ALGORITHMIQUE ET PROGRAMMATION
EXERCICES

I. Fonctions

Exercice I.1 Tableau de valeurs

Écrire un programme qui affiche un tableau de valeurs de la fonction f sur $[0;5]$ avec un pas de 0,5, avec $f(x)=\begin{cases}2x+1 \text{ si } x\leqslant -1\\ -3x+7 \text{ si } -1<x<3\\ -x+8 \text{ si } x\geqslant 3\end{cases}$.

Exercice I.2 Signe d'images

Soit f la fonction affine définie par $f(x)=3x-7$.
Écrire un programme qui affiche le signe de l'image d'un nombre *x* entré par l'utilisateur.
Affichage souhaité du type : f(-4) est positif.

Exercice I.3 La rareté fait le prix

Une librairie vend des livres extrêmement rares. Selon l'exemplaire acheté, elle propose des remises à ses clients les plus fidèles. Elle effectue une remise sur le prix hors taxes (HT), noté *ht*, selon la règle suivante :

– si $ht<2\,500$ alors il n'y a pas de remise ;
– si $2\,500\leqslant ht<4\,000$ alors la remise est de 5 %;
– dans les autres cas la remise est de 8 %.

La TVA est de 20 %.

Écrire un algorithme qui permette de donner le prix toutes taxes comprises (TTC) que devra payer un client fidèle, en fonction du prix HT du livre qu'il souhaite acheter.

Exercice I.4 Tracé d'une courbe

1. Rappeler la définition de la courbe représentative d'une fonction *f* définie sur un intervalle I.

2. Écrire un programme qui trace la courbe d'une fonction *f* sur un intervalle $[a;b]$, avec plus ou moins de précision (nombre de points).

Exercice I.5 Résolution approchée de f(x)=0 par dichotomie

On considère la fonction f définie sur $[0;2]$ par $f(x)=x^3+3x-5$.
On admet que f est strictement croissante sur $\mathbb{R}$.

1. a) Calculer $f(0)$ et $f(2)$ et donner leur signe.

b) Que peut-on en déduire sur l'équation $f(x)=0$?

2. a) On propose l'algorithme suivant :
Expliquer son fonctionnement.
Pourquoi la fonction s'appelle dichotomie ?

b) Tester le programme et donner une valeur approchée à 10^{-8} près de la solution sur $[0;2]$ de l'équation $f(x)=0$.

```
def dichotomie(f,a,b,precision):
    while b-a>precision:
        c = (a+b)/2
        if f(a)*f(c)<=0:
            a,b = a,c
        else:
            a,b = c,b
    return (a+b)/2
```

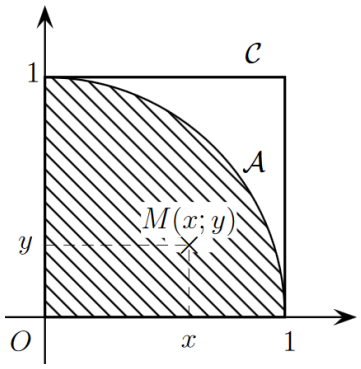
APPROFONDISSEMENTS

3. Modifier l'algorithme ci-dessus pour qu'il détermine une valeur approchée de $f(x)=k$ avec k réel et $f(x)=x^3+3x-5$.

4. Modifier l'algorithme afin qu'il détermine une valeur approchée de toutes les solutions de $f(x)=k$ avec k réel.

Exercice I.6 Monte-Carlo

On considère le carré $\mathcal{C}$ de côté 1 ci-contre, et le quart de disque $\mathcal{A}$ de centre l'origine O du repère.



1. On prend au hasard un point M à l'intérieur du carré $\mathcal{C}$.
Quelle est la probabilité que M soit dans le quart de disque $\mathcal{A}$?

