TP05 : Introduction aux graphismes

L'objectif de ce TP est la découverte des commandes graphiques élémentaires en Python, permettant d'obtenir des représentations graphiques de fonctions ou de suites.

1 La bibliothèque graphique de Python

Pour réaliser des figures en Python, il est courant de se tourner vers la bibliothèque graphique standard, nommée `matplotlib` et plus précisément vers le module `pyplot` de cette bibliothèque. Avant toute chose, il est nécessaire d'importer ce module, ce que l'on fait avec la commande ci-dessous :

```
import matplotlib.pyplot as plt
```

Le module importé par cette commande s'appelle `matplotlib.pyplot` et l'utilisation du mot-clef `as` permet de lui donner un **alias** plus court, à savoir `plt`. Ceci permet de raccourcir toutes les commandes graphiques dépendant de ce module, en remplaçant partout `matplotlib.pyplot` par `plt`.

Une fois que la bibliothèque graphique est importée, on peut commencer à réaliser des figures très rapidement avec les fonctions `plot` et `show`, dont le fonctionnement est présenté ci-dessous.

Définition 1.1 : Les fonctions `plt.plot` et `plt.show`

Dans sa version la plus basique, la fonction `plt.plot` prend en paramètres deux listes de même longueur :

- `listeX` contenant les abscisses des points à placer
- `listeY` contenant les ordonnées des points à placer

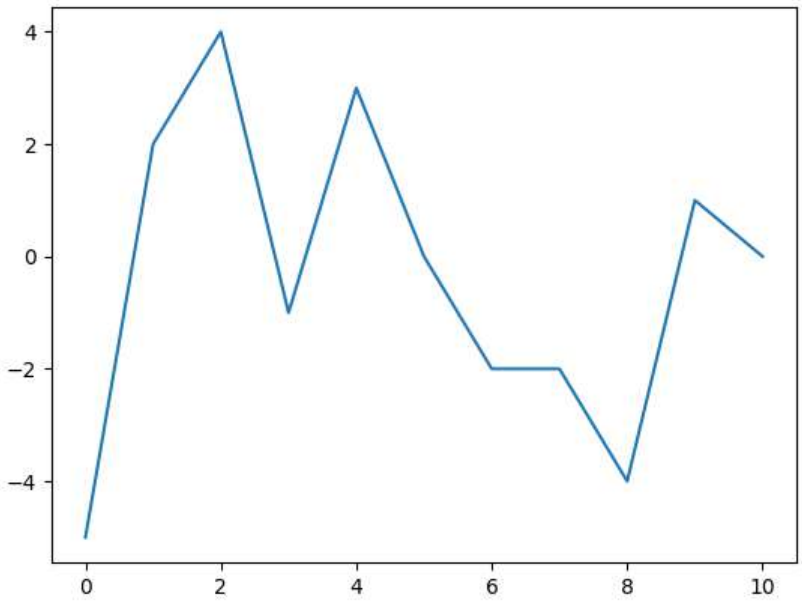
À partir de telles listes, la commande :

`plt.plot(listeX,listeY)`

construit par défaut une figure en reliant successivement par des segments de droite les points du plan dont les abscisses et ordonnées apparaissent dans les listes `listeX` et `listeY`.

La commande `plt.show()` permet ensuite de visualiser la figure obtenue.

```
import matplotlib.pyplot as plt
listeX = [0,1,2,3,4,5,6,7,8,9,10]
listeY = [-5,2,4,-1,3,0,-2,-2,-4,1,0]
plt.plot(listeX,listeY)
plt.show()
```



Il existe de nombreuses manières additionnelles de paramétrer la figure obtenue : on peut changer la position de la fenêtre graphique, faire apparaître un quadrillage, ajouter un titre, des légendes pour les axes ou les courbes tracées, modifier les paramètres de style des courbes tracées avec des choix de marqueurs, de couleur ou de type de ligne, et encore beaucoup d'autres choses.

Définition 1.2 : Paramètres de style

La fonction `plt.plot` accepte un paramètre de style en plus des listes de coordonnées, sous la forme :

```
plt.plot(listeX,listeY,"mlc")
```

où `m` est un paramètre de style de marqueur, `l` un paramètre de style de ligne et `c` un paramètre de couleur.

