PSI 1&2 *125

TD 1 Révisions

2018-2019

On dispose des modules numpy, matplotlib et random.

Exercice 1 Révisions

Les primitives suivantes permettent de modifier le type d'un objet :
`dict, bin, int, str, bool, ord, float, complex, set, tuple, chr, list.`

1. Que fait la primitive `isinstance` ?
2. On désigne par c_n le nombre de chiffres de l'entier n dans son écriture décimale.
   Afficher les 8 premières valeurs de la somme partielle de la série $\sum_{n\geq 1} \frac{c_n}{n(n+1)}$.
3. Soit $(u_n)_{n\in\mathbb{N}^*}$ la suite définie par

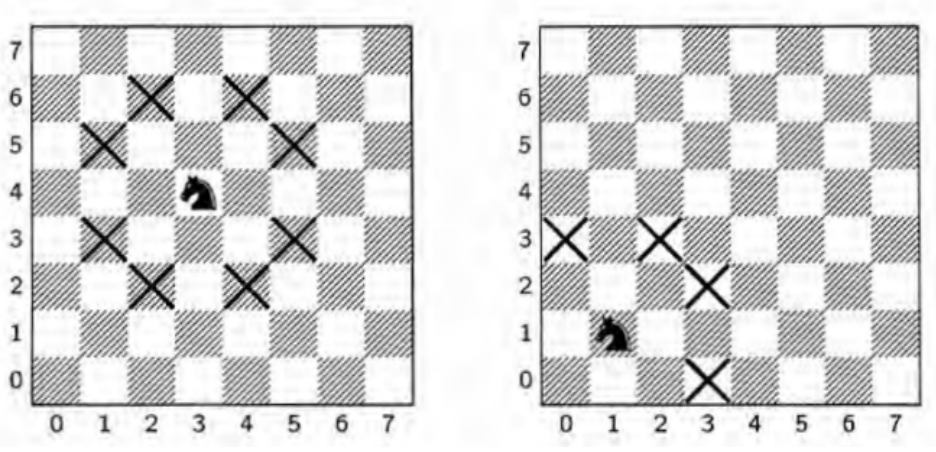
$$u_n = \begin{cases} \frac{1}{n^2} \text{ si } n \text{ ne contient pas de 9 dans son écriture en base 10} \\ 0 \text{ sinon} \end{cases}$$

   Donner une valeur approchée de la somme de la série (on ne demande pas de précision).

Exercice 2 Marche du cavalier

Un échiquier est un plateau avec 8 lignes et 8 colonnes. Ces lignes et ces colonnes seront numérotés de 0 à 7. Une position sur l'échiquier est un couple d'entiers (i, j) compris entre 0 et 7 inclus, i est le numéro de ligne, j le numéro de colonne.

Un cavalier placé sur l'échiquier se déplace en bougeant de 2 cases dans une direction (verticale ou horizontale) et de 1 case perpendiculairement. Comme le cavalier ne peut sortir de l'échiquier, il a moins de possibilités de se déplacer.



1. Écrire une fonction **Valide** prenant en argument deux entiers i et j et vérifiant que le couple (i, j) est bien une position de l'échiquier. **Valide** renvoie un booléen.

Exemple :

```
>>> Valide(4,3)      >>> Valide(4,9)
True                 False
```

2. Écrire une fonction **CoupSuivant** prenant en argument une position (i, j) et renvoyant la liste des positions que peut atteindre un cavalier placé en (i, j) en un seul coup.

Exemple :

```
>>> CoupSuivant(1,1)
[(3, 2), (3, 0), (2, 3), (0, 3)]
>>> CoupSuivant(4,3)
[(6, 4), (6, 2), (5, 5), (5, 1), (3, 5), (3, 1), (2, 4), (2, 2)]
```

3. On veut écrire une fonction **Cavalier** prenant en argument une case (i, j) et renvoyant une matrice M de taille 8×8 telle que $M[i][j]$ est le nombre minimum de coups nécessaires à un cavalier situé initialement en (i_0, j_0) pour arriver à la position (i, j).

a) Écrire une fonction **Initialisation**(i_0, j_0) renvoyant une matrice (une liste de liste) 8×8 contenant la valeur par défaut **None** partout sauf la case de coordonnées (i_0, j_0) contenant la valeur 0.

b) La fonction principale est fondée sur l'algorithme suivant :

- On initialise une matrice M par la fonction précédente ;
- on initialise une variable **NombreCoups** à 1. La variable compte le nombre de coup minimal pour que le cavalier arrive sur une case ;
- on initialise une liste **ListeCasesAtteintes** à la valeur (i_0, j_0) ;
- tant que **ListeCasesAtteintes** est non vide, on crée une liste **SuccesseursNonRencontres** contenant tous les successeurs des couples stockés dans **ListeCasesAtteintes** et non encore rencontrés : on incrémente **NombreCoups**, on affecte aux entrées de M associées à ces nouvelles cases la valeur (nouvelle) de **NombreCoups** et on remplace **ListeCasesAtteintes** par **SuccesseursNonRencontres**.