2. On note $M(x;y)$ les coordonnées de M.
a) Quelle(s) condition(s) doivent vérifier les coordonnées pour que M soit dans le carré $\mathcal{C}$?

b) Quelle(s) condition(s) doivent vérifier les coordonnées pour que M soit dans le quart de disque $\mathcal{A}$?

c) On propose l'algorithme suivant, en langage naturel :

```
N ← 1000
C ← 0
Pour i allant de 1 à N
        Affecter à x une valeur aléatoire de [0;1]
        Affecter à y une valeur aléatoire de [0;1]
        Si x²+y² < 1 :
                C ← C+1
        Fin Si
Fin Pour
Afficher C/N
```

Expliquer cet algorithme (à quoi correspond la valeur de la variable C ? A quelle valeur peut-on s'attendre en sortie ? Etc.).

d) Programmer cet algorithme en Python et le tester.
En déduire une très bonne valeur approchée de π (vérifier sur internet).

Exercice I.7 Volume maximal

On considère une pyramide à base carrée de côté 10 cm et de hauteur 12 cm.
On y inscrit un pavé droit à base carrée.
On souhaite déterminer la hauteur du pavé qui aura un volume maximal.

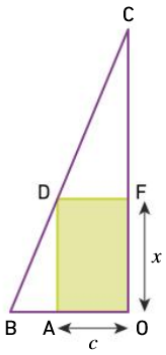
On note x la hauteur du pavé et c la longueur du côté du carré de sa base.

1. x prend ses valeurs dans quel intervalle ?

2. On considère la fonction suivante.

```
def volume(x) :
        return(x*(5/6*(12-x))**2)
```

Expliquer cette fonction à l'aide du triangle ci-contre.



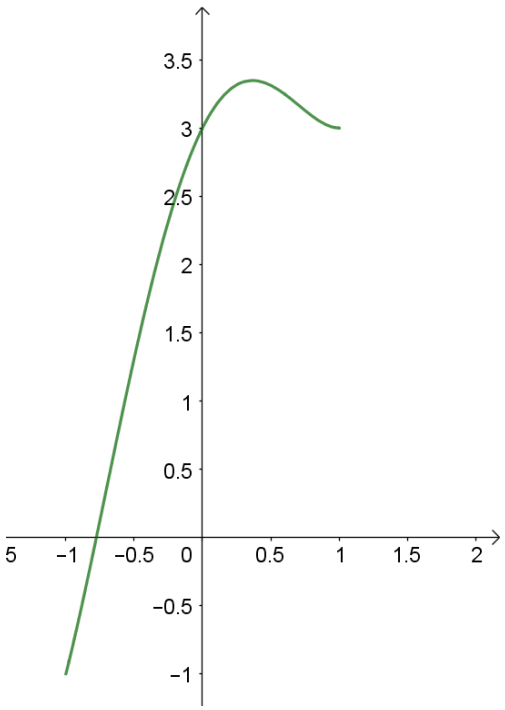
3. Compléter le programme Python suivant qui permet de calculer une valeur approchée du volume maximal du pavé :

```
max=volume(0)
while x<12 :
        if volume(x) > ……
                max = ……
                x_max = x
        x = x + 0.1
```

4. En déduire alors une valeur approchée du volume maximal et de la valeur de x correspondante.

Exercice I.8 Longueur d'un arc de courbe

Écrire un algorithme qui permette de calculer la longueur de la courbe représentative de la fonction f définie sur $[-1;1]$ par $f(x)=x^4-3x^2+2x+3$.



On pourra utiliser les deux fonctions suivantes :

```
def f(x):
        return(x**4-3*(x**2)+2*x+3)
```

```
from math import sqrt
def distance(x1,x2,x3,x4):
        return( sqrt( (x1-x2)**2+(y1-y2)**2 ) )
```

II. Solution(s) entière(s) d'équation(s)

Exercice II.1 Solutions entières d'un système

On considère le système d'équations suivant : $\begin{cases} 2x+y+z=18 \\ x-2y-z=-7 \\ -4x+5y+3z=16 \end{cases}$.

Écrire un programme qui détermine les solutions entières comprises entre -100 et 100.

