Une introduction à Python 3

Hubert Raymondaud - Yannick Le Bastard

20 mars 2019

1 Exercices d'initiation

Exercice 0 :

1. type(56+9) type int
2. type(3*4) type int
3. type(3/4) type float
4. type(3.*4) type float
5. type('5') type str
6. type('5'+2) message d'erreur
7. type('ABC'*2) type str : 'ABC'*2 donne 'ABCABC'
8. type('5'+str(2)) type str :'5'+str(2) donne '52'
9. type([11,'Hello',56.3]) type list

Exercice 1 : Pour bien comprendre l'affectation parallèle !
Question 1 : a prend pour valeur 7 et b prend pour valeur 3.
Question 2 : a prend pour valeur 7 et b prend pour valeur 1.

Exercice 2 : Voici un script :

```
from math import sqrt
AB=float(input("AB= ? "))
AC=float(input("AC= ? "))
print("BC=",sqrt(AB**2+AC**2))
```

Exercice 3 : Voici un script :

```
N=int(input("N= ? "))
for i in range(11):
    print(i,"fois",N,"egale",N*i)
```

Exercice 4 : Nous obtenons :

1. $[2, 5, 8, 11]$ et pas 14, exclus.
2. $[14, 11, 8, 5]$ et pas 2, exclus.
3. nous obtenons la liste vide.
4. $[-2, -5, -8, -11]$ et pas -14, exclus.
5. $[-2, 1, 4, 7, 10]$ et pas 13, exclus.

Exercice 5 : Une solution...

```
#question 1
S,P=1,1
for i in range(2,21,2):
    P=P*i
    S=S+1/P
print(S) #on obtient 1.6487212706873655

#question 2
N=int(input("Nombre de barres de fractions ? "))
S=1
for i in range(N):  #ou range(1,N+1)
    S=1+1/S
print(S) #avec N=5 on obtient 1.625

#question 3
S=0
for i in range(1,10): #ou range(1,11) ! pourquoi ?
    for j in range(i+1,11): #j>i
        S=S+i*j
print(S)  #on obtient 1320
```

Nous devons résoudre l'équation : $\begin{pmatrix} 2 & 0 & 0 & 0 \\ -1 & 2 & 0 & 0 \\ 0 & -1 & 2 & 0 \\ 0 & 0 & -1 & 2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$ qui se traduit par le système :

$$\begin{cases} 2x_1 & = 1 \\ -x_1 + 2x_2 & = 1 \\ -x_2 + 2x_3 & = 1 \\ -x_3 + 2x_4 & = 1 \end{cases} \Leftrightarrow -x_i + 2x_{i+1} = 1 \ (0 \leq i \leq 3) \text{ en posant} x_0 = 0$$

On en déduit que $x_{i+1} = (1+x_i)/2$, ce qui nous permet de calculer pas à pas les x_i. La solution est (x_1, x_2, x_3, x_4). Nous renvoyons le lecteur au paragraphe sur les listes pour ce qui suit.

```
#question 4
sol=[0,0,0,0,0]
for i in range(0,4):
    sol[i+1]=(1+sol[i])/2
print("Solution :",sol[1:]) #on a Solution : [0.5, 0.75, 0.875, 0.9375]
```

Un petit exercice du même ordre est de programmer un script dans le cas où la matrice est triangulaire supérieure avec des -1 au-dessus de la diagonale.

Exercice 6 : un bon galop d'essai !

```
#question 1
N=int(input("N= ? "))
S,P=1,1
for i in range(N):
    P=P*2**(i+1)
    S=S+1/P
print(S) #avec N=5 on obtient 1.641632080078125

#question 2
```

```
S=0
for i in range(1,4):
    P=1 #Attention a la place de l'initialisation de P
    for k in range(1,i+1):
        P=P*k*(k+1)
    S=S+1/P
print(S) #on obtient 0.5902777777777778

#question 3
u,v=1,1 #initialisation de la suite de Fibonacci
S=u+v
N=int(input("Nombre de termes a sommer ? "))
for i in range(N-2): #Attention au decalage !
    u,v=v,u+v #Bien comprendre ceci
    S=S+v
print(S) #avec N=7 on trouve 33
```

