Nom :

Mathématiques approfondies
Cours et exercices
ECG 2

Python - partie I

Thèmes

1. Révisions
2. Exemples en algèbre linéaire
3. Variables aléatoires discrètes
4. Variables aléatoires à densité
5. Fonctions de deux variables

Lycée Saint Louis 2026/2027

Python - semestre I

La théorie, c'est quand on sait tout et que rien ne fonctionne. La pratique, c'est quand tout fonctionne et qu'on ne sait pas pourquoi. Ici, nous avons réuni théorie et pratique : rien ne fonctionne... et on ne sait pas pourquoi !

citation attribuée à ALBERT EINSTEIN

1 Révisions

1.1 Boucles for, boucles while

Premiers exemples

Exercice 1. ✧
Que fait le code suivant ?

Editeur

```
x=float(input("Entrez un réel positif"))
n=0
while n<=x-1 :
    n+=1
print(n)
```

PY1

Exercice 2. ✦ **Le grand théorème de Fermat** # PY2

1. Écrire un programme Python qui calcule le nombre de triplets $(a,b,c) \in \mathbb{N}^3$ solutions de

$$a^2+b^2=c^2 \quad \text{et} \quad 1 \leqslant a \leqslant b \leqslant c \leqslant 50.$$

2. Même question en modifiant l'exposant 2 par $3,4,5\ldots$ Que remarque-t-on ?

Exercice 3. ✦ **Une preuve à l'aide de Python** # PY3

L'objectif est de déterminer tous les entiers naturels a, b, c et n tels que $a!+b!+c!=n!$. On suppose qu'une telle solution existe.

1. Prouver que $n! \leqslant 3(n-1)!$.
2. En déduire que n est inférieur ou égal à 3 et a, b, c inférieurs ou égaux à 2.
3. Écrire un programme qui teste toutes les solutions avec $a,b,c \in [\![0;2]\!]$ et $n \in [\![0;3]\!]$.
4. Conclure sur l'ensemble des solutions.

Exercice 4. ✦✦ **Un peu d'ordre !** # PY4

On s'interdit dans cet exercice d'utiliser les commandes préprogrammées `np.min` *et* `np.max`.

1. Écrire une fonction `maxi` qui prend comme argument deux nombres réels et renvoie le maximum de ces deux nombres.
2. En déduire une fonction `maximum` qui prend en argument une matrice ligne x et renvoie le maximum des éléments de x.
3. En utilisant seulement la fonction `maximum`, comment obtenir un programme qui calcule le minimum ?
4. Donner un programme qui teste si tous les coefficients x ne sont pas identiques et dans ce cas renvoie le deuxième maximum.

Exercice 5. ✦✦ **Énigme** # PY5

In the Battle of Hastings that occurred on October 14, 1066 Harold's forces formed 13 similar squares with exactly same number of soldiers in each square. When Harold himself joined the fray and was added to the number of his soldiers in those thirteen squares a single huge square could be arranged. How many men there must have been in Harold's force ? *Sam Loyd*

La broderie de Bayeux, longue de près de 70 mètres et confectionnée quelques années après la bataille raconte la victoire du futur Guillaume Le Conquérant sur Harold, le dernier roi anglo-saxon.

Cas particuliers des suites récurrentes, des sommes et produits

Exercice 6. ✦ 🎵 **Calcul d'un n-ième terme via Python** # PY6

1. Écrire une fonction qui prend n en argument et qui renvoie les n-ièmes termes des suites u et v définies par les relations de récurrence :
$$\forall n\in\mathbb{N},\quad \begin{cases} u_0 &= 3\\ u_{n+1} &= \sqrt{u_n{}^2+1}\end{cases}\quad\text{et}\quad \begin{cases} v_0 &= 1\\ v_{n+1} &= v_n+2n+1.\end{cases}$$
2. Conjecturer et prouver une formule simple pour u_n et v_n.

Exercice 7. ✦ **Sinus hyperbolique** *D'après EDHEC 2022* # PY7

Écrire une fonction Python qui prend en argument un entier n, un réel x non nul et renvoie la somme partielle d'ordre n de la série $\sum_{k\geqslant 2} 1/\operatorname{sh}(kx)$. Que dire de la convergence ?

La fonction sh *est la fonction sinus hyperbolique définie sur* $\mathbb{R}$ *par* $\operatorname{sh}(x)=(\mathrm{e}^x-\mathrm{e}^{-x})/2$.

Exercice 8. ✦ **Variante de la suite de Syracuse**

Soit $a\in\mathbb{N}$. On définit la suite u par son premier terme $u_0=a$ et # PY8

$$\forall n\in\mathbb{N},\quad u_{n+1}=\begin{cases} u_n/2 & \text{si } u_n \text{ pair}\\ u_n+5 & \text{sinon.}\end{cases}$$

1. Écrire une fonction qui prend en argument $u_0\in\mathbb{N}^*$ et n et affiche les $n+1$ premiers termes de la suite.
2. Tester pour $a\in[\![1;6]\!]$. Que constatez vous ?

Exercice 9. ✦ 🎵 **Conjecture et limite** # PY9

On étudie dans cet exercice la suite définie par : $\forall n\in\mathbb{N},\quad u_n=\dfrac{n!}{\sum_{k=0}^{n} k!}$.

1. Écrire une fonction `facto` qui prend en argument un entier n et renvoie la matrice ligne
$$\begin{bmatrix} 0! & 1! & 2! & 3! & \cdots & (n-1)! & n!\end{bmatrix}.$$
2. En déduire un programme qui prend en argument n et renvoie u_n.
3. Tester et conjecturer la limite de la suite u. Prouver votre conjecture.

Exercice 10. ✦✦ 🎵 La série $\sum 1/n^3$ est convergente. Écrire un programme qui calcule les termes successifs de la suite $(S_n)_{n\in\mathbb{N}^*}$ des sommes partielles de cette série jusqu'à ce que $S_n-S_{n-1}<10^{-10}$ et renvoie le dernier S_n calculé. # PY10

Exercice 11. ✦✦✦ On pose pour tout $n\in\mathbb{N}$ # PY11

$$u_n=\int_0^{\frac{\pi}{4}}\tan(t)^n\,\mathrm{d}t.$$

1. Calculer u_0, u_1 et pour tout $n\in\mathbb{N}$, u_n+u_{n+2}.
2. En déduire un programme qui prend en argument n et renvoie la matrice ligne
$$\begin{bmatrix} u_0 & u_1 & u_2 & \ldots & u_{2n-1} & u_{2n}\end{bmatrix}.$$

Exercice 12. ✦ 🎵 On sait que la suite de terme général $u_n=\sum_{k=1}^{n}\frac{1}{k}$ tend vers $+\infty$. # PY12

Écrire un programme qui prend en argument un réel A et renvoie le plus petit entier n tel que $u_n\geqslant A$.

Exercice 13. ✦ On définit la suite $(u_n)_{n\in\mathbb{N}^*}$ par # PY13

$$u_n=\prod_{k=1}^{n}\left(1-\frac{1}{2k}\right).$$

1. Écrire un programme qui prend en argument n et renvoie u_n.
2. On admet que la suite u converge vers 0. Écrire un programme qui renvoie la plus petite valeur n pour laquelle $u_n \leqslant 10^{-3}$.

Exercice 14. ✦ 🎵 Écrire un programme qui prend en argument une fonction f, un entier naturel non nul n et deux réels a, b (avec $a < b$) et renvoie la somme de Riemann d'ordre n de f. C'est-à-dire : # PY14

$$S_n(f) = \frac{b-a}{n}\sum_{k=0}^{n-1} f\left(a + k\frac{b-a}{n}\right).$$

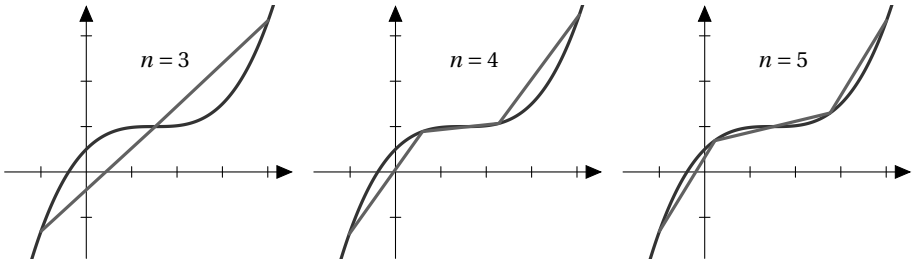
Tester ensuite votre programme en approximant $\pi = \displaystyle\int_0^1 \frac{4\,\mathrm{d}t}{1+t^2}$.

Exercice 15. ✦✦✦ **Approximation de la longueur d'une courbe** # PY78

1. Écrire une fonction qui prend en argument deux points A (x_A, y_A) et B (x_B, y_B) de $\mathbb{R}^2$ et renvoie la distance entre A et B.
   Pour rappel, $AB^2 = (x_B - x_A)^2 + (y_B - y_A)^2$.
2. Soit $f : [a;b] \to \mathbb{R}$.
   Afin d'approximer la longueur de la courbe représentative de f, on découpe l'intervalle $[a;b]$ régulièrement

   $$a = x_0 < x_1 < \cdots < x_{n-1} < x_n = b.$$

   Puis, on somme les distances entre les points $\big(x_i, f(x_i)\big)$ et $\big(x_{i+1}, f(x_{i+1})\big)$ pour tout indice i.
   Écrire un programme qui prend en entrée une fonction f, les réels a et b, un entier n et renvoie une approximation de la longueur de la courbe représentative de f.



Exercice 16. ✦✦ **Calculs des coefficients binomiaux** # PY15

1. *Méthode 1.*
   a) Écrire une fonction `facto` qui prend un entier naturel n en argument et renvoie $n!$.
   b) En utilisant la formule explicite des coefficients binomiaux, en déduire une fonction `binom1` qui prend en argument deux entiers p et n et renvoie $\binom{n}{p}$.
   c) Tester votre programme. Que pouvez-vous conclure sur son efficacité ?
2. *Méthode 2.*
   a) Justifier que pour $p, n \in \mathbb{N}$ avec $p \leqslant n$: $\displaystyle\binom{n}{p} = \prod_{i=1}^{p} \frac{n-p+i}{i}$.
   b) Compléter le programme suivant permettant le calcul de $\binom{n}{p}$.

Editeur

```
if p<n :
    coeff= ...
    for i in range( ... ):
        coeff= ...
    return coeff
else :
    return ...
```

3. *Méthode 3.*
   Compléter et expliquer le programme suivant qui permet de construire le triangle de Pascal.

Editeur

```
def binom3(n):
    pascal = eye(n) # On part de la matrice identité
    for i in range(...):
        pascal[i,0] = 1
        for j in range(...):
            pascal[i,j] = ...
    return pascal
```

Rappels. La commande `eye(n)` *renvoie une matrice avec n lignes et n colonnes avec des zéros partout sauf sur la diagonale qui est remplie de 1. De plus, attention au décalage d'indices,* `pascal[i,j]` *renvoie le coefficient à l'intersection de la $(i+1)$-ème ligne et de la $(j+1)$-ème colonne du tableau.*

Exercice 17. ✦✦ On définit une suite $(u_n)_{n\geqslant 1}$ par $u_1 = 1$, et pour tout $n \geqslant 1$: # PY76

$$u_{n+1} = \frac{1 + {u_1}^2 + {u_2}^2 + \cdot + {u_n}^2}{n}.$$

1. Étudier la monotonie de cette suite et donner sa limite.
2. Construire une fonction Python nommée u prenant en argument n qui renvoie u_n.
3. On considère le programme informatique suivant :

Editeur

```
i=1
p=1
while u(i)>=p:
      i=i+1
      p=p*2
```

Ce programme affiche 7 lorsqu'on l'exécute. Déterminer la nature de la série de terme général $u_n/2^n$.

4. On considère le programme informatique suivant :

Editeur

```
n=1
S=1
a=1
while u(n)!=int(u(n)):
      n=n+1
print(n)
```

On suppose que ce programme puisse «tourner ». Il affiche 43 lors de son exécution. Que peut-on en déduire ?

1.2 La bibliothèque matplotlib.pyplot

Graphe associé à une suite

Exercice 18. ✦ **Représentation des termes d'une suite - L'effet papillon** # PY16

On considère la suite définie par la récurrence : $u_0 \in]0;1[$, $\forall n \in \mathbb{N}$, $u_{n+1} = \lambda u_n(1-u_n)$.

1. *Étude théorique pour $\lambda \in]0;1]$.*
   - **a)** Justifier que pour tout $x \in [0;1]$, $x(1-x) \in [0;1/4]$. En déduire que pour tout $n \in \mathbb{N}$, $u_n \in [0;1]$.
   - **b)** Montrer que la suite u est monotone.
   - **c)** En déduire la convergence. Préciser la limite.
2. *Représentation des termes de la suite.*
   - **a)** Écrire une fonction `suite` qui prend en argument un entier n, u_0 et $\lambda \in \mathbb{R}$ et renvoie la liste des n premiers termes de la suite u définie avec le paramètre λ et la condition initiale u_0.
   - **b)** Afficher les premières valeurs sur un graphique à l'aide de la commande `plt.plot( .. , .. ,'ro')`.
3. *Simulation.*

a) On choisit $\lambda = 2$. Que dire de la convergence de la suite u pour les conditions initiales :

$$u_0 = 1/4, \quad u_0 = 0.1, \quad u_0 = 0.0001\ldots \ ?$$

b) Tester pour $\lambda = 3$ et $u_0 = 0.1$. Que dire si on modifie la condition initiale ?

c) Même question avec $\lambda = 4$ et $u_0 = 0.1$, puis $u_0 = 0.1001$ et $u_0 = 0.100001$. Conclusion ?