Exercice II.2 Équation diophantienne

Cédric Grolet est né le 28 août 1985 à Firminy, à côté de Saint-Étienne.
*En 2011, il intègre Le Meurice1 à Paris (hôtel ***** et restaurant ***), en tant que sous-chef et devient rapidement Chef Pâtissier. Élu Meilleur chef pâtissier 2016, il est sacré meilleur chef pâtissier de restaurant du monde le mardi 17 octobre 2017.*



Les organisateurs d'un mariage ont souhaité un buffet de desserts. Mais pas n'importe lesquels…
Des desserts de Cédric Grolet bien sûr !
Les mariés ont chacun choisi leur préféré, et il sera donc demandé à C. Grolet de préparer des « Rubik's Cake » (40 €) et des « 100 % vanille » individuels (35 €).

Le budget est de 2600 € maximum pour 67 invités.
A l'aide d'un algorithme, déterminer combien de desserts de chaque le génial C. Grolet pourra faire.

Exercice II.3 Rédaction (système d'équations)

On souhaite créer un programme qui rédige entièrement la résolution (par combinaisons linéaires) d'un système d'équations du type $\begin{cases} ax+by=c \\ dx+ey=f \end{cases}$.

III. Droites

Exercice III.1 Si proches et pourtant…

On considère la fonction affine *f* définie par $f(x)=2{,}3x-4$.
On souhaite écrire une fonction qui détermine si un point $A(x_A;y_A)$ appartient ou non à la représentation graphique de *f*.

1. a) On considère les points $A(1;-1{,}7)$ et $B(1{,}739130435;0)$.
Montrer que A appartient à la courbe mais que B non.

b) On propose le programme suivant. Le tester avec les points $A(1;-1{,}7)$ et $B(1{,}739130435;0)$.

```
def appartient(xA,yA) :
    if yA == 2.3*xA-4 :
        return("A appartient à la courbe")
    else :
        return("A n'appartient pas à la courbe")
```

Qu'observe-t-on ? Pourquoi ?

2. On propose alors d'utiliser la fonction isclose (bibliothèque numpy) qui permet de tester si deux valeurs sont « très proches ». Modifier le programme ci-dessus, et le tester avec A et B.
Qu'observe-t-on ?

Conclusions sur isclose : ……

……

……

……

……

……

Exercice III.2 Équation réduite

Écrire un algorithme qui affiche l'équation réduite d'une droite (AB), où A et B sont deux points dont on connaît les coordonnées.

IV. Probabilités & statistiques

Exercice IV.1 Des indics discrets

1. Écrire un programme qui calcule la moyenne d'une série statistique discrète rentrée dans une liste.

2. Expliquer le programme suivant. Quel calcule-t-il ?

```
def mediane(serie):
    n = len(serie)
    serie = sorted(serie)
    if n%2 == 0:
        return (serie[n//2]+serie[n//2 - 1])/2
    else:
        return serie[n//2]
```

Exercice IV.2 Rien ne sert de courir...

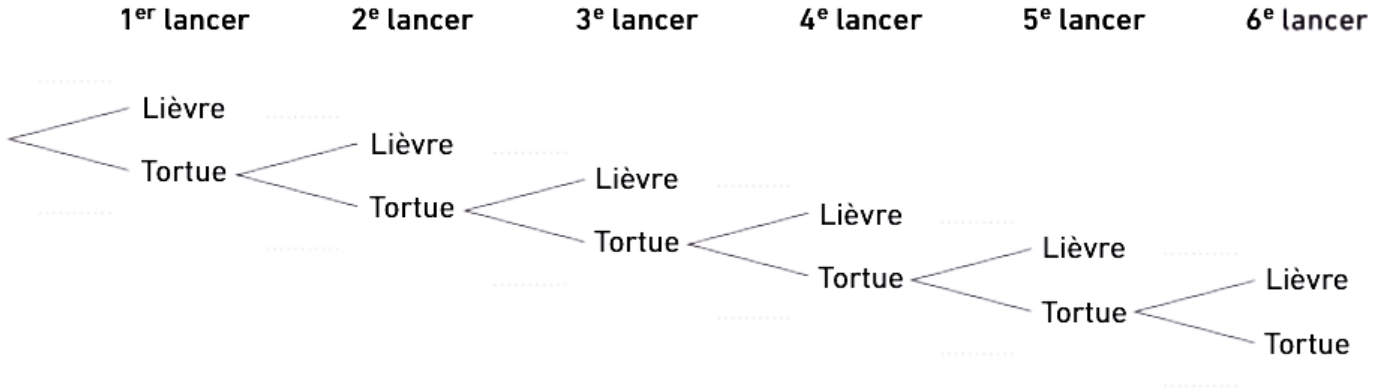
1. Méthode « bourrin »

