Lycée Faidherbe, 2019-2020

T.P. d'informatique

MP, PC & PSI

Version du 20 janvier 2020

Table des matières

TP I

Révisions

Résumé

Les exercices ici proposés sont classés en fonction de leur niveau de difficulté. Tous les étudiants doivent savoir faire les exercices de niveaux 1 sans problème et parvenir à une solution pour les exercices de niveau 2. Pour chaque algorithme proposé, on précisera sa complexité en fonction des paramètres fournis.

1 Niveau 1

Exercice 1
Proposer une fonction `testPres(L,x)` qui teste la présence de l'élément x dans la liste L.
Exemple : `testPres([1,2,27,3],-7) -> False.`

Exercice 2
Écrire une fonction `maxPerso(L)` qui retourne le maximum de la liste L sans utiliser la fonction max de python.

Exercice 3
Écrire une fonction `IndiceMax(L)` qui retourne un indice du maximum de la liste L.

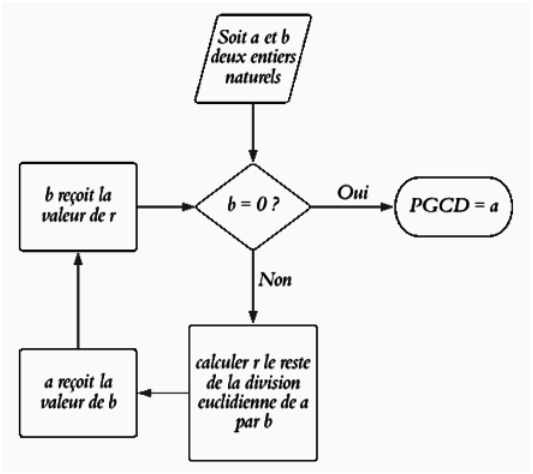
Exercice 4
Proposer une fonction qui calcule les coefficients binomiaux en utilisant un calcul de factorielles.
Exemple : `binomial(3,10) -> 120.`

Exercice 5
Proposer une fonction renvoyant la liste obtenue en retournant une liste L fournie en paramètre.
Proposer ensuite une procédure réalisant cela "sur place".
Exemple : `inverse([2,4,7,13]) -> [13,7,4,2].`

Exercice 6
On rappelle que l'opérateur `%` calcule le reste de la division (entière).
Comment cet opérateur permet-il de tester la divisibilité ?
Proposer une fonction `pgcd(n, p)` qui détermine le plus grand diviseur commun de n et p entiers positifs. On utilisera un algorithme simple qui teste tous les nombres de 1 au minimum de n et p.

Exercice 7

L'algorithme d'Euclide offre une amélioration avec une complexité logarithmique en a et b. Son principe est symbolisé dans le schéma ci-joint.
Expliciter les valeurs de a et b après un passage lorsque, au départ de la boucle, on a $b > a$.
Avec cette méthode, écrire une fonction `Euclide(a,b)` qui retourne le PGCD des entiers naturels a et b.



Exercice 8
Proposer une fonction `phi` qui prenant en paramètre un entier naturel non nul n, retourne le nombre d'entiers compris entre 1 et n, premiers avec n.
Exemple : `phi(6) -> 2`.

Exercice 9
Proposer une fonction permettant d'obtenir l'écriture en base 2 de tout nombre entier positif (sous forme de liste de 0 et de 1), puis proposer une fonction réciproque.
On mettra la liste sous forme des puissances de 2 croissantes c'est-à-dire que $L = [a_0, ..., a_{k-1}]$ représente $n = a_0 + 2a_1 + 4a_2... + 2^{k-2}a_{k-2} + 2^{k-1}a_{k-1}$.
Exemple : `base10To2(13) ->[1, 0, 1, 1] ; base2To10([1, 0, 0, 1, 1]) -> 25`

Exercice 10
Déterminer tous les triplets pythagoriciens $[a, b, c]$ tels que a et b sont des entiers de $[\![1, 100]\!]$ tels que $a \leqslant b$ et $a^2 + b^2 = c^2$.
Réponse : il y en a 63.

Exercice 11
Proposer une fonction `testPrime` qui teste si un nombre fourni en paramètre est premier.
Exemple : `testPrime(47) -> True`

Exercice 12
On considère une suite récurrente linéaire d'ordre trois donnée par : u_0, u_1, u_2 puis la relation

$$u_{n+3} = au_{n+2} + bu_{n+1} + cu_n$$

. Proposer une fonction qui prenant en paramètres $u_0, u_1, u_2, a, b, c, n$ retourne u_n.
Exemple : `suiteRec(1,2,3,1,1,1,3) -> 6.`

Exercice 13
On fixe n, un entier supérieur à 100. Tester sur les nombres p premiers inférieurs à 100 le petit théorème de Fermat : si p ne divise pas n, alors l'entier $n^{p-1} - 1$ est divisible par p.

2 Niveau 2

Exercice 14
Écrire une fonction `DeuxMax(L)` qui retourne les deux plus grandes valeurs de la liste L supposées sans doublon.

Exercice 15
Proposer une fonction `testPresDicho(L,x)` qui teste la présence de l'élément x dans la liste triée dans l'ordre croissant L, en procédant à une dichotomie.

Exercice 16
Proposer une fonction qui calcule les coefficients binomiaux en utilisant le triangle dit de Pascal. Pour calculer $\binom{n}{p}$ on calculera la liste des $\binom{m}{k}$, k variant de 0 à m, pas-à-pas pour les entiers m allant de 0 à n.

Exercice 17
Une liste L de booléens représente les résultats successifs d'une expérience de Bernoulli. Proposer une fonction qui détermine les longueurs des suites de succès consécutifs.
Exemple : `listeSucces([False,True,True,True,False,True,True]) -> [3,2]`

Exercice 18
Une application de l'ensemble $[\![1,n]\!]$ dans l'ensemble $[\![1,p]\!]$ est modélisée par la donnée de l'entier p ainsi que la liste ordonnée des images des éléments de $[\![1,n]\!]$.
Proposer deux fonctions déterminant si une telle application est injective, puis surjective.
Exemple : `testInj(5,[1,3]) -> True ; testSurj(3,[1,3,1]) -> False.`

Exercice 19
Proposer une fonction qui prenant en paramètre une liste contenant au moins deux entiers, détermine les deux éléments les plus proches. On pourra proposer deux complexités différentes.
Exemple : `plusProches([2,13,7,15,28]) -> [13,15]`.

Exercice 20
Proposer une fonction qui détermine si un nombre est égal à la somme des cubes de ses chiffres.
Trouver les entiers égaux à la somme des cubes de leurs chiffres (toutes les solutions ont 3 chiffres).

Exercice 21
On se donne une liste de couples $LVal$ (donnés sous formes de listes de deux éléments).
Par exemple `LVal=[[1,3],[2,7],[3,15]]`. On dit d'une fonction f qu'elle valide $LVal$ si elle est compatible avec chaque couple de valeur. Dans notre exemple, cela se traduirait par $f(1) = 3, f(2) = 7, f(3) = 15$. On se donne également une liste de fonctions $LFct$. Proposer une fonction `testFonction(LVal,LFct)` prenant en paramètres $LVal$ et $LFct$ et déterminant la sous-liste de $LFct$ constituée des fonctions qui valident $LVal$.

Exercice 22
Définir une fonction qui, prenant en paramètre un entier n, retourne le n-ième nombre entier positif supérieur à 10 dont l'écriture en base 10 est symétrique.

Exercice 23 — Mines-Ponts 2018
Une liste $Lalt$ modélise la liste des altitudes d'une succession de points sur un chemin. Proposer une fonction qui retourne les hauteurs des dénivelés successifs des montées et des descentes.
Exemple : `listeDeniveles([0,2,7,13,12,7,15,22,21,25,37]) -> [13,-6,15,-1,16].`

Exercice 24
Proposer une fonction qui détermine la liste de nombres premiers inférieurs ou égaux à un entier n fourni en paramètre, en utilisant la méthode du crible d'Eratosthène.
Exemple : `crible(22) -> [2,3,5,7,11,13,17,19]`

Exercice 25
Écrire une fonction `goldbach(n)` qui retourne un couple de nombres premiers (a, b) tels que $a+b = n$ lorsque n pair. On pourra utiliser l'exercice 2.2.
Exemple : `goldbach(232) -> (3, 229)` et `goldbach(252) -> (11, 241)`
Tester la conjecture selon laquelle tout entier pair supérieur à trois est la somme de deux nombres premiers, jusque 10.000.000 (Conjecture de Golbach).

3 Niveau 3

Exercice 26
Proposer une fonction qui détermine si un nombre n fournis en paramètre peut s'écrire comme la somme de cubes d'entiers strictement supérieurs à 1.
On pourra introduire une matrice de booléens B telles que $B[i, j]$ vaut vrai si et seulement si i est somme de j cubes d'entiers strictement supérieurs à 1.
Exemple : `testSommeCubes(24) -> True`

Exercice 27
On se donne un entier n et trois entiers $a < b < c$ non nuls. écrire une fonction qui donne le nombre de listes à valeurs $\{a, b, c\}$ dont la somme des termes vaut n. Par exemple, les 7 listes $[1, 1, 1, 1], [1, 1, 2], [1, 2, 1], [2, 1, 1], [2, 2], [1, 3], [3, 1]$ sont à dénombrer lors de l'appel à `nbrSommes(4,[1,2,3])`.

Exercice 28
On appelle permutation de l'ensemble $[\![1, n]\!]$ une bijection de cet ensemble dans lui-même. Proposer une fonction qui, prenant en paramètre n, retourne la liste des permutations de l'ensemble $[\![1, n]\!]$.
Exemple : `listePerm(3) -> [[1,2,3],[1,3,2],[2,1,3],[2,3,1],[3,1,2],[3,2,1]]`

Exercice 29 — Exercice des olympiade US, USACO
La plupart des sujets des Usaco relatent les aventures de John le fermier et de ses vaches, dont Bessie est la star incontestée.
Bessie suit un régime tel qu'elle ne peut pas manger plus de C ($10 \leqslant C \leqslant 35000$) calories par jour. John le fermier la taquine en plaçant B ($1 \leqslant B \leqslant 21$) seaux de fourrage, chacun ayant un nombre (pas nécessairement unique) de calories (dans l'intervalle $[\![1, 35000]\!]$). Bessie ne sait pas se contrôler : une fois qu'elle commence à manger le contenu d'un seau, elle mange tout ce qu'il contient.
Bessie n'est pas très douée pour la combinatoire. Déterminer la combinaison optimale de seaux de fourrage qui donne à Bessie autant de calories que possible sans dépasser la limite C.
Par exemple, considérer une limite de 40 calories et 6 seaux de dimensions 7, 13, 17, 19, 29 et 31. Bessie peut manger $7 + 31 = 38$ calories mais peut en manger encore plus en consommant trois seaux $7 + 13 + 19 = 39$ calories. Elle ne peut pas trouver de meilleure combinaison.

4 Solutions

Solution de l'exercice 1 -

```
def testPres(L,x):
    n=len(L)
    for k in range(n):
        if L[k] == x:
            return True
    return False
```

Solution de l'exercice 2 -

```
def maxPerso(L):
    indice=0
    max = L[0]
    for j in range(1,len(L)):
        if L[j]> L[indice]:
            indice = j
            max = L[j]
    return indice
```

Solution de l'exercice 3 - C'est en fait plus simple.

```
def IndiceMax(L):
    indice=0
    for j in range(1,len(L)):
        if L[j]> L[indice]:
            indice = j
    return indice
```

Solution de l'exercice 4 -

```
def factoriel(n):
    f=1
    for k in range(2,n+1):
        f=f*k
    return f

def binomial(p,n):
    return factoriel(n)//(factoriel(p)*factoriel(n-p))
```

Solution de l'exercice 5 -

```
def inverse(L):
    n=len(L)
    miroir =[]
    for k in range(n):
        miroir.append(L[-k-1])
    return miroir
```

```
def inverseSurPlace(L):
    '''L est modifie après appel a la fonction'''
    n=len(L)
    k=0
    while k<n-k-1:
        temp=L[k]
        L[k] = L[-k-1]
        L[n-k-1] = temp
        k=k+1
```

Solution de l'exercice 6 -

```
def pgcd(a,b):
    commun = 1
    for k in range(1, min(a,b)+1):
        if (a%k == 0) and (b%k == 0 ):
            commun = k
    return commun
```

Solution de l'exercice 7 -

```
def Euclide(a,b):
    while b>0:
        r=a%b
        a=b
        b=r
    return a
```

Solution de l'exercice 8 -

```
def phi(n):
    compteur = 1
    for k in range(2,n):
        if pgcd(k,n) == 1:
            compteur = compteur + 1
    return compteur
```

Solution de l'exercice 9 -

```
def base10To2(n):
    nb = n
    liste = []
    while nb > 0:
        liste.append(nb%2)
        nb = nb//2
    return liste

def base2to10(L):
    nb = 0
    n = len(L)
    for k in range(n):
        nb = nb + (2**k)*L[k]
    return nb
```

Solution de l'exercice 10 -

```
def testCarre(n):
    racine = int(n**0.5)
    return (racine**2 == n, racine)

def triplet():
    Liste=[]
    for a in range(1,100):
        for b in range(a,100):
            (booleen, racine) = testCarre(a*a+b*b)
            if  booleen:
                Liste.append([a,b,racine])
    return(Liste)
```