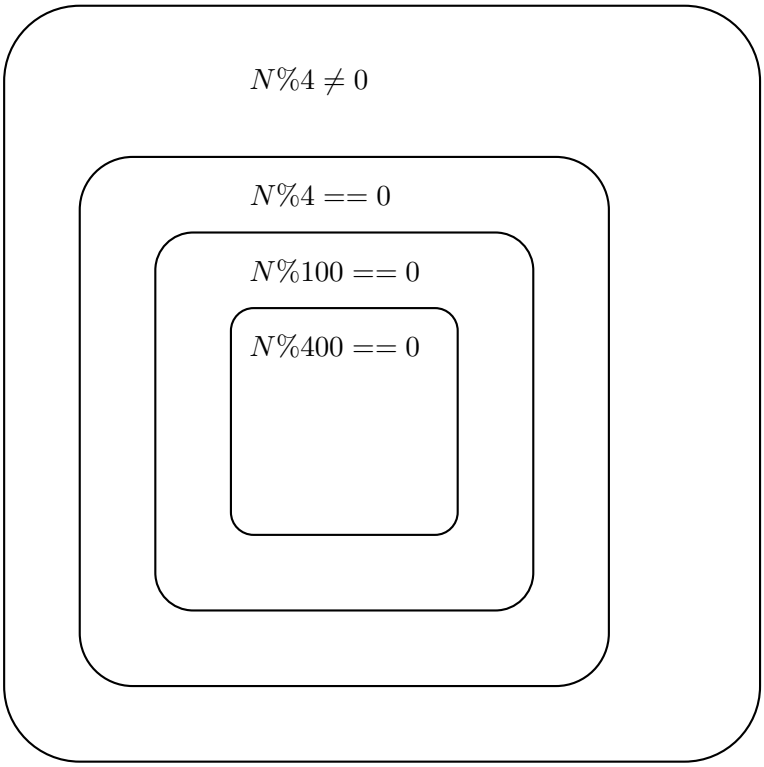
Exercice 7 : Soyez for ! while not ?

```
#question 1
#solution 1 (avec for) : il suffit de remarquer que 3*(2**16) = 196608
S=3
for i in range(1,17):
    S=S+3*2**i
print(S) #on trouve 393213

#solution 2 (avec while)
S=3
i=1
while 3*2**i<=196608: #le dernier terme donne la condition
    S=S+3*2**i
    i=i+1
print(S) #meme resultat : 393213

#question 2
S=1
N=2
while S<5:
    S=S+1/N
    N=N+1
print(N-1) #N-1 et pas N ; on trouve N=83
```

Exercice 8 : Test de saisie.

```
ch=""
while len(ch)!=10:
    ch=input("Saisir exactement 10 caracteres (espaces compris) ")
    if len(ch)==10:
        print("Saisie correcte")
        break
```

Exercice 9 : Dessinons la situation à l'aide d'un diagramme de Venn :



Ainsi, l'année N est bissextile si et seulement si :
$N\%4 == 0$ or ($N\%100! = 0$ and $N\%400 == 0$) : "le noyau et une couronne". Dès lors le script devient simple :

```
N=int(input("Entrez une annee N : "))
if (N%4==0 and N%100!=0) or N%400==0:
    print("Annee bissextile")
else:
    print("Annee non bissextile")
```

Exercice 10 : Le jeu du devin.

```
#question 1
from random import randint
alea=randint(1,100)
N=int(input("Choisir un entier entre 1 et 100 : "))
compteur=1
while N!=alea:
    if N>alea:
        print("Plus petit")
    else:
        print("Plus grand")
    N=int(input())
    compteur=compteur+1
if N==alea:
    print("Gagne en",compteur,"essais")

```

```
#question 2
from random import randint
alea=randint(1,100)
N=int(input("Choisir un entier entre 1 et 100 : "))
compteur=1
while N!=alea and compteur<5:
    if N>alea:
        print("Plus petit")
    else:
        print("Plus grand")
    N=int(input())
    compteur=compteur+1
if N==alea:
    print("Gagne en",compteur,"essais")
else:
    print("Perdu ! il fallait trouver",alea)
```

Exercice 11 : ordre alphabétique.

```
#test de bonne saisie
ch1=input("Saisir une chaine 1 de caracteres en minuscules : ")
ch2=input("Saisir une chaine 2 de caracteres en minuscules : ")
condition=(ch1=="" or ch2=="")
while condition==True:
    ch1=input("Saisir une chaine 1 de caracteres en minuscules : ")
    ch2=input("Saisir une chaine 2 de caracteres en minuscules : ")
    condition=(ch1=="" or ch2=="")

m=min(len(ch1),len(ch2))
for i in range(m):
    if ch1[i]<ch2[i]:
        print(ch1,"<",ch2)
        break
    elif ch1[i]>ch2[i]:
        print(ch2,"<",ch1)
        break
else: #pour le cas abc < abcd par exemple
    if len(ch1)<len(ch2):
        if ch2[m]!="":
            print(ch1,"<",ch2)
    elif len(ch2)>len(ch1):
        if ch1[m]!="":
            print(ch2,"<",ch1)
    else:
        print("Les chaines sont identiques")
```

Exercice 12 : Chiffres romains.

```
N=int(input("saisissez un entier entre 1 et 3999 "))
Nbis=N #copie de N
R=""
while N>=1000:  #car M peut etre ecrit plusieurs fois consecutivement
    R=R+"M"
    N=N-1000
        #une fois tous les milliers epuises, on regarde les centaines
if N>=900:
```

```
    R=R+"CM"
    N=N-900

elif N>=500:    #ou if
    R=R+"D"
    N=N-500

elif N>=400:    #ou if
    R=R+"CD"
    N=N-400

while N>=100:   #car C peut etre ecrit plusieurs fois consecutivement
    R=R+"C"
    N=N-100
            #une fois les centaines epuisees, on regarde les dizaines
if N>=90:
    R=R+"XC"
    N=N-90

elif N>=50:     #ou if
    R=R+"L"
    N=N-50

elif N>=40:     #ou if
    R=R+"XL"
    N=N-40

while N>=10: #car X peut etre ecrit plusieurs fois consecutivement
    R=R+"X"
    N=N-10
            #une fois les dizaines epuisees, on regarde les unites
if N>=9:
    R=R+"IX"
    N=N-9

elif N>=5:  #ou if
    R=R+"V"
    N=N-5

elif N>=4:  #ou if
    R=R+"IV"
    N=N-4

while N>=1: #car I peut etre ecrit plusieurs fois consecutivement
    R=R+"I"
    N=N-1

print (Nbis,"s'ecrit ",R, "en chiffres romains")
```