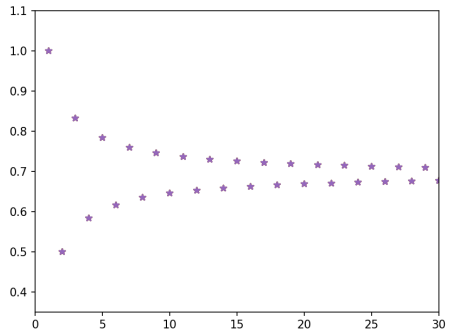
Exercice 19. ✦✦ Suites adjacentes

PY17

Editeur

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

n=30
x=np.arange(1,n+1); y=np.zeros(n)
eps=1
for i in range(n):
    y[i]=eps/(i+1)
    eps=eps*(-1)
z=np.cumsum(y)

plt.axis([0, 30, 0.35, 1.1])
plt.plot(x,z,'*'); plt.show()
```

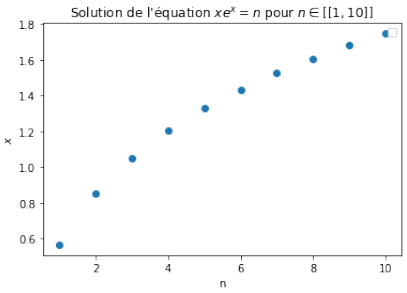
On considère le programme ci-contre et, ci-dessous, le résultat obtenu.



1. Préciser le contenu des variables y et z après l'exécution du programme.
2. Ce graphe suggère que deux suites sont adjacentes. Lesquelles ?
3. Démontrer cette conjecture.

Exercice 20. ✦✦ 🎵 Dichotomie et suite définie de manière implicite

PY18



1. Soit $n \in \mathbb{N}^*$. Justifier qu'il existe une unique solution à l'équation $xe^x = n$ d'inconnue $x \in \mathbb{R}$. Notons u_n cette solution.
2. **a)** Écrire une fonction qui prend en argument n puis renvoie une approximation de u_n à 10^{-3} près.
   b) Modifier la fonction précédente pour retourner une approximation des n premières valeurs de la suite u.
3. Voici le résultat pour les 10 premières valeurs. Conjecturer la monotonie et la limite de la suite u. Prouver votre conjecture.

Exercice 21. ✦ Suites de Michel Mendès-France

PY19

Soient $k \in \mathbb{R}^+$ et $f : \mathbb{N} \to \mathbb{R}$. On définit les suites $(x_n)_n$ et $(y_n)_n$ par les récurrences :

$$x_0 = y_0 = 0 \quad \text{et} \quad \forall\, n \in \mathbb{N}, \quad \begin{cases} x_{n+1} = x_n + k^n \cos\big(2\pi f(n)\big) \\ y_{n+1} = y_n + k^n \sin\big(2\pi f(n)\big). \end{cases}$$

On définit ensuite les points M_n par $\mathrm{M}_n : (x_n, y_n)$.

1. Écrire un programme qui prend en arguments $\mathrm{N} \in \mathbb{N}^*$, k, f et relie par un segment les points M_0, M_1, M_2, …, M_N.
2. *Exemples.* Tester pour :
   a) $k = 1$, $\mathrm{N} = 5000$ et $f(x) = x^{1.18649}$;
   b) $k = 1$, $\mathrm{N} = 1550$ et $f(x) = \sin\left(x^{1/2}\right)$;
   c) $k = 1$, $\mathrm{N} = 1000$ et $f(x) = x^{3/2}$;
   d) $k = 1$, $\mathrm{N} = 6000$ et $f(x) = 0.141593x^2$;
   e) $k = 1$, $\mathrm{N} = 30000$ et $f(x) = 0.0666818x^2$.
3. Comment modifier le programme précédent pour s'arrêter dès qu'on sort du carré $[-100; 100]^2$?

Michel Mendès-France est un mathématicien français (1936-2018), fils de Pierre Mendès-France, homme d'État français, président du Conseil des ministres du 18 juin 1954 au 23 février 1955.

Graphe de fonctions

Exercice 22. ✧ Quelles courbes trace le code ci-contre? # PY20

Editeur
```
x=np.linspace(-1,2,100)
y=np.exp(x)
plt.plot(x,y), plt.plot(x,x), plt.plot(y,x)
plt.show()
```

Exercice 23. ✦✦ **Polynômes de Lagrange** # PY21

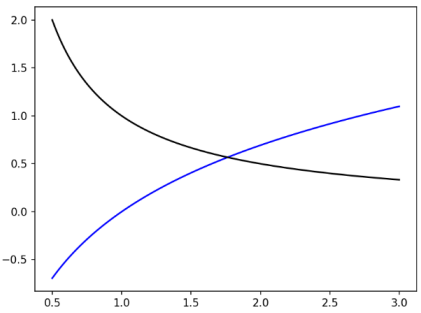
Soient $n \in \mathbb{N}^*$ et $(x_0, x_1, \dots, x_n)$, $n+1$ nombres réels distincts deux à deux. Pour $k \in [\![0, n]\!]$, on définit le polynôme L_k par

$$\forall x \in \mathbb{R}, \qquad L_k(x) = \prod_{\substack{0 \leqslant j \leqslant n \\ j \neq k}} \frac{x - x_j}{x_k - x_j}.$$

1. Écrire une fonction `Lagrange(X,k,x)` qui prend en entrée une matrice ligne `X`$= [x_0, x_1, \dots, x_n]$, un entier $k \in [\![0; n]\!]$ et un réel x, puis renvoie le nombre $L_k(x)$.
2. Tracer sur l'intervalle $[-3;3]$ chacune des courbes des polynômes L_k correspondant à `X`$= [-2, -1, 0, 1, 2]$

Exercice 24. ✦✦ # PY22

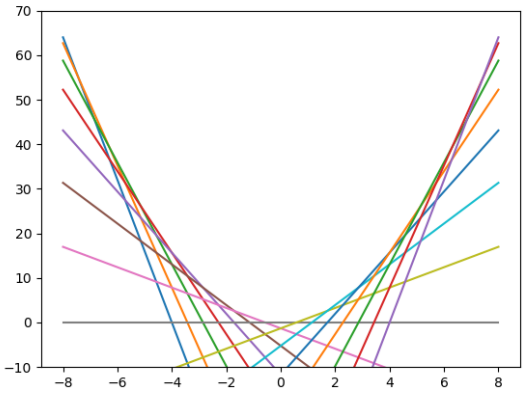
Expliquer l'intérêt du programme suivant.
Test : `>>> mystere(np.log,0.0001)`



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

def mystere(f,e):
# f une fonction, e est un réel "petit"
    n=200
    x=np.linspace(0.5,3,n)
    y=np.zeros([n,1])
    z=np.zeros([n,1])
    for i in range(n):
        y[i]=f(x[i])
        z[i]=(f(x[i]+e)-f(x[i]))/e
    plt.plot(x,y,'b')
    plt.plot(x,z,'k')
    plt.show()
```

Exercice 25. ✦✦ **Enveloppe convexe** # PY23

Compléter le code ci-dessous afin qu'il trace les tangentes à la courbe représentative de $f : x \in \mathbb{R} \mapsto x^2$ aux points d'abscisse -8, -7, …, 0, 1, 2, …, 8.



Editeur
```
n=17 # Nombre de tangentes
a=np.linspace(-8,8,n)

for i in range(n) :
    x= ...
    y= ...

     ...

    plt.ylim(-10,70)
    # Limite l'axe des ordonnées à
        [-10,70]
plt.show() #Affichage
```

Remarque. La commande `plt.clf()` pour « Clear Figure » permet de supprimer un graphe précédemment tracé.

Compléments

Exercice 26. ✦ Soit $n \in \mathbb{N}$, on pose : # PY75

$$u_n = \sin\left(2\pi(2+\sqrt{3})^n\right).$$

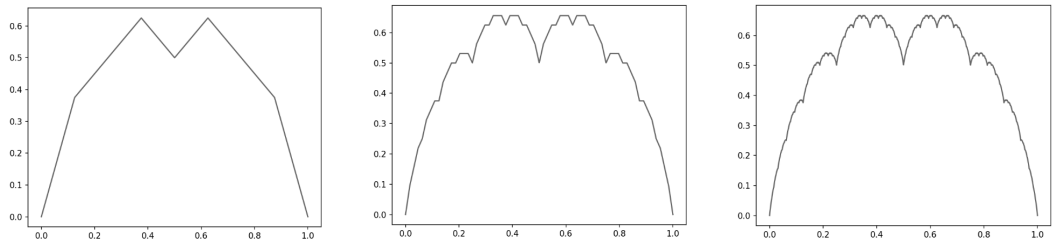
1. **a)** Tracer le graphe de $x \in [0;\pi] \mapsto \sin(x)$ et $x \in [0;\pi] \mapsto x$. Quelle courbe est au-dessus ?
   b) Calculer $(2+\sqrt{3})^n + (2-\sqrt{3})^n$ pour $n \in [\![0;20]\!]$. Que remarque-t-on ?
   c) Écrire un programme qui calcule u_n. Que conjecturer sur la convergence de la suite u ?
2. À l'aide de ces trois informations, prouver que la suite u est bien convergente et préciser la limite.

Exercice 27. ✦✦ ⚙ La courbe Blanc-manger

PY54

Pour tout réel x, on note $d(x)$ la distance de x avec le plus proche entier. On montre que $d(x) = \dfrac{1-\left|1-2x+2\lfloor x\rfloor\right|}{2}$.

1. Tracer la courbe de d sur $[0;1]$.
2. Écrire un script qui prend en argument n et renvoie la courbe représentative de S_n définie par
$$S_n(x) = \sum_{k=0}^{n} \frac{d\left(2^k x\right)}{2^k}.$$
Commenter. *Ci-dessous quelques graphes pour* $n \in \{3,6,10\}$.

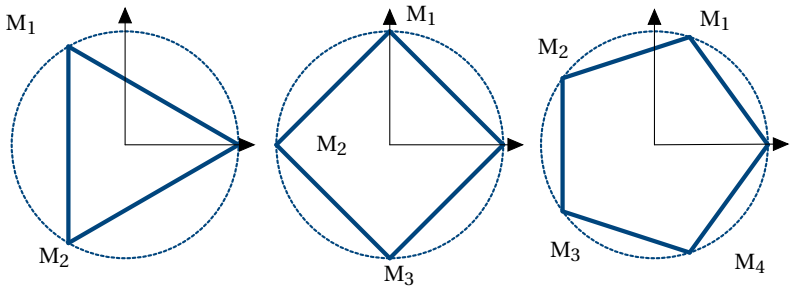


3. Justifier, pour tout $x \in \mathbb{R}$, la convergence de la suite $\left(S_n(x)\right)_{n\in\mathbb{N}}$.
4. Notons $S(x)$ la limite obtenue. Que peut-on conjecturer sur la régularité de la fonction S ?

Exercice 28. ✦✦ ⚙ Les tables de multiplication à l'aide de polygones réguliers convexes

PYS1

Soit $n \in \mathbb{N}\setminus\{0;1\}$. Pour tout $k \in \mathbb{N}$, on considère le point
$$M_k : \left(\cos\left(\frac{2\pi k}{n}\right), \sin\left(\frac{2\pi k}{n}\right)\right).$$
On montre que les points $M_0, M_1, M_2, \ldots, M_{n-1}, M_n = M_0$ constituent les sommets d'un polygone régulier à n côtés inscrit dans le cercle unité. Pour $n=3$, $n=4$, $n=5$, on a un triangle équilatéral, un carré et un pentagone.



1. Écrire un programme qui prend en argument n et renvoie les matrices lignes
$$X = \left[\cos\left(\frac{2\pi}{n}\cdot 0\right), \cos\left(\frac{2\pi}{n}\cdot 1\right), \cos\left(\frac{2\pi}{n}\cdot 2\right) \ldots \cos\left(\frac{2\pi}{n}\cdot n\right)\right] \quad \text{et} \quad Y = \left[\sin\left(\frac{2\pi}{n}\cdot 0\right), \sin\left(\frac{2\pi}{n}\cdot 1\right), \sin\left(\frac{2\pi}{n}\cdot 2\right) \ldots \sin\left(\frac{2\pi}{n}\cdot n\right)\right].$$
2. En déduire un programme qui prend en argument un entier n et trace un polygone régulier à n côtés inscrit dans le cercle unité. Tester le programme en traçant un triangle, un pentagone, un hexagone et finalement un chiliogone.
3. *Visualisation des tables de multiplication*
   a) Donner un programme qui prend en entrée un couple d'entiers (n,a) et qui, sur un même graphique, commence par tracer le polygone régulier à n côtés, puis, à l'intérieur de ce polygone, trace pour tout $k \in [\![0;n]\!]$ le segment entre M_k et M_{ak}. *Indication. Pour tracer un segment entre* $A(x_A,y_A)$ *et* $B(x_B,y_B)$*, il suffit d'écrire*
```
ab=[xA,xB]; ord=[yA,yB]
plt.plot(ab,ord)
```
   b) Observer les figures obtenues pour les couples (n,a) avec $(300,2)$, $(300,3)$, $(300,6)$, $(84,55)$, $(324,28)$, $(405,28)$ et $(993,399)$.
4. *Périmètre*
   a) Écrire une fonction qui prend en argument deux points $A(x_A,y_A)$ et $B(x_B,y_B)$ de $\mathbb{R}^2$ et renvoie la distance entre A et B. *Pour rappel,* $AB^2 = (x_B-x_A)^2 + (y_B-y_A)^2$.

b) Calculer ou donner avec Python une approximation du périmètre du polygone régulier à n côtés inscrit dans le cercle unité. Que dire lorsque $n \to +\infty$?

c) Écrire un programme qui prend en argument n et renvoie un polygone inscrit dans le cercle unité dont les n points sont choisis au hasard.
Indication. On pourra utiliser la commande `np.sort(A)` *qui, à partir de la matrice ligne* A*, renvoie une matrice ligne dont les coefficients sont les coefficients de* A *ordonnés dans l'ordre croissant.*

d) Adapter le programme pour afficher en plus le périmètre du polygone. Vérifier que le périmètre obtenu est systématiquement plus petit que le périmètre du polygone régulier à n côtés.