Les (principaux) paramètres de style de marqueur possibles sont :

Code	.	o	v ou ^ ou < ou >	s	*	+	P
Marqueur	Petits points	Gros points	Triangles	Carrés	Étoiles	Plus	Gros Plus

Code	x	X	p	h	D	| ou _
Marqueur	Croix	Grosse croix	Pentagones	Hexagones	Diamants	Trait vertical ou horizontal

Les paramètres de style de ligne possibles sont :

Code	-	--	-.	:
Ligne	Trait plein	Tirets	Tirets et points alternés	Pointillés

Les paramètres de couleur possibles sont :

Code	w	k	r	g	b	c	m	y
Couleur	Blanc	Noir	Rouge	Vert	Bleu	Cyan	Magenta	Jaune

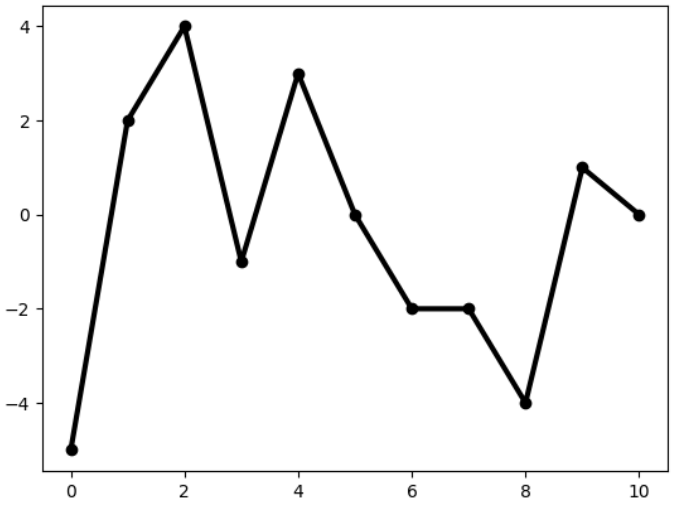
On peut tout à fait ne pas spécifier tous ces paramètres et n'en définir qu'un ou deux. Par exemple :

- `plt.plot(listeX,listeY,".r")` produira une figure avec des petits points rouges
- `plt.plot(listeX,listeY,"*-b")` produira une figure avec des étoiles bleues reliées par des traits
- `plt.plot(listeX,listeY,"D:g")` produira une figure avec des diamants verts reliés par des pointillés

Enfin, il est également possible de modifier la largeur du trait avec le paramètre `lw` (pour linewidth).

Par exemple, la commande `plt.plot(listeX,listeY,"o-k",lw=3)` produira une figure avec des gros points noirs reliées par des traits de largeur 3, comme on le voit ci-dessous.

```
import matplotlib.pyplot as plt

listeX = [0,1,2,3,4,5,6,7,8,9,10]
listeY = [-5,2,4,-1,3,0,-2,-2,-4,1,0]

plt.plot(listeX,listeY,"o-k",lw=3)
plt.show()
```



Il est également possible d'ajouter des éléments de lisibilité à une figure parmi lesquels on compte le titre, les légendes pour les axes ou les courbes ainsi que le quadrillage.

Définition 1.3 : Paramètres descriptifs

On peut ajouter des paramètres descriptifs en utilisant certaines des commandes ci-dessous après l'utilisation de la fonction `plt.plot` et avant l'utilisation de la commande `plt.show()`.

Pour ajouter un titre à la figure, on utilise la commande :

```
plt.title("Titre de la figure")
```

Pour ajouter un quadrillage à la figure, on utilise la commande :

```
plt.grid()
```

Pour ajouter une légende à l'axe des abscisses, on utilise la commande :

```
plt.xlabel("Légende en abscisse")
```

Pour ajouter une légende à l'axe des ordonnées, on utilise la commande :

```
plt.ylabel("Légende en ordonnée")
```

Enfin, pour ajouter une légende pour la courbe, on utilise la fonction `plt.plot` avec le paramètre additionnel `label` au moment de la création de la courbe, par exemple sous la forme

```
plt.plot(listeX,listeY,label="Légende")
```

puis on utilise la commande :

```
plt.legend()
```

De plus, il est possible de régler les coordonnées de la fenêtre d'affichage manuellement, même si le choix par défaut est souvent tout à fait pertinent et nécessite rarement une modification.

Définition 1.4 : Paramètres d'affichage

Pour choisir les valeurs minimales et maximales des abscisses et des ordonnées de la fenêtre graphique, on utilisera, après la fonction `plt.plot` mais avant la commande `plt.show()`, la commande :

```
plt.axis([xmin, xmax, ymin, ymax])
```

où `xmin` est la valeur de l'abscisse minimale, `xmax` est la valeur de l'abscisse maximale, `ymin` est la valeur de l'ordonnée minimale et `ymax` est la valeur de l'ordonnée maximale.

Il est également possible d'agir uniquement sur les abscisses en utilisant la commande :

```
plt.xlim(xmin,xmax)
```

De même, il est également possible d'agir uniquement sur les ordonnées en utilisant la commande :