Le lièvre est plus rapide que la tortue.
Pour donner plus de chance à la tortue de gagner une course de 5 km, on adopte la règle de jeu suivante :
On lance un dé.
Si le 6 sort, le lièvre est autorisé à démarrer et gagne la course en quelques secondes; sinon on laisse la tortue avancer d'un kilomètre.
Recommencer le procédé jusqu'à la victoire du lièvre ou de la tortue.

Le jeu est-il à l'avantage du lièvre ou de la tortue ?

2. Méthode élégante

Utilisons nos connaissances en probabilités pour résoudre ce problème.
Compléter l'arbre ci-dessous (« lièvre » signifie que 6 est sorti).



En déduire la probabilité que la tortue gagne.

Exercice IV.3 Joyeux anniversaire

Dans cet exercice, on se propose de répondre à la question suivante :

Quelle est la probabilité que, dans une classe de N élèves,
il y ait au moins deux élèves qui partagent la même date d'anniversaire ?

1. Méthode algorithmique

On se propose d'effectuer un grand nombre de simulations.
Il s'agit dans un premier temps de tirer N dates d'anniversaires au sort (parmi 365 jours, en supposant les dates d'anniversaire uniformément réparties sur l'année civile).
Il faudra ensuite chercher si deux dates coïncident (on pourra utiliser les listes - voir MEMENTO PYTHON).
Écrire un algorithme qui permette de répondre à la question posée.

2. Méthode mathématique

Sauriez-vous retrouver le résultat ci-dessus à l'aide d'un raisonnement probabiliste ?
Aide : un arbre de probabilités peut être utile.

APPROFONDISSEMENT

3. L'attaque des anniversaires

Jusqu'en 2004, les échanges Wifi étaient sécurisés par le protocole WEP.
Ce protocole fonctionnait à partir d'une clé aléatoire de 24 bits (1 bit peut prendre deux valeurs : 0 ou 1).
Il y avait donc 2^{24} clés aléatoires possibles, soit plus de 16 millions : $16\,777\,216$.
Mais ce protocole avait des failles : il ne faut jamais utiliser deux fois la même clé aléatoire, sinon il est possible d'en déduire la clé utilisée…
Imaginons qu'un hacker souhaite « pirater » la connexion, c'est-à-dire décrypter les informations qui transitent dans le réseau. Il lui faudra pour cela analyser des paquets de données pour obtenir deux informations cryptées avec une même clé aléatoire.
Combien de paquets devra-t-il analyser pour avoir une probabilité de 95 % de pirater la connexion ?

Exercice IV.4 Monty-Hall

Le paradoxe de Monty Hall trouve son origine dans le jeu télévisé *Let's Make a Deal*, diffusé aux États-Unis à partir de 1963. L'animateur y proposait le choix suivant.

> Un candidat est présenté face à 3 portes : derrière une seule de ces portes se trouve un cadeau, alors que derrière chacune des deux autres portes se trouve un objet sans intérêt (typiquement : une chèvre).
> Le candidat choisit une de ces 3 portes, mais sans l'ouvrir.
> L'animateur (qui sait où se trouve le cadeau) ouvre une des 2 portes restantes, en prenant soin (si besoin) d'éviter la porte qui contient le cadeau (la porte ouverte par l'animateur révèle donc toujours une chèvre).
> Le candidat a alors le choix entre conserver sa porte initiale, ou changer pour pour prendre l'autre porte restante.