2 Exemples en algèbre linéaire

2.1 Les bibliothèques

numpy

La bibliothèque `numpy` doit d'abord être importée via `import numpy as np`.
Elle permet d'accéder à la plupart des fonctions et constantes mathématiques comme pour la bibliothèque `math`, on peut par exemple utiliser π avec la commande `np.pi` ou la fonction exponentielle avec `np.exp(.)`.
Elle permet surtout de créer des tableaux de type `ndarray`, parfait pour représenter des matrices. Pour créer une matrice, il suffit de donner ses différentes lignes séparées par des virgules :

Console
```
>>> B = np.array([[1,2,3],[4,5,6],[7,8,9]])
>>> B
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

Pour accéder au coefficient en position (i, j), il suffit de taper `B[i,j]` en prenant garde qu'en Python, les indices commencent à 0.

- **Matrices prédéfinies**

Il existe des matrices usuelles prédéfinies dans la bibliothèque `numpy`.

- `np.ones(n)` crée une matrice ligne de longueur n remplie 1.
- `np.ones((m,n))` crée une matrice de taille `(m,n)` remplie de 1.
- `np.zeros(n)` crée une matrice ligne de longueur n remplie 0.
- `np.zeros((m,n))` crée une matrice de taille `(m,n)` remplie de 0.
- `np.eye(n)` crée la matrice identité de taille `(n,n)`.

- **Opérations sur les matrices**

La plupart des opérations sur les tableaux `numpy` se font terme à terme, par exemple :

Editeur
```
A=np.array([[1,2,3],[4,5,6],[7,8,9]])
print(A==2)
# On teste si chaque coefficient de la
    matrice vaut 2
```

Console
```
[[False  True False]
 [False False False]
 [False False False]]
# Un seul 2, en position (1,2)
```

⚠ **Attention.** La somme matricielle est une opération terme à terme. Pour effectuer la somme entre deux matrices `A`, `B` de même taille, il suffit donc d'écrire `A+B`. Par contre, la commande `A*B` renvoie le produit coefficient par coefficient et non le produit matriciel.

- `np.dot(A,B)` effectue le produit matriciel usuel AB.
- `shape()` renvoie la taille d'un tableau `numpy`.
- `transpose(A)` renvoie la transposée de la matrice A.

Exercice 29. ✧ Comment obtenir avec Python la matrice carrée $\begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}$? # PY24

Exercice 30. # PY25

✧ Prévoir la réponse de la machine.

Console
```
n=10
A=np.arange(1,n+1)
B=np.ones((n,1))
print(np.dot(A,B))
```

numpy.linalg

Pour avoir plus de possibilités, on pourra importer la bibliothèque numpy.linalg avec la commande : `import numpy.linalg as al`

- `al.inv` pour obtenir l'inverse d'une matrice (s'il existe) ;
- `al.matrix_rank` pour le rang ;
- `np.linalg.det` pour le déterminant ;
- `al.matrix_power` pour calculer les puissances ;
- `al.solve` pour résoudre une équation matricielle AX = B d'inconnue X ;
- `al.eig` pour obtenir le spectre de la matrice.

2.2 Les exercices

Exercice 31. ✦ Écrire une fonction Python qui prend en argument n et renvoie les n premières lignes du triangle # PY26

$$\begin{array}{ccccc} 1 & 0 & 0 & 0 & 0 \\ 2 & 3 & 0 & 0 & 0 \\ 4 & 5 & 6 & 0 & 0 \\ 7 & 8 & 9 & 10 & 0 \\ 11 & 12 & 13 & 14 & 15 \\ & & \dots & & \end{array}$$

Exercice 32. ✦ Résoudre, à l'aide de Python, les deux systèmes linéaires suivants. Que constatez-vous ? # PY27

$$\mathscr{S}_1 : \begin{cases} x_1+9x_2+10x_3 &= -50 \\ 9x_1+5x_2+x_3 &= 180 \\ 5x_1+10x_2+9x_3 &= 40 \end{cases} \quad \text{et} \quad \mathscr{S}_2 : \begin{cases} x_1+9x_2+10x_3 &= -50 \\ 9x_1+5x_2+x_3 &= 180 \\ 5x_1+10x_2+9x_3 &= 41. \end{cases}$$

Exercice 33. ✦ On dit qu'une matrice $A = \left(a_{i,j}\right) \in \mathscr{M}_n(\mathbb{R})$ est à diagonale dominante si : # PY31

$$\forall i \in [\![1;n]\!], \qquad |a_{i,i}| \geqslant \sum_{\substack{j=1 \\ j\neq i}}^{n} |a_{i,j}|.$$

Écrire une fonction Python `DiagoD` qui prend en argument une matrice A et retourne « True » si A est à diagonale dominante et « False » sinon.

Exercice 34. ✦✦ Soit $A = \begin{bmatrix} 1 & -6 & 0 \\ 2 & -6 & 2 \\ 2 & -4 & 3 \end{bmatrix}$. En s'aidant des instructions suivantes, calculer les puissances de A. # PY28

Editeur
```
>>> import numpy as np
>>> A = np.array([[1, -6, 0], [2, -6, 2], [2, -4, 3]])
>>> P = np.array([[6, 2, -1], [2, 1, 0], [-1, 0, 1]])
>>> P_inv = np.linalg.inv(P)  # calcule l'inverse de la matrice P
>>> P_inv
array([[ 1., -2.,  1.],
       [-2.,  5., -2.],
       [ 1., -2.,  2.]])
>>> np.dot(P_inv,np.dot(A, P))
array([[-1.00000000e+00,  0.00000000e+00,  0.00000000e+00],
       [ 6.66133815e-16, -2.00000000e+00,  0.00000000e+00],
       [ 4.44089210e-16,  0.00000000e+00,  1.00000000e+00]])
```

Exercice 35. ✦ Écrire un programme qui compte le nombre de matrices non inversibles parmi les matrices de taille $(25,25)$: # PY29

$$\mathrm{M}(a,b)=\begin{bmatrix} a & b & \cdots & \cdots & b \\ b & a & \ddots & \cdots & b \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & a & b \\ b & b & \cdots & b & a \end{bmatrix} \quad \text{avec} \quad -50 \leqslant a,b \leqslant 50.$$

Exercice 36. ✧ Expliquez l'intérêt de la fonction Python suivante. # PY30

Editeur

```
import numpy as np
import numpy.linalg as al

def bidule(A,x) : # A est une matrice carrée et x, un réel
    [n,p]=np.shape(A)
    if n!=p :
        print('Non, la matrice n est pas carrée ...')
    elif np.linalg.matrix_rank(A-x*np.eye(n))<n :
        print('Oui !')
    else :
        print('Non !')
```

Exercice 37. ✦✦ 🎵 **Permutations** # PY73

Soient $n \in \mathbb{N}^*$ et $\mathrm{E}_n = [\![0; n-1]\!]$. Une permutation f de E_n est une application bijective de E_n dans E_n. On peut la coder par une matrice ligne

$$\mathrm{F} = \begin{bmatrix} f(0) & f(1) & \ldots & f(n-1) \end{bmatrix}.$$

1. Écrire une fonction qui prend en argument une matrice ligne et teste si on a bien une permutation.
   On pourra utiliser la commande `np.sort` *qui permet d'ordonner la matrice ligne.*
2. Écrire une fonction Python prend en argument deux matrices lignes F, G associées à deux permutations f, g et renvoie la matrice ligne associée à la composition $f \circ g$.
3. Écrire une fonction Python qui prend une matrice ligne associée à f et renvoie la matrice ligne associée à la réciproque f^{-1}.

3 Variables aléatoires discrètes

3.1 Histogrammes, diagrammes en bâtons

• Un histogramme donne une idée graphique de la répartition des valeurs d'un **échantillon** $\mathrm{E} = \{x_1, \ldots, x_n\}$. Pour cela, on découpe l'ensemble des valeurs possibles en un certain nombre d'intervalles. Notons $p \in \mathbb{N}^*$ le nombre d'intervalles choisis. Ces intervalles doivent être deux à deux disjoints, et leur réunion est égale à (ou contient) l'ensemble des valeurs possibles. Plus précisément, en notant $a_1, a_2, \ldots, a_{p+1} \in \mathbb{R}$, les bornes de tous ces intervalles avec $a_1 < a_2 < \cdots < a_{p+1}$, on a

$$\mathrm{E} \subset \bigcup_{i=1}^{p} [a_i; a_{i+1}[.$$

L'intervalle $[a_i; a_{i+1}[$ est une **classe**. On peut ainsi définir l'**effectif** d'une classe comme le nombre d'éléments de E appartenant à la classe puis sa fréquence f_i comme le rapport de l'effectif sur le nombre total d'éléments de E.
Ensuite, le graphique est obtenu en traçant, pour chaque $i \in [\![1, p]\!]$, le rectangle de base $[a_i, a_{i+1}]$ sur l'axe des abscisses, et

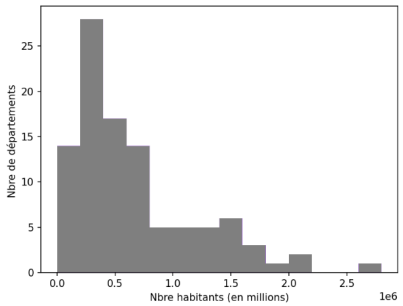
- de hauteur égale à m_i (on parle d'histogramme en effectif) ;
- d'aire égale à f_i (on parle d'histogramme en fréquence). Dans ce cas, la somme des aires des rectangles est égale à 1.

Sous Python, les histogrammes s'obtiennent par la commande `hist` obtenu dans la bibliothèque `matplotlib.pyplot`. Elle prend en argument l'échantillon `E`, puis on indique les bornes de tous les intervalles $a_1 < a_2 < \cdots < a_{p+1}$ ou simplement le nombre p d'intervalles souhaité (Python se chargeant de créer alors p intervalles de même longueur entre la plus petite et la plus grande valeur de l'échantillon).

Exemple. Répartition du nombre d'habitants par département.
Le site de L'Insee (Institut national de la statistique et des études économiques) contient de très nombreuses statistiques en libre accès. On y trouve par exemple la liste du nombre d'habitants par département dont on trace ci-dessous l'histogramme en effectif.

Editeur

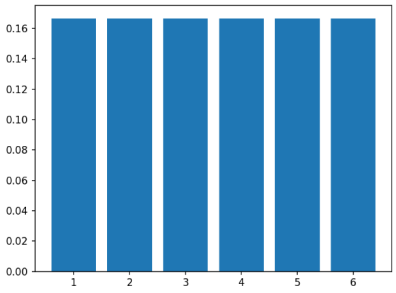
```
import matplotlib.pyplot as plt
# E désigne dans la suite la liste
# du nombre d'habitants par département
# Création d'un tableau avec les intervalles de
    longueurs 200 000 (habitants)
inter =np.linspace(0,2.8*10**6,15)
plt.hist(E, bins=inter)
plt.xlabel('Nbre habitants (en millions)')
plt.ylabel('Nbre de départements ')
plt.show()
```



• On peut aussi utiliser la commande `bar` (toujours dans la bibliothèque `matplotlib.pyplot`) pour tracer de diagramme en bâtons. Voici un exemple qui permet de représenter la loi d'une variable aléatoire uniforme sur $\mathscr{U}([\![1;6]\!])$.

Editeur

```
# La loi de la variable
val=[1,2,3,4,5,6]
loi=[1/6,1/6,1/6,1/6,1/6,1/6]