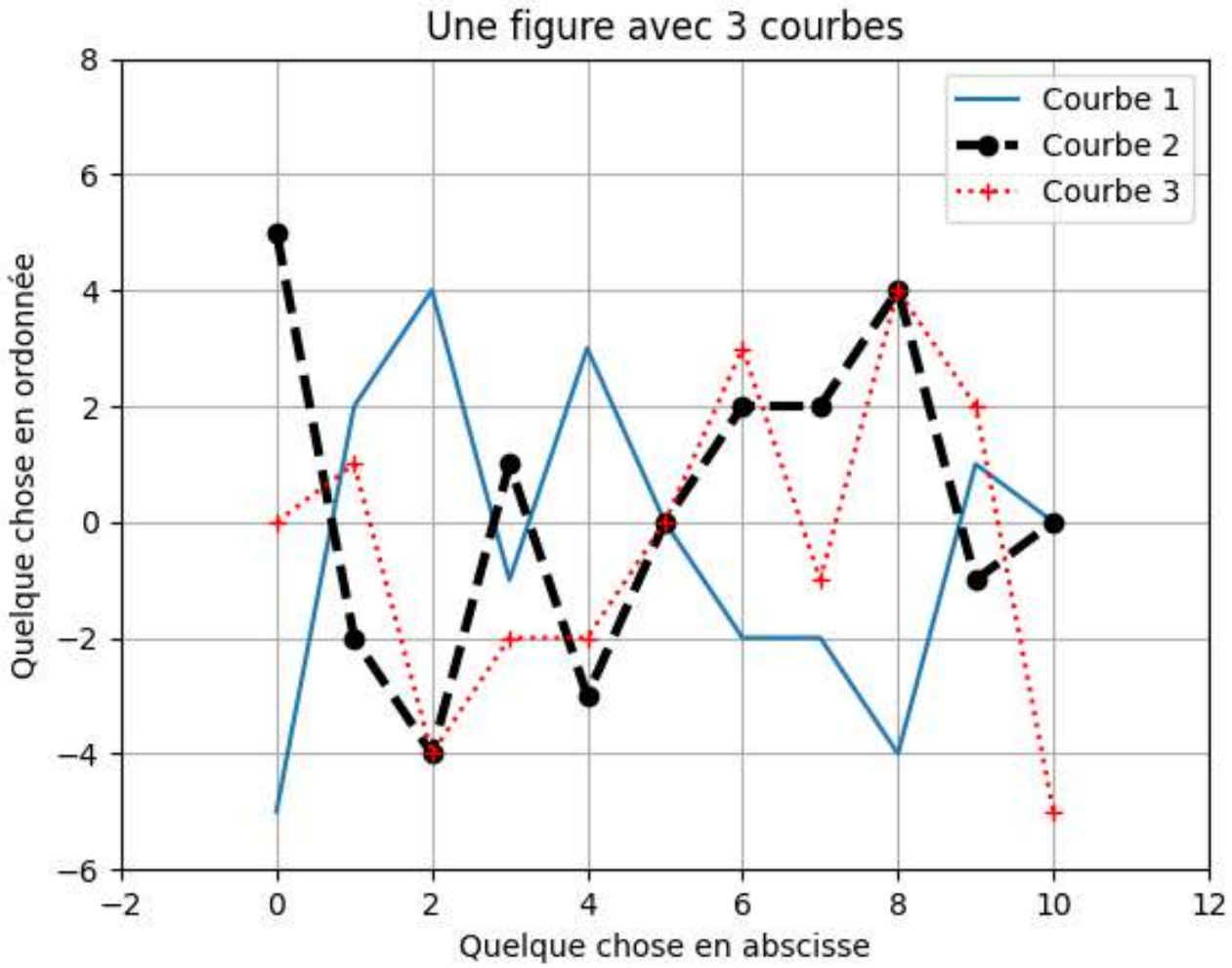
```
plt.ylim(ymin,ymax)
```

Enfin, il est tout à fait possible d'utiliser plusieurs fois la fonction `plt.plot` pour tracer plusieurs courbes simultanément sur une même figure, tout en appliquant des paramètres de style indépendants à ces différentes courbes.

On notera qu'en dehors des fonctions `plt.plot` et `plt.show`, aucune des commandes présentées ci-dessus n'est exigible pour les concours. Leur utilisation en TP pourra toutefois s'avérer très pratique, il est donc tout de même recommandé de s'entraîner à les manipuler.

Pour conclure cette introduction, on présente ci-dessous un tracé de plusieurs courbes illustrant l'utilisation de divers paramètres descriptifs.

```
import matplotlib.pyplot as plt

listeX = [0,1,2,3,4,5,6,7,8,9,10]
listeY1 = [-5,2,4,-1,3,0,-2,-2,-4,1,0]
listeY2 = [5,-2,-4,1,-3,0,2,2,4,-1,0]
listeY3 = [0,1,-4,-2,-2,0,3,-1,4,2,-5]

plt.plot(listeX,listeY1,label="Courbe 1")
plt.plot(listeX,listeY2,"o--k",lw=3,label="Courbe 2")
plt.plot(listeX,listeY3,"+:r",label="Courbe 3")

plt.axis([-2, 12, -6, 8])
plt.grid()
plt.legend()

plt.title("Une figure avec 3 courbes")
plt.xlabel("Quelque chose en abscisse")
plt.ylabel("Quelque chose en ordonnée")

plt.show()
```



Enfin, on pourra noter que le paramètre `listeX` de la fonction `plt.plot` est facultatif : en son absence, la fonction fabriquera elle-même cette liste de valeurs en abscisses, qui seront par défaut les entiers naturels compris entre `0` et `len(listeY)-1`, ce qui est équivalent à l'utilisation de `listeX=range(0,len(listeY))`.

