

# Chapitre 1

## Nombres, opérations et fonctions dans Python

### 2. Représentation des nombres

#### 2. 2. La représentation binaire des entiers naturels

```
.  
# Ecriture binaire d'un entier naturel  
from math import*  
N=eval(input("Valeur de l'entier naturel N ? "))  
n=int(log(N)/log(2))  
binaire=[]  
for i in range(n,-1,-1):  
    q=N//(2**i)  
    r=N%(2**i)  
    binaire=binaire+[q]  
    N=r  
print(binaire)
```

#### 2. 6. La précision des calculs

```
from math import*  
a,b,c=sqrt(2),sqrt(3),sqrt(5)  
if c*c==a*a+b*b:  
    print("Ce calcul est exact.")  
else:  
    print("Ce calcul est faux !")  
print("valeur de c*c : ", c*c)  
print("Valeur de a*a+b*b : ",a*a+b*b)
```

### 3. Fonctions disponibles dans Python

#### 3.3. Comment définir ses propres fonctions ?

```
# Définition d'une fonction f telle que  $f(x)=\sqrt{1+x^2}$ 
from math import*
def f(x):
    return sqrt(x*x+1)
# Utilisation de la fonction f pour constituer une table de valeurs
x=0
for i in range(1,11):
    y=f(x)
    print("Pour x=",x," f(x)=",y)
    x=x+1
```

#### 3.4. Définir une fonction par une suite d'actions

```
from math import*
def f(x):
    if x>=1/3:
        return 3*x-1
    else:
        return 1-3*x
x=-5
for i in range(1,11):
    y=f(x)
    print("Pour x=",x," f(x)=",y)
    x=x+1
```

Le résultat du calcul est le suivant :

```
Pour x= -2 f(x)= 7    Pour x= 2 f(x)= 5
```

#### 3.5. Le mot réservé « lambda »

```
from math import*
f=lambda x : sqrt(x+1)
x=eval(input("Valeur de x ?"))
y=f(x)
print("Pour x=",x," f(x)=",y)
```

```
# Définition et emploi de deux fonctions affines
f=lambda x: 2*x-3
g= lambda x : 5-3*x
```

```
print("Valeurs de f(x) :")
for i in range (-4,4) :
    print(f(i))
print("Valeurs de g(x) :")
for i in range (-4,4) :
    print(g(i))
```

## 4. La récursivité des fonctions

### 4.1 Factorielles

```
# Calcul itératif de n !
def factorielle(n) :
    f=1
    for i in range(1,n+1) :
        f=f*i
    return f
# Utilisation de la fonction
n=eval(input("Quelle est la valeur de l'entier n ;"))
print("factorielle ",n, "=", factorielle(n))
```

```
# Calcul récursif de n!
def fact(n):
    if n==0:
        return (1)
    else :
        return(n*fact(n-1))
# Utilisation de la fonction
n=eval(input("Quelle est la valeur de l'entier n ?"))
print("factorielle ",n," =", fact(n))
```

### 4.2 Une application de la récursivité : les tours de Hanoï

```
# On définit la fonction « déplace » qui décrit le déplacement des disques
# d'un piquet à l'autre
def déplace(n,A,C,B):
    if n==1:
        print("de ",A, " vers ", C)
    else :
        déplace(n-1,A,B,C)
        déplace(1,A,C,B)
        déplace(n-1, B,C,A)
# Le programme indique les déplacements de disques quand le nombre n des
# disques est choisi
A=1
B=2
C=3
n=int(input("Choisissez le nombre des disques : "))
déplace(n,A,B,C)
```

## Chapitre 2

### Suites de nombres réels

#### 1. Suites et racines carrées

##### 1.1 La méthode d'Archytas de Tarente

```
# Calcul d'une racine carrée avec la méthode d'Archytas de Tarente
A=eval(input("Valeur de A : "))
x,y=2,A/2
for i in range(1,6):
    m=(x+y)/2
    h=x*y/m
    print("x=",x," et y=",y)
    x,y=m,h
```

```
Valeur de A : 2
x= 2 et y= 1.0
x= 1.5 et y= 1.3333333333333333
x= 1.4166666666666665 et y= 1.411764705882353
x= 1.4142156862745097 et y= 1.41421143847487
x= 1.4142135623746899 et y= 1.4142135623715
```