# Tracé du diagramme en bâtons
          # bar(abscisses,ordonnées)
plt.bar(val,loi)
plt.show()
```



Exercice 38. ✧ Comment modifier le programme précédent afin qu'il affiche la loi d'une variable aléatoire X, somme des valeurs de deux dés équilibrés? # PY32

i	2	3	4	5	6	7	8	9	10	11	12
$\mathbf{P}([X=i])$	1/36	1/18	1/12	1/9	5/36	1/6	5/36	1/9	1/12	1/18	1/36

3.2 Représentations des lois de probabilité

Exercice 39. ✦✦ **Diagramme en bâton d'une loi géométrique** # PY33

1. Soient $p \in]0;1[$ et $X \hookrightarrow \mathscr{G}(p)$.
   Écrire un programme qui prend en argument p et renvoie la plus petite valeur entière n_p telle que $\mathbf{P}\big([X > n_p]\big) \leqslant 1\%$.
2. En déduire un second programme qui prend en argument p et renvoie le diagramme en bâtons sur $[\![1;n_p]\!]$ associé à la loi géométrique $\mathscr{G}(p)$.

Exercice 40. ✦✦✦ **Représentation d'une loi de Poisson** # PY34

1. Écrire un programme qui prend en argument $n \in \mathbb{N}^*$, $\lambda \in \mathbb{R}_*^+$ et renvoie la matrice-ligne
$$\begin{bmatrix} p_0 & p_1 & \dots & p_n \end{bmatrix} \quad \text{où} \quad p_i = \frac{\lambda^i}{i!}\mathrm{e}^{-\lambda}.$$
   On pourra remarquer que $p_{i+1} = \lambda p_i/(i+1)$.
2. Soient $\lambda \in \mathbb{R}_*^+$ et $X \hookrightarrow \mathscr{P}(\lambda)$. Écrire un programme qui prend en argument λ et renvoie la plus petite valeur entière n_λ telle que $\mathbf{P}\big([X \geqslant n_\lambda]\big) \leqslant 1\%$.
3. En déduire un second programme qui prend en argument λ et renvoie le diagramme en bâtons sur $[\![0;n_\lambda]\!]$ associé à la loi de Poisson $\mathscr{P}(\lambda)$.

Exercice 41. ✦✦✦ **Calcul des probabilités d'une loi binomiale** # PY35

Soit $n \in \mathbb{N}$. Soit P un polynôme de degré inférieur ou égal à n défini par
$$P(x) = \sum_{i=0}^{n} a_i x^i = a_0 + a_1 x + \cdots + a_n x^n.$$
En Python, on peut coder le polynôme P à l'aide d'une matrice-ligne formée de ses coefficients : $L = \begin{bmatrix} a_0, & a_1, & \cdots, & a_n \end{bmatrix}$.
Dans ce cas, la commande L[k] renvoie le coefficient de P de degré k.

1. Soient $p, q \in \mathbb{R}$. Donner la relation entre les coefficients du polynôme P et les coefficients du polynôme $Q_{p,q}$ défini par
$$\forall\, x \in \mathbb{R}, \qquad Q_{p,q}(x) = (q + px)P(x).$$

2. Compléter le programme ci-contre qui prend en argument la matrice L des coefficients d'un polynôme P, les réels p et q et renvoie la liste des coefficients de $Q_{p,q}$.

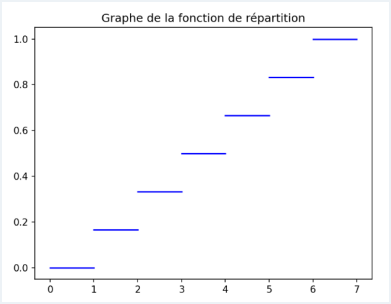
3. À l'aide de la formule du binôme, développer $(q+px)^n$.
En déduire une fonction qui prend en argument un réel $p \in [0;1]$, un entier naturel n et renvoie les probabilités associées à la loi $\mathscr{B}(n;p)$.

Editeur
```
def Bino(liste,p,q) :
    m = len(liste)
    B = ....
    for i in range(1,m) :
        ....
    return B
```

Complément : tracé d'une fonction de répartition, cas discret

Ci-dessous, un code pour tracer la fonction de répartition d'une variable aléatoire discrète finie à partir de sa loi.

Editeur
```
def FctionRepartition(val,loi) :
    plt.clf()
    n=len(val)
    val=[val[0]-1]+val+[val[-1]+1]
    LoiCumulee=[sum(loi[0:i:1]) for i in range(0, n+1)]
    for i in range(n+1) :
        x=[val[i],val[i+1]]
        y=[LoiCumulee[i],LoiCumulee[i]]
        plt.title("Graphe de la fonction de répartition")
        plt.plot(x,y,'b')
    plt.show()

# On teste avec la loi uniforme sur [1,6]
val=[1,2,3,4,5,6];loi=[1/6,1/6,1/6,1/6,1/6,1/6]
FctionRepartition(val,loi)
```



3.3 Simulation des variables aléatoires discrètes

Premiers exemples avec les lois usuelles

Pour simuler des variables aléatoires, on peut utiliser la bibliothèque `random` que l'on importe par :

Editeur
```
import numpy.random as rd
```

- **rd.random()**

La commande `rd.random()` renvoie un réel choisi au hasard dans l'intervalle [0;1[suivant une loi de probabilité uniforme continue sur [0;1[.
En particulier, pour simuler à l'aide de Python une variable aléatoire qui suit une loi de Bernoulli de paramètre p, on peut écrire `rd.random() < p` qui renverra `True` avec une probabilité p. On identifie alors `True` avec 1 et `False` avec 0.

- **rd.randint(debut,fin)**

De la même façon, on peut tirer au hasard (et uniformément) un entier plutôt qu'un réel. Dans ce cas, la commande à utiliser est `rd.randint(début,fin)` qui tire au hasard avec une probabilité uniforme un entier dans l'intervalle `[début,fin[`.
Noter que l'on peut aussi simuler directement une variable aléatoire $X \hookrightarrow \mathscr{U}([\![a;b]\!])$ en tapant :

Console
```
np.floor((b-a+1)*rd.random())+a
```

- **rd.geometric(p)**

La fonction `rd.geometric(p)` permet de simuler une variable aléatoire qui suit une loi géométrique de paramètre p. Pour rappel, si X renvoie le rang du premier succès dans une infinité d'expériences de Bernoulli mutuellement indépendantes, $X \hookrightarrow \mathscr{G}(p)$ où p est la probabilité d'un succès.

- **rd.binomial(n,p)**

De la même façon, on peut simuler une variable aléatoire qui suit une loi binomiale de paramètres n, p avec la commande `rd.binomial(n,p)`. Pour rappel, le nombre de succès obtenus lors d'une répétition de n expériences de Bernoulli mutuellement indépendantes de probabilité de succès p suit une loi binomiale de paramètres n, p.

- **rd.poisson(lambda)**

Enfin, `rd.poisson(lambda)` simule une variable qui suit une loi de Poisson de paramètre `lambda`.

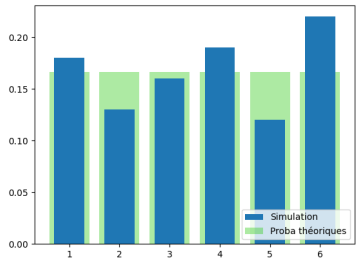
Remarque. Toutes les méthodes précédentes peuvent aussi renvoyer une liste de valeurs tirées selon les différentes lois de probabilité plutôt qu'une seule valeur. Pour faire cela, il suffit de donner comme argument supplémentaire à l'instruction la taille de la liste voulue. Par exemple, `rd.randint(1,101,200)` renvoie un tableau `numpy` contenant 200 entiers pris aléatoirement entre 1 et 100. De plus, la commande `rd.randint(1,101,[200,10])` renvoie une matrice de taille (200,10).

Exemple. Reprenons l'exemple d'un lancer de dé et comparons les probabilités théoriques avec les valeurs d'un échantillon par une simulation informatique. Plus l'échantillon est important, plus l'histogramme a de chance d'être très proche du diagramme en bâtons représentant la loi.

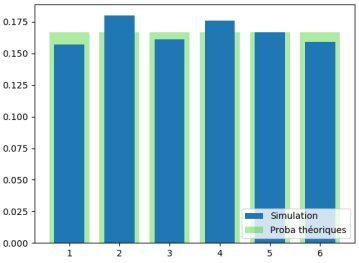
Editeur

```
# La loi de la variable
val=[1,2,3,4,5,6]
loi=[1/6,1/6,1/6,1/6,1/6,1/6]

# Simulation de la variable aléatoire
ech=np.floor(6*np.random.rand(m))+1

# Tracé du diagramme en bâtons
          # bar(abscisses,ordonnées)
plt.bar(val,loi,color=(0.2, 0.8, 0.1, 0.4),label=
    'Proba théoriques')

# Tracé de l'histogramme
          # hist(echantillon, bins=classe)
classe=[0.5,1.5,2.5,3.5,4.5,5.5,6.5]
plt.hist(ech,bins=classe,density='true',rwidth
    =0.6,label='Simulation')

# density='true' pour un histogramme en fréquence

plt.legend(loc = 'lower right')
plt.show()
```

Échantillon de taille $m = 100$:



Échantillon de taille $m = 1000$:



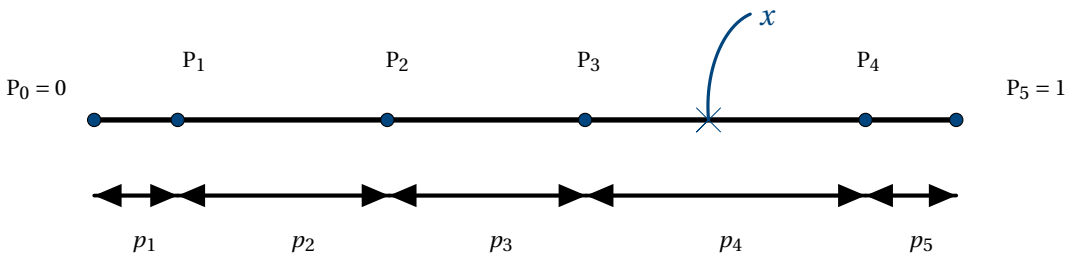
Exercice 42. ✧ Comment modifier le programme précédent pour une variable X donnant la somme de deux dés? # PY36

Simulation pour une loi discrète finie

Soit X une variable aléatoire finie. Pour simplifier le programme, on suppose $X(\Omega) = [\![1;n]\!]$. On note

$$\forall\, i \in [\![1;n]\!], \qquad p_i = \mathbf{P}\big([X = i]\big) \quad \text{et} \quad P_i = \sum_{k=1}^{i} p_k.$$

On souhaite simuler la variable X uniquement avec la commande `rand()`. L'idée est la suivante : on simule la loi de X en tirant un réel x, au hasard dans $[0;1]$ et en renvoyant l'entier k si x appartient à l'intervalle $\big[P_{k-1};P_k\big[$. La probabilité que l'entier k soit choisi est alors $P_k - P_{k-1} = p_k = \mathbf{P}\big([X = k]\big)$ (avec la convention $P_0 = 0$).



Exercice 43. ✦✦✦ 🎵 **Simulation des v.a finies** # PY37

1. Que représentent les réels P_i en termes de probabilités et de X?
2. Écrire un programme qui prend en argument la matrice-ligne $(p_i)_{i\in[\![1;n]\!]}$ correspond à la loi de X et simule X.
3. Tester votre programme sur la loi uniforme $\mathscr{U}([\![1;10]\!])$ et vérifier la cohérence du programme à l'aide d'un histogramme.

≫ Solution p. 44

Exemple de simulation dans le cas infini dénombrable - la loi de Poisson

On peut toutefois étendre la méthode précédente au cas où X prend ses valeurs dans $\mathbb{N}$. Prenons le cas de $X \hookrightarrow \mathscr{P}(\lambda)$. Notons pour $k \in \mathbb{N}$,

$$p_k = \mathbf{P}([X=k]) = e^{-\lambda}\frac{\lambda^k}{k!} \quad \text{et} \quad P_k = \sum_{j=0}^{k} p_j.$$

En particulier, on a $$P_{k+1} = P_k + \frac{\lambda}{k+1}p_k.$$

Exercice 44. ✦✦ **Simulation d'une loi de Poisson** # PY38

1. Adapter la méthode précédente pour simuler $X \hookrightarrow \mathscr{P}(\lambda)$.
2. Comparer l'histogramme obtenu aux valeurs théoriques (diagramme en bâtons construit à l'exercice **??**).

≫ Solution p. 44

3.4 Quelques programmes de référence

Estimation d'une probabilité

Lorsque le calcul d'une probabilité d'un événement A est délicat, on peut se contenter d'une approximation. D'après la loi des grands nombres, il suffit de simuler informatiquement « un grand nombre de fois l'expérience » et de regarder la fréquence empirique du nombre de succès.

$$\mathbf{P}(A) \simeq F_N = \frac{\text{Nombre de succès}}{\text{Nombre d'essais}}.$$

Exemple. Le code permet d'avoir une approximation de la probabilité de faire un double avec deux dés.

Editeur

```
# rd.randint(1,7) renvoie
# un nombre au hasard dans [1,6]

Compteur=0
for i in range(N):
    if rd.randint(1,7)==rd.randint(1,7):
        Compteur+=1
print(Compteur/N)
```

Plusieurs essais à comparer à la valeur théorique 1/6.

Console

```
>>> # script executed
0.161
>>> # script executed
0.1695
>>> # script executed
0.1765
```

Exercice 45. ✧ **Paradoxe du prince de Toscane** # PY39

1. Écrire une fonction Python qui prend en argument $k \in [\![3;18]\!]$ et renvoie une approximation de la probabilité que la somme de trois dés équilibrés donne k.
2. Voici deux tests :

Console