Solution de l'exercice 11 -

```
def testPrime(n):
    k = 2
    while k*k <= n:
        if n%k == 0:
            return False
        k = k+1
    return(True)
```

Solution de l'exercice 12 -

```
def suiteRec(u0, u1, u2, a, b, c, n):
    v0, v1, v2 = u0, u1, u2
    for k in range(n):
        temp= a*v2 + b*v1 + c*v0
        v0 = v1
        v1 = v2
        v2 = temp
    return v0
```

Solution de l'exercice 13 -

```
def testFermat(n):
```

```
    Liste = []
    for k in range(2,100):
        if testPrime(k):
            Liste.append(k)
    for p in Liste:
        if (n%p != 0):
            if not((n**(p-1)-1)% p ==0):
                return False
    return True
```

Solution de l'exercice 14 -

```
def DeuxMax(L):
    '''indice1 contient l'indice du max et indice2 contient l'
       indice du 2e max'''
    if L[0] <L[1]:
        indice1=1
        indice2=0
    else:
        indice1=0
        indice2=1
    for k in range(2,len(L)):
        if L[k]>L[indice1]:
            indice2=indice1
            indice1=k
        elif L[k]>L[indice2 ]: # dans ce cas L[indice2]<L[k]<L
           [indice1]
            indice2=k
    return (L[indice2], L[indice1])
```

Solution de l'exercice 15 -

```
def dicho(L,x):
    '''teste la presence de x dans L'''
    i = 0
    j = len(L)
    while i < j:       # recherche dans L[i:j] (tranche non
       vide)
        m = (i+j)//2  # indice du milieu de tranche
        if L[m] == x:
            return True
        if x < L[m]:
            j = m      # recherche dans la partie basse de la
               tranche
        else:
            i = m + 1 # recherche dans la partie haute de la
               tranche
    # en sortie i >= j donc la tranche L[i:j] est vide
    return False
```

Solution de l'exercice 16 -

```
def binom(n,p):
    liste=[1,1] # pour n=1
    for k in range(n-1): # ici liste est celle de k+1
        listeTemp = [1]
        for j in range(len(liste)-1):
            listeTemp.append(liste[j]+liste[j+1])
        liste=listeTemp
        liste.append(1)# ici liste est celle de k+2
    return liste[p]
```

On peut n'utiliser qu'une seule liste.

```
def binom(n,p):
    liste=[1]*(n+1)
    for m in range(n): on calcule les k parmi m
        for k in range(k-1, 0, -1):
            liste[k] = liste[k] + liste[k-1]
    return liste[p]
```

Solution de l'exercice 17 -

```
def listeSucces(L):
    liste = []
    lg = 0
    for k in range(len(L)):
        if L[k]:
            lg = lg + 1
        else:
            if lg > 0:
                liste.append(lg)
                lg = 0
    if lg > 0:
        liste.append(lg)
    return liste
```

Solution de l'exercice 18 -

```
def testInj(n,p,L):
    arrivee = [False]*(p+1) # la valeur d'indice 0 ne compte
       pas
    for y in L:
        if arrivee[y]:
            return False   # la valeur y est deja une image
        arrivee[y] = True  # desormais y est une image
    return True
```

```
def testSurj(n,p,L):
    arrivee = [False]*(p+1)
    for y in L:
        arrivee[y]=True # desormais  y est connue comme image
    for k in range(1,len(arrivee)):
        if not(arrivee[k]):
            return False # k n'a pas d'antecedent
    return True
```

Solution de l'exercice 19 -

```
def plusProches(L):
    n = len(L)
    indice1=0
    indice2=1
    distance = abs(L[1] - L[0])
    for i in range(n):
        for j in range(i+1,n):
            if abs(L[i]-L[j])< distance:
                distance=abs(L[i] - L[j])
                indice1=i
                indice2=j
    return (distance, L[indice1], L[indice2])
```

La complexité est quadratique, on peut l'améliorer en quasi linéaire si on trie la liste.

```
def plusProches2(L):
    n = len(L)
    liste=L*1
    liste.sort()
    indice=0
    distance = abs(liste[1] - liste[0])
    for i in range(n-1):
        if abs(liste[i] - liste[1+i])< distance:
                distance=abs(liste[i] - liste[1+i])
                indice=i
    return (distance, liste[indice],liste[1+indice])
```

Solution de l'exercice 20 -

```
def listeChiffres(n):
    nb = n
    liste = []
    while nb >0:
        liste.append(nb % 10)
        nb = nb//10
    return liste

def sommeCube(L):
    s = 0
    for x in L:
        s = x**3 + s
    return s

def testCube(n):
    return n == sommeCube(listeChiffre(n))

def cherche():
    for k in range(2, 1000):
        if testCube(k):
            print(k)
```

Solution de l'exercice 21 -

```
def testFonction(LVal,LFct):
    liste = []
    for f in LFct:
        test=True
        for x in LVal:
            if f(x[0]) != x[1]:
                test = False
        if test:
            liste.append(f)
    return liste
```

Solution de l'exercice 22 - Exercice 20 pour `listeChiffres`, exercice 5 pour `inverse`.

```
def est_miroir(n):
    chf = listeChiffres(n)
    fhc = inverse(chf)
    return chf == fhc

def sym(n):
    compteur=0
    k=10
    while compteur < n:
        k=k+1
        if est_miroir(k):
            compteur = compteur + 1
    return k
```

Solution de l'exercice 23 -

```
def listeDeniveles(L):
    liste=[]
    n=len(L)
    signe = L[1] - L[0]
    indiceDebut = 0
    for k in range(2, n):
        if signe*(L[k]-L[k-1]) < 0:
            signe =-signe
            liste.append(L[k-1]-L[indiceDebut])
            indiceDebut = k-1
    liste.append(L[n-1]-L[indiceDebut])
    return liste
```

Solution de l'exercice 24 -

```
def crible(n):
    tableau=[True]*(n+1) # l'indice 0 sera inutile
    for k in range(2, n+1):
        if tableau[k]:
            indice = 2*k
            while indice <= n:
                tableau[indice] = False
                indice = indice+k
    liste=[k for k in range(2, n+1) if tableau[k]]
    return liste
```

Solution de l'exercice 25 -

```
def goldbach(n):
    liste = crible(n)
    for j in liste:
        if n-j in liste:
            return j,n-j
```

Solution de l'exercice 26 -

```
def sommeCubes(n):
    somme3 = [False]*(n+1)
    somme3[0] = True
    for i in range(1, n+1):
        k = 2
        while k**3 <= i:
            somme3[i] = somme3[i] or somme3[i - k**3]
            k = k + 1
    return somme3[n]
```

Solution de l'exercice 27 -

```
def nbrSommesIter(n,L):
    [a,b,c]=L
    import numpy as np
    B=np.zeros((n+1,n+1),int)
    B[a][1]=1
    B[b][1]=1
    B[c][1]=1
    for j in range(2,n+1):
        for i in range(a,n+1):
            B[i][j] =   B[max(0,i-a)][j-1]
                      + B[max(0,i-b)][j-1]
                      + B[max(0,i-c)][j-1]
    return sum(B[n])
```

Solution de l'exercice 28 -

```
from copy import deepcopy

def listePerm(n):
    liste=[[1]]
    for k in range(2,n+1):
        newL = []
        for permutation in liste:
            for j in range(len(permutation)+1):
                newL.append(  permutation[:j]
                            + [k]
                            + permutation[j:])
        liste = deepcopy(newL)
    return liste
```

Solution de l'exercice 29 -

```
def bessie(CalMax, nbSeaux, Lprime):
    L = [0] + Lprime
    L.sort()
    B=np.zeros((1+nbSeaux, 1+CalMax))
    for p in range(1, 1+CalMax):
        if L[1] <= p:
            B[1][p] = L[1]
    for k in range(1+nbSeaux):
        for p in range(1,1+CalMax):
            if p<L[k]:
                B[k][p] = B[k-1][p]
            else:
                B[k][p] = max(B[k-1][p], B[k-1][p-L[k]]+L[k])
    return B[nbSeaux][CalMax]
```

TP II

Images couleurs

1 Rappel : images monochromes

Une image est représentée dans un ordinateur sous la forme d'une mosaïque de petits carrés appelés `pixels`[1]. Ces éléments sont codés dans un tableau dont les lignes et les colonnes correspondent aux positions dans l'image. On peut remarquer que cette décomposition en points est aussi celle qu'utilise notre œil qui reçoit la lumière dans des éléments appelés bâtonnets dans la rétine.

Python possède une interface qui permet de traiter ces images sous la forme de tableaux gérés par le module `numpy`.
Nous avons étudié en première année cette interface dans le cas des images monochrome qui sont en fait l'écriture d'une fonction de deux variables dont la valeur peut représenter la luminosité.

Les outils

- On doit charger les bibliothèques

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

- Pour afficher une image que l'on a définie par une matrice on utilise `plt.imshow(image1)`.
- Par défaut les niveaux ne sont pas visualisés en niveau de gris. On peut ajouter l'échelle des couleurs employées avec `plt.colorbar()`
- On change l'échelle des couleurs avec le paramètre `cmap` (pour Color MAP)
  `plt.imshow(image1,cmap='gray')`
- On peut voir les différentes échelles de couleurs à l'adresse

  `http://matplotlib.org/examples/color/colormaps_reference.html`

- Python peut effectuer un lissage utile pour améliorer le rendu. On peut l'activer avec le paramètre `interpolation`
  `plt.imshow(image1,cmap='gray',interpolation='bilinear')`
- On peut lire une image depuis la mémoire de masse :
  `img = plt.imread("/chemin/vers/l/image/mon_image.png")`
  `img` est alors connu par python sous la forme d'un tableau numpy.
- Le nom complet du chemin est accessible dans le gestionnaire de fichiers (file browser) de Pyzo par un clic droit sur le nom du fichier.
  On choisit "Copier le nom complet (chemin) du fichier".

1. **pic**ture **el**ements

- Il y a une image monochrome dans le module `scipy` :

```
from scipy.misc import ascent

a = ascent()
plt.imshow(a,cmap='gray')
```

- La taille d'une image est celle de la matrice associée que l'on obtient par `np.shape`.

2 Images en couleur

2.1 Première exploration

`scipy` contient aussi une image en couleurs, un mignon raton-laveur.

```
from scipy.misc import face

image1 = face()
```

Exercice 1
Afficher l'image.
Quelle est sa taille ?
On remarque que la matrice a trois dimensions.
De plus l'image est tout de suite vue en "vraies couleur", `cmap` n'a aucun effet.

2.2 Séparation des couleurs

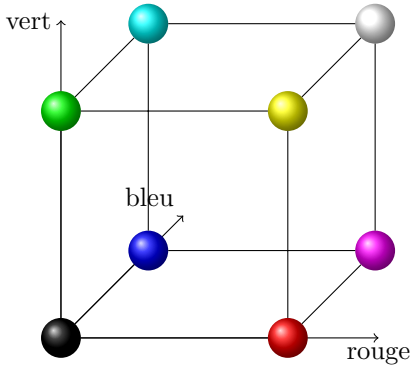
En effet une couleur est définie par 3 composantes donc chaque point d'une image couleur est représenté par 3 valeurs. On a donc affaire à une structure de données à 3 dimensions : la hauteur, la largeur et la profondeur de couleur qui peut être 3 [2]. Dans l'interface `matplotlib` les valeurs de ce tableau sont le plus souvent des flottants compris entre 0 et 1 ou des entiers sur 8 bits, compris entre 0 et 255.
Pour voir l'effet de chaque couche nous allons créer des image.
On crée la matrice à l'aide de la fonction `np.zeros`, par exemple `imageR = np.zeros((300,300,3))` ; on rappelle que les dimensions sont définies par une variable unique, c'est un tuple, d'où les doubles parenthèses.
Pour donner une valeur à une partie de la matrice on peut employer l'extraction, vue pour les listes, mais qui se généralise.
Par exemple on peut donner la valeur 1 dans un carré au centre de l'image pour la première couche avec l'instruction `imageR[100:200,100:200,0] = 1`
Les trois dimensions sont séparées par des virgules, pour la première dimension (la hauteur) on prend les valeurs de 100 à 199, pour la deuxième (la largeur) on a le même intervalle et on ne transforme que la première couche.

Exercice 2
Afficher l'image correspondante.
Faire de même pour les autres couches.
On peut afficher plusieurs images en même temps :

```
plt.clf()
plt.subplot(1,3,1)
# 1 ligne de 3 colonnes, 1 pour la première image
plt.imshow(imageR)
plt.subplot(1,3,2)
plt.imshow(imageV)
```

2. la profondeur peut être 4 quand on ajoute une composante de transparence.

```
plt.subplot(1,3,3)
plt.imshow(imageB)
```



On voit donc que les couleurs sont séparées en 3 composantes : rouge, vert et bleu.
On notera (r, v, b) les composantes de la couleur d'un point elles sont obtenues par `image[i,j,:]`.

La composition des couleurs se fait additivement.

Exercice 3

Définir une image de taille 300x300 nulle initialement dans laquelle on mettra les valeurs à 1 dans les partie décrites ci-après :

- *le cercle de centre (135,124) de rayon 50 pour la première couche,*
- *le cercle de centre (135,176) de rayon 50 pour la deuxième couche,*
- *le cercle de centre (180,150) de rayon 50 pour la troisième couche.*

On obtient une image classique d'illustration de la synthèse additive.