##### 1.2 La méthode de Héron d'Alexandrie

```
# Calcul d'une racine carrée avec la méthode de Héron d'Alexandrie
A=eval(input("Valeur du nombre A ? "))
n=eval(input("Valeur de n ? "))
u=A/2
for i in range(1,n+1):
    v=(u+A/u)/2
    print(v)
    u=v
```

```
from math import*
a=10
n=8
u=a
for i in range(1,n+1):
    v=(u+a/u)/2
    print(v)
    u=v
```

### 1.3 Le calcul d'une racine cubique

```
# Calcul d'une racine cubique. Méthode de Héron d'Alexandrie
from math import*
a=eval(input("Valeur du nombre positif a ? "))
m=3
u=a/3
v=u/2+a/(2*u*u)
e=abs(u-v)
while e>0.000001:
    u=v/2+a/(2*v*v)
    e=abs(v-u)
    v=u
print("Valeur approchée de la racine cubique de ",a," =",v)
print("Valeur exacte selon Python = ", a**(1/3))
```

## 2. Comment définir une suite ?

### 2.2 Suites définies par $u_n=f(n)$

```
from math import*
def f(n):
    return (2*n-1)/(n+1)
n=eval(input("Choisissez n :"))
for i in range(1,n+1):
    print("terme de rang ",i,"=",f(i))
```

```
from math import*
def f(n):
    return (2*n-1)/(n+1)
n=eval(input("Choisissez n :"))
print("Le terme de rang ", n," est ",f(n))
```

### 2.3 Suites récurrentes

```
from math import*
def f(x):
    return sqrt(1+x)
n=eval(input("Choisir l'entier n : "))
u=1
print("rang 1:" , " ",u)
for i in range(2,n+1):
    v=f(u)
    print("rang ",i," : ",v)
    u=v
```

```
n=eval(input("Choisir l'entier n : "))
u=0
v=1
print("terme de rang 1 :",u)
print("terme de rang 2 :",v)
for i in range(3, n+1):
    w=(u+v)/2
    print("terme de rang ",i," :",w)
    u=v
    v=w
```

### 3. Quand $n$ devient de plus en plus grand

#### 3.1 Une suite peut être convergente

```
n=eval(input("Valeur de n ?"))  
u=(2*n-1)/(n+1)  
print(" pour n=",n,"u=",u)
```

```
from math import*  
n=10  
for i in range(1,n+1):  
    u=i*sin(1/i)  
    print ("u",i,"=",u)
```

## 4. Une suite célèbre : la suite de Fibonacci

### 4.2 Le problème des lapins

```
# Programme Fibonacci itératif
u,v=1,1
liste=[u,v]
n=eval(input("Choisir l'entier n : "))
for i in range(3,n+1):
    w=u+v
    liste=liste+[w]
    u=v
    v=w
print(liste)
```

```
# Programme Fibonacci récursif
liste=[]
def fibo(n) :
    if n==1 or n==2:
        return 1
    else:
        return fibo(n-1)+fibo(n-2)
n=eval(input("Choisir l'entier : "))
for i in range(1,n+1):
    liste=liste+[fibo(i)]
print (liste)
```

```
# Rapport de 2 termes successifs de la suite de Fibonacci
u,v=1,1
liste=[]
n=eval(input("Choisir l'entier n : "))
for i in range(3,n+1):
    w=u+v
    r=w/v
    liste=liste+[r]
    u=v
    v=w
print(liste)
```

### 4.3 Etude du nombre $|r_n - \varphi|$

```
from math import*
u,v=1,1
e=eval(input("Valeur du nombre epsilon ? "))
dr=0
```

```
n=2
while dr==0:
    w=u+v
    r=w/v
    n=n+1
    if abs(r-(sqrt(5)+1)/2)<e:
        dr=1
    else:
        u=v
        v=w
print("n=",n)
```

#### 4.5 La formule de Binet

```
# Formule de Binet
from math import*
n=eval(input("Valeur de n ? "))
a,b=1+sqrt(5),1-sqrt(5)
F=((a**n)-(b**n))/2**n/sqrt(5)
print(F)
```

## 5. Suites définies par des sommes

### 5.1 Historique

```
# Somme des entiers de 1 à n
n=eval(input("Choisissez l'entier n : "))
S=0
for i in range(1,n+1):
    S=S+i
print (" Somme des entiers de 1 à", n, " =",S)
```

Le programme donne  $\sum_1^{1000} i = 500500$ .