Exercice 13 : Attention à l'indexation !

```
ch=input("Saisir une chaine de caracteres : ")
#question 1
neo=""
for i in range(len(ch)):
    neo=neo+ch[len(ch)-1-i] #Attention a l'indice de fin
```

```
print(neo)

#question 2
neo=""
for i in range(len(ch)-1,-1,-2): #le retour de range(a,b,k)
    neo=neo+ch[i]
print(neo)
```

Exercice 14 : Nous ne donnons que le script des questions 2, 3 et 4.

```
#En toute rigueur, nous devrions effectuer un test de bonne saisie.

#question 2
ch=input("Saisir un mot en minuscules ")
CH=""
for i in range(len(ch)):
    CH=CH+chr(ord(ch[i])-32)
print(CH)

#question 3
CH=input("Saisir un mot en majuscules ")
ch=""
for i in range(len(CH)):
    ch=ch+chr(ord(CH[i])+32)
print(ch)

#question 4
mot=input("Saisir un mot avec majuscules et minuscules melangees ")
mot2=""
for el in mot:
    if ord(el)<=90:
        mot2=mot2+chr(ord(el)+32)
    else:
        mot2=mot2+chr(ord(el)-32)
print(mot2)
```

Exercice 15 : Plus court, tu meurs !

```
#question1
print(ch[::-1])

#question 2
print(ch[::-2])
```

Exercice 16 : Complétons le script entamé.

```
from random import randint

#creation du mot mystere choisi par l'ordinateur
L=['a','e','i','o','u','y'] #liste de voyelles
mot_mystere=""              #mot vide au depart
for i in range(5):
    mot_mystere=mot_mystere+L[randint(0,5)]
Lmot=[""]*5                 #votre mot en liste (pourquoi ?)
```

```
#print(mot_mystere)
while Lmot!=list(mot_mystere):
    essai=input("Saisir une voyelle : ")
    if essai in mot_mystere:
        for i in range(5):
            if essai==mot_mystere[i]:
                Lmot[i]=essai
    print(Lmot)
print("Bravo !", Lmot)
```

La variable de sortie est de type list. Pour améliorer le script, écrivez-la sous forme de chaine de caractères.

Exercice 17 : Récréation de re-créations !

```
#Question 1 : minimum d'une liste
L=[]
print("Saisir cinq nombres ") #saisie des elements
for i in range(5):
    print("Nombre",i+1,"? ")
    N=float(input())
    L.append(N)

min=L[0]
for j in range(1,5):
    if L[j]<min:
        min=L[j]
print("Le minimum de",L,"est",min)

#Question 2 : tri dans l'ordre croissant
#La notion de fonction informatique est tres utile ici
#Mais nous la verrons plus tard !
L=[]
print("Saisir cinq nombres ") #initialisation de L
for i in range(5):
    print("Nombre",i+1,"? ")
    N=float(input())
    L.append(N)

Lbis=[]
for j in range(5): #construction de Lbis
    min=L[0]    #L'initialisation de min est ici
    for k in range(1,len(L)):
        if L[k]<min:
            min=L[k]
    Lbis.append(min)
    L.remove(min)
print(Lbis)

#Question 3 : tri dans l'ordre decroissant
#Il suffit de changer min en max
#Le lecteur le fera aisement !
```

Exercice 18 : Il faut s'en **sort()**ir !

```
#Question 1
from random import randint
```

```
L=[randint(1,6) for i in range(5)]
print(L)

#Question 2
M=[]
for a in L:
    if L.count(a)>1:
        M.append(a)
M.sort()
print(M)

#Question 3
pairs,impairs=[ ],[ ]
for a in L:
    if a%2==0:
        pairs.append(a)
    else:
        impairs.append(a)
pairs.sort()
impairs.sort()
print(pairs,impairs)
```

Exercice 19 : Compréhension et indices !

```
#Question 1
print(sum([i for i in range(1,202,2)]))
#on trouve 10201

#Question 2
N=15
print(sum([1/i**2 for i in range(1,N+1)]))
#on trouve 1.580440283444987

#Question 3
print("S=",sum([1/(2*i)**2 for i in range(4,11)]))
print("T=",sum([1/(2*i+1)**2 for i in range(4,11)]))
#on trouve S= 0.047164155013857394, puis
#T= 0.039469610441116035

#Question 4
N=30 #on prend un N "grand" (N=22 suffit)
L=[1/i**2 for i in range(1,N)]
print("S=",sum(L[7:20:2])) #Attention au decalage !
print("T=",sum(L[8:22:2]))

#Question 5
from math import sqrt
N=int(input("Saisir un entier naturel N non nul : "))
A=[]
print("Saisie des coordonnees du vecteur : ")
for k in range(N) :
    print("a_"+str(k+1)+"= ? ")
    a=float(input())
    A.append(a)
print("norme euclidienne de A :",sqrt(sum([a**2 for a in A])))
```