```
compte(9)                compte(10)
>>> 0,1133                >>> 0,1257
```

On constate que la probabilité de faire 10 est légèrement supérieure à celle de 9. Pourtant, comme l'avait remarqué le prince de Toscane au XVII^{e} siècle, il y a autant de façons d'écrire 9 et 10 comme sommes de nombres compris entre 1 et 6 :

$$\begin{aligned} 9 &= 1+2+6 \\ &= 1+3+5 \\ &= 1+4+4 \\ &= 2+2+5 \\ &= 2+3+4 \\ &= 3+3+3 \end{aligned} \quad \text{et} \quad \begin{aligned} 10 &= 1+3+6 \\ &= 1+4+5 \\ &= 2+2+6 \\ &= 2+3+5 \\ &= 2+4+4 \\ &= 3+3+4 \end{aligned}.$$

Pouvez-vous expliquer ce paradoxe ?

Exercice 46. ✦✦ *d'après oraux HEC 2023* # PY40

Soient X, Y deux variables aléatoires indépendantes qui suivent une loi uniforme sur $[\![1;n]\!]$. Écrire un programme Python qui permet de donner une valeur approchée de

$$\mathbf{P}(\mathrm{E}_n) \quad \text{où } \mathrm{E}_n \text{ est l'événement "X/Y est un entier."}$$

Estimation d'une espérance

Soit X, une variable aléatoire réelle admettant une espérance.
En utilisant une nouvelle fois la loi faible des grands nombres, on peut approximer $\mathbf{E}(\mathrm{X})$ en simulant un grand nombre de fois la variable aléatoire X et en effectuant la moyenne arithmétique de l'échantillon obtenu.

$$\mathbf{E}(\mathrm{X}) \simeq \frac{x_1 + \cdots x_\mathrm{N}}{\mathrm{N}} \quad \text{où N représente la taille de l'échantillon } (x_1, \ldots, x_\mathrm{N}).$$

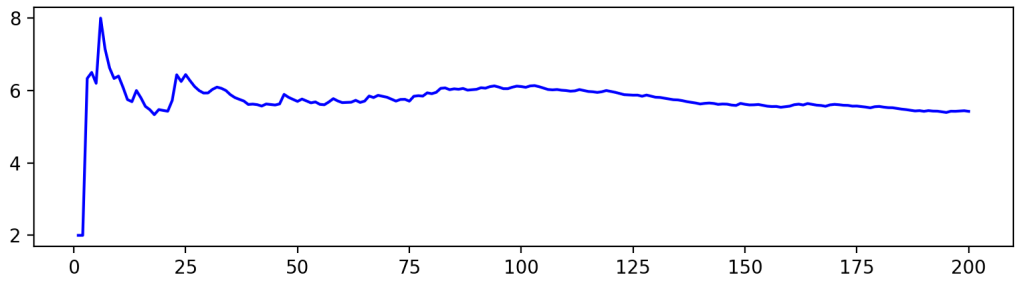
Exercice 47. ✦ 🎵 *d'après Ecricome 2021, voie E* # PY41

On lance indéfiniment une pièce équilibrée. On s'intéresse au rang du lancer auquel on obtient pour la première fois deux «Pile» consécutifs.

1. Recopier et compléter la fonction Python ci-contre afin qu'elle simule les lancers de la pièce jusqu'à l'obtention de deux «Pile» consécutifs, et qu'elle renvoie le nombre de lancers effectués.
2. Écrire une fonction `moyenne` d'argument n qui simule n fois l'expérience ci-dessus et renvoie la moyenne des résultats obtenus.
3. On calcule `moyenne(n)` pour chaque entier n de $[\![1,200]\!]$, et on trace les résultats obtenus dans le graphe suivant. Que pouvez-vous conjecturer sur la variable aléatoire X ?

Editeur

```
def simulX():
    tirs=0
    pile=0
    while pile ... :
        if rd.random()<1/2 :
            pile ...
        else :
            pile ...
        tirs ...
    return ...
```



Exercice 48. ✦✦ **Estimation de l'espérance et de la variance** # PY42

Une urne contient trois boules : une rouge, une bleue et une blanche. On tire avec remise dans l'urne jusqu'à l'apparition de la première boule blanche. Soit X, la variable aléatoire égale au nombre de boules rouges tirées.
Écrire un programme qui permet d'approximer l'espérance et la variance de X.

3.5 Les exercices

Exercice 49. ✦ **Moments** # PY43

Soit X une variable aléatoire sur un univers fini. On rappelle que donner la loi de X signifie donner

$$\mathrm{X}(\Omega) = \{x_1, x_2, \cdots, x_m\} \quad \text{et} \quad \forall i \in [\![1;m]\!], \quad \mathbf{P}\left([\mathrm{X} = x_i]\right).$$

Dans ce cas, on définit les listes val et Loi par :

$$\text{val} = \begin{bmatrix} x_1 & x_2 & \cdots & x_m \end{bmatrix} \quad \text{et} \quad \text{Loi} = [\mathbf{P}([\mathrm{X} = x_1]) \quad \mathbf{P}([\mathrm{X} = x_2]) \quad \cdots \quad \mathbf{P}([\mathrm{X} = x_m])]$$

1. Écrire une fonction Python, nommée `moment`, qui prend en argument les deux listes (val, Loi), un entier s et renvoie le moment d'ordre s : $\mathbf{E}(\mathrm{X}^s)$.
2. En utilisant uniquement la fonction `moment`, écrire une nouvelle fonction qui calcule la variance.

Exercice 50. ✦ 🎵 **Lois usuelles uniquement avec la commande random()** # PY44

Une urne contient 5 boules (1 rouge et 4 bleues). On considère l'expérience suivante :

On tire une boule au hasard et on note la couleur. On replace ensuite la boule dans l'urne.

1. Soit X la variable aléatoire qui renvoie 1 si la boule est rouge et 0 sinon.
   Préciser la loi de X.
   En utilisant uniquement la commande `random`, écrire un programme qui simule la variable X.
2. On répète maintenant n fois l'expérience élémentaire. On suppose les tirages mutuellement indépendants. On note Y le nombre de boules rouges obtenues.
   Quelle est la loi de Y?
   Avec la commande `random`, écrire un programme qui prend en argument n et simule Y.
3. On répète maintenant une infinité de fois l'expérience élémentaire. On suppose toujours les tirages mutuellement indépendants.
   On note Z le numéro du tirage où on a obtenu la première boule rouge.
   Quelle est la loi de Z? Écrire un programme qui simule la variable Z.
4. Modifier le programme précédent pour simuler la variable X_2 qui donne le numéro du tirage où on obtient la seconde boule rouge.
   Soit $r \in \mathbb{N}^*$. Généralisez la question avec X_r, la variable aléatoire qui renvoie le tirage de la r-ième boule rouge.

Exercice 51. ✧ **Le problème du chevalier de Méré** # PY45

Le chevalier de Méré est un noble et écrivain français très amateur de jeu d'argent. Contemporain de Blaise Pascal, il s'opposa à ce dernier sur un problème de jeu de dés :

Sur un lancer de 4 dés, le chevalier gagne si au moins un « 6 » apparaît.

Méré remarque expérimentalement que le jeu lui est favorable et en profite pour s'enrichir.

1. **a)** Écrire un programme qui simule $m = 1000$ parties et compte le nombre de parties gagnantes.
   b) Afficher la fréquence empirique et confirmer l'intuition du chevalier de Méré.
   c) Prouver, par le calcul, que le jeu est effectivement favorable.

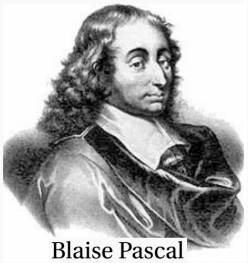
   Malheureusement pour notre chevalier, celui-ci trouva de moins en moins de candidats. Il proposa la variante suivante :

   On lance 24 *fois une paire de dés et le chevalier gagne si un « double* 6 *» apparaît.*

   Basé sur le raisonnement suivant : la probabilité d'obtenir un « double 6 » est de 1/36 soit 6 fois moins de chance que d'obtenir un simple « 6 ». Donc en jouant six fois plus longtemps, c'est-à-dire en lançant $6 \times 4 = 24$ paires de dés, on doit obtenir un jeu tout aussi favorable que le premier. Pascal et Fermat, dans leur correspondance, montrèrent que le raisonnement du Chevalier était faux.
2. **a)** Modifier le programme précédent pour simuler 1000 parties.
   b) Vérifier expérimentalement que le second jeu est défavorable.
   c) Justifier, par le calcul, que le jeu est effectivement défavorable.

Extrait de la lettre du 29 juillet 1654 de Pascal à Fermat

Je n'ai pas eu le temps de vous envoyer la démonstration d'une difficulté qui étonnait fort M. de Méré, car il a très bon esprit, mais il n'est pas géomètre (c'est, comme vous savez, un grand défaut) et même il ne comprend pas qu'une ligne mathématique soit divisible à l'infini et croit fort bien entendre qu'elle est composée de points en nombre fini, et je n'ai jamais pu l'en tirer. Si vous pouviez le faire, on le rendrait parfait. Il me disait donc qu'il avait trouvé fausseté dans les nombres par cette raison :

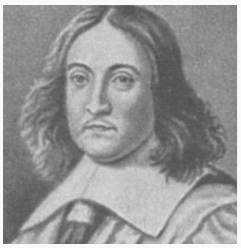


Blaise Pascal

Si on entreprend de faire un six avec un dé, il y a avantage de l'entreprendre en 4, comme de 671 à 625. Si on entreprend de faire Sonnez avec deux dés, il y a désavantage de l'entreprendre en 24. Et néanmoins 24 est à 36 (qui est le nombre des faces de deux dés) comme 4 à 6 (qui est le nombre des faces d'un dé). Voilà quel était son grand scandale qui lui faisait dire hautement que les propositions n'étaient pas constantes et que l'arithmétique se démentait : mais vous en verrez bien aisément la raison par les principes où vous êtes.

Cette correspondance épistolaire marque le début de l'étude des probabilités.

Le saviez-vous



Pierre de Fermat

Exercice 52. ✦✦ **Le problème du collectionneur** # PY46

Afin de relancer ses ventes, une marque met en ligne une nouvelle série de 20 figurines Schtroumpfs dans ses paquets de céréales. Blaise, père attentionné, souhaite obtenir la collection complète pour son enfant tout en maîtrisant son budget petit-déjeuner. Plutôt que de faire les calculs, Blaise simule le problème pour répondre à la question :

Quelle est la probabilité d'obtenir la collection complète avec 50 paquets ?

Pour simuler l'achat de n paquets, Blaise propose le pseudo-code :

- Les figurines sont numérotées de 0 à 19.
On suppose qu'elles ont la même probabilité d'apparition.

- `L` est une matrice ligne avec 20 coefficients. Chaque coefficient `L[k]` peut prendre deux valeurs : 1 si Blaise possède la figurine `k`, 0 sinon.
Au début de la simulation, `L` est donc la matrice nulle $0_{1,20}$.

- Pour i variant de 1 à n, Blaise tire au hasard une figurine. Notons `j` le numéro de la figurine. On affecte 1 au `j`-ème coefficient de `L`.

- Au bout des n tirages, on affiche `L`.
Si `L` ne contient que des 1, Blaise possède la collection complète.

1. a) Écrire une fonction, qui prend comme argument n et renvoie la matrice `L`.
Rappel. La commande `randint(p,q)` *renvoie un entier dans* $[\![p;q-1]\!]$ *choisi au hasard.*

b) Écrire une fonction qui prend en arguments deux entiers n, m et simule m tirages indépendants de n paquets et renvoie la fréquence empirique de l'événement « La collection est complète ».
Indication. Pour vérifier que la collection est complète, on pourra vérifier que la somme des coefficients de `L` *vaut 20.*

2. a) Donner une réponse à la question de Blaise à l'aide de Python.

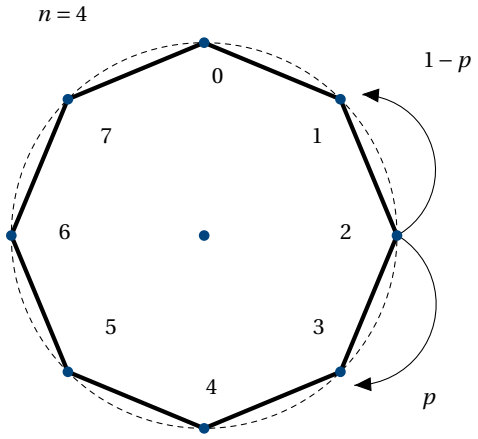
b) Combien de paquets faut-il acheter pour avoir la collection complète avec une probabilité supérieure à 90% ?

Exercice 53. ✦✦ **Mobile sur un polygone** # PY47

On place $2n$ points numérotés de 0 à $2n-1$ sur un cercle. Les points sont répartis uniformément.
Un mobile part initialement de 0 et à chaque étape, il avance d'une unité avec une probabilité p et recule avec probabilité $1-p$.
On note :

→ Pour tout $k \in [\![0;N]\!]$, on note X_k la position du mobile à la k-ième étape.

→ T_p la variable aléatoire égale au temps de retour à l'origine. Si le mobile ne retourne pas à l'origine au bout des N étapes, on pose $T_p(\omega) = 0$.



1. Compléter ce programme qui prend en argument p, simule les 100 premières étapes du mobile et renvoie la position finale du mobile.

Editeur