2 Représentations graphiques de suites réelles

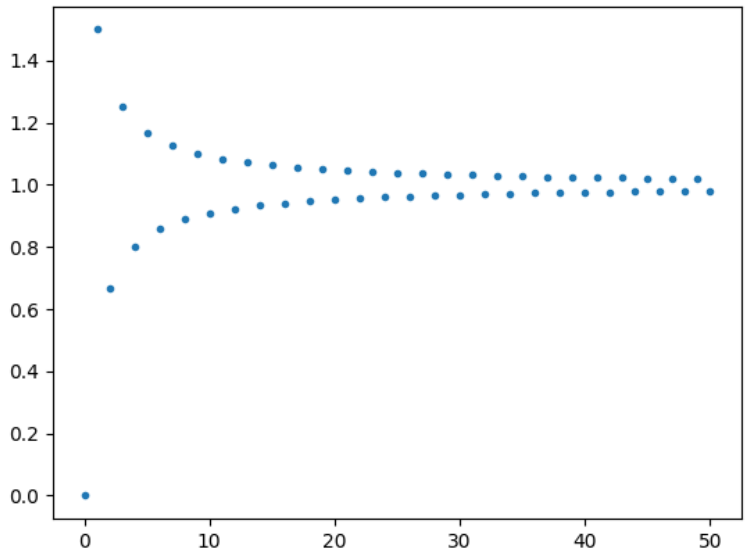
Les commandes de la bibliothèque graphique de Python découvertes dans la section précédente permettent d'afficher la représentation graphique d'un nombre fini de termes d'une suite réelle, qu'elle soit définie explicitement ou par récurrence, à condition d'être capable de produire deux listes :

- `listeX` contenant les abscisses des points à représenter, par exemple les entiers $n \in [\![0, 50]\!]$
- `listeY` contenant les ordonnées des points à représenter, par exemple les termes u_0 à u_{50} de la suite

Pour construire la première liste, la fonction `range` est tout à fait indiquée, tandis que pour construire la seconde, on aura recours à une définition de liste en compréhension dans le cas d'une suite définie explicitement et à l'utilisation d'une boucle `for` dans le cas d'une suite définie par une relation de récurrence.

Considérons par exemple la suite $(u_n)_{n\in\mathbb{N}}$ définie pour tout $n \in \mathbb{N}$ par $u_n = 1 - \dfrac{(-1)^n}{n+1}$. Alors on peut obtenir une représentation graphique, avec un style approprié, des valeurs $(u_n)_{n\in[\![0,50]\!]}$ avec le code ci-dessous :

```
import matplotlib.pyplot as plt

listeX = range(0,51)
listeY = [1-(-1)**n/(n+1) for n in listeX]

plt.plot(listeX,listeY,".")
plt.show()
```



Définition 2.1 : Représentation graphique d'une suite définie explicitement

Soit $(u_n)_{n\in\mathbb{N}}$ une suite réelle et $a, b \in \mathbb{N}$ tels que $a \leq b$.

On suppose que l'on dispose d'une fonction Python, nommée `u`, prenant en paramètre un entier naturel `n` et renvoyant en sortie la valeur de u_n.

Alors on peut afficher la représentation graphique des valeurs $(u_k)_{k\in[\![a,b]\!]}$ avec le code ci-dessous :

```
import matplotlib.pyplot as plt

listeX = range(a,b+1)
listeY = [u(n) for n in listeX]

plt.plot(listeX,listeY,".")
plt.show()
```

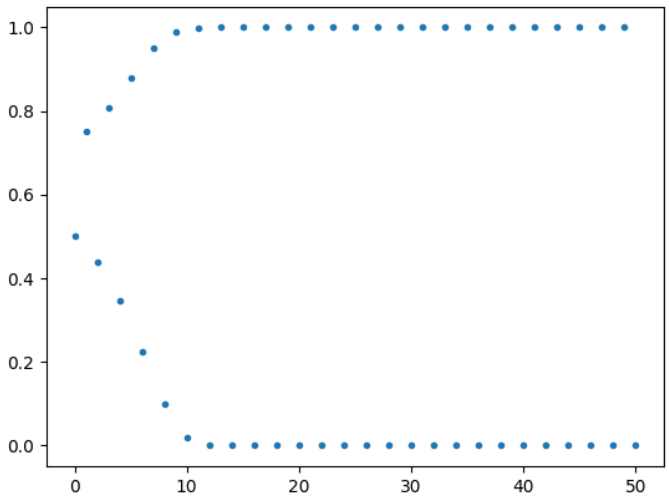
Dans le cas d'une suite définie par une relation de récurrence, on peut procéder de manière identique, mais ce n'est pas recommandé pour des raisons de temps de calcul : en effet, utiliser la syntaxe `[u(n) for n in listeX]` est catastrophique dans ce cas, car pour obtenir chaque valeur `u(n)`, tous les termes de la suite sont calculés jusqu'au rang `n`.