Le résultat devrait ressembler à l'image `spots.png` qui se trouve dans le dossier public.

Ce dossier contient aussi une image aux couleurs plus marquées que celle que fournit `scipy` : `bouquet.png`.

Exercice 4

Charger cette image en mémoire.

Créer une image couleur dont les composantes vertes et bleues (indices 1 et 2) sont nulles mais qui recopie les composantes rouges (indice 0) du bouquet et l'afficher.

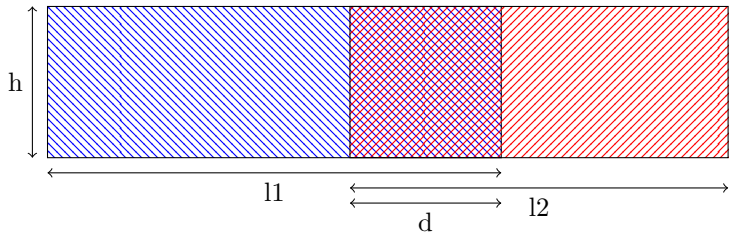
Faire de même avec les autres composantes.

3 Recollements d'images

Le but de cette partie est d'assembler deux images en essayant de ne pas faire apparaître la liaison. Si on applique à deux copies d'une même image on peut prolonger celle-ci ; c'est l'idée à la base du remplissage d'une surface par un motif.

Nous nous plaçons dans le cas particulier d'un placement côte-à-côte de deux images de même hauteur.

On note l1 la largeur de la première image et l2 la largeur de la seconde.



Voici les images qui serviront dans les exemples.


Exercice 5
Écrire une fonction `decoupe(img1,img2,d)` *qui prend l1 -* $\frac{d}{2}$ *pixels de gauche à img1 et l2-*$\frac{d}{2}$ *pixels de droite de img2 et les associe : on découpe au milieu de la zone commune.*

`decoupe(img1,img2,100)` donne 

Exercice 6
Pour obtenir une transition moins brutale on va plutôt mélanger les deux images. Écrire une fonction `melange(img1,img2,d)` *qui calcule les pixel de la bande commune sous la forme* `img[x,y,k] =` $\frac{\texttt{img1[x,y,k] + img2[x,y - l1 + d,k]}}{2}$. *Pourquoi divise-t-on par 2 ?*

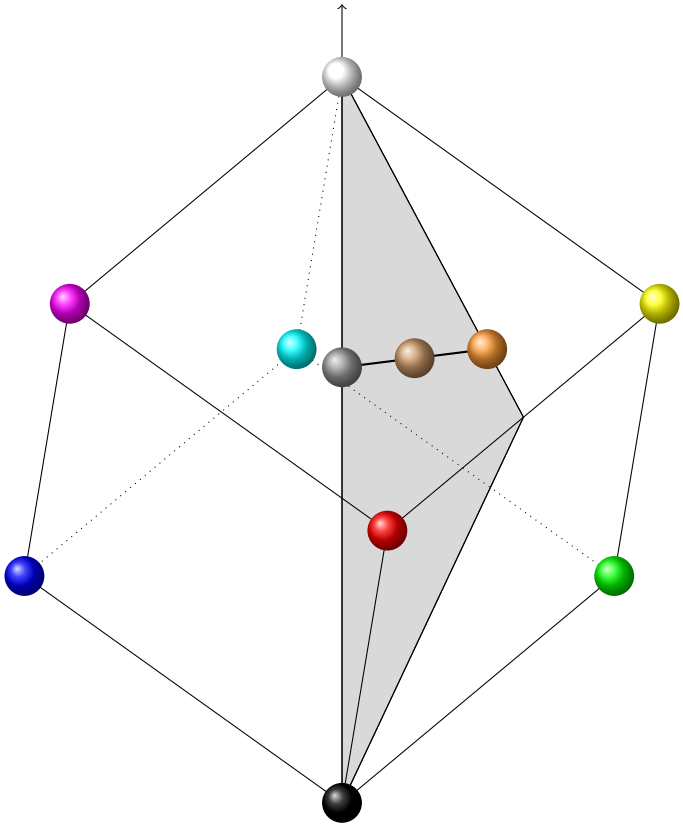
`melange(img1,img2,100)` donne 

Exercice 7
Le mélange ci-dessus manque encore de transition : on va mélanger progressivement.
Écrire une fonction `fondu(img1,img2,d)` *qui calcule les pixel de la bande commune sous la forme* `img[x,y,k] = (1-t)img1[x,y,k] + t *img2[x,y - l1 + d,k]` *avec* $t = \frac{y-l1+d}{d}$ *et* y *compris entre l1 - d et l1.*

`fondu(img1,img2,100)` donne 

4 Autre définition des couleurs

Le modèle RVB employé ci-dessus est pratique pour le calcul des images mais rend difficile une modification cohérente des couleurs. On voudrait pouvoir augmenter/diminuer le caractère coloré des images (on parle de **saturation**), augmenter/diminuer la luminosité et même transformer les couleurs ...
On a vu que les 3 composantes des couleurs décrivent un cube : $[0;1]^3$.
On peut remarquer que la diagonale du cube, de (0,0,0) à (1,1,1) représente les tons de gris et que sa direction semble indiquer la luminosité. On va donc privilégier cette direction.



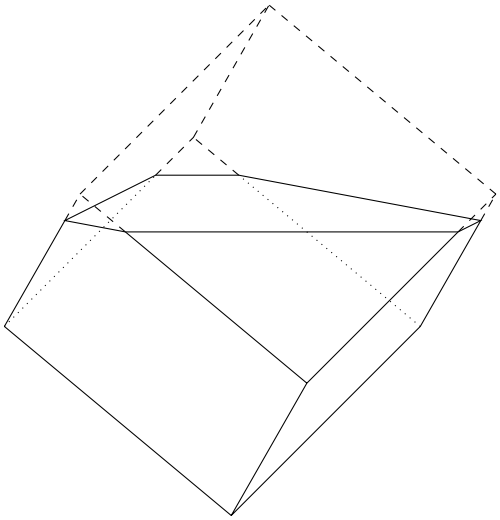
Nous allons choisir une représentation qui ressemble aux représentation HSV et HSL [3] mais dont les calculs sont simplifiés. Pour cela nous utiliserons les coordonnées cylindrique avec un axe porté par $(1,1,1)$. L'angle est déterminé de telle manière que les couleurs rouges, les points (r,0,0) ou (1,s,s), définissent un angle nul.
On commence par définir une nouvelle base orthonormée $(\vec{u}_1, \vec{u}_2, \vec{u}_3)$ dont le troisième vecteur dirige l'axe. Le choix de la couleur rouge comme angle initial impose que le plan engendré par $\vec{u}_1$ et $\vec{u}_3$ contienne le rouge, c'est-à-dire le vecteur $(1,0,0)$.
On aboutit à $\vec{u}_1 = \frac{1}{\sqrt{6}}\begin{pmatrix}2\\-1\\-1\end{pmatrix}$ $\vec{u}_2 = \frac{1}{\sqrt{2}}\begin{pmatrix}0\\1\\-1\end{pmatrix}$ et $\vec{u}_3 = \frac{1}{\sqrt{3}}\begin{pmatrix}1\\1\\1\end{pmatrix}$.
Une couleur de composantes (r,v,b) sera donc définie par les nouvelles coordonnées (X,Y,Z) avec $\begin{pmatrix}r\\v\\b\end{pmatrix} = X\vec{u}_1 + Y\vec{u}_2 + Z\vec{u}_3$ d'où $r = \frac{2X}{\sqrt{6}} + \frac{Z}{\sqrt{3}}$, $v = -\frac{X}{\sqrt{6}} + \frac{Y}{\sqrt{2}} + \frac{Z}{\sqrt{3}}$ et $b = -\frac{X}{\sqrt{6}} - \frac{Y}{\sqrt{2}} + \frac{Z}{\sqrt{3}}$
On pose alors $X = S\cos(H)$ et $Y = S\sin(\varphi)$ en coordonnées polaires donc $S = \sqrt{X^2+Y^2}$ et l'angle H est l'argument du complexe $X + iY$ que l'on peut calculer par une fonction `numpy` : `H = np.arctan2(Y,X)`.

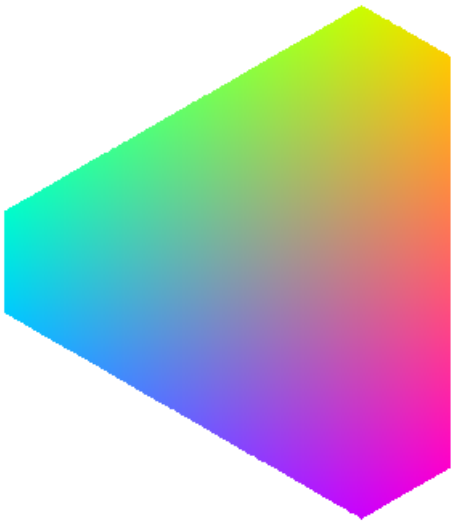
3. Voir `https://en.wikipedia.org/wiki/HSL_and_HSV`

On pose aussi $I = \dfrac{Z}{\sqrt{3}} = \dfrac{r+v+b}{3}$ pour obtenir un réel dans $[0;1]$.

- H (pour **hue**, teinte en anglais) indique la teinte.
- S est l'éloignement du gris, c'est une mesure de la saturation en couleur.
- On a vu que I mesurait la luminosité.

La saturation n'atteint pas la valeur 1 : voici les couleurs qui ont une luminosité $I = 0,6$, cela correspond à la section du cube par un plan perpendiculaire à l'axe de gris.





On remarque toutes les valeurs s'obtiennent à l'aide de fonctions simples donc on pourra les calculer globalement en appliquant les fonctions aux tableaux.
On définira les matrices des couleurs par `R = image[:,:,0]`, de même pour `V` et `B`.
La matrice des X se calcule alors par `X = (R + V + B)/np.sqrt(3)`.

Exercice 8
Écrire une fonction `HSI(image)` *qui renvoie les matrices des 3 composantes des couleurs des points.*

Exercice 9
Écrire une fonction `faireImage(H,S,I)` *qui renvoie l'image reconstituée à partir des composantes* (H,S,I) *des couleurs.*

Des composantes transformées peuvent donner des valeurs de r, g ou b qui ne sont pas dans $[0;1]$. On tronquera les valeurs pour qu'elles restent entre 0 et 1 à l'aide de la fonction `np.clip(x,0,1)` qui renvoie 0 pour $x < 0$, x pour $0 \leqslant x \leqslant 1$ et 1 pour $x > 1$.

Exercice 10
Transformer les images (raton-laveur ou bouquet). On pourra

- *modifier la teinte en ajoutant une constante à H,*
- *modifier la luminosité en multipliant I par une constante positive,*
- *modifier la saturation en multipliant I par une constante positive.*

Dans le cas des multiplications on expérimentera avec un facteur supérieur à 1 et un facteur inférieur à 1.

5 Solutions

Solution de l'exercice 1 -

```
from scipy.misc import face

image1 = face()
plt.clf()
plt.imshow(image1)

print(np.shape(image1))
```

Solution de l'exercice 2 -

```
plt.clf()
plt.imshow(imageR)

imageV = np.zeros((300,300,3))
imageV[100:200,100:200,1] = 1

imageB = np.zeros((300,300,3))
imageB[100:200,100:200,2] = 1

plt.clf()
plt.subplot(1,3,1)
plt.imshow(imageR)
plt.subplot(1,3,2)
plt.imshow(imageV)
plt.subplot(1,3,3)
plt.imshow(imageB)
```

Solution de l'exercice 3 -

```
image2 = np.zeros((300,300,3))
for i in range(300):
    for j in range(300):
        if (i -135)**2 + (j-124)**2 < 50**2:
            image2[i,j,0] = 1
        if (i -135)**2 + (j-176)**2 < 50**2:
            image2[i,j,1] = 1
        if (i -180)**2 + (j-150)**2 < 50**2:
            image2[i,j,2] = 1

plt.clf()
plt.imshow(image2)
```

Solution de l'exercice 4 -

```
image3 = plt.imread("/home/ericd13/Travail/2016-2017/IC2/TP/
   TP1 Images/bouquet.png")

hauteur,largeur,p = np.shape(image3)
image3R = np.zeros(hauteur,largeur,3) # On peut aussi définir
   image3R = image3*0
image3V = np.zeros(hauteur,largeur,3)
```

```
image3B = np.zeros(hauteur,largeur,3)

image3R[:,:,0] = image3[:,:,0]
image3V[:,:,1] = image3[:,:,1]
image3B[:,:,2] = image3[:,:,2]


plt.clf()
plt.subplot(2,2,1)
plt.imshow(image3)
plt.subplot(2,2,2)
plt.imshow(image3R)
plt.subplot(2,2,3)
plt.imshow(image3V)
plt.subplot(2,2,4)
plt.imshow(image3B)
```

Solution de l'exercice 5 -

```
def decoupe(img1,img2,d):
    h,l1,n = img1.shape
    h,l2,n = img2.shape
    ll1 = l1 - d//2
    tout = np.zeros((h,l1+l2-d,n))
    tout[:,0:ll1,:] = img1[:,0:ll1,:]
    tout[:,ll1:,:] = img2[:,d-d//2:,:]
    return tout
```