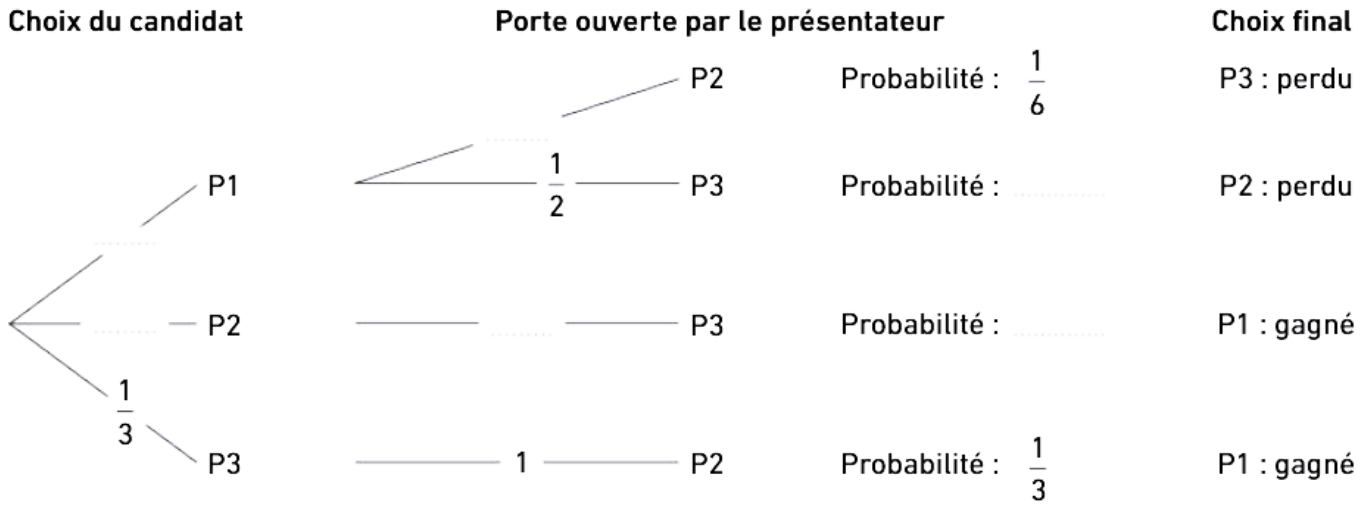
A votre intuition : que doit faire le candidat ? Conserver ou changer ?

1. Méthode algorithmique

Écrire un programme qui permette de simuler N parties aléatoires et donc d'en déduire quel choix semble le plus profitable.

2. Méthode mathématique

On appelle P1, P2 et P3 les trois portes. On suppose que la voiture est derrière la porte P1 et que le candidat change de porte.
Compléter l'arbre suivant et en déduire la probabilité de gagner en changeant de porte.



Exercice IV.5 Calvitie incluse ?

Un être humain possède au maximum 150 000 cheveux.
Écrire un algorithme qui détermine combien de personnes il faut dans un groupe pour que la probabilité qu'il y en ait deux possédant le même nombre de cheveux dépasse 99 %.

Exercice IV.6 Aléatoire = répétition ? O_O

Mon lecteur MP3 contient 7 00 titres. Je les écoute en "mode aléatoire".
Après combien de titres vais-je entendre un morceau déjà passé ?

Exercice IV.7 Jeux de dés, jeux de Méré

Antoine Gombaud, chevalier de Méré, est un écrivain français né dans le Poitou en 1607, mort le 29 décembre 1684. Cet homme d'esprit réputé ne voulait absolument pas reconnaître l'autorité des mathématiciens dans un problème concernant le jeu de dés : il s'était mis dans la tête une autre solution et, étant persuadé de sa justesse, il accusa ouvertement les mathématiques de se contredire.
Voilà ce dont il s'agit.

Pari 1
Si l'on jette 4 fois un dé à six faces, il y a plus de chances qu'on obtienne un 6 plutôt qu'on n'en obtienne pas.
Pari 2
Si l'on jette 24 fois deux dés à six faces, il y a aussi plus de chances qu'on obtienne un double six plutôt qu'on n'en obtienne pas.