Il faut donc dans ce cas préférer l'utilisation d'une fonction qui calcule de proche en proche les termes de la suite jusqu'au rang `n`, tout en les stockant au fur et à mesure dans une liste, comme cela a été vu au TP04.

Considérons par exemple la suite $(u_n)_{n\in\mathbb{N}}$ définie par $u_0 = \dfrac{1}{2}$ et pour tout $n \in \mathbb{N}$, $u_{n+1} = 1 - (u_n)^2$.

Alors on peut obtenir une représentation graphique des valeurs $(u_n)_{n\in[\![0,50]\!]}$ avec le code ci-dessous :

```
import matplotlib.pyplot as plt

def termes_u(n):
  a=1/2
  L=[a]
  for i in range(1,n+1):
    a=1-a**2
    L.append(a)
  return L

listeX = range(0,51)
listeY = termes_u(50)
plt.plot(listeX,listeY,".")
plt.show()
```



Définition 2.2 : Représentation graphique d'une suite définie par récurrence

Soit $(u_n)_{n\in\mathbb{N}}$ une suite réelle.

On suppose que l'on dispose d'une fonction Python, nommée `termes_u`, prenant en paramètre un entier naturel `n` et renvoyant en sortie la liste des valeurs de u_0 à u_n.

Alors on peut afficher la représentation graphique des valeurs $(u_k)_{k\in[\![0,n]\!]}$ avec le code ci-dessous :

```
import matplotlib.pyplot as plt

listeX = range(0,n+1)
listeY = termes_u(n)
plt.plot(listeX,listeY,".")
plt.show()
```

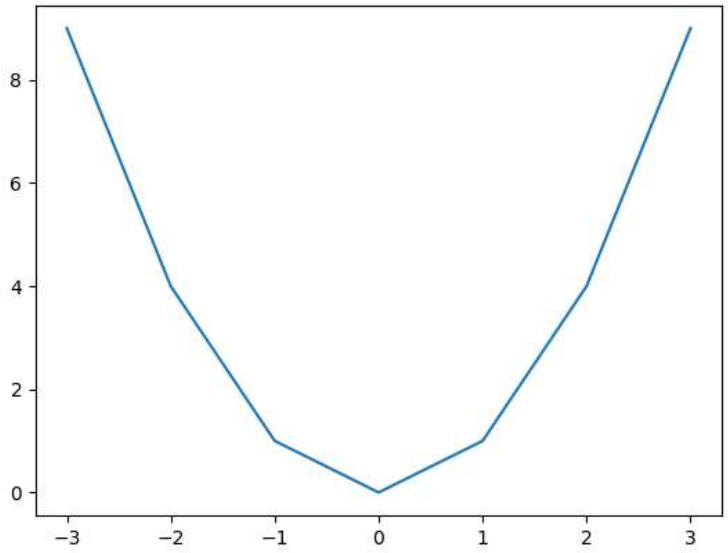
Dans le cas d'une suite $(u_n)_{n\in\mathbb{N}^*}$, il convient d'adapter la numérotation dans la fonction `range`, en remplaçant `range(0,n+1)` par `range(1,n+1)`, la fonction `termes_u` renvoyant alors la liste des valeurs de u_1 à u_n.

3 Représentations graphiques de fonctions

La représentation graphique d'une fonction est tout à fait analogue à celle d'une suite définie par une formule explicite, à ceci près qu'il n'est en général pas satisfaisant de se contenter d'utiliser une liste d'abscisses `listeX` contenant uniquement des valeurs entières, car le tracé de la courbe peut être très imprécis dans ce cas.

Observons ci-dessous le tracé de la fonction carré sur l'intervalle $[-3,3]$ à partir d'une liste de valeurs en abscisse égale à `[-3,-2,-1,0,1,2,3]`, que l'on obtient avec la commande `range(-3,4)`.

```
import matplotlib.pyplot as plt

listeX = range(-3,4)
listeY = [x**2 for x in listeX]

plt.plot(listeX,listeY)
plt.show()
```



Pour améliorer la précision de ce tracé, il est nécessaire de remplacer `listeX` par une liste de valeurs allant de -3 à 3 avec un pas inférieur à 1, sans saisir ces valeurs directement à la main (ce qui serait beaucoup trop long). Pour remplir cet objectif, il est possible d'adapter l'utilisation de la commande `range`, qui présente le défaut de ne manipuler que des bornes et des pas entiers, mais cela requiert un certain nombre de contorsions.

En pratique, on se tourne plutôt dans ce cas vers une commande fournie par une autre bibliothèque classique de Python : la fonction `linspace` du module `numpy`.