### 5.2 Somme des carrés et des cubes des entiers naturels de 1 à $n$

```
# Somme des carrés des entiers de 1 à n
n=eval(input("Choisissez l'entier n : "))
S=0
for i in range(1,n+1):
    S=S+i*i
print ("Somme des carrés des entiers de 1 à",n, " =",S)
print("D'après la formule S=",n**3/3+n**2/2+n/6)
```

```
# Somme des cubes des entiers de 1 à n
n=eval(input("Choisissez l'entier n : "))
S=0
for i in range(1,n+1):
    S=S+i*i*i
print (" Somme des cubes des entiers de 1 à", n, " =",S)
```

### 5.3 Les séries géométriques

```
# Série géométrique de Nicole Oresme
a=eval(input("Choisissez un nombre a : "))
n=eval(input("Choisissez un entier n : "))
u=1/a
S=u
for i in range(1,n+1):
    v=u/a
    S=S+v
    u=v
print("Somme de la série : ",S)
```

### 5.4 La série de Swineshead

```
# Série de Richard Swineshead
```

```
n=eval(input("Choisissez un entier n : "))
S=0
for i in range(1,n+1):
    u=i/(2**i)
    S=S+u
print("Somme de la série : ",S)
```

### 5.5 La série harmonique et la série harmonique alternée

```
# Série harmonique
n=eval(input("Choisissez un entier n : "))
S=0
for i in range(1,n+1):
    u=1/i
    S=S+u
print("Somme de la série : ",S)
```

```
# Série harmonique alternée
from math import*
n=eval(input("Choisissez un entier n : "))
S=0
for i in range(1,n+1):
    if i%2==1:
        u=-1/i
    else:
        u=1/i
    S=S+u
print("Somme des n premiers termes :",S)
print("-Ln(2)=", -log(2))
```

### 5.6 Le problème de Bâle

```
# Série des inverses des carrés
from math import*
n=eval(input("Choisissez un entier n : "))
S=0
for i in range(1,n+1):
    u=1/(i*i)
    S=S+u
print("Somme des n premiers termes :",S)
print(pi*pi/6)
```

### 5.7 Le nombre $e$

```
# Calcul du nombre e
```

```
from math import*
n=eval(input("Valeur de n ? "))
print(factorial(n))
u,s=1,1
for k in range(1,n+1):
    u,s=u/k,s+s+u
print("Somme=",s)
```

## Chapitre 3

# La fonction exponentielle et les fonctions logarithmes

### 1. La fonction exponentielle

#### 1.2 Définition de la fonction exponentielle par Euler

```
# Calcul de exp(x) avec la définition d'Euler
x=eval(input("Valeur de x ? "))
n=eval(input("Valeur de l'entier n ? "))
y=(1+x/n)**n
print("exp(",x,")=",y)
```

```
# Calcul de la dérivée de exp(x)
from math import*
x=eval(input("Choisir x : "))
h=0.001
y=exp(x)
d=(exp(x+h)-exp(x-h))/(2*h)
print("exp(",x,")=",y)
print(" dérivée de exp(x) en x=",x,"=",d)
```

#### 1.4 Une autre définition de exp(x)

```
# Calcul de exp(x) avec la deuxième définition d'Euler
from math import*
# Calcul de la factorielle
def fact(n):
    f=1
    for i in range(2,n+1):
        f=f*i
    return f
# Calcul de exp(x)
x=eval(input("Choisir x : "))
n=eval(input("Valeur de n ? "))
y=1
for i in range(1,n+1):
    y=y+(x**i)/fact(i)
print("Ce programme donne exp(",x,")=",y)
print("D'après Python, exp(",x,")=",exp(x))
```

## 2. Les logarithmes décimaux

### 2.3 La fonction logarithme décimal dans Python

```
#Logarithme décimal d'un nombre positif  
from math import*  
x=eval(input("Valeur de x ? "))  
y=log10(x)  
print("logarithme décimal de ",x,"=",y)
```