Le chevalier de Méré, qui était un grand joueur, avait remarqué que le premier jeu était avantageux.
Il considérait que le deuxième pari était aussi avantageux : il pensait que le rapport 4/6 (4 lancers, 6 possibilités) du pari 1, supérieur à ½, déterminait une probabilité supérieure à ½, et donc la probabilité plus forte d'obtenir un 6 (ou n'importe quel autre nombre) que ne pas en obtenir ; il en déduisait, dans le pari 2, en faisant intervenir le même rapport 24/36 (24 lancers, 36 possibilités) égal à 4/6, que la probabilité était plus forte d'obtenir un double six que ne pas en obtenir.

1. a) Ecrire un algorithme qui simule 20 000 parties liées à chaque pari.

b) Conclure : Méré avait-il raison ?

2. Sauriez-vous retrouver les résultats de la question 1.b) à l'aide d'outils mathématiques.
Aide : un arbre de probabilités peut être utile.

V. Divers

Exercice V.1 Suite de Syracuse/Collatz

La suite de Syracuse d'un entier N non nul est définie ainsi : on choisit un nombre entier N non nul.
Si N est pair, on divise N par 2. Si N est impair, on effectue $3N+1$.
On recommence l'opération avec le nouveau nombre entier obtenu. Et ainsi de suite…
Il existe une conjecture célèbre sur ces nombres :

Quel que soit l'entier N, la suite de Syracuse d'un entier N finira par atteindre 1
puis alternera indéfiniment entre 4, 2 et 1.

Écrire un programme qui écrit tous les termes d'une suite de Syracuse d'un entier N rentré par l'utilisateur, jusqu'à apparition du premier 1.

> Cette conjecture porte le nom de *conjecture de Collatz*, du nom de son découvreur dans les années 1950.
> A ce jour, elle n'est toujours pas démontrée !

Exercice V.2 Triplets pythagoriciens

Un triplet pythagoricien est un triplet d'entiers non nuls (a, b, c) tels que $a^2+b^2=c^2$.
Écrire un programme qui affiche tous les triplets pythagoriciens avec $c \leqslant 50$.

Exercice V.3 Jeu du nombre à deviner

Cet exercice propose la programmation d'un petit jeu.
L'ordinateur choisit un nombre pseudo-aléatoire compris entre 10 et 100 inclus.
On demande à l'utilisateur de deviner ce nombre en moins de six essais.
On lui indique à chaque fois si le nombre proposé est supérieur ou inférieur au nombre cherché.

Programmez votre algorithme dans un langage de programmation, et jouez...
Puis essayez de trouver une stratégie qui vous permette de gagner à tous les coups !

Exercice V.4 Diviser pour mieux régner

On souhaite créer une fonction `diviseur` qui compte le nombre de diviseurs d'un nombre entier.
Pour cela, on a écrit un algorithme en langage naturel :

```
Fonction diviseur(n)
compteur ← 0
Pour div allant de 1 à n
        Si n est divisible par div alors
                compteur ← compteur + 1
        Fin Si
Fin Pour
Retourner compteur
```

1. Écrire le script correspondant en langage Python.

2. Avec le programme, compléter les phrases suivantes :

12 admet …… diviseurs	36 admet …… diviseurs
13 admet …… diviseurs	60 admet …… diviseurs
15 admet …… diviseurs	90 admet …… diviseurs

3. a) Modifier le programme pour qu'il retourne l'entier inférieur à 1 000 qui a le plus de diviseurs.

b) Modifier alors le programme pour qu'il affiche ces diviseurs.

4. Modifier le programme afin de pouvoir compléter le tableau suivant, qui donne le nombre d'entiers inférieurs à 1 000 qui ont *x* diviseurs.

Nombre de diviseurs	1	2	3	4	5	6	7	8	9	10
Effectif										

> Si ce sujet vous intéresse, vous pouvez lire un de mes articles sur mon site :
> www.mathemathieu.fr/art/articles-maths/33-theorie-probabiliste-nombres-thm-fondateurs