```
...
def simulation(p,n) :
    X=0
    for i in range(100) :
        ...
    X=X%(2*n)    # renvoie le reste de la division par 2n
    return X
```

2. Modifier le programme pour afficher le temps T_p de premier retour à l'origine 0.

3. En déduire une approximation de l'espérance de T.

4. Comparer $\mathbf{E}(T_p)$ et $\mathbf{E}(T_{1-p})$.

Exercice 54. ✦✦ *vu à l'oral HEC 2024* # PY48

Un pion se trouve à l'origine de l'axe des abscisses. Il se déplace vers la gauche ou vers la droite d'une unité à chaque fois de manière équiprobable.
Écrire une fonction Python qui prend en argument $n \in \mathbb{N}$ et qui simule son déplacement en renvoyant :

- un tableau à une ligne de $n+1$ colonnes dont le coefficient en position k est la position du pion après k déplacements ;

- un tableau à deux lignes et $2n+1$ colonnes où la première ligne contient les entiers de $-n$ à n, la deuxième ligne indique le nombre de fois où le pion est passé par cette case.

Exercice 55. ✦✦ Soient X, Y deux variables aléatoires indépendantes suivant des lois géométriques de même paramètre p. # PY49
Soit $f(p)$, la probabilité que la matrice suivante soit inversible

$$A = \begin{bmatrix} X & Y \\ Y & X \end{bmatrix}.$$

1. Proposer un programme qui prend en argument p et donne une approximation de $f(p)$ (notée dans la suite $\tilde{f}(p)$).
2. Tracer le graphe de $p \mapsto \tilde{f}(p)$ pour $p \in]0;1[$.
3. Faire le calcul exact de $f(p)$ et comparer avec les résultats précédents.

Exercice 56. ✦✦✦ **Deux méthodes pour approximer une somme** # PY50

1. Justifier la convergence de la série $\sum \sqrt{1+e^k}\frac{2^k}{k!}$. Dans la suite, on pose $S = \sum_{k=0}^{+\infty} \sqrt{1+e^k}\frac{2^k}{k!}$.
2. En calculant les sommes partielles, donner une approximation de S.
3. En utilisant une variable X suivant une loi de Poisson de paramètre 2, exprimer S sous forme d'une espérance. En déduire une approximation de S.

Exercice 57. ✦✦✦ **Résolution et simulation du Monty-Hall** # PY51
Dans le film *Las Vegas 21*, Micky Rosa est professeur au M.I.T. Il soumet à ses étudiants, dont Ben, le problème du Monty-Hall.

Micky Rosa : Ben, vous participez à un jeu télévisé et on vous donne la possibilité de choisir entre trois portes différentes, d'accord ? Derrière l'une des trois portes, il y a une voiture neuve, et derrière les deux autres, des chèvres. Quelle porte choisiriez-vous, Ben ?

Ben : Porte $n^\circ 1$?

Micky Rosa : Ben choisit la porte $n^\circ 1$ et là, l'animateur de jeu télé qui, soit dit en passant, sait ce qu'il y a derrière chacune des trois portes, décide d'ouvrir une autre des trois portes ; disons qu'il choisit la $n^\circ 3$, derrière laquelle se trouve une chèvre. Bien. Ben... L'animateur s'approche et dit "Ben, vous voulez garder la porte $n^\circ 1$ ou prendre la porte $n^\circ 2$?". Alors, est-il dans votre intérêt de changer de choix ?

Ben : Oui.

Micky Rosa : Attendez. Souvenez-vous : l'animateur sait où est la voiture. Alors, comment être sûr qu'il ne joue pas avec vous, qu'il n'essaie pas de vous déstabiliser pour vous faire choisir la chèvre ?

Est-ce que Ben a raison de changer de porte?
Pour cela, on simulera un grand nombre de parties de Monty Hall (avec un choix aléatoire de porte) en distinguant les deux cas : on garde la porte, puis on change de porte.

Lien vers l'extrait du film
Game Theory Scene | 21(2008) | Now Playing

Lien vers un documentaire
Voyages au pays des maths | ARTE

Exercice 58. ✦✦ **Loi de Benford** # PY77
La loi de Benford énonce que pour de nombreux types de listes de données statistiques, le 1er chiffre le plus fréquent est 1 pour près du tiers des observations. Puis le chiffre 2 est lui-même plus fréquent que 3 ... alors que la probabilité d'avoir un 9 comme premier chiffre est inférieure à 5 %.
Cette propriété empirique, mise en évidence par Simon Newcomb en 1881, puis à nouveau par Frank Benford 57 ans plus tard, s'étend à de très nombreux types de listes (nombres dans un journal, longueurs des fleuves en mètres, prix dans les supermarchés, etc.). De plus, ils ont précisé que le premier chiffre suivait une certaine loi, dite de Benford. Une variable aléatoire X suit la loi de Benford si :

$$X(\Omega) = [[1;9]] \quad \text{et} \quad \forall k \in [[1;9]], \quad \mathbf{P}([X=k]) = p_k = \frac{\ln(k+1)-\ln(k)}{\ln(10)}.$$

1. En reprenant le code page 13, afficher le diagramme en bâton d'une loi de Benford?
2. Tester la réponse de la machine à cette succession de commandes :

Console

```
num=2026
NUM=str(num)
print(int(NUM[0]))
```

3. En déduire un programme qui prend en argument une matrice ligne **L** de nombres et renvoie l'histogramme de la répartition de la première décimale.
4. Télécharger les deux listes de comptes des usines (celle Twix gauches et celle des Twix droits) sur le site de la classe. Une des deux usines a falsifié ses comptes. Laquelle ?

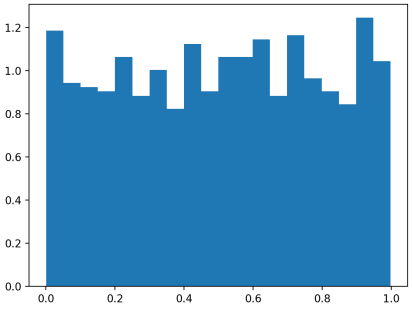
4 Variables aléatoires à densité

4.1 Lien entre histogramme et densité

Commençons par l'exemple d'une simulation d'une loi uniforme.
Le script Python ci-dessous tire m réels uniformément dans $[0;1]$, puis affiche l'histogramme.

Editeur

```
m=1000
L= np.random.rand(m)
classes =np.linspace(0,1,10)

plt.hist(L, bins=20, density=True)
# Création de l'histogramme
plt.show()
```



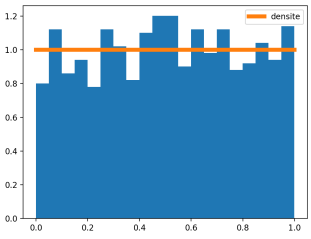
Expliquons les commandes :

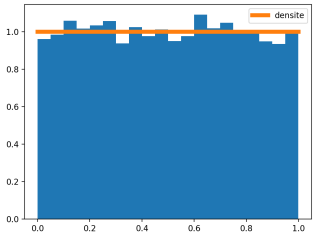
- `density=True` : Trace l'histogramme en fréquence.
- `bins=...` : Si on met une valeur entière en argument, on définit le nombre de classes. Sinon, c'est une liste (ou un tableau) qui définit les différents intervalles des classes.
- `rwidth=...` : Largeur relative de la barre par rapport à l'intervalle (facultatif).

Dans cet exemple, il est utile de rajouter la courbe représentative d'une densité et d'augmenter la taille de l'échantillon. On constate alors que **plus la taille de l'échantillon augmente, plus on a de chance que l'histogramme « se rapproche » de la courbe représentative de la densité.**

Editeur

```
x=[0,1]; y=[1,1]
plt.plot(x,y,linewidth=5,label="
    densite")
plt.legend()
plt.show()
# test avec m=1000, puis m=10000
```





4.2 Simulation de variables aléatoires à densité

Premiers exemples

Comme pour les variables discrètes, la bibliothèque `numpy.random` importée via `rd` permet de simuler les lois à densité usuelles.

- La commande `rd.random()` simule une loi uniforme continue sur $[0;1[$.
- `rd.exponential(a)` renvoie la simulation d'une variable aléatoire de loi exponentielle de paramètre `1/a`.
- `rd.normal(mu,sigma)` renvoie la simulation d'une variable aléatoire de loi normale d'espérance `mu` et d'écart-type `sigma`.

- `rd.gamma(a)` renvoie la simulation d'une variable aléatoire de loi gamma de paramètre `a`.

Exercice 59. ✧ 🎵 En utilisant uniquement la commande `rd.random()`, expliquer comment simuler une variable aléatoire suivant une loi uniforme continue sur $[a,b]$. # PY52

La méthode d'inversion

L'idée repose sur l'énoncé suivant :

Si
- U suit la loi uniforme sur $]0;1[$.
- X est une variable aléatoire à densité dont la fonction de répartition F est une bijection de $]a;b[$ avec $-\infty \leqslant a < b \leqslant +\infty$ sur $]0;1[$.

Alors La variable $Y = F^{-1}(U)$ suit la même loi que X.

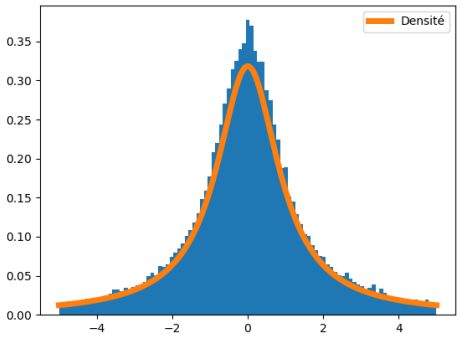
Exercice 60. ✦ 🎵 Prouver l'énoncé précédent. # PY53

On dit qu'une variable aléatoire X à densité suit une loi de Cauchy si une densité est donnée par $f : t \in \mathbb{R} \mapsto 1/\big(\pi(1+t^2)\big)$. On a vu que la fonction de répartition est définie sur $\mathbb{R}$ par $F(x) = \arctan(x)/\pi + 1/2$. La fonction F est strictement croissante, continue de $\mathbb{R}$ dans $]0;1[$. Le théorème de la bijection s'applique

$$\forall\, t \in]0;1[, \qquad F^{-1}(t) = \tan\big(\pi(t-1/2)\big).$$

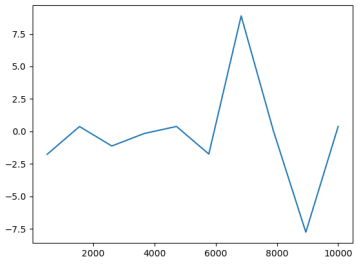
D'après l'exercice précédent, si U suit une loi uniforme continue sur $]0;1[$, alors la variable $F^{-1}(U)$ suit une loi de Cauchy. Ci-dessous, un histogramme d'un échantillon obtenu par simulation par la méthode d'inversion. On vérifie bien que l'histogramme obtenu est très proche d'une densité de la loi de Cauchy.

Editeur
```
U= np.random.rand(50000)
L=np.tan(np.pi*(U-1/2))
inter=np.linspace(-5,5,100)
plt.hist(L, bins=inter, density=True)
# Création de l'histogramme
x=np.linspace(-5,5,200)
y=1/(np.pi*(1+x**2))
plt.plot(x,y,linewidth=5,label="Densité")
plt.legend()
plt.show()
```



Exercice 61. ✦✦ 🎵 Expliquer l'intérêt du code suivant ? Qu'illustre-t-on sur la loi de Cauchy ? # PY55

Editeur
```
Liste=100*np.linspace(5,100,10)
E=np.zeros(10)
for i in range(10) :
    U= np.random.rand(int(Liste[i]))
    L=np.tan(np.pi*(U-1/2))
    e=sum(L)/Liste[i]
    E[i]=e
plt.plot(Liste,E)
plt.show()
```



Exercice 62. ✦ 🎵 Adapter la méthode précédente pour simuler une loi exponentielle de paramètre λ. # PY56

Méthode par rejet

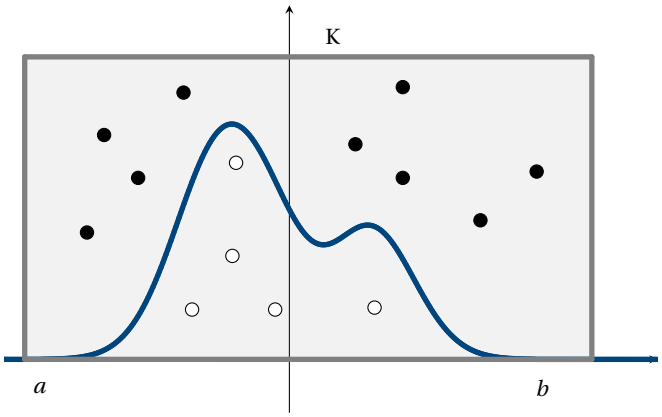
Soit X, une variable à densité et f, une densité. On suppose que f est bornée et à support borné. C'est-à-dire, qu'il existe $K \in \mathbb{R}^+$, $a, b \in \mathbb{R}$ avec $a < b$ tels que

$$\forall\, x \in \mathbb{R}, \qquad 0 \leqslant f(x) \leqslant K \quad \text{et} \quad f(x) = 0 \text{ si } x \notin [a;b].$$

Rappelons que l'aire comprise entre la courbe, l'axe des abscisses et les droites d'équation $x = a$, $x = b$ correspond exactement à la probabilité que la variable aléatoire prenne les valeurs comprises entre a et b.

$$\text{Aire} = \int_a^b f(t)\,\mathrm{d}t = \mathbf{P}\Big(\big[a \leqslant \mathrm{X} \leqslant b\big]\Big).$$

Ainsi, pour simuler la variable X, on peut tirer un point au hasard sous la courbe représentative de f de manière uniforme. On prend alors pour réalisation l'abscisse de ce point.
Précisons que pour tirer un point sous la courbe, on tire un point au hasard dans le rectangle $[a;b] \times [0;\mathrm{K}]$ à l'aide de deux lois uniformes. Si ce dernier est sous la courbe, on le garde. Sinon, on recommence.



Exercice 63. ✦ Quelle est la loi du nombre d'essais avant de tirer un point qui est bien situé sous la courbe de f ? # PY57

Exercice 64. ✦✦ **Exemple de simulation par la méthode de rejet** # PY58
Soit X une variable aléatoire dont une densité f est définie sur $\mathbb{R}$ par

$$f(x) = \begin{cases} 6x(1-x) & \text{si } x \in [0;1] \\ 0 & \text{sinon.} \end{cases}$$

1. Tracer le graphe de f. Quelles valeurs choisir pour a, b et K ?
2. a) Comment tirer un point au hasard dans le rectangle $[a;b] \times [0;\mathrm{K}]$ à l'aide de la commande `rd.random` ?
   b) En déduire un programme Python qui simule la variable X.

4.3 Compléments sur les variables à densité

Exercice 65. ✦✦ **Simulation d'une loi géométrique à partir d'une loi exponentielle** # PY59
Soit $\lambda \in \mathbb{R}_+^*$. Soit X une variable aléatoire sur un espace probabilisé $(\Omega, \mathscr{A}, \mathbf{P})$ de loi $\mathscr{E}(\lambda)$. On pose $\mathrm{Y} = \lfloor \mathrm{X} \rfloor + 1$.

1. Montrer $\mathrm{Y} \hookrightarrow \mathscr{G}(1 - \mathrm{e}^{-\lambda})$.
2. En déduire un programme qui simule une loi géométrique de paramètre p à partir de la loi exponentielle.

Exercice 66. ✧ **Simulation d'une loi de Pareto** # PY60
Soient $\theta \in \mathbb{R}_*^+$ et la fonction f définie sur $\mathbb{R}$ par :

$$f(x) = \begin{cases} \dfrac{1}{\theta x^{1+1/\theta}} & \text{si } x \geqslant 1 \\ 0 & \text{si } x < 1. \end{cases}$$

1. Montrer que f peut être considérée comme une densité de probabilité.
   On considère dans la suite une variable aléatoire X strictement positive de densité f et on note F sa fonction de répartition.
2. On pose $\mathrm{Y} = \ln(\mathrm{X})$ et on admet que Y est une variable aléatoire définie sur le même espace probabilisé que X. On note G sa fonction de répartition.
   Justifier que Y suit une loi exponentielle dont on précisera le paramètre.
3. On rappelle qu'en Python, la commande `rd.exponential(a)` simule une variable aléatoire suivant la loi exponentielle de paramètre `1/a`. Écrire une fonction Python `simuP` prenant en argument θ et permettant de simuler la variable X.
4. Commenter les résultats des lignes suivantes.

Editeur