Solution de l'exercice 6 -

```
def melange(img1,img2,d):
    h,l1,n = img1.shape
    h,l2,n = img2.shape
    tout = np.zeros((h,l1+l2-d,n))
    tout[:,0:l1-d,:] = img1[:,0:l1-d,:]
    tout[:,l1:,:] = img2[:,d:,:]
    tout[:,l1-d:l1]=(img1[:,l1-d:,:]+img2[:,:d,:])/2
    return tout
```

Solution de l'exercice 7 -

```
def fondu(img1,img2,d):
    h,l1,n = img1.shape
    h,l2,n = img2.shape
    tout = np.zeros((h,l1+l2-d,n))
    tout[:,0:l1-d,:] = img1[:,0:l1-d,:]
    tout[:,l1:l1+l2-d,:] = img2[:,d:l2,:]
    for i in range(0,d):
        tout[:,l1-d+i,:] = (d-i)/d*img1[:,l1-d+i,:]+i/d*img2
           [:,i,:]
    return tout
```

Solution de l'exercice 8 -

```
def HSI(image):
    R = image[:,:,0]
    V = image[:,:,1]
    B = image[:,:,2]
    X = (2*R-V-B)/np.sqrt(6)
    Y = (V-B)/np.sqrt(2)
    I = (R+V+B)/3
    H = np.arctan2(Y,X)
    S = np.sqrt(X*X+Y*Y)
    return (H,S,I)
```

Solution de l'exercice 9 -

```
def faireImage(H,S,I):
    X = S*np.cos(H)
    Y = S*np.sin(H)
    haut, larg = np.shape(H)
    image = np.zeros((haut,larg,3))
    image[:,:,0] = 2*X/np.sqrt(6) + I
    image[:,:,1] = -X/np.sqrt(6) + Y/np.sqrt(2) + I
    image[:,:,2] = -X/np.sqrt(6) - Y/np.sqrt(2) + I
    return np.clip(image,0,1)
```

Solution de l'exercice 10 -

TP III

ÉQUATIONS DIFFÉRENTIELLES

1 Équations scalaires

Dans cette partie on considère des équations différentielles linéaires du premier ordre à variable réelle dont on connaît la solution. Pour chaque exercice, on demande

1. d'écrire la méthode de résolution de l'exercice,
2. de calculer et afficher la solution approchée avec les nombres de points indiqués,
3. de calculer et afficher la solution exacte (avec 1000 points).

On représentera les approximations et la solution sur le même graphe en mettant une légende pour identifier les courbes.

Exercice 1
Approcher les solutions de $y'(t) - 2y(t) = 4$, $y(0) = 0$ *sur* $[0; 2]$ *en utilisant la méthode d'Euler avec 10, 100 et 1000 points. La solution exacte est* $t \mapsto 2e^{2t} - 2$.

Exercice 2
Approcher les solutions de $y' + \tan(t)y = \sin(2t)$, $y(0) = 1$ *sur* $[0; \pi]$ *en utilisant la méthode d'Euler avec 10, 100 et 1000 points. La solution exacte est* $t \mapsto -2\cos^2(t) + 3\cos(t)$.
Approcher la solution sur $[-\pi; \pi]$.

Exercice 3
Approcher les solutions de $t^2y'(t) - (2t - 1)y(t) = t^2$, $y(1) = 1$ *sur* $[1; 2]$ *en utilisant la fonction* `odeint` *avec 3 points. La solution exacte est* $t \mapsto t^2$.

Exercice 4
Après avoir écrit une fonction `heun(phi, y0, T)` *qui utilise la méthode de Heun, l'utiliser pour résoudre* $(t + 1)y'(t) + y(t) = e^t$, $y(0) = 1$ *sur* $[0; 2]$ *avec 5, 10 et 20 points. La solution exacte est* $t \mapsto \frac{e^t}{t+1}$.

Pour écrire une méthode d'Euler implicite, on pourra définir l'équation dont y_{k+1} est solution en définissant une fonction dans la fonction dont on cherchera un zéro avec `scipy.optimize.fsolve`. Une bonne valeur initiale pour rechercher y_{k+1} est y_k.

Exercice 5
Écrire une fonction `eulerImplicite(phi, y0, T)` *qui donne une approximation de la solution de* $y' = \varphi(y,t)$ *avec la condition initiale* $y(t_0) = y_0$ *aux points de la liste* T *par la méthode d'Euler implicite.*
Approcher les solutions de $(t+1)y'(t) - ty(t) + 1 = 0$, $y(0) = 2$ *sur* $[0;2]$ *en utilisant la méthode d'Euler implicite avec 5, 10 et 100 points. La solution exacte est* $t \mapsto \frac{e^t+1}{t+1}$.

Exercice 6
Écrire une fonction `RK(phi, y0, T)` *qui donne une approximation de la solution de* $y' = \varphi(y,t)$ *avec la condition initiale* $y(t_0) = y_0$ *aux points de la liste* T *par la méthode de Runge-Kutta.*
Approcher les solutions de $y'(t) + y(t) = t.\big(y(t)\big)^2$, $y(0) = \frac{1}{2}$ *sur* $[0;5]$ *en utilisant la méthode de Runge-Kutta avec 10, 50 et 200 points. La solution exacte est* $x \mapsto \frac{1}{t+1+e^t}$.

Exercice 7
On considère l'équation $y' - y = \sin(t)$ *de solution* $y(t) = \frac{-1}{2}\big(\sin(t) + \cos(t)\big)$ *pour* $y(0) = \frac{-1}{2}$.
Tracer les solutions sur $[0;50]$ *données par différentes méthodes avec 1000 points*[1] *; on pourra limiter verticalement les valeurs affichées avec, par exemple,* `plt.ylim([-2,2])`.
Comment expliquer le phénomène ?

2 Équations d'ordre 2

Pour une équation d'ordre 2, on travaille avec $u(t) = (y(t), y'(t))$qui donne $u'(t) = (y'(t), y''(t))$.
On est ramené à une équation vectorielle d'ordre 2.
On emploiera des tableaux `numpy` pour coder les vecteurs.

Exercice 8
On considère l'équation $y'' + y'\tan(t) - y\cos^2(t) = 0$ *sur* $[0,\pi]$ *avec* $y(0) = 1$ *et* $y'(0) = 0$.

1. *Écrire le code de la fonction* `phi(u,t)`.
2. *Avec la méthode d'Euler, donner une solution approchée* y *du système avec une liste d'abscisses contenant 10 valeurs, 50 valeurs, 100 valeurs et 1000 valeurs.*
3. *Comparer avec la solution exacte,* $y(t) = \frac{1}{2}e^{\sin(t)} + \frac{1}{2}e^{-\sin(t)}$.

Exercice 9
Tracer les solutions de $y'' = -2y' + 3y$ *avec* $y(0) = 1$ *et* $y'(0) = -3$.
Remarquez et essayez d'expliquer ce qui se passe pour les grandes valeurs de t.

Exercice 10
Tracer les solutions de l'équation de van der Pol : $y'' = (1-y^2)y' - y$ *pour différentes conditions initiales, par exemple* $y_0 = \frac{k}{2}$, $y_0' = 0$ *pour* $k \in \{0, 1, \ldots, 9\}$.
On pourra tracer la trajectoire dans l'espace des phases (y, y'). *Que remarque-t-on ?*

Exercice 11
Tracer les solutions de l'équation du pendule non amorti, $y'' = -\sin(y)$, *pour différentes conditions initiales en utilisant la méthode d'Euler.*
Écrire une fonction `verlet(psi, y0, v0, T)` *qui renvoie une liste* `Y` *des valeurs approchées des* $y(t_i)$ *en appliquant le schéma de Verlet.*
Résoudre l'équation du pendule amorti avec la méthode de Verlet.
Quelle méthode préférer sachant que les solutions doivent être périodiques ?

1. Augmenter le nombre de points n'améliorera pas vraiment le phénomène.

3 Systèmes différentiels

Dans le cadre des systèmes différentiels, on obtient une liste de vecteurs comme solution, il sera souvent plus signifiant de représenter les trajectoires, c'est-à-dire l'ensemble des points parcourus. Onperd alors la vitesse du parcours.

Exercice 12
Résoudre le système $\begin{cases} x' = (2-t)x + (t-1)y \\ y' = 2(1-t)x + (2t-1)y \end{cases}$ *avec* $x(0) = 0$ *et* $y(0) = 1$ *pour* $t \in [0, 1]$.
On comparera les solutions obtenues avec `euler` *et* `odeint` *avec une liste de temps de 10 points.*
La solution exacte est $x(t) = -e^t + e^{t^2/2}$, $y(t) = -e^t + 2e^{t^2/2}$.

Exercice 13 — Équations de Lotka-Volterra
Un système classique pour modéliser les interactions entre proies et prédateurs est $\begin{cases} x' &= x - xy \\ y' &= xy - y \end{cases}$.
Tracer les solutions de solutions du système pour différentes conditions initiales $(a, 1)$ *avec* $a > 1$.

On peut obtenir les courbes intégrales de manière interactive.

1. On se place en mode interactif : `plt.ion()`.
2. On nomme l'image que python va dessiner : `image = plt.figure()`.
3. On écrit une fonction, ici `agir`, qui indique ce qu'il faut faire quand on clique sur l'image.

```
def agir(evenement):
  # Si on clique en dehors de la zone de tracé
  if evenement.xdata == None:
    # On stoppe l'action
    image.canvas.mpl_disconnect(action)
  else:
    # On donne la position comme condition initiale
    position = [evenement.xdata, evenement.ydata]
    # On calcule la nouvelle solution
    U = odeint(proiesPred, position, T)
    # On ajoute le tracé
    plt.plot(U[ : , 0], U[ : , 1])
```

   `evenement` est le nom donné à une variable qui reçoit les indications de l'événement.
   `action` est le nom, choisi ensuite, de l'interaction.
4. On trace une trajectoire `plt.plot(x,y)`. On pourra imposer des bornes plus grandes pour pouvoir voir plus de trajectoires : `plt.xlim([0, 8])` et `plt.ylim([0, 8])`
5. On fait suivre d'une fonction système qui attend les événements et exécute la fonction en réponse à un événement donné, ici ce sera le clic de la souris : *"button_press_event"*.

```
action = image.canvas.mpl_connect("button_press_event", agir)
```

   La fonction `agir` sera exécutée avec un paramètre qui contient les informations du clic (position, double clic, ...).

Exercice 14
Essayer.

Exercice 15 — Équations de Lorenz
L'équation de Lorenz est définie par $\begin{cases} x' &= \sigma(y - x) \\ y' &= \rho x - y - xz \\ z' &= xy - \beta z \end{cases}$
avec $\sigma = 10$, $\rho = 28$ *et* $\beta = 8/3$.
Tracer la solution des équations de Lorenz pour la condition initiale $(0,1; 0; 0,1)$.
On pourra tracer une représentation en 3 dimension par

```
from mpl_toolkits.mplot3d import Axes3D

dessin = Axes3D(plt.figure()) # Un cadre contenant les tracés
dessin.plot(x, y, z) # On y ajoute un tracé
```

4 Sujets d'oraux de l'épreuve Centrale 2

Exercice 16
Soit (E) *:* $y'' + w^2 y = 0$ *pour* $w > 0$.

1. *Montrer qu'il y a une bijection entre les solution de* (E) *et celles de* (E') *:* $X' = \begin{pmatrix} 0 & 1 \\ -w^2 & 0 \end{pmatrix} X$.
2. *Montrer que les suites* (y_i) *et* (z_i) *telles que* $\begin{cases} y_{i+1} = y_i + hz_i \\ z_{i+1} = z_i - hw^2 y_i \end{cases}$ *permettent de résoudre avec la méthode d'Euler.*
3. *Programmer la méthode d'Euler pour résoudre* (E)*, puis vérifier que l'on obtient la même chose avec* `odeint`*. Indication : fixer par exemple :* $w = 3, y(0) = 1, y'(0) = 2, t \in [0, \pi]$.
4. *Programmer une deuxième méthode d'Euler avec le système* $\begin{cases} y_{i+1} = y_i + hz_i \\ z_{i+1} = z_i - hw^2 y_{i+1} \end{cases}$

Exercice 17
On pose, pour tout $t \neq 0$,

$$A(t) = \begin{pmatrix} 1 - \frac{1}{t} & 1 \\ \frac{1}{2t} & 0 \end{pmatrix} \text{ et } (S) \begin{cases} x'(t) = \left(1 - \frac{1}{t}\right) x(t) + y(t) \\ y'(t) = \frac{x(t)}{2t} \end{cases}$$

1. Résoudre numériquement pour plusieurs valeurs de $x(1) = a$ et $y(1) = b$, les tracer pour $t \in [1, 4]$. Commenter.
2. **Pour les cubes** : Pour $t \in \{0,5; 1; 1,5; 2\}$, trouver les valeurs propres et des vecteurs propres de $A(t)$. Que peut-on conjecturer ?

Exercice 18
Soit $F : x \mapsto \displaystyle\int_0^{+\infty} \frac{e^{-tx}}{1+t^2} \mathrm{d}t$ et (E) l'équation différentielle $y'' + y = \frac{1}{x}$.