Le module `numpy` (qui sera étudié en détails plus tard) s'importe via la commande :

```
import numpy as np
```

Définition 3.1 : La fonction `linspace`

Soient $a,b \in \mathbb{R}$ tels que $a<b$ et $n \in \mathbb{N}\setminus\{0,1\}$. Alors la commande

`np.linspace(a,b,n)`

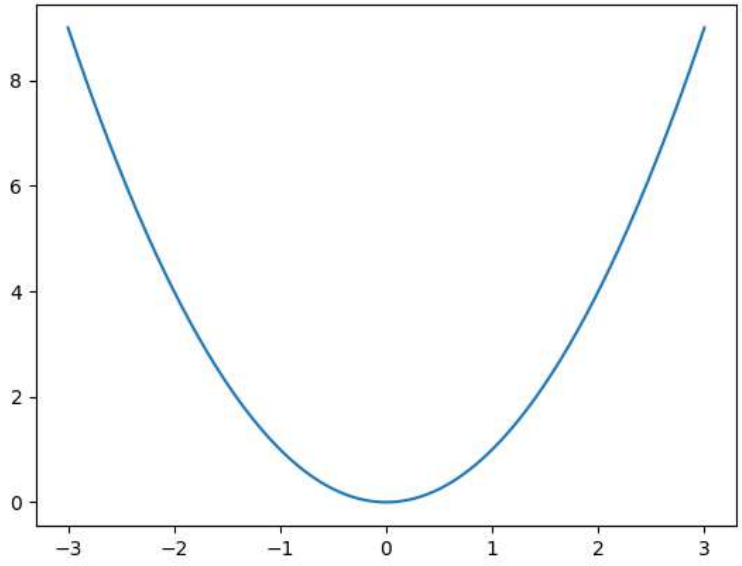
renvoie la liste des n premières valeurs de l'unique suite arithmétique $(u_k)_{k\in\mathbb{N}^*}$ telle que $u_1=a$ et $u_n=b$.

Par exemple, la liste `[-3,-2,-1,0,1,2,3]` obtenue par la commande `range(-3,4)` peut aussi être obtenue par la commande `np.linspace(-3,3,7)` (car cette liste est de longueur 7).

Pour obtenir une liste de valeurs allant de -3 à 3 avec un pas non-entier de $\dfrac{1}{100}$, ce qui fait en tout 601 valeurs, on peut donc utiliser la commande `np.linspace(-3,3,601)`.

On peut donc obtenir un tracé plus précis de la courbe représentative de la fonction carré sur l'intervalle $[-3,3]$ avec le code ci-dessous :

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

listeX = np.linspace(-3,3,601)
listeY = [x**2 for x in listeX]

plt.plot(listeX,listeY)
plt.show()
```



Définition 3.2 : Représentation graphique d'une fonction définie sur un segment

Soient $a,b\in\mathbb{R}$ tels que $a<b$, $n\in\mathbb{N}\setminus\{0,1\}$ et $f:[a,b]\longrightarrow\mathbb{R}$ une fonction.

On suppose que l'on dispose d'une fonction Python, nommée `f`, prenant en paramètre un nombre réel `x` appartenant à l'intervalle $[a,b]$ et renvoyant en sortie la valeur de $f(x)$.

Alors on peut afficher la représentation graphique de f sur l'intervalle $[a,b]$ construite avec n points en utilisant le code ci-dessous :

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

listeX = np.linspace(a,b,n)
listeY = [f(x) for x in listeX]

plt.plot(listeX,listeY)
plt.show()
```

Enfin, notons que pour obtenir une représentation graphique d'une fonction f définie sur un domaine qui n'est pas un segment, comme $]0,1]$ ou $\mathbb{R}_+$, ou $\mathbb{R}^*$, ou encore $[-2,-1]\cup[1,2]$, il faut procéder à des ajustements :

- Si f est définie sur un intervalle I borné mais ne contenant pas l'une de ses bornes, alors on effectue la représentation graphique sur un segment contenu dans I, par exemple en remplaçant l'intervalle $I=]0,1]$ par l'intervalle $[0.001,1]$.
- Si f est définie sur un intervalle I non-borné, alors on choisit d'effectuer la représentation graphique sur un sous-intervalle borné de I, par exemple en remplaçant $I=\mathbb{R}_+$ par $I=[0,100]$.
- Si f est définie sur un domaine $\mathcal{D}$ qui n'est pas un intervalle, mais une union d'intervalles, alors on effectue un tracé pour chaque intervalle maximal contenu dans le domaine de f, par exemple, si $\mathcal{D}=[-2,-1]\cup[1,2]$, alors on effectuera un tracé sur l'intervalle $[-2,-1]$ et un tracé sur l'intervalle $[1,2]$ dans la même figure.