### 3. L'algorithme de Briggs

#### 3.2 Un programme Python

```
# Calcul de log10(x) pour 1 < x < 10 par la méthode de Briggs
from math import*
x=eval(input("Choisir un nombre entre 1 et 10: "))
A=1
B=10
lA=0
lB=1
p=0.0000001
# Calculs
while B-x>p:
    if sqrt(A*B)<=x:
        A=sqrt(A*B)
        lA=(lA+lB)/2
    else:
        B=sqrt(A*B)
        lB=(lA+lB)/2
# Résultat
print("Le logarithme décimal de ",x," est ",lB)
```

## 4. Les logarithmes népériens

### 4.3 Un programme pour calculer un encadrement de $\ln(x)$

```

# Calcul approché de ln(x) par la méthode des rectangles
from math import*
def f(x):
    return 1/x
a=1
x=eval(input("Valeur de x ? "))
n=eval(input("Valeur de l'entier n ? "))
long=(x-1)/n
s1=0
s2=0
h=0.001
dr=0
if x<0:
    dr=1
    print("La fonction n'est pas définie.")
if dr==0:
    for i in range(1,n+1):
        x=1+(i-1)*long
        y=f(x)
        s1=s1+long*y
    for i in range(2,n+2):
        x=1+i*long
        y=f(x)
        s2=s2+long*y
    s=(s1+s2 )/2
    ds=abs((s1-s2)/2)
    print("Valeur approchée par défaut du logarithme=",s2)
    print("valeur approchée par excès du logarithme=",s1)
    print("Valeur moyenne du logarithme=",s)
    print("Valeur calculée avec la fonction log() de Python=",log(x))

```

## Chapitre 4

### Dérivation numérique et équations différentielles

#### 1. Dérivée d'une fonction numérique

##### 1.3 Calculs approchés de $f'_d(x)$ et de $f'_g(x)$

```
# Nombre dérivé à droite
from math import*
def f(x) :
    return abs(x*x-1)
x=eval(input("Choisissez x : "))
h=eval(input("Choisissez h : "))
d=(f(x+h)-f(x))/h
print(" Pour x=",x," la dérivée à droite de x prend la valeur ",d)
```

```
# Nombre dérivé à gauche
from math import*
def f(x) :
    return abs(x*x-1)
x=eval(input("Choisissez x : "))
h=eval(input("Choisissez h : "))
d=(f(x)-f(x-h))/h
print("Pour x=",x," la dérivée à gauche de f prend la valeur ",d)
```

## 2. Calcul approché de $f'(x)$ et de $f''(x)$

### 2.2 Calcul d'une valeur approchée de $f'(x)$

```
# Calcul de f'(x)
from math import*
def f(x) :
    return 3*x**2-5*x+4
x=eval(input("Choisissez x : "))
h=eval(input("Choisissez h : "))
d=(f(x+h)-f(x-h))/(2*h)
print("Le nombre dérivé en x=",x," est égal à ",d)
```

### 2.3 Application à la fonction exponentielle

```
# Calcul approchée de la dérivée de la fonction exponentielle
from math import*
x=eval(input("Valeur de x : "))
print("valeur exacte de exp(x) : ",exp(x))
h=0.01
d=(exp(x+h)-exp(x-h))/2/h
print("Valeur approchée de la dérivée : ",d)
```

### 2.5 Calcul approché de $f''(x)$

```
# Calcul de f''(x)
from math import*
def f(x) :
    return x*exp(x)
x=eval(input("Choisissez x : "))
h=eval(input("Choisissez h : "))
d=(f(x+h)+f(x-h)-2*f(x))/(h*h)
print("Le nombre dérivée en x=",x," est égal à ",d)
```

## 4. La méthode d'Euler

### 4.2 Un programme pour calculer $y(x)$

```

# Méthode d'Euler pour résoudre l'équation différentielle  $y'=F(x,y)$ 
# On connaît  $x_0$  et  $y_0=y(x_0)$ . On choisit  $x$  et on calcule une valeur approchée de
#  $y(x)$  en choisissant un entier  $n$  pour définir un pas  $h$ .
from math import*
def F(x,y):
    return 1/exp(x)-y
x0=eval(input("Valeur de x0 ? "))
y0= eval(input("Valeur de y0 ? "))
x=eval(input("Valeur de x ? "))
n=eval(input("Valeur de l'entier n ? "))
h=(x-x0)/n
x,y=x0,y0
for i in range(1,n+1):
    y=y+h*F(x,y)
    x=x+h
print("y=",y)
print("Valeur exacte de y :", x/exp(x))