Exercice 20 : on mélange ce qu'on vu avant.

```
#Question 1
ch=input("saisir une chaine de caracteres en minuscules : ")
#test de bonne saisie a faire normalement
chtri=""
L=[ord(ch[i]) for i in range(len(ch))]
L.sort() #sinon methode sort() a recreer (vu avant)
for carac in L:
    chtri=chtri+chr(carac)
print(chtri)

#Question 2
from random import randint
#bases=['A','C','T','G']
ch1=input("Saisir votre brin d'ADN : ")
#normalement test de bonne saisie a faire
ch2="" #Le brin complementaire
#script naturel, mais long !
for el in ch1:
    if el=='A':
        ch2=ch2+'T'
    elif el=='C':
        ch2=ch2+'G'
    elif el=='T':
        ch2=ch2+'A'
    elif el=='G': #pourquoi pas else ?
        ch2=ch2+'C'
print(ch1)  #affichage de la base de N nucleotides
print(len(ch1)*"|",end='') #alignement de |
print("")
print(ch2) #Le brin complementaire

#Remarque: ce script peut etre simplifie en remarquant
#quelque chose sur l'indexation des elements de bases !
```

Exercice 21 : Exercice 2A ou 2B : ça se corse !

```
from random import *
voyelles=['a','e','i','o','u','y']
consonnes=['b','c','d','f','g','h','j','k','l','m','n','p','q','r','s'\
,'t','v','w','x','z']
liste=[voyelles[randint(0,5)] for i in range(5)]+[consonnes[randint(0,19)]\
 for i in range(5)]
shuffle(liste)  #la liste de 5 voyelles et de 5 consonnes melangees est creee
#il reste maintenant a les re-ordonner par ordre d'apparition
print("entree=",liste)
#initialisation de la liste alternee L
L,V,C=[],[],[]  #V et C sont les listes de consonnes et de voyelles de liste
for ch in liste:
    test=(ch in voyelles)
    if test:           #on separe de maniere ordonnee
        V.append(ch)
    else:
        C.append(ch)

test=(liste[0] in voyelles)   #pour bien demarrer
```

```
if test:
    for i in range(len(C)):    #on reunit en alternant
        L.append(V[i])
        L.append(C[i])
else:
    for i in range(len(C)):
        L.append(C[i])
        L.append(V[i])

#print(V,C)
print("sortie=",L)
```

Améliorez ce script : attendre la section Fonctions...

Exercice 22 : Comme avant, la section Fonctions rendra plus lisible ces scripts.

```
from turtle import *
#Question 1
for i in range(4):
    forward(80)
    left(90)

reset() #Pour effacer la figure d'avant
#Question 2
for i in range(5):
    forward(70)
    left(72)

reset()
#Question 3
for i in range(6):
    forward(100)
    left(60)
for j in range(3):
    forward(100)
    left(120)
    forward(200)
    left(120)

reset()
#Question 4
for i in range(10):  #Attention : motif=carre + espace
    color('red')  #couleur des carres : rouge
    for j in range(4):
        forward(10)  #taille d'un carre : 10
        left(90)
    up()
    forward(10+10)  #espacement entre carres de 10
    down()

reset()
#Question 5
t=10  #taille du premier carre
for i in range(4):
    for j in range(4):
        color('blue')
        forward(t)
```

```
        left(90)
    up()
    forward(t+10)   #espacement entre carres de 10
    down()
    t=t+15

reset()
#Question 6
t=10   #mesure d'un cote du carre
e=10   #ecart entre carres
for y in range(5):
    for x in range(5):
        for i in range(4):
            color('violet') #pour changer...
            forward(t)
            left(90)
        up()
        forward(t+e)
        down()
    up()
    goto(0,(y+1)*(t+e))
    down()

mainloop()
```

Exercice 23 : Une fonction servira à toutes les questions.

```
from turtle import *
#fonction commune a toute les questions
def polygoneG(N,T,C):
    color(C)
    begin_fill()
    for i in range(N):
        forward(T)
        left(360/N)
    end_fill()

#motif en couleurs plein
def motif():
    from math import sqrt
    polygoneG(N=4,T=60,C='blue')
    forward(30)
    color('yellow')
    begin_fill()
    circle(30)
    end_fill()
    left(45)
    polygoneG(N=4,T=30*sqrt(2),C='green')

motif()
exitonclick()
```

Le motif composé de carrés bleus et de triangles rouges autour d'un hexagone peut s'obtenir par le script qui suit. Petit bonus : on a rempli les figures.

```
#second motif (carres et triangles autour d'un hexagone)
#Attention, il convient de definir une autre fonction polygoneD(N,T,C)
```

```
#qui trace les figures en tournant a droite !
def polygoneD(N,T,C):
    color(C)
    begin_fill()
    for i in range(N):
        forward(T)
        right(360/N)
    end_fill()

def figure():
    polygoneD(N=4,T=80,C='blue')
    forward(80)
    left(60)
    polygoneD(N=3,T=80,C='red')
    forward(80)
    left(60)

for i in range(3):
    figure()
mainloop()
```