1. Montrer que F est définie sur $\mathbb{R}^+$.
2. à l'aide de Python, tracer la courbe représentative de F sur l'intervalle $[0.1, 20]$. Que peut-on conjecturer pour F en $+\infty$?
3. Justifier qu'il existe une unique solution f de (E) telle que $f(1) = 1$ et $f'(1) = 1$.
4. Tracer la courbe représentative de f sur l'intervalle $[1, 20]$. Avec la méthode d'Euler que vous aurez réécrite. Cumuler sur le même graphique la solution obtenue avec `integr.odeint`.
5. Tracer la courbe représentative de $F - f$ sur l'intervalle [1,20].
6. Calculer $F(1)$ et $F'(1)$ puis tracer sur $[1, 20]$ la solution g de (E) telle que $g(1) = F(1)$ et $g'(1) = 1$. Comparer au graphe de F.
   Réponse : $F(1) = 0.62144, F'(1) = -0.34337$.

Exercice 19
On considère le système différentiel $\begin{cases} x' = -x + 2y - 1 \\ y' = 2x - y + 2 \\ z' = 3x \end{cases}$
On définit $\mathcal{C}$ l'arc paramétré $t \mapsto (u(t), v(t), w(t))$ solution de ce système différentiel. Représenter l'arc paramétré $\mathcal{C}$ pour $t \in [0, 2]$. On utilisera la fiche `Python-plot.pdf`.

Exercice 20

On considère le système différentiel $\forall t \in \mathbb{R}, X'(t) = AX(t)$ avec $A = \begin{pmatrix} 0 & 1 & 2 \\ -1 & 0 & 2 \\ -2 & -2 & 0 \end{pmatrix}, X(t) = \begin{pmatrix} x(t) \\ y(t) \\ z(t) \end{pmatrix}$.

Tracer pour $t \in [0, 10]$ avec un pas de $0,01$ les fonctions $u : t \mapsto 2x(t) - 2y(t) + z(t)$ et et $v : t \mapsto x^2(t) + y^2(t) + z^2(t)$, avec X solution du système.

Pour les cubes : Donner les éléments propres de A avec Python.

Exercice 21

1. On considère l'équation $(E) : (1-x)y'' = y$ sur $]-\infty, 1[$.
   (a) Montrer qu'il existe une unique solution de (E) sur $]-\infty, 1[$ vérifiant $f(0) = 0$ et $f'(0) = 1$.
   (b) Représenter graphiquement f sur $[-2; 0,95]$.
2. Soit $(a_n) \in \mathbb{R}^{\mathbb{N}}$, définie par : $a_0 = 0, a_1 = 1$ et, pour tout $n \in \mathbb{N} - \{0, 1\}, a_n = \frac{n-2}{n} a_{n-1} + \frac{1}{n(n-1)} a_{n-2}$.
   (a) Représenter graphiquement a_n pour $n \in [\![0, 100]\!]$.
   (b) Montrer que le rayon de convergence de $\sum a_n x^n$ est supérieur ou égal à 1.
   (c) Représenter graphiquement $s : x \mapsto \sum_{k=0}^{100} a_k x^k$ sur $[-1, 1; 0, 95]$. Que constate-t-on ? Démontrer le résultat.

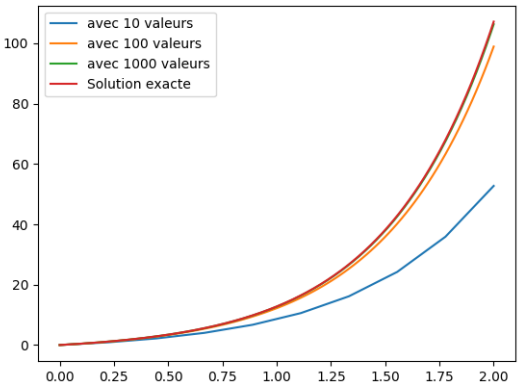


Figure III.1 – Exercice 1

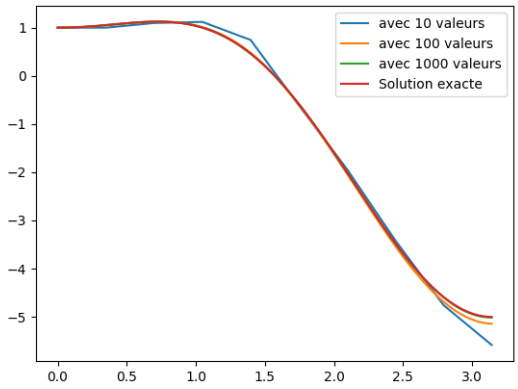


Figure III.2 – Exercice 2

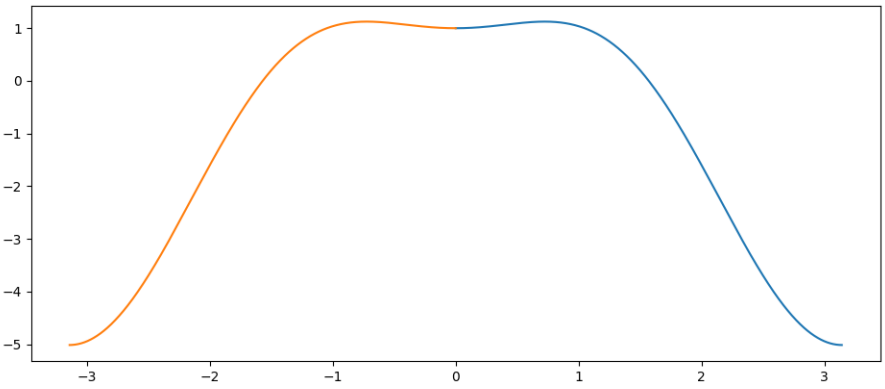


Figure III.3 – Exercice 2, intervalle symétrique

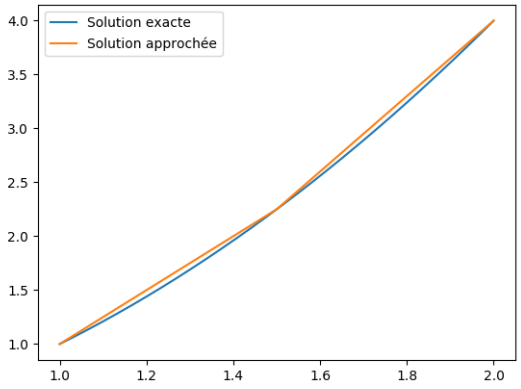


Figure III.4 – Exercice 3

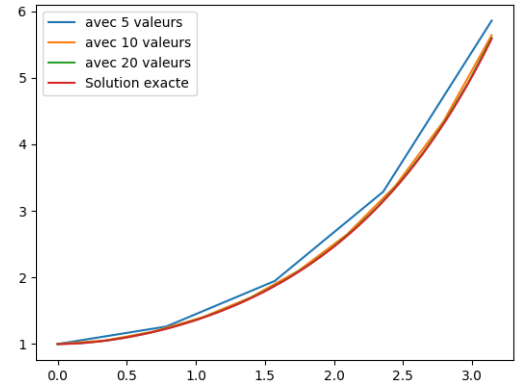


Figure III.5 – Exercice 4

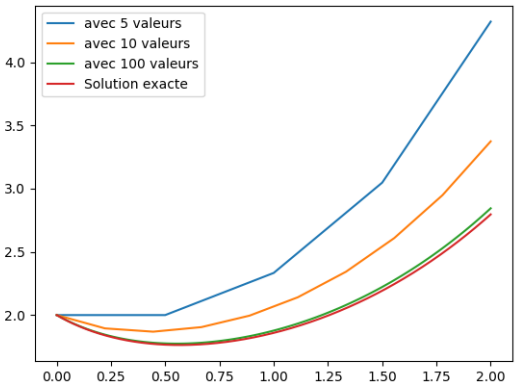


Figure III.6 – Exercice 5

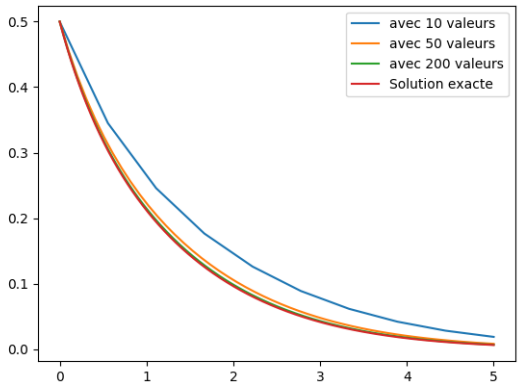


Figure III.7 – Exercice 6

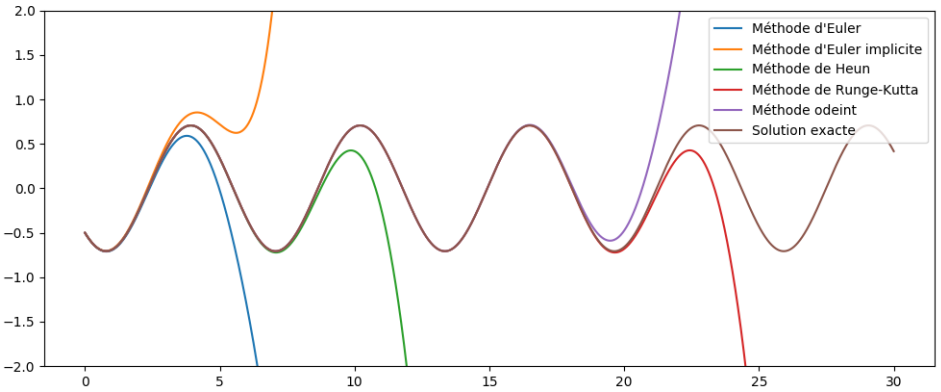


Figure III.8 – Exercice 7

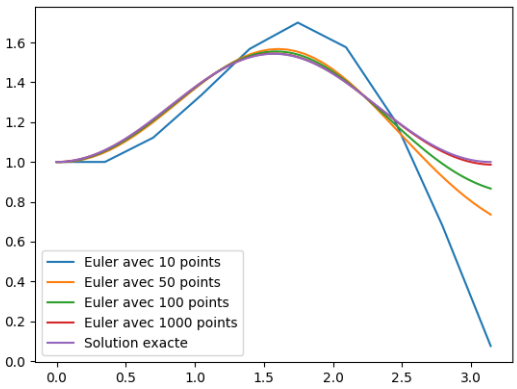


Figure III.9 – Exercice 8

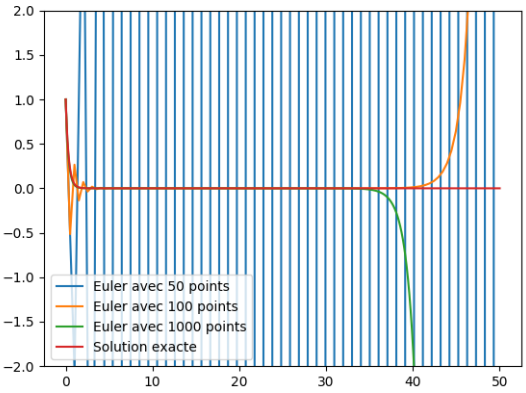


Figure III.10 – Exercice 9

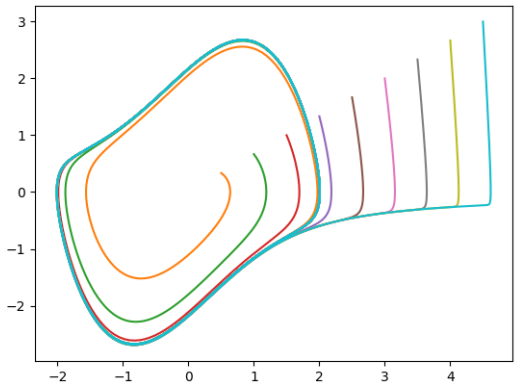


Figure III.11 – Exercice 10

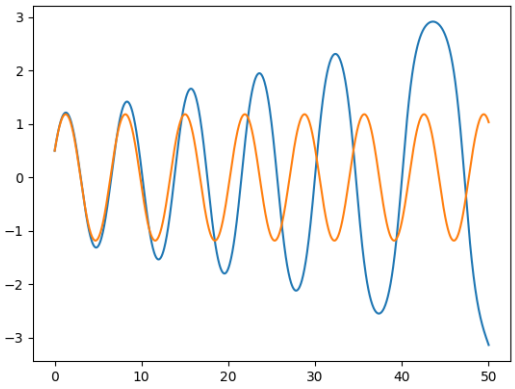


Figure III.12 – Exercice 11

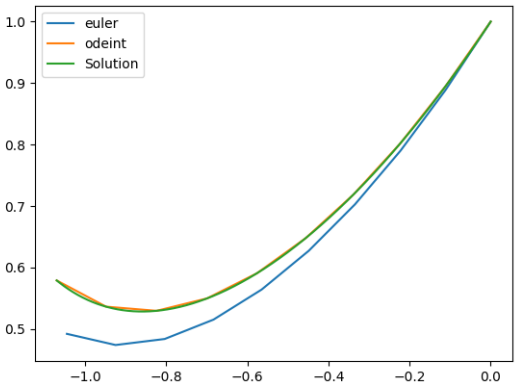


Figure III.13 – Exercice 12

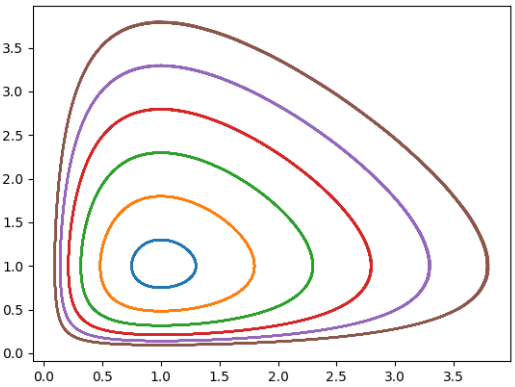


Figure III.14 – Exercice 13

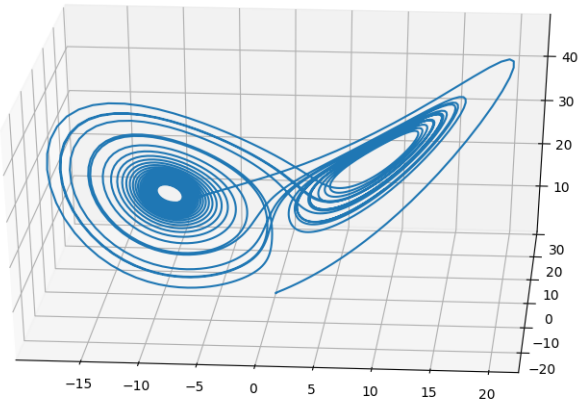


Figure III.15 – Exercice 15

5 Solutions

Solution de l'exercice 1 -

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

def euler(phi, y0, T):
    n = len(T)
    Y = [0]*n
    Y[0] = y0 # ordonnée initiale
    for i in range(n-1): # il reste n-1 valeurs à calculer
        y = Y[i]
        pas = T[i+1] - T[i]
        pente = phi(y, T[i])  # la pente est approchée par la
            dérivée
        Y[i+1] = y + pas*pente
    return Y

