emporte g euros, et s'il perd, il doit donner p euros. Il joue n parties. Représenter sur un graphe l'évolution de ses gains au cours du temps pour différentes valeurs de g, p et n. On utilisera la fonction `randint` du module `random`.

Exercice 10. Le diagramme de factorisation d'un entier n est un diagramme où n points sont représentés par paquets de taille identique, disposés régulièrement sur un cercle, et où chaque paquet lui-même est découpé en paquets de taille identique disposés régulièrement sur un cercle.

1. En s'adant d'un exercice précédent, représenter n points en cercle pour différentes valeurs de n. Pour représenter des points et non des segments, on utilisera la fonction `scatter` à la place de la fonction `plot`.
   L'option `s=100` permet de contrôler la taille des points.
2. Écrire une fonction `plus_grand_diviseur_premier(n)` renvoyant le plus grand diviseur premier d de n.
   On s'aidera de la commande `is_prime(d)`.
3. Écrire une fonction `diagramme(n)` renvoyant les listes `Xn` et `Yn`, respectivement, des abscisses et des ordonnées des points du diagramme de factorisation de n, sachant que :
   — si n est premier, les points de ce diagramme sont disposés sur le cercle trigonométrique.
   — sinon, en notant d le plus grand diviseur premier de n : `Xn` est la liste des $x_d + \frac{x}{d}$ où x_d parcourt `Xd` et x parcourt $X_{\frac{n}{d}}$; et de même pour `Yn`.

   On s'aidera pour cela de récursivité.
4. Ajouter comme titre l'écriture de la décomposition de n, sous la forme $15 = 3 \times 5$.
5. Afficher le diagramme de 2026.