La rosace (nouveau motif) s'obtient par :

```
#motif en rosace (carres et pentagones entremeles)
def rosace():
    for i in range(6):
        polygoneG(N=4,T=30,C='blue')
        left(30)
        up()
        forward(30)
        down()
        polygoneG(N=5,T=30,C='red')
        left(30)
        up()
        forward(30)
        down()

rosace()
mainloop()
```

Nous laissons à la sagacité du lecteur le soin de tracer la suite de 8 rosaces, et pourquoi pas de l'enrouler !

Exercice 24 : Attention à la portée des variables !

```
# on se donne un R.O.N (O;i;j)et deux points A(xA;yA) et B(xB;yB)
# on souhaite savoir si un point M(xM ; yM) appartient au cercle de diametre [AB]

def saisie():
    xA=float(input("Abscisse de A ? "))
    yA=float(input("Ordonnee de A ? "))
    xB=float(input("Abscisse de B ? "))
    yB=float(input("Ordonnee de B ? "))
    xM=float(input("Abscisse de M ? "))
    yM=float(input("Ordonnee de M ? "))

def distance2(x1,y1,x2,y2):
    return (x1-x2)**2+(y1-y2)**2
```

```
def Est_sur_C():
    saisie()
    if distance2(xA,yA,xB,yB)==distance2(xA,yA,xM,yM)+distance2(xM,yM,xB,yB):
        return 1
    else:
        return 0

if Est_sur_C()==1:
    print("M appartient au cercle de diametre [AB]")
else:
    print("M n'appartient pas au cercle de diametre [AB]")
```

Exercice 25 : Nous donnons un script utilisant une sortie de boucle : l'instruction break.

```
from math import sqrt

def liste_premiers(n):
    liste,impairs=[2],[i for i in range(3,n+1,2)]
    for k in impairs:           #on ne teste que les entiers impairs k <= n
        for div in range(3,int(sqrt(k))+1,2):     #diviseurs impairs
            if k%div==0:        #si k a au moins un diviseur div<=racine(k)
                break
        else:                           #signifie que k est premier
            liste.append(k)
    return(liste)


#decomposition en facteurs premiers
def decompositionP(n):
    decomposition=[]
    for p in liste_premiers(n):
        test=(n%p==0)
        while test:                 #Tant que test est vrai
            decomposition.append(p) #liste des facteurs premiers de n
            n=n//p
            test=(n%p==0)
    return decomposition


#Programme principal
n=int(input("Saisir un entier n : "))
while n<=1:                         #test de bonne saisie
    n=int(input("Saisir un entier n : "))
print("decomposition de n en facteurs premiers :")
print(decompositionP(n))
```

2 Exercices thématiques

2.1 Géométrie

Exercice x : On se prend des droites ?

1. Par soucis de rigueur, on effectue un test pour vérifier que les trois points A, B et C sont bien distincts. La condition d'alignement repose sur la colinéarité des vecteurs $\overrightarrow{AB}$ et $\overrightarrow{AC}$, que l'on exprime analytiquement à l'aides des coordonnées des points A, B et C.

```
#question 1 : determiner si 3 points distincts A, B et C sont alignes.
#on prend le point de vue vectoriel

def sontDistincts(xA,yA,xB,yB,xC,yC):
    if (xA==xB and yA==yB) or (xA==xC and yA==yC) or (xB==xC and yB==yC):
        return False
    else:
        return True

def alignement(xA,yA,xB,yB,xC,yC):
    if sontDistincts(xA,yA,xB,yB,xC,yC):
        delta=(xB-xA)*(yC-yA)-(xC-xA)*(yB-yA)
        if delta==0:
            return True
        else:
            return False

xA=float(input("Saisir l'abscisse de A "))
yA=float(input("Saisir l'ordonnee de A "))
xB=float(input("Saisir l'abscisse de B "))
yB=float(input("Saisir l'ordonnee de B "))
xC=float(input("Saisir l'abscisse de C "))
yC=float(input("Saisir l'ordonnee de C "))

if alignement(xA,yA,xB,yB,xC,yC):
    print("A, B et C sont alignes")
else:
    print("A, B et C ne sont pas alignes")
```