4 Exercices

Exercice 1 Soit $(u_n)_{n\in\mathbb{N}}$ la suite réelle définie pour tout $n \in \mathbb{N}$ par $u_n = \dfrac{1+(-1)^n}{2}$.

Représenter graphiquement les valeurs $(u_n)_{n\in[\![0,50]\!]}$.

Exercice 2 Soient $(u_n)_{n\in\mathbb{N}}$ et $(v_n)_{n\in\mathbb{N}}$ les suites réelles définies pour tout $n \in \mathbb{N}$ par $u_n = n$ et $v_n = 2\left\lfloor \dfrac{n}{2} \right\rfloor$.

Représenter graphiquement les valeurs $(u_n)_{n\in[\![0,20]\!]}$ et $(v_n)_{n\in[\![0,20]\!]}$ sur une même figure.

Quelle propriété mathématique est illustrée par cette figure ? Démontrer cette propriété.

Exercice 3 Soit $(u_n)_{n\in\mathbb{N}}$ la suite réelle définie par $u_0 = 0$ et pour tout $n \in \mathbb{N}$, $u_{n+1} = \dfrac{1}{1+u_n}$.

Représenter graphiquement les valeurs $(u_n)_{n\in[\![0,20]\!]}$.

Que peut-on conjecturer grâce à cette figure ?

Exercice 4 Soit $(u_n)_{n\in\mathbb{N}}$ la suite réelle définie par $u_0 = 1$, $u_1 = 2$ et pour tout $n \in \mathbb{N}$, $u_{n+2} = u_{n+1} - u_n$.

Représenter graphiquement les valeurs $(u_n)_{n\in[\![0,30]\!]}$.

Que peut-on conjecturer grâce à cette figure ? Démontrer cette conjecture.

Exercice 5 Soit $(u_n)_{n\in\mathbb{N}}$ une suite réelle telle que pour tout $n \in \mathbb{N}$, $u_{n+1} = 3 + 2\sqrt{u_n - 3}$.

Représenter sur une même figure les valeurs $(u_n)_{n\in[\![0,25]\!]}$ dans chacun des cas suivants :

(a) $u_0 = 3$ (b) $u_0 = 4$ (c) $u_0 = 7$ (d) $u_0 = 8$

Exercice 6 Soient $(S_n)_{n\in\mathbb{N}^*}$ et $(T_n)_{n\in\mathbb{N}^*}$ les suites définies pour tout $n \in \mathbb{N}^*$ par $S_n = \displaystyle\sum_{k=1}^{n} \frac{1}{k}$ et $T_n = \displaystyle\sum_{k=1}^{n} \frac{1}{k^2}$.

Représenter graphiquement les valeurs $(S_n)_{n\in[\![0,50]\!]}$ et $(T_n)_{n\in[\![0,50]\!]}$ sur une même figure.

Quels résultats vus en cours sont illustrés par cette figure ?

Exercice 7 Représenter sur une même figure les courbes représentatives des fonctions $x \longmapsto e^x$, $x \longmapsto 1+x$ et $x \longmapsto 1 + xe^x$ sur l'intervalle $[-2, 2]$ avec un quadrillage.

Quelle propriété mathématique est illustrée par cette figure ? Démontrer cette propriété.

Exercice 8 Représenter sur une même figure les courbes représentatives des fonctions $x \longmapsto \ln(x)$, $x \longmapsto x-1$ et $x \longmapsto \dfrac{x-1}{x}$ sur l'intervalle $[0.1, 2]$ avec un quadrillage.

Quelle propriété mathématique est illustrée par cette figure ? Démontrer cette propriété.

Exercice 9 Représenter sur une même figure les courbes représentatives des fonctions $x \longmapsto \lfloor x \rfloor$, $x \longmapsto x-1$ et $x \longmapsto x$ sur l'intervalle $[-3, 3]$ avec 601 points et un quadrillage.

Quelle propriété mathématique est illustrée par cette figure ?

Exercice 10 Tracer la représentation graphique de la fonction `signe` sur l'intervalle $[-3, 3]$ (voir TP02 p.5).

Exercice 11 Pour tout $x \in \mathbb{R}$, posons $f(x) = x^3 - 3x - 1$ et $g(x) = |f(x)|$.

Utiliser Python pour représenter sur une même figure les courbes représentatives de f et de g, avec des épaisseurs différentes, sur l'intervalle $[-3, 3]$, en faisant figurer un quadrillage.

Exercice 12 Pour tout $x \in \mathbb{R}$, posons $f(x) = \dfrac{e^x - e^{-x}}{6}$.

Il est possible de démontrer que f réalise une bijection de $\mathbb{R}$ sur $\mathbb{R}$, mais pour traiter cet exercice, on ne cherchera pas à déterminer explicitement f^{-1}.

Utiliser Python pour représenter sur une même figure les courbes représentatives de f et de f^{-1} sur l'intervalle $[-3, 3]$, en faisant figurer également la droite d'équation $y = x$ en pointillés et un quadrillage.