```

### 4.4 Construction d'une fonction Euler( $x_0,y_0,x,n$ )

```

# Méthode d'Euler. On connaît  $x_0$  et  $y_0=y(x_0)$ . On choisit  $x$  et on calcule une
# valeur approchée de  $y(x)$  en choisissant un entier  $n$  pour définir le pas  $h$ .
from math import*
def F(x,y):
    return 1/exp(x)-y
def Euler(x0,y0,x,n):
    h=(x-x0)/n
    x,y=x0,y0
    for i in range(1,n+1):
        y=y+h*F(x,y)
        x=x+h
    return y
# Entrée des données
x0=eval(input("Valeur de x0 ? "))
y0=eval(input("Valeur de y0 ? "))
x=eval(input("Valeur de x ? "))
n=eval(input("Valeur de l'entier n ? "))
# Résultats
y=Euler(x0,y0,x,n)
print("y=",y)

```

#### 4. 5 Un cas particulier : l'équation différentielle $y'=y$

# On partage  $[a;b]$  en 10 parties égales et on calcule les 10 valeurs de  $y$  par  
# la méthode d'Euler. On prend  $x_0=a$ . On connaît  $y_0=y(x_0)$ .

```
from math import*
def F(x,y):
    return y
def Euler(x0,y0,x,n):
    h=(x-x0)/n
    x,y=x0,y0
    for i in range(1,n+1):
        y=y+h*F(x,y)
        x=x+h
    return y
# Entrée des données
a=eval(input("Valeur de a ? " ))
ya=eval(input("Valeur de y(a)? "))
b=eval(input("Valeur de b ? " ))
n=3000
k=(b-a)/10
# Calculs des y et affichage des résultats
for j in range(0,10):
    x0=a+j*k
    y0=ya
    x=a+(j+1)*k
    y=Euler(x0,y0,x,n)
    print("x=",x," y=",y)
ya=y
```

## 5. Les méthodes de Runge-Kutta

### 5.2 Cas d'une équation différentielle du 1<sup>er</sup> ordre

```
# Méthode de Runge-Kutta pour résoudre une 'équation différentielle du 1er ordre
# On choisit x et on calcule une valeur approchée de
# y(x) en choisissant un entier n pour définir un pas h
from math import*
def F(x,y):
    return 1/exp(x)-y
x0=eval(input("Valeur de x0 ? "))
y0= eval(input("Valeur de y0 ? "))
x=eval(input("Valeur de x ? " ))
n=eval(input("Valeur de l'entier n ? "))
h=(x-x0)/n
x,y=x0,y0
for i in range(1,n+1):
    k1=h*F(x,y)
    k2=h*F(x+h/2,y+k1/2)
    k3=h*F(x+h/2,y+k2/2)
    k4=h*F(x+h,y+k3)
    y=y+(k1+2*k2+2*k3+k4)/6
    x=x+h
print("y=",y)
```

### 5.3 Cas d'une équation différentielle du second ordre

```
# Méthode de Runge-Kutta pour résoudre une équation différentielle d'ordre 2
# On choisit x et on calcule une valeur approchée de y(x) en choisissant
# un entier n pour définir un pas h
from math import*
def F(x,y,dy):
    return 3*dy+2-6*x
x0=eval(input("Valeur de x0 ? "))
y0= eval(input("Valeur de y0 ? "))
dy0=eval(input("Valeur de y'0 ? " ))
x=eval(input("Valeur de x ? " ))
n=eval(input("Valeur de l'entier n ? "))
h=(x-x0)/n
x,y,dy=x0,y0,dy0
for i in range(1,n+1):
    k1=h*F(x,y,dy)
    k2=h*F(x+h/2,y+h*dy/2,dy+k1/2)
    k3=h*F(x+h/2,y+h*dy/2+h*k1/4,dy+k2/2)
    k4=h*F(x+h,y+h*dy+h*k2/2,dy+k3)
    y=y+h*(dy+(k1+k2+k3)/6)
```

```
dy=dy+(k1+2*k2+2*k3+k4)/6  
x=x+h  
print("y=",y)
```

