Feuille de cours 6 (informatique) : Bibliothèques et graphiques

1 Généralités sur les bibliothèques

Si on construit souvent nos propres fonctions en Python, on utilise aussi des fonctions préexistantes comme par exemple :

Cependant, la plupart des fonctions mathématiques usuelles (cos, sin, exp, ln, etc) ne sont pas directement accessibles dans Python. Il faut les "importer" depuis des *bibliothèques* qui mettent à disposition des fonctions créées par des professionnels.

Il existe beaucoup de bibliothèques différentes. Chacune rassemble des fonctions utiles dans un domaine particulier. En BCPST nous aurons besoin essentiellement de trois bibliothèques :

- `numpy` pour manipuler des tableaux de nombres, et des fonctions mathématiques ;
- `random` pour simuler des expériences aléatoires ;
- `matplotlib.pyplot` pour dessiner des graphiques.

Signalons aussi la bibliothèque `math` pour utiliser des fonctions mathématiques (mais, sauf exception, on recommande d'utiliser `numpy`).

Pour importer une bibliothèque, il faut taper en début de script, la commande :

```
1 import ma_bibliotheque
```

où `ma_bibliotheque` est le nom de la bibliothèque voulue.

Si vous travaillez sur votre ordinateur portable, la première fois que vous utilisez une bibliothèque, il faut la télécharger depuis Internet en tapant directement dans le shell :

```
1 pip install ma_bibliotheque
```

Cette étape de téléchargement n'est nécessaire qu'à la première utilisation de la bibliothèque. Ensuite, vous pouvez utiliser simplement la commande `import`.

Une fois cette bibliothèque téléchargée, on a accès aux fonctions qu'elle contient. Toutefois, il faut rappeler à Python d'aller chercher ces fonctions dans la bibliothèque concernée. Pour cela, si l'on souhaite utiliser une fonction `fun` issue de la bibliothèque `ma_bibliotheque`, il faut utiliser la syntaxe `ma_bibliotheque.fun`.

Par exemple, la bibliothèque dont le nom est `math` contient la fonction exponentielle `exp`. Pour calculer exp(5) on écrit donc `math.exp(5)`.

Important : Il suffit d'importer la bibliothèque une seule fois dans le script pour pouvoir l'utiliser (inutile de la réimporter à chaque fonction).

Il peut être fastidieux de recopier le nom de la bibliothèque devant le nom des fonctions qu'on souhaite utiliser. On peut donc renommer la bibliothèque à l'aide d'un *alias* plus court. La syntaxe est la suivante :

```
1 import ma_bibliotheque as alias
```

pour utiliser la fonction `fun` issue de cette biblithèque, il suffit alors d'écrire `alias.fun`.

Les noms d'alias sont libres. Toutefois, certains noms se sont imposés par l'usage ; et il n'est pas recommandé de faire de fantaisie. Par exemple, la bibliothèque `numpy`, contenant elle aussi les fonctions mathématiques usuelles, s'importe généralement avec l'alias `np`. On écrit donc en préambule : `import numpy as np`

Exercice 1
Importer la bibliothèque `math` avec l'alias `m`. Cette bibliothèque contient la constante `pi` et la fonction racine carrée `sqrt`. Écrire alors une fonction `f` prenant en argument un réel x et renvoyant $\sqrt{\pi x}$.

Remarque
Il est également possible de n'importer qu'une fonction en particulier depuis une bibliothèque avec la syntaxe :

```
1 from ma_bibliotheque import ma_fonction
```

Par exemple, si je n'ai besoin que de la fonction racine carrée issue de la bibliothèque `math`, il me suffit d'écrire : `from math import sqrt`. Dans ce cas, il n'est plus nécessaire de rappeler le nom de la bibliothèque pour utiliser la fonction. On pourra écrire directement `sqrt(5)` pour accéder à $\sqrt{5}$.
On peut aussi importer toutes les focntions d'une bibliothèque et ne plus avoir besoin de rappeler le nom de la bibliothèque ou son alias. Pour cela, on utilise la syntaxe :

```
1 from ma_bibliotheque import *
```

Par exemple, en écrivant `from math import *`, on peut ensuite écrire `sqrt(sin(5))` pour calculer $\sqrt{\sin(5)}$.

Toutefois, ces deux derniers usages ne sont pas recommandés. Même si cela peut paraître plus long, utiliser un alias permet de se rappeler dans quelle bibliothèque se trouve quelle fonction, et surtoût qu'il faut penser à importer cette bibliothèque !

À retenir : la bibliothèque `numpy` contient la fonction `log` qui désigne le logarithme népérien. Attention, en Python $\ln(x)$ s'écrit donc :

Exercice 2
Importer la bibliothèqe `numpy` puis écrire une fonction `g` prenant en arguments deux réels x et y et renvoyant $\ln(\cos(x) + 2 + |y|)$.

2 Graphiques

Pour tracer des graphiques en Python, on utilise la bibliothèque `matpllotlib.pyplot` qui s'importe usuellement avec l'alias `plt`. On écrira donc en préambule :

Pour tracer un graphique, on doit disposer de deux listes `L_absi` et `L_ordo` contentant les valeurs à placer en abscisse et en ordonnées. On utilise alors la commande

```
plt.plot(L_absi,L_ordo)
```

pour tracer le graphe, puis la commande

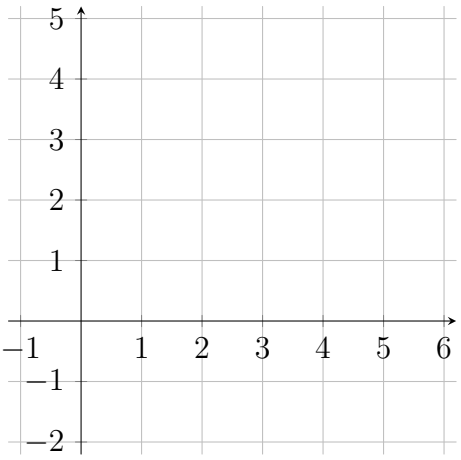
```
plt.show()
```

pour afficher ce graphe.