Compléter les lignes manquantes ou incomplètes dans le script proposé ci-dessous :

```
def Cavalier(i0,j0):
    M=Initialisation(i0,j0)
    NombreCoups=1
    ListeCasesAtteintes=...
    while ListeCasesAtteintes != ...:
        SuccesseursNonRencontres=[]
        for couple in ListeCasesAtteintes:
            i,j=couple
            Liste=CoupSuivant(i,j)
            for coordonne in Liste:
                ii,jj=coordonne
                if M[ii][jj] == None:
                    ...
                    ...
        ListeCasesAtteintes=SuccesseursNonRencontres
        ...
    return(M)
```

Exemple :

```
>>> Cavalier(0,0)
[[0, 3, 2, 3, 2, 3, 4, 5],
 [3, 4, 1, 2, 3, 4, 3, 4],
 [2, 1, 4, 3, 2, 3, 4, 5],
 [3, 2, 3, 2, 3, 4, 3, 4],
 [2, 3, 2, 3, 4, 3, 4, 5],
 [3, 4, 3, 4, 3, 4, 5, 4],
 [4, 3, 4, 3, 4, 5, 4, 5],
 [5, 4, 5, 4, 5, 4, 5, 6]]
```

Exercice 3 Carré de Polybe

Le carré de Polybe est une ancienne technique de chiffrement consistant à remplacer une lettre de l'alphabet par un couple d'entiers correspondant aux coordonnées (ligne,colonne) dans le tableau suivant :

	0	1	2	3	4	5
0	a	b	c	d	e	é
1	è	f	g	h	i	j
2	k	l	m	n	o	p
3	q	r	s	t	u	v
4	w	x	y	z	à	ê
5	ù	.	,	"		!

1. Écrire une fonction **Coord** qui prend en argument un caractère du carré et renvoie les deux entiers (ligne,colonne) qui sont les coordonnées du caractère dans le carré.

2. Écrire une fonction **Chiffrage** qui prend en argument une chaîne de caractère et qui renvoie la liste de chiffres issus du carré.

3. Écrire une fonction **Dechiffrage** qui prend en argument une liste de chiffres et qui renvoie la chaîne de caractère correspondant au carré.

Déchiffrer [0, 1, 3, 1, 0, 0, 3, 5, 2, 4, 5, 4, 4, 4, 5, 4, 3, 3, 2, 4, 3, 4, 3, 2, 5, 4, 0, 2, 0, 4, 3, 4, 4, 1, 5, 4, 3, 0, 3, 4, 1, 4, 5, 4, 2, 4, 2, 3, 3, 3, 5, 4, 3, 1, 0, 5, 3, 4, 3, 2, 3, 2, 1, 4, 5, 4, 0, 2, 0, 4, 3, 3, 5, 4, 0, 4, 4, 1, 0, 4, 3, 1, 0, 2, 1, 4, 0, 2, 0, 4, 5, 4, 5, 5].

4. Une variante un peu plus complexe consiste à chiffrer un caractère selon la parité de la position. Si la position est paire, elle est chiffrée par le numéro de ligne suivi du numéro de colonne ; si la position est impaire, elle est chiffrée par le numéro de colonne suivi du numéro de ligne.

Écrire les fonctions **ChiffrageParite** et **DechiffrageParite** selon la méthode décrite.

Déchiffrer [3, 5, 4, 2, 3, 4, 2, 3, 5, 4, 5, 4, 3, 3, 4, 0, 3, 2, 4, 5, 3, 5, 1, 3, 0, 0, 4, 1, 2, 2, 4, 0, 2, 3, 3, 3, 5, 4, 3, 3, 3, 1, 0, 1, 3, 2, 4, 5, 1, 1, 4, 2, 3, 1, 3, 3, 5, 1, 4, 5, 1, 1, 5, 0, 2, 1, 4, 1, 0, 2, 4, 1, 3, 3, 0, 0, 3, 3, 4, 1, 2, 4, 3, 2, 3, 2, 4, 5, 5, 5].