## Chapitre 5

# Résolution approchée des équations $f(x)=0$

### 1. La recherche d'une solution par dichotomie

#### 1. 2. Deux programmes

```
# Résolution de l'équation  $f(x)=0$  par dichotomie
from math import*
def f(x):
    return x*x*x-6*x*x+10*x-4
a=eval(input("Valeur initiale de a ? "))
b=eval(input("Valeur initiale de b ? "))
e=0.000001
n=1
if f(a)*f(b)<=0:
    while b-a>=e:
        c=(a+b)/2
        n=n+1
        if f(a)*f(c)<=0:
            b=c
        else:
            a=c
    print("Résultats :")
    print("Dernière valeur calculée de a= ",a)
    print("Dernière valeur calculée de b= ",b)
    print("Nombre des itérations = ",n)
else:
    print("Pas de solution dans cet intervalle.")
```

```
# Résolution de l'équation  $f(x)=0$  par dichotomie
from math import*
def f(x):
    return x*x*x-6*x*x+10*x-4
a=eval(input("Valeur initiale de a ?"))
b=eval(input("Valeur initiale de b ?"))
n=eval(input("Combien d'itérations ?"))
if f(a)*f(b)<=0:
    for i in range(1,n+1):
        c=(a+b)/2
        if f(a)*f(c)<=0:
            b=c
        else:
            a=c
```

```
print("Dernière valeur calculée de a: ",a)
print("Dernière valeur calculée de b: ",b)
else:
    print("Pas de solution dans cet intervalle.")
```

## 2. La méthode des approximations successives

### 2.1 Historique

```
# Le calcul de sin 1° par al-Kashi
def g(x):
    return (4*x*x*x+0.0523336)/3
n=10
x1=eval(input("Valeur de x1 ? "))
print("Valeurs approchées calculées successivement :")
for i in range(1,n):
    x1=0
    x2=g(x1)
    print("x",i+1,"= ",x2)
    x1=x2
```

### 2.2 Etude d'un exemple

```
def f(x):
    return x*x*x+2*x-2
def g(x):
    return 1-x*x*x/2
x1=0.5
for i in range (1,100):
    x2=g(x1)
    x1=x2
print("Valeur approchée de r=",x2,"Valeur approchée de g(r)=",g(x2))
```

```
# Méthode des approximations successives
from math import*
def f(x):
    return x-cos(x)
def g(x):
    return cos(x)
x1=eval(input("Valeur de x1 ? "))
n=eval(input("Combien d'itérations? "))
for i in range(1,n+1):
    x2=g(x1)
    x1=x2
print("Valeur de la solution r :", x2)
print("Valeur de g(r): ", g(x2))
print("Valeur de f(r): ", f(x2))
```

```
# Méthode des approximations successives
from math import*
def f(x):
```

```
    return x-cos(x)
def g(x):
    return cos(x)
x1=eval(input("Valeur de x1 ? "))
e=0.000001
dr=0
compteur=1
while dr==0:
    compteur=compteur+1
    x2=g(x1)
    d=abs(x2-x1)
    if d<e:
        dr=1
    else:
        x1=x2
    if compteur>100:
        dr=2
if dr==1:
    print("Vérification")
    print("Valeur r de la solution :", x2)
    print("Valeur de g(r): ", g(x2))
    print("Valeur de f(r): ", f(x2))
if dr==2:
    print("Trop d'itérations")
```