Remarque : sous Spyder, pour visualiser le graphe tracé, il faut aller dans l'onglet "Plot" au-dessus de la console.

Exercice 3
Dessiner le graphe affiché par Python lorsqu'on exécute le code suivant :

```
import matplotlib.pyplot as plt
L_absi = [1,2,5]
L_ordo = [2,4,-1]
plt.plot(L_absi,L_ordo)
plt.show()
```



Remarques :
Il existe de nombreuses options de dessin de graphe. Signalons les points suivants :

- par défaut, Python relie les points du graphe entre eux par des segments ;
- on peut utiliser une option pour ne pas relier les points entre eux, et mettre par exemple une croix sur chaque point : `plt.plot(absi,ordo,'x')`
- on peut utiliser des couleurs : `plt.plot(absi,ordo,'couleur')` en remplaçant `couleur` par : `r` pour rouge, `b` pour bleu, etc.
- la commande `plt.grid()` ajoute une grille au dessin.
- on peut ajouter des titres avec les commandes : `plt.title("Mon titre")`, `plt.xlabel("Titre des abscisses")`, et `plt.ylabel("Titre des ordonnés")`.
- beaucoup d'autres options existent. N'hésitez pas à taper des mots clés correspondant à ce que vous souhaitez faire dans un moteur de recherche, vous trouverez très rapidement la syntaxe appropriée et des exemples.
- Toutes ces commandes sont surtoût utiles lorsqu'on manipule des données réelles, en physique ou en biologie (et en TIPE). Sauf indication contraitre, on ne demandera pas de les utiliser en mathématiques lorsqu'on trace des graphes de fonctions.

Exercice 4
Que se passe-t-il lorsqu'on exécute le code suivant ?

```
1 import matplotlib.pyplot as plt
2 L_absi = [1,2,5]
3 L_ordo = [2,4,-1,3]
4 plt.plot(L_absi,L_ordo)
5 plt.show()
```

Python signale cette erreur par le message suivant (à ne pas ignorer) :
`ValueError: x and y must have same first dimension, but have shapes (3,) and (4,)`

Un exercice classique consiste à tracer le graphe d'une suite (u_n), c'est-à-dire à représenter u_n en fonction de n. On place donc en abscisse les valeurs de et en ordonnées les valeurs de .

Exercice 5
Soit (u_n) la suite définie par $u_0 = 3$ et $\forall n \in \mathbb{N},\ u_{n+1} = u_n^2$.

1. Écrire une fonction prenant en argument n et renvoyant la liste `[`u_0`,`u_1`,`u_2`,...,`u_n`]`.
2. Écrire un programme Python permettant de tracer u_n en fonction de n pour $n \in [\![0, 100]\!]$.

Un autre exercice classique consiste à tracer le graphe d'une fonction réelle $f : [a, b] \longrightarrow \mathbb{R}$, c'est-à-dire à représenter $f(x)$ en fonction de x. Cette fois-ci, on ne peut pas calculer *toutes* les valeurs de $f(x)$ pour $x \in [a, b]$. Il faut se contenter de choisir un nombre fini de valeurs de $x \in [a, b]$, régulièrement espacées.

Par exemple, pour tracer le graphe d'une fonction f définie sur $[1, 2]$ on pourra choisir de prendre en abscisse la liste

```
L_absi = [1, 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9, 2]
```

et alors il faudra prendre pour ordonnées la liste

```
L_ordo =
```

De manière générale, pour tracer le graphe d'une fonction f sur $[a, b]$ on peut fixer un entier N et considérer une liste d'abscisses régulièrement espacées entre a et b, chacunes séparées de la suivante par la longueur :

On obtient alors le graphe de f sur $[a, b]$ par le code suivant :

```
a =
b =
N =

L_absi =

L_ordo =

plt.plot(L_absi,L_ordo)
plt.show()
```

Remarque : combien d'éléments comportent les listes `L_absi` et `L_ordo` ?

Exercice 6
Écrire un programme Python permettant de tracer le graphe de la fonction exponentielle sur $[-2, 5]$ avec un nombre de points de tracé $N(+1)$ de votre choix. Que se passe-t-il lorsqu'on fait varier N ?

3 Entraînement

Exercice 7

Soit (u_n) la suite définie par $u_1 = 10$ et $\forall n \in \mathbb{N},\ u_{n+1} = \sin(u_n)$.

1. Écrire une fonction prenant en argument n et renvoyant la liste `[`u_1`,`u_2`,`u_3`,...,`u_n`]`.
2. Écrire un programme Python permettant de tracer u_n en fonction de n pour $n \in [\![1, 20]\!]$.

Exercice 8

1. Écrire une fonction f prenant en argument un réel x et renvoyant $\ln(x^2 + 1)$.
2. Écrire un programme Python premettant de tracer le graphe de f sur $[-2, 2]$ avec le nombre de points de tracé de votre choix.