5. Une autre variante consiste à chiffrer une phrase à l'aide d'une autre phrase considérée comme une clé. Le principe est le suivant :

- les coordonnées du premier caractère de la phrase sont additionnées modulo 6 aux coordonnées du premier caractère de la clé ;

- on fait de même avec les deuxièmes caractères de la phrase et de la clé puis les troisièmes, etc.
- lorsque tous les caractères de la clé ont été utilisés, on reprend au premier caractère de la clé ;
- on renvoie la liste des coordonnées ainsi calculées.

Écrire la fonction **ChiffrageCle** qui prend en argument la clé et la phrase à chiffrer et qui renvoie la liste d'entiers calculée selon l'algorithme précédent.

6. Écrire une fonction de déchiffrage **DechiffrageCle** avec une clé donnée.

Déchiffrer [5, 0, 1, 2, 0, 1, 5, 2, 2, 0, 2, 1, 2, 3, 0, 0, 4, 0, 5, 0, 4, 1, 0, 1, 0, 2, 5, 0, 2, 2, 1, 1, 3, 3, 3, 2, 4, 2, 2, 2, 3, 0, 4, 0, 2, 3, 4, 3, 4, 3, 2, 2, 1, 5, 4, 5, 2, 3, 2, 0, 1, 3, 3, 5, 1, 2, 2, 3, 3, 5, 2, 2, 0, 4, 2, 2, 5, 4, 2, 4, 1, 0, 4, 2, 3, 0, 0, 2, 3, 0, 3, 5, 3, 3, 3, 0, 2, 0, 2, 5, 2, 3, 2, 1, 1, 3, 5, 4, 3, 2, 5, 2, 0, 2, 3, 1, 2, 2, 5, 2, 5, 5, 3, 1, 4, 2, 2, 2, 2, 5, 4, 2, 5, 2, 1, 2, 2, 3, 2, 5, 5, 4, 4, 4, 1, 2, 2, 2, 2, 3, 0, 0, 4, 5, 2, 3, 3, 0, 5, 3, 2, 1, 1, 2, 5, 4, 2, 4, 1, 1, 2, 3, 2, 3].

La clé est 'vive la psi'.

Exercice 4 Polynômes

La bibliothèque Numpy propose un type informatique pour gérer les polynômes d'une variable. Cf la documentation centrale.

```
>> import numpy as np
>> from numpy.polynomial import Polynomial
   >> p = Polynomial([1,2,0,3])
   représente 3X^3 + 2X + 1
   >> p
```

1. Prévoir les réponses de Python aux lignes de commande suivantes :

```
>> 2*p
>> (2*p).coef
>> p**2
>> print p**2 - 2*p
```

   Il existe également `p.deriv()` et `p.integ()` pour dériver ou intégrer `p`.

2. On pose

```
>> >> q = Polynomial([3,2,1,5,4])
```

   Interpréter la réponse de Python au calcul de `q//p`.

3. Expliquer les lignes de commande suivantes :

```
import math
import numpy as np
import scipy as sp
from numpy.polynomial import Polynomial
from scipy.integrate import quad
```

```
import matplotlib.pyplot as plt
n = 30
cano = []
cano.append(Polynomial([1]))
cano.append(Polynomial([0,1]))
for i in range(2,n+1):
cano.append(cano[1]*cano[i-1])
```

4. On définit le produit scalaire suivant sur l'ensemble des polynômes (on ne demande pas de justifier que c'est un produit scalaire) :

$$\langle P, Q\rangle = \int_{-1}^{1} P(t)Q(t)\,dt$$

   Ecrire une fonction `scalaire` prenant en argument un couple de deux polynômes `(p,q)` et renvoyant la valeur de leur produit scalaire, et une fonction `norme` prenant en argument un polynôme `p` et renvoyant la valeur de sa norme (racine carrée de $\langle P, P\rangle$).

5. On note $(L_i)_{i\in\mathbb{N}}$ la famille de polynômes obtenue en orthonormalisant, par le procédé de Schmidt, la base canonique. Les L_i sont appelés les polynômes de Legendre. On a donc :

$$L_i = \frac{X^i - \sum_{j=0}^{i-1}\langle L_j, X^i\rangle L_j}{\|X^i - \sum_{j=0}^{i-1}\langle L_j, X^i\rangle L_j\|}$$

   Calculer les n premiers polynômes de Legendre (n fixé) : on pourra renvoyer le résultat sous forme d'une liste `Legendre` puisque le calcul de `Legendre[i]` fait appel aux valeurs de `Legendre[j]` pour $j < i$.