2. Il faut prendre garde ici qu'une, voire les deux droites peuvent être verticales (faire un dessin pour les différents cas). Comme précédemment, on vérifiera que les points sont tous distincts.

```
#question 2
def sontDistincts2(xA,yA,xB,yB,xC,yC,xD,yD):
    if (xA==xB and yA==yB) or (xA==xC and yA==yC) or (xB==xC and yB==yC)\
    or (xA==xD and yA==yD) or (xD==xC and yD==yC) or (xB==xD and yB==yD):
        return False
    else:
        return True

def Positions(xA,yA,xB,yB,xC,yC,xD,yD):
    if sontDistincts2(xA,yA,xB,yB,xC,yC,xD,yD):
        if (xA==xB and xC==xD):
            print("(AB) et (CD) sont verticales et paralleles")
        elif (xA!=xB and xC!=xD):
            m1,m2=(yB-yA)/(xB-xA),(yD-yC)/(xD-xC)
            p1,p2=yA-m1*xA,yC-m2*xC
```

```
            if (m1==m2 and p1==p2):
                print("(AB) et (CD) sont confondues")
            elif (m1==m2 and p1!=p2):
                print("(AB) et (CD) sont strictement paralleles")
            else:
                print("(AB) et (CD) sont secantes")
        else:
            print("(AB) et (CD) sont secantes")

xA=float(input("Saisir l'abscisse de A "))
yA=float(input("Saisir l'ordonnee de A "))
xB=float(input("Saisir l'abscisse de B "))
yB=float(input("Saisir l'ordonnee de B "))
xC=float(input("Saisir l'abscisse de C "))
yC=float(input("Saisir l'ordonnee de C "))
xD=float(input("Saisir l'abscisse de D "))
yD=float(input("Saisir l'ordonnee de D "))

Positions(xA,yA,xB,yB,xC,yC,xD,yD)
```

3. Il suffit d'appliquer le script précédent. On trouve que les droites (AB) et (CD) sont strictement parallèles.

Exercice y : Nous rappelons qu'un trapèze est un quadrilatère qui possède deux côtés parallèles. En particulier, tout parallélogramme est un trapèze.
La condition qui fait de ABCD un trapèze est donc :
$condition0 = ((xB - xA) * (yC - yD) - (yB - yA) * (xC - xD) == 0$ or $(xD - xA) * (yC - yB) - (yD - yA) * (xC - xB) == 0)$. Dès lors le script est immédiat :

```
#Saisie des coordonnees des 4 points
x_A=float(input("Abscisse de A ? "))
y_A=float(input("Ordonnee de A ? "))
x_B=float(input("Abscisse de B ? "))
y_B=float(input("Ordonnee de B ? "))
x_C=float(input("Abscisse de C ? "))
y_C=float(input("Ordonnee de C ? "))
x_D=float(input("Abscisse de D ? "))
y_D=float(input("Ordonnee de D ? "))

def trapeze(xA,yA,xB,yB,xC,yC,xD,yD):
    condition00=(xB-xA)*(yC-yD)-(yB-yA)*(xC-xD)==0
    condition01=(xD-xA)*(yC-yB)-(yD-yA)*(xC-xB)==0
    condition0=(condition00 or condition01)
    if condition0:
        print("ABCD est un trapeze")
    else:
        print("ABCD n'est pas un trapeze")

trapeze(x_A,y_A,x_B,y_B,x_C,y_C,x_D,y_D)
```

Exercice z : Nous allons nous servir de la condition 0 de l'exercice précédent ainsi que des conditions 1, 2 et 3 du cours qui nous ont servi à définir un parallélogramme, un rectangle, un losange et a fortiori un carré. Rappelons-les. Considérons donc un quadrilatère ABCD.

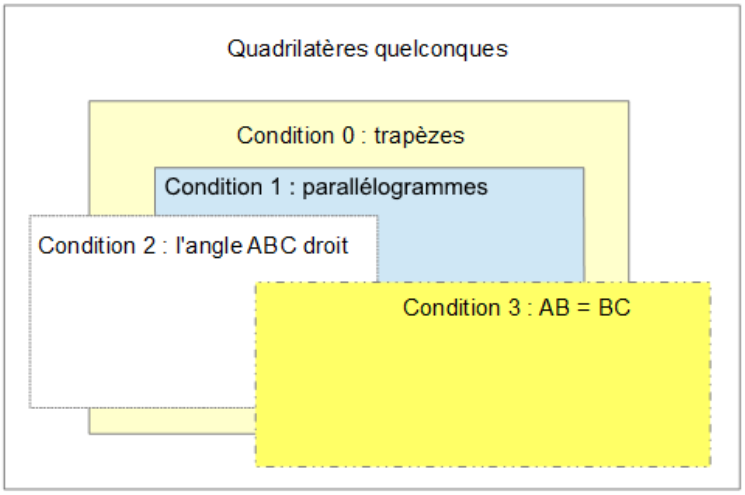
- condition 0 : "ABCD a (au moins) deux côtés parallèles", ce que l'on a traduit par : $condition0 = ((xB - xA) * (yC - yD) - (yB - yA) * (xC - xD) == 0$ or $(xD - xA) *$

$(yC - yB) - (yD - yA) * (xC - xB) == 0)$

- condition 1 : "ABCD a ses côtés parallèles deux à deux", ce que l'on traduit par : $condition1 = (xB - xA == xC - xD \text{ and } yB - yA == yC - yD)$
- condition 2 : "L'angle $\widehat{ABC}$ est droit", ce que l'on a traduit par : $condition2 = ((xB - xA) * *2 + (yB - yA) * *2 + (xB - xC) * *2 + (yB - yC) * *2 == (xC - xA) * *2 + (yC - yA) * *2)$
  Attention : c'est la conjonction des conditions 1 et 2 qui permettait d'affirmer que ABCD, en tant que parallélogramme, était en fait un rectangle.
- condition 3 : "Les côtés AB et BC sont de même longueur", ce que l'on a traduit par : $condition3 = ((xB - xA) * *2 + (yB - yA) * *2 == (xB - xC) * *2 + (yB - yC) * *2)$
  Attention : c'est la conjonction des conditions 1 et 3 qui permettait d'affirmer que ABCD, en tant que parallélogramme, était en fait un losange.