def phi1(y, t) :
    return 4 + 2*y

def f(x) :
    return 2*np.exp(2*x) - 2

for n in [10, 100, 1000]:
    T = np.linspace(0, 2, n)
    Y = euler(phi1, 0, T)
    plt.plot(T, Y, label='avec {} valeurs'.format(n))
T=np.linspace(0, 2, 1000)
Y = f(T) # ou Y = [f(t) for t in T]
plt.plot(T, Y, label = 'Solution exacte')
plt.legend()
plt.show()
```

Solution de l'exercice 2 - On modifie la fonction `phi`.
Pour l'intervalle symétrique il suffit de faire une liste des temps de 0 à $-\pi$.

Solution de l'exercice 3 -

```
from scipy.integrate import odeint

T=np.linspace(1, 2, 1000)
Y=[t*t for t in T]
plt.plot(T, Y, label='Solution exacte')

def phi(y, t) :
  return(1+(2*t-1)*y/(t**2))

T=np.linspace(1, 2, 3)
Ybis=odeint(phi, 1, T)

plt.plot(T, Ybis, label='Solution approchée')
plt.legend()
plt.show()
```

Solution de l'exercice 4 -

```
def heun(phi, y0, T):
    n = len(T)
    Y = [0]*n
    Y[0] = y0
    for i in range(n-1):
        y = Y[i]
        pas = T[i+1] - T[i]
        pente_g = phi(y, T[i])
        z = y + pas*pente_g
        pente_d = phi(z, T[i+1])
        pente = (pente_g  + pente_d)/2
        Y[i+1] = y + pas*pente
    return Y
```

```
def phi(y, t) :
    return (np.exp(t) - y )/(t+1)

def f(t) :
    return np.exp(t)/(t+1)

for n in [5, 10, 20]:
    T = np.linspace(0, np.pi, n)
    Y = heun(phi, 1, T)
    plt.plot(T, Y, label='avec {} valeurs'.format(n))
T = np.linspace(0, np.pi, 1000)
Y = f(T) # ou Y = [f(t) for t in T]
plt.plot(T, Y, label = 'Solution exacte')
plt.legend()
plt.show()
```

Solution de l'exercice 5 -

```
from scipy.optimize import fsolve

def eulerImplicite(phi, y0, T) :
    n = len(T)
    Y = [0]*n
    Y[0] = y0
    for k in range(n-1):
        # On féfinit une fonction dans la fonction
        def g(x):
            return x - Y[k] - (T[k+1]-T[k])*phi(x, T[k+1])
        yNew = fsolve(g,Y[k])
        Y[k+1] = yNew
    return Y

def phi(y,t) :
    return (t*y-1)/(t+1)

def f(x) :
    return (np.exp(x)+1)/(x+1)
```

```
for n in [5, 10, 100]:
    T = np.linspace(0, 2, n)
    Y = eulerImplicite(phi, 2, T)
    plt.plot(T, Y, label='avec {} valeurs'.format(n))
T=np.linspace(0, 2, 1000)
Y = f(T)
plt.plot(T, Y, label = 'Solution exacte')
plt.legend()
plt.show()
```

Solution de l'exercice 6 -

```
def RK(f, y0, T):
    n = len(T)
    Y = [0]*n
    Y[0] = y0
    for i in range(n-1):
        y = Y[i]
        t = T[i]
        t1 = T[i+1]
        pas = t1 - t
        pente1 = f(y,t)
        a = y + pente1*pas/2 # 1ère valeur au milieu
        pente2 = f(a, t+pas/2) # 1ère pente au milieu
        b = y + pente2*pas/2 # 2ème valeur au milieu
        pente3 = f(b, t+pas/2) # 2ème pente au milieu
        c = y + pente3*pas  # valeur à droite
        pente4 = f(c,t1)  # pente à droite
        pente = (pente1 + 2*pente2 + 2*pente3 + pente4)/6
        Y[i+1] = y + pas*pente
    return Y
```

Solution de l'exercice 7 -

```
def phi(y, t):
    return np.sin(t) + y

T = np.linspace(0, 30, 3000)
Y = (-np.cos(T)-np.sin(T))/2
YE = euler(phi, -0.5, T)
YI = eulerImplicite(phi, -0.5, T)
YH = heun(phi, -0.5, T)
YRK = RK(phi, -0.5, T)
YO = odeint(phi, -0.5, T)
plt.plot(T, YE, label = "Méthode d'Euler")
plt.plot(T, YI, label = "Méthode d'Euler implicite")
plt.plot(T, YH, label = "Méthode de Heun")
plt.plot(T, YRK, label = "Méthode de Runge-Kutta")
plt.plot(T, YO, label = "Méthode odeint")
plt.plot(T, Y, label = "Solution exacte")
plt.legend()
plt.ylim((-2, 2))
plt.show()
```

La solution générale est $y(t) = \frac{-1}{2}\big(\sin(t) + \cos(t)\big) + Ke^t$; un petit écart, engendré par les erreurs d'arrondi, par rapport à la solution exacte introduit une fonction exponentielle qui tend vers l'infini rapidement.
Rappel : **tous les calculs avec des flottants sont faux.**

Solution de l'exercice 8 -

```
def phi(u, t):
    y, dy = u
    return np.array([dy, -dy*np.tan(t) + y*(np.cos(t))**2])

u0 = np.array([1, 0])

for N in [10, 50, 100, 1000]:
    T = np.linspace(0, np.pi, N)
    U = euler(phi, u0, T)
    Y = [u[0] for u in U]
    plt.plot(T, Y, label = "Euler avec {} points".format(N))

T = np.linspace(0, np.pi, 1000)
Y = 0.5*np.exp(np.sin(T)) + 0.5*np.exp(-np.sin(T))
plt.plot(T, Y, label = 'Solution exacte')
plt.legend()
plt.show()
```

Solution de l'exercice 9 - La solution devrait être $y(t) = e^{-3t}$ mais les erreurs d'arrondi et d'imprécision font que la solution calculée comporte un terme en e^t.

Solution de l'exercice 10 -

```
def vdPol(u,t) :
  (x,y) = u
  xPrime = y
  yPrime = y*(1-x*x)-x
  return np.array([xPrime, yPrime])

T = np.linspace(0, 30, 1000)
for i in range(10):
  y0 = np.array([i/2, i/3])
  sol = heun(vdPol, y0, T)
  X = [u[0] for u in sol]
  X = [u[0] for u in sol]
  plt.plot(T, X)
plt.show()
```

Toutes les solutions se rapprochent d'un cycle limite.

Solution de l'exercice 11 -

```
def verlet(psi, u0, T):
  n = len(T)
  y0, dy0 = u0
  dt = T[1] - T[0]
  y1 = y0 + dt*dy0 + dt**2/2*psi(y0, T[0])
  Y = [0]*n
  Y[0] = y0
  Y[1] = y1
  for i in range(n-2): # il reste n-2 valeurs à calculer
    y_avant = Y[i]
    y = Y[i+1]
    Y[i+2] = 2*y - y_avant + dt**2*psi(y, T[i+1])
  return Y
```

```
def psi(y, t):
  return -np.sin(y)

def phi(u, t):
    y, dy = u
    return np.array([dy, -np.sin(y)])

T = np.linspace(0, 50, 1000)
u0 = np.array([0.5, 1])
U = euler(phi, u0, T)
Y = [u[0] for u in U]
plt.plot(T, Y)
V = verlet(psi, u0, T)
plt.plot(T, V)
plt.show()
```

Solution de l'exercice 12 -

```
def phi(u, t):
    x, y = u
    dx = (2-t)*x+(t-1)*y
    dy = 2*(1-t)*x+(2*t-1)*y
    return np.array([dx, dy])

u0 = np.array([0, 1])
T = np.linspace(0, 1, 10)
U = euler(phi, u0, T)
X = [u[0] for u in U]
Y = [u[1] for u in U]
plt.plot(X, Y, label = 'euler')
U = odeint(phi, u0, T)
X = [u[0] for u in U]
Y = [u[1] for u in U]
plt.plot(X, Y, label = 'odeint')
T = np.linspace(0, t_max, 1000)
X = -np.exp(T) + np.exp(T**2/2)
Y = -np.exp(T) + 2*np.exp(T**2/2)
plt.plot(X, Y, label = 'Solution')
plt.legend()
plt.show()
```

Solution de l'exercice 13 -

```
def proiesp(u,t):
  x, y = u
  dx = x - x*y
  dy = x*y - y
  return np.array([dx,dy])

T = np.linspace(0,100,1000)
for i in range(6):
  y0 = np.array([1.0, 1.3 + 0.5*i])
  U = heun(proiesp,y0,T)
  X = [u[0] for u in U]
  Y = [u[1] for u in U]
  plt.plot(X, Y)
```

Solution de l'exercice 14 -

```
plt.ion()
image = plt.figure()
T = np.linspace(0, 30, 1000)
y0 = np.array([1.0, 2.0])
U = RK(proiesp, y0, T)
X = [u[0] for u in U]
Y = [u[1] for u in U]
plt.plot(X, Y)
plt.xlim([0, 8])
plt.ylim([0, 8])


def agir(evenement):
    # Si on clique en dehors de la zone de tracé
    if evenement.xdata == None:
        # On stoppe l'action
        image.canvas.mpl_disconnect(action)
    else:
        # On donne la position comme condition initiale
        position = np.array([evenement.xdata, evenement.ydata
           ])
        # On calcule la nouvelle Uution
        U = odeint(proiesp, position, T)
        X = [u[0] for u in U]
        Y = [u[1] for u in U]
        # On ajoute le tracé
        plt.plot(X, Y)
action = image.canvas.mpl_connect("button_press_event", agir)
```

Solution de l'exercice 15 -

```
def lorenz(u, t):
    sigma = 10
    rho = 28
    beta = 8/3
    x, y, z = u
    dx = sigma*(y - x)
    dy = rho*x - y - x*z
    dz = x*y - beta*z
    return np.array([dx, dy, dz])