```
def mystere(theta):
    A=np.zeros(5)
    for p in range(2,7):
        S=0
        for k in range(1,10**p):
            S+=simuP(theta)
        A[p-2]=round(S/10**p,3)
        # round(..,3) arrondit
        # le résultat à 10**(-3)
    return A
```

Console

```
>>> mystere(1/2)
array([2.197, 2.103, 2.011, 1.992, 2.   ])

>>> mystere(1)
array([ 4.892,  8.204, 15.007, 10.577,
    13.309])

>>> mystere(1.2)
array([  7.318,  12.865,  29.35 , 502.559,
    187.655])
```

Exercice 67. ✦ **Simulation d'une loi de Laplace** # PY61
Une variable aléatoire à densité X suit la loi de Laplace si une densité f est définie sur $\mathbb{R}$ par $f(x) = \dfrac{1}{2}\mathrm{e}^{-|x|}$.

1. Soit $U \hookrightarrow \mathscr{U}(]0;1])$. Donner la loi de $X = -\ln(U)$.

2. *Méthode 1.*

a) Justifier que si X et Y sont deux variables aléatoires indépendantes définies sur un même espace probabilisé qui suivent une loi exponentielle de paramètre 1 alors la différence $X - Y$ suit une loi de Laplace.

b) En déduire un programme Python qui simule une loi de Laplace en uniquement la commande `rd.random()`.

c) Tester votre programme en comparant les histogrammes obtenus avec la densité.

3. *Méthode 2.*
On considère deux variables aléatoires X et Y, définies sur un espace probabilisé $(\Omega, \mathscr{A}, \mathbf{P})$, et indépendantes. On suppose que X suit une loi exponentielle de paramètre 1. On suppose par ailleurs que la loi de Y est donnée par

$$\mathbf{P}(Y=1) = \mathbf{P}(Y=-1) = \frac{1}{2}.$$

L'indépendance de X et Y se traduit par les égalités suivantes, valables pour tout réel x :

$$\mathbf{P}([X \leqslant x] \cap [Y=1]) = \mathbf{P}(X \leqslant x)\mathbf{P}(Y=1) \quad \text{et} \quad \mathbf{P}([X \leqslant x] \cap [Y=-1]) = \mathbf{P}(X \leqslant x)\mathbf{P}(Y=-1).$$

On pose $Z = XY$ et on admet que Z est, elle-aussi, une variable aléatoire définie sur $(\Omega, \mathscr{A}, \mathbf{P})$.

a) Vérifier que Z suit une loi de Laplace.

b) Comment simuler la variable Y ?

c) En déduire un nouveau programme qui simule une loi de Laplace.

4. *Compléments sur la loi de Laplace.* *d'après Essec 2017 E*
Soient $\alpha \in \mathbb{R}$ et $\beta \in \mathbb{R}_*^+$. On dit qu'une variable aléatoire réelle à densité suit une loi de Laplace de paramètre (α, β), notée $\mathscr{L}(\alpha, \beta)$, si elle admet comme densité la fonction f donnée par :

$$\forall t \in \mathbb{R}, \quad f(t) = \frac{1}{2\beta} \exp\left(-\frac{|t-\alpha|}{\beta}\right)$$

a) Montrer que si U suit la loi $\mathscr{L}(0;1)$, alors $V = \beta U + \alpha$ suit la loi $\mathscr{L}(\alpha;\beta)$.

b) Conclure en donnant un programme qui simule une variable de loi $\mathscr{L}(\alpha;\beta)$.

Exercice 68. ✦✦✦ **Simulation d'une loi bêta** # PY62

Soient $\alpha, \beta \in \mathbb{R}_*^+$. On pose

$$B(\alpha,\beta) = \int_0^1 t^{\alpha-1}(1-t)^{\beta-1}\,\mathrm{d}t.$$

On considère la fonction $f_{\alpha,\beta}$ définie sur $\mathbb{R}$ par $f_{\alpha,\beta}(x) = \dfrac{1}{B(\alpha,\beta)} x^{\alpha-1}(1-x)^{\beta-1}\,\mathbf{1}_{]0,1[}(x)$.

1. **a)** Vérifier que $f_{\alpha,\beta}$ est une densité de probabilité. On parle alors d'une loi bêta de paramètre (α,β).

b) Écrire un programme qui prend en argument le paramètre (α,β) et renvoie la courbe de $f_{\alpha,\beta}$ sur $[0;1]$.
On pourra utiliser la commande `sp.beta(alpha,beta)` après avoir importé la bibliothèque `scipy.special` *via* `sp`.

2. *Statistiques d'ordre et loi uniforme*
Soient $n \in \mathbb{N}^*$ et $X_1, X_2, \ldots, X_n$, n variables aléatoires indépendantes et de loi uniforme sur $[0;1]$.
On admet l'existence de variables aléatoires à densité $Y_1, Y_2, \ldots, Y_n$ telles que, pour tout ω de Ω, les réels $Y_1(\omega), Y_2(\omega), \ldots, Y_n(\omega)$ constituent un réarrangement par ordre croissant des réels $X_1(\omega), X_2(\omega), \ldots, X_n(\omega)$, de telle sorte que, pour tout ω de Ω :

$$Y_1(\omega) \leqslant Y_2(\omega) \leqslant \cdots \leqslant Y_n(\omega).$$

En particulier, $Y_1 = \min(X_1, X_2, \ldots, X_n)$ et $Y_n = \max(X_1, X_2, \ldots, X_n)$.

a) Écrire un programme qui prend en arguments $n \in \mathbb{N}^*$ et $k \in [\![1;n]\!]$ et renvoie une simulation de Y_k.
Indication. On pourra utiliser la commande `np.sort(A)` *qui prend en argument une matrice ligne et renvoie une matrice mais avec les coefficients de* `A` *ordonnés par ordre croissant.*

b) En déduire un programme qui prend en arguments $n \in \mathbb{N}^*$ et $k \in [\![1;n]\!]$ et affiche l'histogramme associé à 5000 réalisations de la variable Y_k ainsi que la courbe représentative de la fonction $f_{k,n+1-k}$. Tester et commenter.

3. On suppose ici $\alpha, \beta \in]1;+\infty[$ afin que la densité $f_{\alpha,\beta}$ soit bornée. Comment simuler une loi bêta de paramètres $(\alpha;\beta)$ par la méthode de rejet ?
Indication. On pourra s'inspirer de l'exercice 64.

4. **a)** Écrire un programme Python qui prend en argument α, β et renvoie $\overline{x}$ (resp. v), la moyenne arithmétique (resp. la variance empirique) de 5000 réalisations d'une loi bêta de paramètre $(\alpha;\beta)$.

b) *Facultatif.* Donner la valeur exacte de l'espérance et la variance d'une loi bêta de paramètre (α,β).
On pourra admettre que pour tous $a, b \in \mathbb{R}_*^+$

$$B(a,b) = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)} \quad \textit{et} \quad \Gamma(a+1) = a\Gamma(a).$$

5. Modifier le programme précédent en rajoutant le calcul des quantités suivantes. Tester et commenter.

$$\overline{x}\left(\frac{\overline{x}(1-\overline{x})}{v} - 1\right) \quad \text{et} \quad (1-\overline{x})\left(\frac{\overline{x}(1-\overline{x})}{v} - 1\right).$$

Exercice 69. ✦✦ Pour tout entier naturel $n \geqslant 2$, on pose $J_n = \displaystyle\int_0^1 \frac{\ln(t)}{1+t^n}\,dt$. L'objectif est d'obtenir une valeur approchée de l'intégrale J_n à l'aide de Python. # PY63

1. Soit X une variable aléatoire suivant la loi exponentielle de paramètre 1. On pose :
$$Y_n = \frac{-X}{1+e^{-nX}}.$$
Exprimer, à l'aide du changement de variable $u = -\ln(t)$, $\mathbf{E}(Y_n)$ en fonction de J_n.
2. On rappelle que dans la bibliothèque Python `numpy.random` importée par `rd` se trouve l'instruction `rd.exponential(a)` qui renvoie une réalisation d'une variable aléatoire suivant la loi exponentielle de paramètre `1/a`.
Écrire une fonction qui prend en argument n et permet d'approximer J_n.

Exercice 70. ✦ Écrire un programme qui prend en argument une matrice ligne $x \in \mathcal{M}_{1,n}(\mathbb{R})$ et un entier p et renvoie # PY74
$$S_p(x) = \sqrt[p]{\sum_{i=1}^{n} |x_i|^p}.$$
Tirer au hasard une matrice $x \in \mathcal{M}_{1,4}(\mathbb{R})$ et tracer $\left(S_p(x)\right)_{p\in[\![1;20]\!]}$. Que constatez-vous?

5 Fonctions de deux variables

Pour définir et tracer les surfaces représentatives, lignes de niveau, gradient, etc. On commence par importer :

Editeur
```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D  #  pour pouvoir utiliser la fonction Axes3D
ax=Axes3D(plt.figure())
```

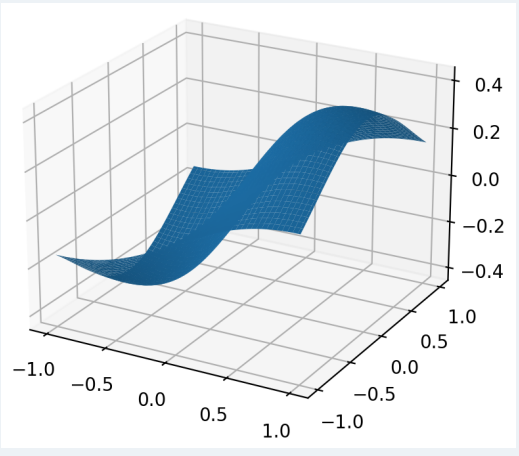
5.1 Graphe d'une fonction de deux variables

Dans un premier temps, on définit la fonction. Par exemple, pour la fonction définie sur $\mathbb{R}^2$ par $f(x,y) = \sin(x)\cos(y)/(1+x^2+y^2)$, on peut écrire :

Editeur
```
def f(x,y):  return np.sin(x)*np.cos(y)/(1+x**2+y**2)
```

Suivant le même principe que pour le tracé de la courbe représentative d'une fonction d'une variable réelle, le code Python suivant permet de tracer la surface représentative d'une fonction de deux variables.

Editeur
```
x = np.linspace(-1, 1, 100)
 # 100 valeurs pour la variable x espacées
 # régulièrement entre -1 et 1
y = np.linspace(-1, 1, 100)
 # De même pour la variable y