En schématisant à l'aide d'un diagramme de Venn, c'est ce dernier qui va nous donner l'idée du script :

screenshot001.png



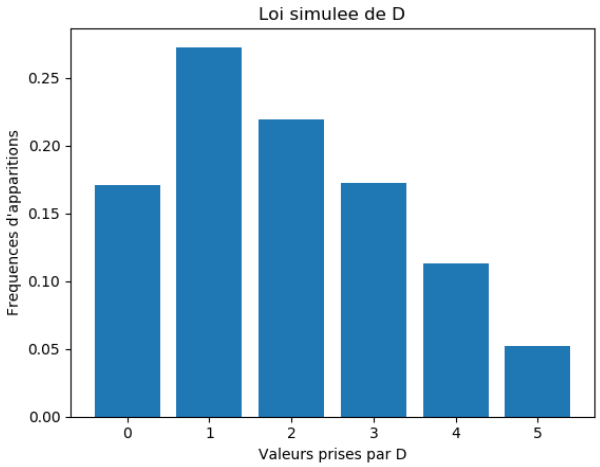

```
xA=float(input("abscisse de A ? "))
yA=float(input("ordonnee de A ? "))
xB=float(input("abscisse de B ? "))
yB=float(input("ordonnee de B ? "))
xC=float(input("abscisse de C ? "))
yC=float(input("ordonnee de C ? "))
xD=float(input("abscisse de D ? "))
yD=float(input("ordonnee de D ? "))

def Quadrilatere(xA,yA,xB,yB,xC,yC,xD,yD):
    condition00=((xB-xA)*(yC-yD)-(yB-yA)*(xC-xD)==0)
    condition01=((xD-xA)*(yC-yB)-(yD-yA)*(xC-xB)==0)
    condition0=( condition00 or condition01)
    condition1=((xB-xA==xC-xD) and (yB-yA==yC-yD))
    condition2= ((xB-xA)**2+(yB-yA)**2+(xC-xB)**2+(yC-yB)**2==(xC-xA)**2+(yC-yA)**2)
    condition3=((xB-xA)**2+(yB-yA)**2==(xC-xB)**2+(yC-yB)**2)
    if condition0:
        print("ABCD est un trapeze")
    if condition1:
```

```
        print("ABCD est aussi un parallelogramme")
    if condition1 and condition2:
        print("ABCD est un rectangle")
    if condition1 and condition3:
        print("ABCD est meme un losange")
    if condition1 and condition2 and condition3:
        print("ABCD est meme plus : un carre !")
    if not condition0:
        print("ABCD est un quadrilatere quelconque")

Quadrilatere(xA,yA,xB,yB,xC,yC,xD,yD)
```

2.2 Probabilités - statistiques

Exercice x : La variable aléatoire D : "différence entre le plus grand et le plus petit des deux nombres affichés" prend clairement pour valeurs : $\{0;1;2;3;4;5\}$ (à proposer aux élèves en réflexion préliminaire). Notons que le langage Python dispose des fonctions min et max comparant deux réels donnés, mais il est pédagogiquement intéressant de les reconstruire ad hoc, ce que nous ne ferons pas ici.

```
def unePartie():
    from random import randint
    alea1,alea2=randint(1,6),randint(1,6)
    D=max(alea1,alea2)-min(alea1,alea2)
    return D

def frequence(N):
    L=[0 for i in range(6)] #liste des valeurs de D initialisee
    for j in range(N):
        L[unePartie()]+=1
    for k in range(6):
        L[k]/=N
    return(L)

#Programme principal
import numpy as np
import matplotlib.pylab as plt

N=int(input("Combien de parties ? "))
x=list(range(6))
y=[el for el in frequence(N)]
plt.title('Loi simulee de D')
plt.xlabel('Valeurs prises par D')
plt.ylabel("Frequences d'apparitions")
plt.bar(x,y)
plt.show()
```



Exercice xx : Les puristes vont se précipiter pour créer un arbre de probabilité et en déduire la loi de variable aléatoire gain afin de calculer son espérance. C'est tout à fait adapté car le problème est simple ! Il est néanmoins très formateur de décomposer ce problème dynamique à l'aide de fonctions informatiques et de faire tourner la "machine fréquentiste" ! Tout à fait présentable en classe.

Stratégie : nous allons créer cinq fonctions correspondant :

- au tirage de la pièce truquée,
- au tirage dans l'urne 1 contenant 4 boules blanches et 6 boules noires,
- au tirage dans l'urne 2 contenant 2 boules blanches et 8 boules noires,
- au gain algébrique obtenu par partie,
- au gain moyen obtenu lors de N parties.