T = np.linspace(0, 30, 3000)
y0 = np.array([0.1, 0.0, 0.1])
U = odeint(lorenz, y0, T)
X = [u[0] for u in U]
Y = [u[1] for u in U]
Z = [u[2] for u in U]
dessin = Axes3D(plt.figure())
dessin.plot(X, Y, Z)
plt.show()
```

TP IV

SQL : RÉVISIONS

Nous allons utiliser une base de données qui contient de nombreuses tables.
Certaines sont des tables qui contiennent des descriptions, par exemple ***Pays****.*
D'autres représentent des associations entre des données, par exemple ***Frontiere****.*

Voici les tables tables utilisées et leurs attributs. La clé primaire est soulignée.

- **Appartenance** *: Pays, Continent, Pourcentage*
- **Continent** *: Nom, Superficie*
- **Frontiere** *: Pays1, Pays2, Longueur*
- **Ile** *: Nom, Iles, Superficie, Altitude, Type, Longitude, Latitude*
- **IleDans** *: Ile, Mer, Lac, Riviere*
- **IlePays** *: Ile, Pays, Province*
- **Langage** *: Pays, Nom, Percentage*
- **MembreOrganisation** *: Organisation, Pays, Type*
- **Montagne** *: Nom, Chaine, Altitude, Type, Longitude, Latitude*
- **MontagnePays** *: Montagne, Pays, Province.*
- **Organisation** *: Abbreviation, Nom, Ville, ...*
- **Pays** *: Nom, Code, Capitale, Province, Superficie, Population*
- **Politique** *: Pays, DateIndependance, AncienneDependance, Dependance, ...*
- **Riviere** *: Nom, Riviere, Lac, Mer, ...*
- **RivierePays** *: Riviere, Pays, Province*

Dans les tables d'associations de nom du pays est le code de la table **Pays***.*

1 Requêtes sur une seule table

Exercice 1
Déterminer la liste des continents et de leur superficie, par ordre croissant.
Continent

Exercice 2
Quels sont les pays de plus de 10^8 habitants ? Il y en a 10.
Pays

Exercice 3
Quels sont les volcans (`type = "volcano"`) de plus de 6000 m ? Il y en a 7.
Montagne

Exercice 4
Quels sont les affluents du Rhin (**Rhein**) ? Il y en a 4.

Riviere
La rivière se jette dans une autre rivière (elle est un affluent), dans un lac ou dans une mer. Elle peut aussi passer dans un lac avant d'atteindre une autre rivière ou une mer.

2 Fonctions d'agrégation

Exercice 5
Quelle est la longueur de la frontière (terrestre) de la France (code "F") ? (2892.4 km)
Frontiere

Exercice 6
Quelles sont les îles faisant partie des Caraïbes (`iles = "Caraibes"`) avec leur superficie ?
Combien y en a-t-il ? `count` (22)
Quelle est la superficie totale ? (12165 km^2)
Quelle est la moyenne des points culminants ? (825 m)
Ile

Exercice 7
Quelles sont les mers dans lesquelles se jettent plus 5 rivières ou plus (`having`) ? Il y en a 5.
On excluera les rivières qui ne sont pas des fleuves par la condition : `mer is not null`.
Riviere

Exercice 8
Quelles sont les villes qui sont le siège de plus de 5 organisations ? Il y en a 6.
Organisation

Exercice 9
Quelles sont les langues parlées dans 5 pays au moins par plus de 25 % de la population ?
Il y en a 4.
Langage

3 Jointures

Exercice 10
Quels sont les pays membres de l'UNESCO ? Il y en a 192 (188 vrais membres).
MembreOrganisation, **Pays**

Exercice 11
Quels sont les pays qui ont été des colonies de la France (`AncienneDependance = "F"`) ? (24)
Politique, **Pays**

Exercice 12
Quels sont les pays qui dans lesquels l'anglais est la langue pour plus de 10 % de la population ?
Langage, **Pays**

Exercice 13
Quels sont les pays qui sont traversés par le Danube (**Donau**) ? Il y en a 10.
Pour éviter les répétitions on utilisera `select distinct`.
RiverePays, **Pays**

Exercice 14
Quelles sont les montagnes d'Amérique de plus de 6000 m avec le nom du pays dans lequel elles sont situées et leur altitude ? Il y en a 16.
MontagnePays, **Appartenance**, **Pays**, **Montagne**

Exercice 15
Quels sont les affluents des affluents du Rhin ? Il y en a 3.
Riviere 2 fois

Exercice 16
Quels sont les pays dont la plus grande partie est en Europe ? Il y en a 51.
Pays, **Appartenance**

Exercice 17
Quels sont les organisations qui rassemblent des pays avec un total de plus de 5.10^9 habitants ? Il y en a 27.
Organisation, **Pays**, **MembreOrganisation**

Exercice 18
Dans quels pays existe-t-il une île dans un Lac ? Il y en a 4 (2 îles au Canada).
IleDans, **IlePays**, **Pays**

Exercice 19
Quels sont les pays qui ont une montagne dans la chaîne des Alpes (**"Alps"**) ? Il y en a 5.
Pays, **MontagnePays**, **Montagne**

Exercice 20
Quels sont les fleuves qui passent en France ?
Riviere, **RivierePays**, **Pays**

4 Sous-requêtes et combinaisons

Exercice 21
Quels sont les pays qui font partie de l'ONU (**UN**) mais pas de l'UNESCO ? Il y en a 6.
Pays, **MembreOrganisation**

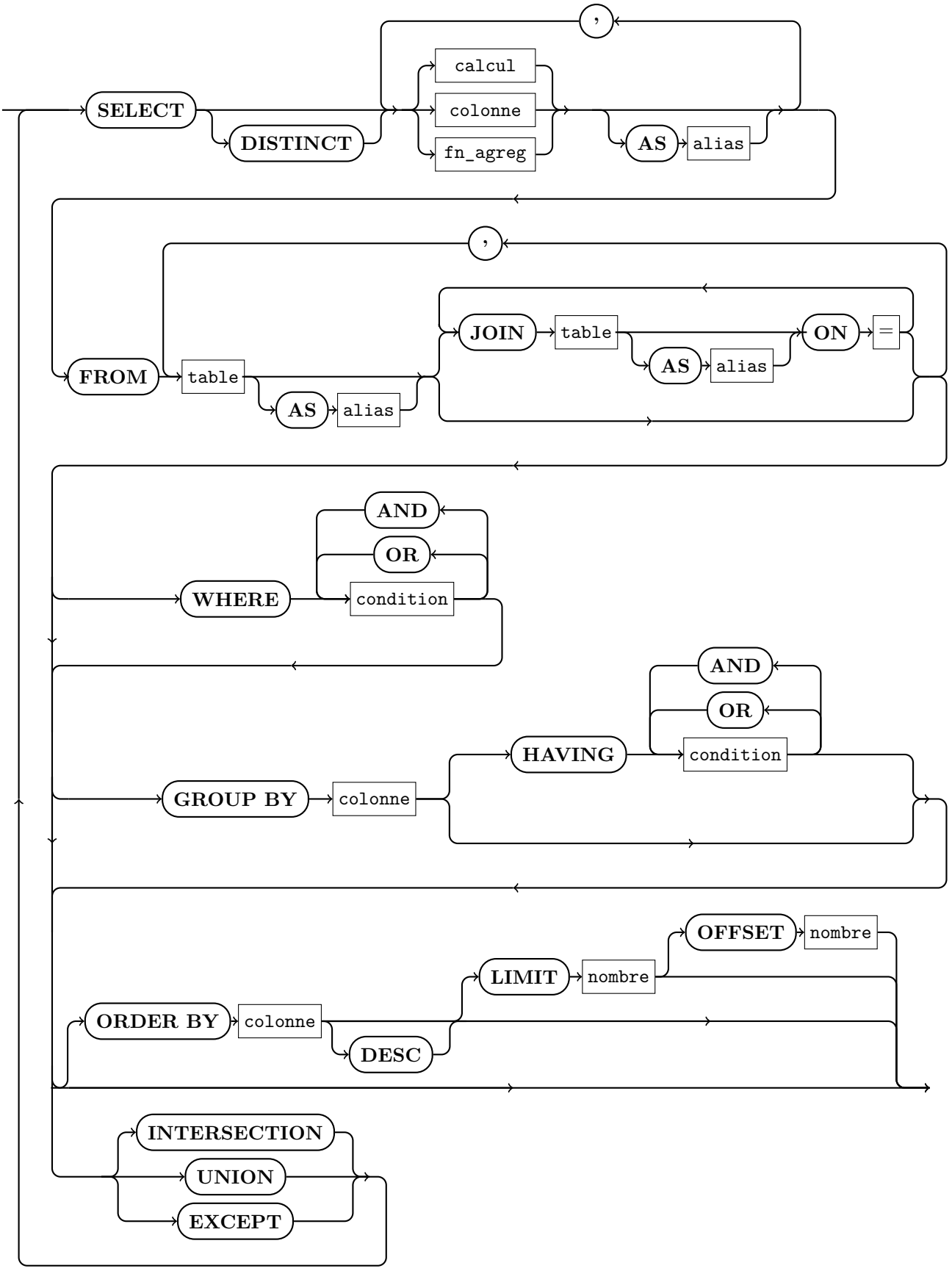
Exercice 22
Quels sont les pays qui ont une frontière avec la Russie (**Russia**, code **R**) ? Il y en a 14.
Frontiere, **Pays**

Exercice 23
Combien ont d'affluents les fleuves qui passent en France ? (Seuls 3 ont des affluents dans la base)
Riviere deux fois, **RivierePays**, **Pays**
La réponse n'est pas que la Seine a quatre affluents dans la base.

Exercice 24
Quels sont les langues parlées par plus de 100.000.000 locuteurs ? (7)
Pays, **Langage**

Exercice 25
Quels sont les pays qui partagent une même montagne ? Il y en a 69.
Pays, **MontagnePays**, **Montagne**

Exercice 26
Quels sont les pays d'Europe qui ont une montagne dans la base ? On donnera aussi le nom et l'altitude de la plus haute montagne. Il y en a 20.
Pays, **MontagnePays**, **Montagne**, **Appartenance**


SELECT
DISTINCT
,
calcul
colonne
fn_agreg
AS
alias
FROM
,
table
AS
alias
JOIN
table
AS
alias
ON
=
WHERE
AND
OR
condition
GROUP BY
colonne
HAVING
AND
OR
condition
ORDER BY
colonne
DESC
LIMIT
nombre
OFFSET
nombre
INTERSECTION
UNION
EXCEPT

5 Complément : championnat de France

Nous allons travailler avec une base de données qui recense les renseignements sur les matchs de division 1 d'une année. La base ne contient qu'une table, nommée `statFoot`.

En voici quelques lignes, la saison est 2018-2019

jour	equDom	equExt	butsDom	butsExt
10/08/2018	Marseille	Toulouse	4	0
11/08/2018	Angers	Nimes	3	4
11/08/2018	Lille	Rennes	3	1
11/08/2018	Montpellier	Dijon	1	2
11/08/2018	Nantes	Monaco	1	3

Il y a 20 équipes ; chaque équipe reçoit toutes les autres une fois et une seule.

- `jour` est la date du match, sous forme d'une chaîne de caractères.
- Les autres attributs vont par paire : le suffixe `Dom` indique que l'attribut concerne l'équipe qui joue à domicile, le suffixe `Ext` indique que l'attribut concerne l'équipe qui joue à l'extérieur.
- `equDom` ou `equExt` indique le nom de l'équipe, sous forme d'une chaîne de caractères.
- `butsDom` ou `butsExt` indique le nombre de buts marqués pendant le match.

Exercice 27
Écrire une requête qui permet de connaître le nombre de matchs gagnés par chaque équipe, le nombre de matchs nuls et le total des points de l'année.
Un match nul vaut 1 point, un match gagné vaut 3 points.
N.B. Bien que la base ne contienne qu'une table, la requête fait intervenir plusieurs jointures.

6 Solutions

Solution de l'exercice 1 -

```
select *
from continent
order by Superficie
```

Solution de l'exercice 2 -

```
select nom, population
from Pays
where population > 100000000
order by 2 desc
```

Solution de l'exercice 3 -

```
select nom, altitude
from Montagne
where type = 'volcano'  AND altitude > 6000
```

Solution de l'exercice 4 -

```
select nom
from Riviere
where riviere = "Rhein"
```

Solution de l'exercice 5 -

```
select sum(longueur)
from Frontiere
where Pays1 = "F" or Pays2 = "F"
```

Solution de l'exercice 6 -

```
select nom,superficie
from Ile
where Iles = "Caraibes"
order by 1

select count() as nombre, sum(Superficie) as surface, avg(
   Altitude) as altitude
from ile
where iles = "Caraibes"
```

Solution de l'exercice 7 -

```
select count() as nombre, mer
from Riviere
where mer is not null
group by mer
having nombre >4
```

Solution de l'exercice 8 -

```
select Ville, count() as nbSieges
from Organisation
where Ville is not Null
group by Ville
having nbSieges >= 5
```

Solution de l'exercice 9 -

```
select nom, count() as nb
from langage
where percentage >25
group by nom
having nb >4
```

Solution de l'exercice 10 -

```
select p.nom
from Pays as p join MembreOrganisation as mo on p.Code = mo.
   Pays
where mo.Organisation = "UNESCO"
```

Solution de l'exercice 11 -

```
select p.nom
from Pays as p join Politique as pol on p.code = pol.Pays
where pol.AncienneDependance = "F"
```

Solution de l'exercice 12 -

```
select p.nom
from langage as l join pays as p on p.code = l.pays
where l.percentage > 10 and l.nom = "English"
```

Solution de l'exercice 13 -

```
select distinct p.nom
from Pays as p join RivierePays as r on p.code = r.Pays
where r.Riviere = "Donau"
```

Solution de l'exercice 14 -

```
select distinct m.Nom, p.Nom, m.Altitude
from Appartenance as a join MontagnePays as mp on a.Pays = mp.
   Pays
                       join Montagne as m on m.Nom = mp.
                          Montagne
                       join Pays as p on p.Code = a.Pays
where m.Altitude > 6000 and a.Continent = "America"
```

Solution de l'exercice 15 -

```
select r.nom
from Riviere as r join (select nom, riviere from Riviere) as
   aff
                  on r.riviere = aff.nom
where aff.riviere = "Rhein"
```

On peut aussi composer les requêtes avec **in**.

```
select nom
from Riviere
where riviere in (select nom
                  from Riviere
                  where riviere = "Rhein")
```

Solution de l'exercice 16 -

```
select p.Nom
from Pays as p join Appartenance as a on a.Pays = p.code
where a.Continent = "Europe" and a.Pourcentage > 50
```

Solution de l'exercice 17 -

```
select o.Nom, sum(p.Population) as Pop
from Organisation as o join MembreOrganisation
                            as mo on o.Abbreviation=mo.
                               Organisation
                       join Pays as p on p.Code = mo.Pays
group by mo.Organisation
having pop > 5000000000
```

Solution de l'exercice 18 -

```
select p.Nom, i.Ile as Ile, i.Lac
from Pays as p Join IlePays as ip on p.code=ip.Pays
               Join IleDans as i on ip.Ile=i.Ile
where i.Lac is not Null
```

Solution de l'exercice 19 -

```
select distinct p.Nom
from Pays as p join MontagnePays as mp on p.Code = mp.Pays
               join Montagne as m on m.Nom = mp.Montagne
where m.Chaine = "Alps"
```

Solution de l'exercice 20 -

```
select distinct r.nom
from Riviere as r join RivierePays as rp on r.nom = rp.riviere
                                 join Pays as p on rp.pays = p.
                                    code
where p.nom ="France" and r.mer is not null
```

Solution de l'exercice 21 -

```
select p.Nom
from Pays as p join MembreOrganisation as mo on p.Code = mo.
   Pays
where mo.Organisation = "UN" AND type = "member"

except

select p.Nom
from Pays as p join MembreOrganisation as mo on p.Code = mo.
   Pays
where mo.Organisation = "UNESCO" AND type = "member"
```

Solution de l'exercice 22 -

```
select p.Nom
from Pays as p join Frontiere as f on p.Code = f.Pays1
where f.Pays2 = "R"

union

select p.Nom
from Pays as p join Frontiere as f on p.Code = f.Pays2
where f.Pays1 = "R"
```

Solution de l'exercice 23 - La base `RivierePays` indique les rivière plusieurs fois, pour chaque région parcourue. Il faut donc ne prendre que les fleuves distincts.