Exercice 13 Pour tout $x \in \mathbb{R}$, posons $f(x) = 2x^3 - 5x^2 + x - 5$.

Il n'est pas déraisonnable d'estimer que pour tout $x \in \mathbb{R}$, $f'(x) \simeq \dfrac{f(x+h) - f(x)}{h}$ avec $h = 10^{-6}$.

Utiliser Python pour représenter sur une même figure les courbes représentatives de f et f' sur l'intervalle $[-2, 4]$, sans déterminer explicitement f', en faisant figurer un quadrillage.

Exercice 14 Le but de cet exercice est l'étude d'une famille de suites récurrentes de nombres entiers naturels très célèbre en mathématiques par la difficulté des problèmes qu'elle soulève et pourtant très abordable du point de vue de la programmation.

Suites de Syracuse

Soit $N \in \mathbb{N}$. La suite de Syracuse de paramètre N est la suite réelle $(u_n)_{n\in\mathbb{N}}$ définie par $u_0 = N$ et :

$$\forall n \in \mathbb{N}, \quad u_{n+1} = \begin{cases} \dfrac{u_n}{2} & \text{si } u_n \text{ est pair} \\ 3u_n + 1 & \text{si } u_n \text{ est impair} \end{cases}$$

Par exemple, si $N = 24$, alors on obtient la suite dont les premiers termes sont donnés par le tableau :

n	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
u_n	24	12	6	3	10	5	16	8	4	2	1	4	2	1	4	2	1

(*a*) Écrire un code Python permettant de construire automatiquement une liste contenant les valeurs $(u_n)_{n\in[\![0,16]\!]}$ de la suite de Syracuse de paramètre $N = 24$ (sans les recopier à la main grâce au tableau ci-dessus, bien entendu).
Produire une représentation graphique des termes de cette suite, en choisissant un style adapté.

(*b*) Écrire le code d'une fonction Python, nommée `Syracuse`, prenant en paramètres deux entiers naturels `N` et `n` et renvoyant en sortie une liste contenant les termes u_0, u_1, …, u_n de la suite de Syracuse de paramètre `N`.

(*c*) À l'aide de la fonction `Syracuse`, tracer les représentations graphiques des n premiers termes de la suite de Syracuse de paramètre N :

- pour $N = 3$ et $n = 25$
- pour $N = 15$ et $n = 25$
- pour $N = 9$ et $n = 25$
- pour $N = 7$ et $n = 25$
- pour $N = 27$ et $n = 25$
- pour $N = 27$ et $n = 120$

Quelle conjecture peut-on émettre sur le comportement de long terme d'une suite de Syracuse ?

Durée et altitude maximale

Soit $(u_n)_{n\in\mathbb{N}}$ une suite de Syracuse.

On appelle **durée de vol** la valeur $d = \min\{n \in \mathbb{N}, u_n = 1\}$.

On appelle **altitude maximale de vol** la valeur $A = \max\{u_0, u_1, \ldots, u_d\}$.

(*d*) Que valent d et A pour les suites de Syracuse de paramètres $N = 3$ et $N = 15$?

(*e*) Écrire le code d'une fonction Python, nommée `dA`, prenant en paramètre un entier naturel `N` et affichant en sortie la valeur de la durée de vol et de l'altitude maximale de vol de la suite de Syracuse de paramètre `N`.
Tester cette fonction pour $N = 3$, $N = 15$ et $N = 27$.

(*f*) À l'aide de Python, déterminer la plus grande durée de vol d'une suite de Syracuse de paramètre $N \leq 1\,000$, ainsi que la valeur du paramètre correspondant.
Tracer la représentation graphique des valeurs $(u_n)_{n\in[\![0,d+10]\!]}$ pour le paramètre N obtenu.

(*g*) À l'aide de Python, déterminer la plus grande altitude maximale de vol d'une suite de Syracuse de paramètre $N \leq 1\,000$, ainsi que la valeur du paramètre correspondant.
Tracer la représentation graphique des valeurs $(u_n)_{n\in[\![0,d+10]\!]}$ pour le paramètre N obtenu.

(*h*) À l'aide de Python, déterminer le nombre de suites de Syracuse pour lesquelles $d \leq 10$ et construire une liste contenant les paramètres de toutes les suites de Syracuse vérifiant cette condition.

(*i*) À l'aide de Python, déterminer le nombre de suites de Syracuse pour lesquelles $A \leq 100$ et construire une liste contenant les paramètres de toutes les suites de Syracuse vérifiant cette condition.

Culture générale : La démonstration de la conjecture émise après la question (*c*) est une question posée en 1937 qui demeure, en 2021, une question ouverte en Mathématiques : personne ne sait prouver que pour toute suite de Syracuse, il existe $n \in \mathbb{N}$ tel que $u_n = 1$. Ceci a été vérifié informatiquement pour tout paramètre initial $N \leq 2^{68}$, mais aucune preuve formelle de la conjecture n'est connue à ce jour.