Nous allons ensuite les imbriquer de manière précise pour s'en servir dans le programme principal.

```
def piece():
    alea=random()
    if alea<1/3:
        return 1        #on tombe sur pile, donc tirage dans l'urne 1
    else:
        return 0        #on tombe sur face, donc tirage dans l'urne 2

def urne1():
    alea=randint(1,10)
    if alea<=4:         #on tire une boule blanche dans l'urne 1
        return 5
    else:               #on tire une boule noire dans l'urne 1
        return -2

def urne2():
    alea=randint(1,10)
    if alea<=2:
        return 5
    else:
        return -2

def gain():
    piece()
```

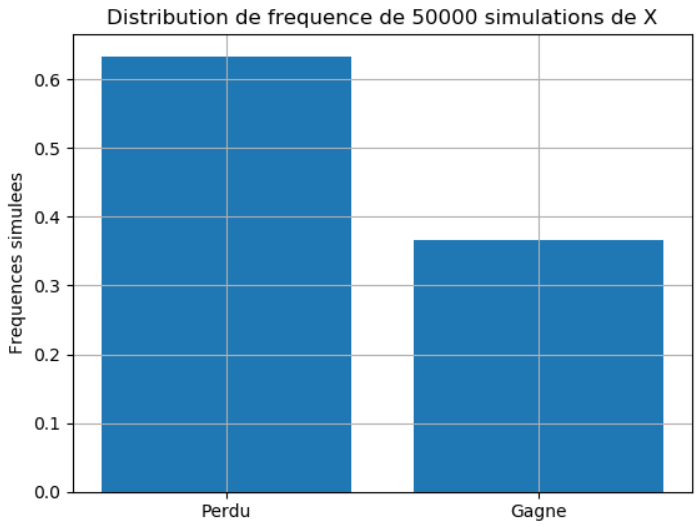
```
    if piece()==1:      #choix de l'urne en fonction du tirage de la piece
        g=urne1()
    else:
        g=urne2()
    return g

def moyenne(N):
    G=0
    for i in range(N):
        G=G+gain()
    return G/N

#Programme principal
from random import *
N=int(input("Nombre d'experiences ? "))
print("Gain moyen : ",moyenne(N))
```

Exercice xxx : Donnons d'abord le script des fonctions demandées aux trois premières questions :

```
def unePartie():
    from random import randint
    S=sum([randint(1,6) for i in range(3)])
    while 8<=S<=14:
        S=sum([randint(1,6) for i in range(3)])
    if S<=7:
        return 0
    else:
        return 1

def TempsAttente():
    from random import randint
    S=sum([randint(1,6) for i in range(3)])
    compteur=1
    while 8<=S<=14:
        S=sum([randint(1,6) for i in range(3)])
        compteur+=1
    return compteur

def frequence(N):
    L=[0,0]
    for i in range(N):
        L[0]+=unePartie()
        L[1]+=TempsAttente()
    L[0]/=N;L[1]/=N
    return L

#Programme principal
N=int(input("Combien de parties ? "))
print("Frequence de gain :",frequence(N)[0])
print("Temps moyen d'une partie :",frequence(N)[1])

#En testant sur 10 000 parties, on obtient :
#Frequence de gain : 0.3684
#Temps moyen d'une partie : 3.9093
#Bien sur, il y a fluctuation d'echantillonnage.
```

Puis la loi simulée de X :

```
"Diagramme en barres de la distribution des frequences simulees de X."
"Utilise frequence(N) et la bibliotheque matplotlib.pylab."

import matplotlib.pylab as plt
N = 50000
x = (0, 1)
freqX1 = frequence(N)[0]
y = (1 - freqX1, freqX1)
titre = 'Distribution de frequence de '+ str(N) + ' simulations de X'
plt.figure()
plt.bar(x, y, tick_label = ('Perdu', 'Gagne'), align = 'center')
plt.title(titre)
#plt.xticks((0, 1), ('Perdu', 'Gagne'))
plt.ylabel('Frequences simulees')
plt.grid()
```



Nous donnerons pour la représentation graphique de la loi simulée de Y un script un peu orienté numpy :

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

def unePartie():
    from random import randint
    S=sum([randint(1,6) for i in range(3)])
    compteur=1      #nombre de coups par partie
    while 8<=S<=14:
        S=sum([randint(1,6) for i in range(3)])
        compteur+=1
    return compteur

def loi(N):
    L=[unePartie() for i in range(N)]
    return np.bincount(L,minlength=20)[1:]/N #explications plus tard
```

```
def frequence(N):
    coups=0      #duree d'une partie
    for i in range(N):
        coups+=unePartie()
    return coups/N

#Programme principal
N=int(input("Nombre de parties ? "))
print("La duree moyenne d'une partie est de : ",frequence(N))
Y=loi(N)
X=[i for i in range(1,len(Y)+1)]
plt.figure()
plt.bar(X,Y,width=0.5, color='b')
plt.title("Loi de probabilite simulee de Y")
plt.xlabel("Duree d'une partie")
plt.ylabel('Frequences simulees')
plt.grid()
plt.show()
```

Puis la loi simulée de Y :