```
select f.fleuve, count()
from (select distinct r.nom as fleuve
          from Riviere as r join RivierePays as rp
                                  on r.nom = rp.riviere
                              join Pays as p
                                  on rp.pays = p.code
          where p.nom ="France" and r.mer is not null) as f
          join Riviere as r1 on f.fleuve = r1. riviere
group by f.fleuve
```

Solution de l'exercice 24 -

```
select langue, sum(locuteurs) as total
from (
  select p.nom, l.nom as langue, p.population*l.percentage/100
      as locuteurs
  from langage as l join pays as p on p.code = l.pays)
  group by langue
  having total > 100000000
  order by total desc
```

Solution de l'exercice 25 -

```
select distinct mm.Nom as NomMontagne, p.Nom as NomPays
from (select Montagne as Nom, count() as nb
      from (select distinct Montagne,Pays
            from MontagnePays)
            group by Nom) as mm
           join MontagnePays as mp on  mp.Montagne = mm.Nom
           join Pays as p on p.Code = mp.Pays
where mm.nb > 1
order by 1
```

Solution de l'exercice 26 -

```
select p.Nom, mp.Montagne, max(m.Altitude)
from Pays as p join Appartenance as a on a.Pays = p.code
               join MontagnePays as mp on p.Code = mp.Pays
               join Montagne as m on m.Nom = mp.Montagne
where a.Continent = "Europe" and a.Pourcentage > 50
group by mp.Pays
```

Solution de l'exercice 27 -

```
select d.club, (d.victoiresDom + e.victoiresExt) as victoires,
       n.nuls,
       (38 - n. nuls - d.victoiresDom - e.victoiresExt) as Dé
          faites,
       (3*(d.victoiresDom + e.victoiresExt) + n.nuls) as
          points
from (select equDom as club, count() as victoiresDom
      from stat
      where butsDom > butsExt group by equDom) as d
     join
     (select equExt as club, count() as victoiresExt
      from stat
      where butsDom < butsExt group by equExt) as e
     join
     (select equDom as club, count() as nuls
      from stat
      where butsDom = butsExt group by equDom) as n
     on e.club = d.club and e.club = n.club
order by points desc
```

TP V

Simulation : probabilités

Ce T.P. va faire exécuter des simulations de probabilité.
Un générateur pseudo-aléatoire est défini par python et il sera utilisé pour approcher le comportement de variables aléatoires sous la forme de fonctions qui renvoient un résultat non constant.
Presque paradoxalement, le générateur utilisé simule une probabilité continue alors que nous allons l'employer pour créer des variable aléatoires discrètes.
La fonction `random` *du module* `random` *renvoie un flottant aléatoire dans l'intervalle* $[0;1[$ *: pour* $0 \leqslant a < b \leqslant 1$*, la probabilité que la valeur renvoyée par* `random()` *appartienne à* $[a;b]$ *est* $b-a$*.*

1 Mise en place d'outils généraux

Dans cette partie nous allons définir des fonctions qui existent dans le module `random`

Exercice 1
Implémenter une fonction `randint(a, b)` qui, prenant en paramètres deux entiers a et b, retourne un entier pseudo-aléatoire de appartenant à $\{a, a+1, \ldots, b\}$.

Exercice 2
Implémenter une fonction `choice(liste)`, qui prenant en paramètre une liste non vide, retourne un élément aléatoire de cette liste.

Exercice 3
Par un méthode de votre choix, proposer une fonction `sous_liste(liste)` qui, prenant en paramètre une liste, retourne une sous-liste aléatoire de celle-ci.

Exercice 4
Par un méthode de votre choix, proposer une fonction `shuffle(liste)` qui, prenant en paramètre une liste `liste`, retourne une liste composée des mêmes éléments mais mélangée aléatoirement. On veillera à ne pas modifier la liste initiale.

Exercice 5
Écrire une fonction `sample(liste, k)`, qui prenant en paramètre une liste non vide et un entier k, retourne une liste aléatoire de k éléments **distincts** issus de la liste. Les éléments renvoyés doivent être dans le même ordre que dans la liste initiale.

2 Lois usuelles

Exercice 6
Implémenter une fonction `bernoulli(p)` qui, prenant en paramètre un flottant p de l'intervalle $[0, 1]$, retourne un booléen suivant une loi de Bernoulli de paramètre p.

Exercice 7
En déduire une fonction `binomial(n, p)` simulant une variable aléatoire suivant une loi binomiale de paramètres (n, p).

Exercice 8
Proposer une fonction `geometric(p)` simulant une variable aléatoire suivant une loi géométrique, qui rappelons-le donne le rang du premier succès dans une suite d'expérience de Bernoulli indépendantes et de même paramètre p.

Exercice 9
Écrire une fonction `VA_finie(X, P)` simulant une variable aléatoire prenant les valeurs `X[i]` avec les probabilités `P[i]`. `X` et `P` doivent être des listes de même taille et la somme des `P[i]` doit valoir 1.

Exercice 10
Écrire une fonction `poisson(lam)` simulant une variable aléatoire suivant une loi de Poisson de paramètre `lam`.

3 Simulation d'expériences

Dans les exercices suivants on calcule une valeur approchée de l'espérance d'une variable aléatoire discrète en calculant la moyenne d'une répétition de la loi.

Exercice 11
Un enfant souhaite compléter son album Panini préféré : "Les grand mathématiciens de l'histoire". Pour cela il achète des paquets contenant 5 vignettes toutes différentes. Dans l'album, 100 cases sont à compléter par une vignette.
À l'aide d'une simulation, estimez le nombre moyens de paquets à acheter pour compléter l'album.

Exercice 12
On mélange la liste $[1, 2, \ldots, 20]$ de manière aléatoire. Estimer la probabilté qu'au plus deux éléments se retrouvent placés juste devant un élément strictement plus petit.

Exercice 13
Un rat de laboratoire est placé dans un cage dont il souhaite sortir. La cage dispose de 10 sorties : la première ramène le rat dans la cage au bout de 10 secondes, la seconde fait de même en 20 secondes, ... la neuvième le ramène au bout de 90 secondes et la dixième est la sortie. On souhaite évaluer le temps moyen de sortie dans chacune des situations suivantes :

- Le rat choisit une sortie au hasard sans se souvenir de celles déjà visitées.
- Le rat ne se souvient que de la dernière sortie qu'il a visitée.
- Le rat se souvient de chaque sortie déjà visitée.

Exercice 14
Dans une urne, on place au départ 2 boules blanches et 3 boules noires. Vingt fois de suite on tire au hasard une boule, et on la remet dans l'urne avec deux nouvelles boules de la même couleur. On tire enfin une vingt et unième boule qui nous donne le résultat de l'expérience. Estimez la probabilité que cette dernière boule soit blanche.

Exercice 15
On choisit aléatoirement deux sous-ensembles d'un ensemble à n éléments. Estimer la probabilité que ces deux sous-ensembles aient un nombre impair d'éléments communs.

Exercice 16
On réalise l'expérience suivante : on lance une pièce équilibrée jusqu'à obtenir une séquence attendue (par exemple pile-face, notée (P,F)). On répète ensuite cette expérience de nombreuses fois pour estimer le temps moyen d'attente avant l'apparition de la séquence souhaitée.

1. Par une simulation, estimer le temps d'attente moyen avant obtention de deux piles consécutifs (P,P).
2. On se donne maintenant une séquence de pile-face sous forme d'une liste de booléens, les piles étant modélisés par la valeur `True`. Implémenter une fonction qui, prenant en paramètre une telle liste, estime le temps moyen d'apparition de la séquence associée. Tester avec `seq=[True,True,False,False]`.

Exercice 17
Soit N un entier naturel non nul. On effectue une marche aléatoire dans un espace de dimension N : on part de l'origine et, à chaque instant, on choisit l'une des N coordonnées. On fait alors sur cette coordonnées soit un pas dans le sens positif (+1) soit dans le sens négatif (-1).
On souhaite estimer la probabilité qu'au bout de 20 instants nous soyons revenus à l'origine. Proposer un simulation de cette expérience et comparer les résultats obtenus pour les premières valeurs de N.

Exercice 18
Dans une grande banque, sept files d'attentes sont à la disposition des clients. Le temps de traitement de la demande de chaque client est estimé à 10 minutes, et nous partons de l'idée qu'un nouveau client entre toute les minutes. Au départ, il n'y a aucun client dans la banque.
La probabilité qu'un client choisisse une file d'attente donnée est proportionnelle à $\frac{1}{0.5+n}$ où n désigne le nombre de clients en attente dans cette file. Estimer, à l'aide d'une simulation, le nombre moyen de clients présents dans la banque au bout d'une heure.

Exercice 19
On choisit au hasard deux nombres entre 1 et 1000. Estimer la probabilité que ces deux nombres soient premiers entre eux, c'est à dire que leur PGCD vaille 1.

4 Une étude statistique

On s'intéresse dans cette partie à la moyenne moyenne des résultats obtenus quand on lance les dés de nombreuses fois. Plus précisément on souhaite tirer de l'expérience un intervalle dans lequel on peut placer avec confiance l'espérance du tirage.
Pour effectuer le plus rapidement possible le calcul d'une suite de tirage, on pourra utiliser les fonctions `numpy` :

```
import numpy as np
from numpy.random import randint as tirage

def serie(n):
    np.average(tirage(1, 7, size = 100))
```

4.1 Premiers résultats

Exercice 20
Écrire une fonction `series(N, P)` qui renvoie un tableau de taille N des moyennes de N séries de P lancers.

Exercice 21
Pour les moyennes de de 10^5 séries de 10^2 lancers.

1. Quelle est la moyenne des moyennes ?
2. Quelle est la moyenne maximale (resp. minimale) qui a été atteinte ?
3. Combien de séries ont fourni une moyenne majorée par 3.4 (resp. minoré par 3.6) ?
4. Même chose en remplaçant 3.4 et 3.6 par 3 et 4.

4.2 Loi (faible) des grands nombres

Si on note Y_n la variable aléatoire égale à la moyenne après n lancers aléatoires, alors l'espérance de Y_n vaut $\frac{7}{2}$, et on a même d'après la loi (faible) des grands nombres :

$$\forall \varepsilon > 0 \quad P\left(\left|Y_n - \frac{7}{2}\right| > \varepsilon\right) \xrightarrow[n\to+\infty]{} 0$$

Exercice 22
Écrire une fonction `np_series(N, P)` prenant en entrée deux entiers strictement positifs N et P, réalisant une série de $N.P$ tirages et retournant la liste constituée des N moyennes correspondant aux kP premiers tirages pour k variant de 1 à N.

Exercice 23
Visualiser le résultat : on pourra afficher le résultats de 10 appels de `np_series(1000, 1000)`.

4.3 Théorème central limite

On fixe maintenant à 100 le nombre de tirages et on considère la variable aléatoire Y, égale à la moyenne de ces 100 tirages. C'est une variable aléatoire finie à valeurs dans $\left\{\frac{k}{100}\ ;\ 100 \leqslant k \leqslant 600\right\}$.
Par linéarité Y a une espérance de $\mu = \frac{7}{2}$ et un écart-type de de $\sigma = \sqrt{\frac{35}{12}}\frac{1}{\sqrt{100}} = \sqrt{\frac{35}{120}}$.
Comme $n = 100$ est proche de l'infini, le théorème central limite dit que cette loi de Y peut être approchée par une loi normale centrée $\mathcal{N}(e, \sigma^2)$:

$$P\big(Y = a\big) \simeq \frac{1}{2\sqrt{\pi}} \int_{a-\frac{1}{200}}^{a+\frac{1}{200}} e^{-(t-\mu)^2/(2\sigma^2)} \frac{\mathrm{d}t}{\sigma} \simeq \frac{e^{-(a-\mu)^2/(2\sigma^2)}}{100.2.\sqrt{\pi}.\sigma}$$

Exercice 24
Comparer la distribution de probabilité pour N valeurs de Y avec la loi normale. On choisira $N = 10^3$, $N = 10^4$, $N = 10^5$, $N = 10^6$.

Prolongements

- Les deux théorèmes qu'on vient de voir s'appliquent dès qu'on connaît l'espérance et l'écart-type d'une variable aléatoire. On peut les appliquer à la répétition d'autres variables aléatoires.
- On peut aussi déterminer la vraie distribution de probabilité de la moyenne après 100 tirages.

%chapitreSolMecaQ