X, Y = np.meshgrid(x, y)
 # Tableau contenant les points (xi,yi) où xi
 # et yi sont calculés précédemment
Z = f(X,Y)
 # Calcul des images pour tous les points (xi,yi)

ax = plt.axes(projection='3d')
ax.plot_surface(X, Y, Z)
plt.show()
```

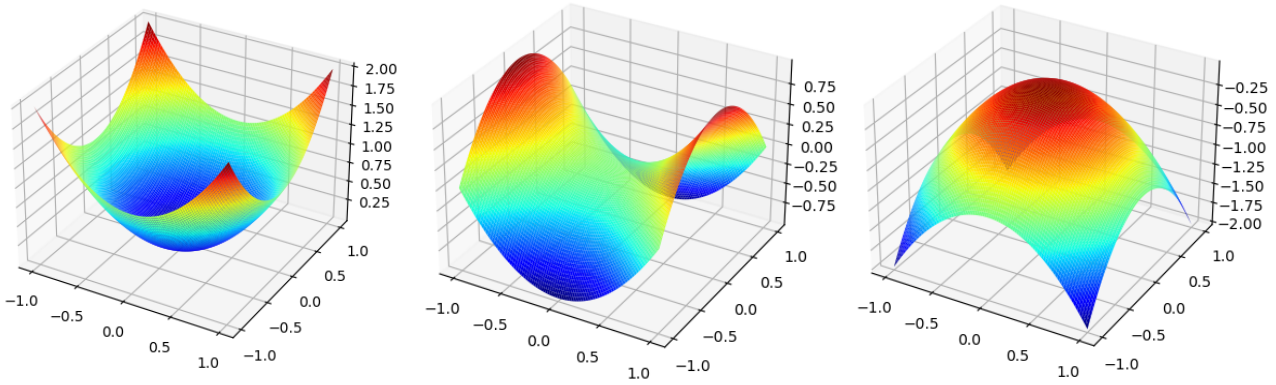


Il existe un grand nombre d'options pour modifier l'affichage. Par exemple, le code suivant permet de faire varier la couleur en fonction de la valeur de la fonction au point considéré.

Editeur

```
ax.plot_surface(X, Y, Z, rstride=1, cstride=1,cmap='jet')

# plusieurs couleurs au choix :
# Remplacer 'jet' par 'cool', 'winter', 'spring', 'summer', 'autumn','hot', 'prism', ...
```

Exemples. Illustrons ce code à l'aide de trois exemples typiques de fonctions polynomiales de degré 2.



Editeur

```
def f1(x, y):
    return x**2 + y**2
```

```
def f2(x, y):
    return x**2 - y**2
```

```
def f3(x, y):
    return -x**2 - y**2
```

Exercice 71. ✧ Tracer la surface représentative de $f : (x, y) \in \mathbb{R}^2 \mapsto \dfrac{\cos\left(x^2+y^2\right)}{\sqrt{1+x^2+y^2}}$ sur $[-\pi;\pi]^2$. # PY64

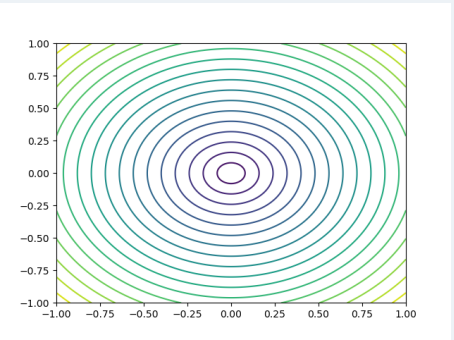
Que peut-on conjecturer sur les extrema locaux? globaux? et le nombre de points où les extrema sont atteints?

5.2 Lignes de niveau

Ci-dessous, un code pour tracer les lignes de niveau d'une fonction de deux variables.

Editeur

```
def f(x,y):
    return np.sqrt(x**2+y**2)

x=np.linspace(-1,1,200); y=np.linspace(-1,1,200)
X , Y = np.meshgrid(x,y)

graphe = plt.contour(X,Y,f(X,Y),20)
 # 20 pour obtenir 20 lignes de niveau
 # ou plt.contour(X,Y,f(X,Y),L)
 # avec L, une liste [.. , ... , ...] avec
 # les valeurs K des lignes de niveaux souhaitées
plt.show()
```

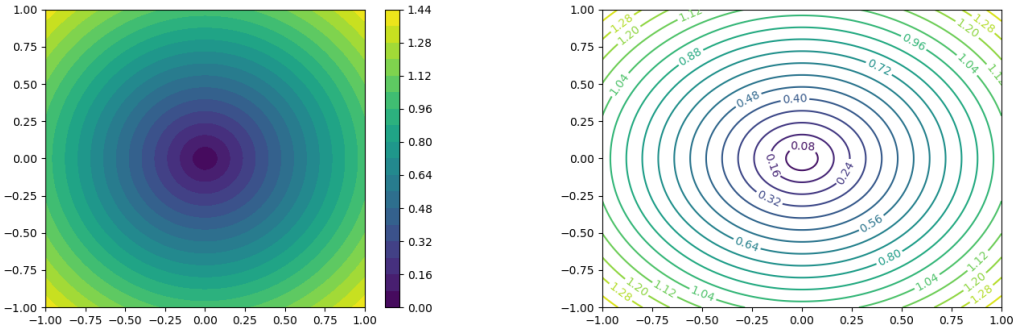


D'autres présentations des lignes de niveaux sont possibles. Par exemple :

Editeur

```
graphe = plt.contourf(x,y,z,20)
plt.colorbar()
plt.show()
```

```
graphe = plt.contour(x,y,z,20)
plt.clabel(graphe,inline=1,fontsize=10)
plt.show()
```



Exercice 72. ✧ Adapter le premier programme pour afficher les lignes de niveau $K \in [\![-4;4]\!]$ de la fonction g définie sur $\mathbb{R}^2$ par $g(x,y) = 4\sin(x) + \cos(3y)$ sur $[-\pi;\pi]^2$. Que dire de la ligne de niveau $K = 5$? # PY65

Exercice 73. ✦ On étudie la fonction $f : (x,y) \in \mathbb{R}^2 \mapsto xy$. # PY66

1. Sans utiliser Python, tracer les lignes de niveau pour $K \in [\![-2;2]\!]$.
2. Vérifier vos calculs en utilisant les codes précédents.

Exercice 74. ✦✦ Pour tous réels x, y tels que $x \neq 0$, on définit le polynôme $P_{x,y}$ sur $\mathbb{R}$ par # PY67

$$P(t) = x^2 \cdot t^2 + |x^2 + y^2 - 1|^{3/2} \cdot t + y^3.$$

À l'aide de Python, représenter dans $\mathbb{R}^2$, l'ensemble des points $(x,y) \in \mathbb{R}^2$ pour lesquels le polynôme $P_{x,y}$ a une unique racine.
Indication. On pourra restreindre l'étude à $[-2;2] \times [-1;2]$.

• **Les symétries**
Dans le cas des fonctions d'une seule variable, nous avons vu que la parité/imparité, la périodicité permettent de simplifier l'étude ou encore de tester la cohérence d'un résultat. Ces idées s'étendent aux fonctions de plusieurs variables.
Voici quelques conditions de symétries.

I	$\forall (x,y) \in \mathbb{R}^2$	$f(x,y) = f(-x,-y)$	II	$\forall (x,y) \in \mathbb{R}^2$	$f(x,y) = -f(-x,-y)$
III	$\forall (x,y) \in \mathbb{R}^2$	$f(x,y) = f(y,x)$	IV	$\forall (x,y) \in \mathbb{R}^2$	$f(x,y) = f(-x,y)$
V	$\forall (x,y) \in \mathbb{R}^2$	$f(x,y) = -f(-x,y)$	VI	$\forall (x,y) \in \mathbb{R}^2$	$f(x,y) = f(x,-y)$
VII	$\forall a \in \mathbb{R}^2$	$f(a) = \varphi(\|a\|)$	avec	$\varphi : \mathbb{R} \to \mathbb{R}$.	

Exercice 75. ✦ Pour chacune des fonctions suivantes, préciser si la fonction vérifie une des symétries parmi I à VII. # PY68

$$f : (x,y) \mapsto \frac{\cos(x^2+y^2)}{4+x^2+y^2}, \qquad g : (x,y) \mapsto 5xy - x^3y^2, \quad h : (x,y) \in \mathbb{R}^2 \mapsto e^{-(x^2+y^2)^2+x+y}$$

$$i(x,y) = -xy\exp(-x^2-y^2), \quad j(x,y) = -3(x+y)/(1+x^2+y^2).$$

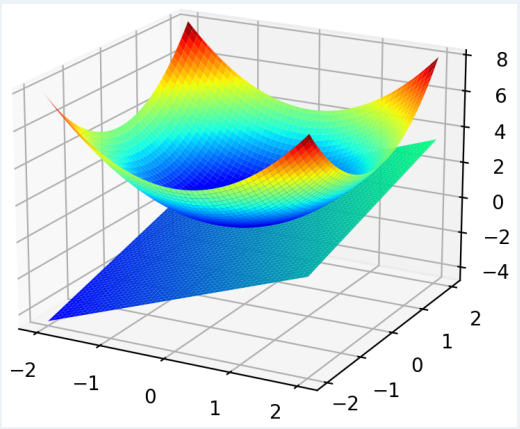
1. Pour chacune des fonctions, tracer quelques lignes de niveau sur $[-2;2] \times [-2;2]$.
2. Comment traduire géométriquement les symétries ?
   On pourra rajouter la commande `plt.axis('equal')` *pour avoir un repère orthonormé.*

5.3 Gradient et plan affine tangent au graphe

Exercice 76. ✧ Tester et expliquer le fonctionnement du code suivant. # PY69

Editeur

```
import numpy.random as rd
import matplotlib.pyplot as plt

def f(x,y): return x**2+y**2

x = np.linspace(-2, 2, 100)
y = np.linspace(-2, 2, 100)
X, Y = np.meshgrid(x, y)

xa=rd.random()
ya=rd.random()
T=2*xa*(X-xa)+2*ya*(Y-ya)+f(xa,ya)

ax = plt.axes(projection='3d')
ax.plot_surface(X, Y, T, cmap='winter')
ax.plot_surface(X, Y, f(X,Y), cmap='jet')
plt.show()
```



Exercice 77. ✦ On définit les fonctions : # PY70

$$f : (x,y) \in \mathbb{R}^2 \mapsto x^2 - 3x + xy + y^2 \quad \text{et} \quad g : (x,y) \mapsto x^2 - x - xy - \frac{y^2}{2} + 2y.$$

1. Calculer le ou les points critiques de f.
2. Tracer la surface représentative de f ainsi que le plan tangent au point critique. Que peut-on en conjecturer sur la nature de ce point critique?
3. Reprendre les questions précédentes avec g.

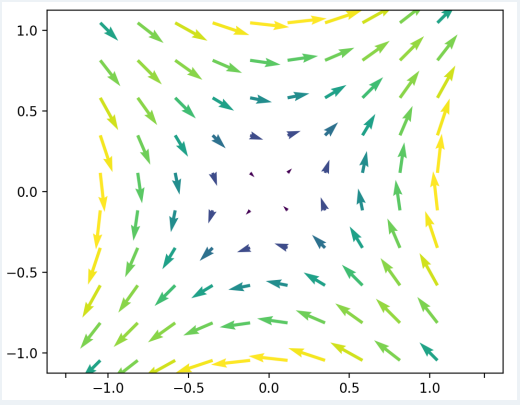
Exercice 78. ✦✦ Soit $f : \mathbb{R}^2 \to \mathbb{R}$ de classe $\mathscr{C}^1$. En s'inspirant de l'exercice 24, page 8, proposer une méthode pour afficher la surface représentative de la dérivée partielle suivant x (resp. suivant y) uniquement en utilisant la fonction f et sans calculer explicitement les dérivées partielles. *On pourra tester le code sur* $[-3;3]^2$ *avec la fonction* # PY71

$$(x,y) \in \mathbb{R}^2 \mapsto \sin(x)\,\mathrm{e}^{-x^2-y^2}.$$

Afin de tracer les vecteurs gradients associés à une fonction de deux variables, on utilise la commande `quiver` dont l'exemple suivant avec la fonction f définie sur $\mathbb{R}^2$ par $f(x,y) = \sin(xy)$ illustre le fonctionnement.

Editeur

```
x = np.linspace(-np.pi/3,np.pi/3,10)
y = np.linspace(-np.pi/3,np.pi/3,10)
X, Y = np.meshgrid(x, y)
Z = np.sin(X*Y)
dx=Y*np.cos(X*Y)
 # expression de la première dérivée partielle
dy=X*np.cos(X*Y)
 # expression de la seconde
color_array = np.sqrt((dx)**2+(dy)**2)
 # Pour que la couleur du vecteur dépende
 # de la norme du gradient
fig, ax = plt.subplots(figsize=(7,7))
ax.quiver(X,Y,dx,dy,color_array)
plt.axis('equal')
# pour avoir un repère orthonormé
plt.show()
```



Exercice 79. ✦ Pour chacune des fonctions suivantes, tracer des lignes de niveau et des gradients. Quelle propriété du gradient illustre-t-on? # PY72

- $f(x,y) = 4\sin(x) + 3\sin(y)$ sur $[-3;3]^2$.
- $g(x,y) = x^2 + y$ sur $[-3;3]^2$.
- $h(x,y) = -xy\mathrm{e}^{-x^2-y^2}$ sur $[0;2]^2$.
- $i(x,y) = -\frac{3\cdot y}{1+x^2+y^2}$ sur $[-5,5] \times [0;10]$.