### 3. La méthode de Newton

#### 3.2 Deux programmes

```
# Méthode de Newton appliquée au polynôme  $x^3-2x-5=0$ 
from math import*
def f(x):
    return x*x*x-2*x-5
def g(x):      # g est la dérivée de la fonction f
    return 3*x*x-2
x1=eval(input("Valeur de x1 ? "))
n=eval(input("Combien d'itérations? "))
for i in range(1,n+1):
    x2=x1-f(x1)/g(x1)
    x1=x2
print("Valeur r de la solution :", x2)
print("Valeur de P(r): ", f(x2))
```

```
# Méthode de Newton
from math import*
def f(x):
    return x-cos(x)
def g(x):      # g est la dérivée de la fonction f
    return 1+sin(x)
x1=eval(input("Valeur de x1 ? "))
dr=0          # la variable dr est un drapeau qui vaut 0 ou 1 ou 2 selon les cas
compteur=1    # la variable compteur indique le nombre des itérations
while dr==0:
    compteur=compteur+1
    x2=x1-f(x1)/g(x1)
    d=abs(x2-x1)
    if d<0.000001:
        dr=1    # cette valeur du drapeau détermine l'arrêt du programme
    else:
        x1=x2
    if compteur>100:
        dr=2    # après 100 itérations, cette valeur du drapeau stoppe le programme
if dr==1:
    print("Vérification")
    print("Valeur r de la solution :", x2)
    print("Valeur de f(r): ", f(x2))
if dr==2:
    print("Trop d'itérations")
```

# Chapitre 6

## Calcul infinitésimal et intégration numérique

### 1. Longueur d'un arc de courbe

#### 1.2 Un programme de calcul

```
# Calcul de la longueur d'un arc de parabole
from math import*
def f(x):
    return x*x
a=eval(input("Valeur de l'abscisse du point A ?"))
b=eval(input("Valeur de l'abscisse du point B ?"))
n=eval(input("Valeur de l'entier n ?"))
d=(b-a)/n
L=0
for i in range(0,n):
    x1, x2=a+i*d, x1+d
    y1, y2=f(x1), f(x2)
    L=L+sqrt((x2-x1)*(x2-x1)+(y2-y1)*(y2-y1))
print("Longueur de l'arc AB=",L)
```

#### 1.3 Application : calcul du nombre $\pi$

```
# Calcul du nombre pi
from math import*
def f(x):
    return sqrt(1-x*x)
a,b=0,1
n=eval(input("Valeur de l'entier n ?"))
d=(b-a)/n
L=0
for i in range(0,n):
    x1=a+i*d
    x2=x1+d
    y1=f(x1)
    y2=f(x2)
    L=L+sqrt((x2-x1)*(x2-x1)+(y2-y1)*(y2-y1))
print("Valeur de pi :",2*L)
*
```

## 2. Aire du disque et calcul de $\pi$

### 2.3 Avec le calcul infinitésimal

```
# Aire d'un disque de rayon R
from math import*
R=eval(input("Valeur du rayon ? "))
n=eval(input("Valeur de l'entier n ? "))
h=R/n
# Aire par défaut d'un quart de disque
A1=0
for i in range(1,n):
    r=sqrt(R*R-i*i*h*h)
    A1=A1+r*h
# Aire par excès d'un quart de disque
A2=0
for j in range(0,n):
    r=sqrt(R*R-j*j*h*h)
    A2=A2+r*h
# Aire du disque
print("L'aire du disque est comprise entre ",4*A1," et ",4*A2)
print("La valeur moyenne de cette aire est : ",2*(A1+A2))
```

### 2.4 Calcul de $\pi$ par la méthode de Monte-Carlo

```
# Calcul de l'aire d'un disque par une méthode de Monte-Carlo
from math import*
from random import*
R=1
N=10000
n=0
for i in range(1,N+1):
    x=randint(0,R)
    y=randint(0,R)
    if x*x+y*y<=R*R:
        n=n+1
A=R*R*n/N
print("Il est tombé",n,"points sur",N,"dans le quart de disque.")
print("Valeur estimée de l'aire du quart de disque : ",A)
print("Valeur estimée de l'aire du disque : ",4*A)
print("Valeur estimée du nombre pi :", 4*A/R/R)
```

La figure qui suit montre les points tombés dans le quart de disque et eux seuls pour  $R=1$  et  $N=1500$ .

### 3. Volume d'une boule

#### 3.3 Le programme de calcul

Le programme de calcul du volume d'une boule de rayon  $R$  est le suivant :

```
# Volume d'une boule de rayon R
from math import*
R=eval(input("Valeur du rayon ? "))
n=eval(input("Valeur de l'entier n ? "))
h=R/n
# Volume V1 par défaut d'une demi-boule
V1=0
for i in range(1,n):
    r=sqrt(R*R-i*i*h*h)
    V1=V1+pi*r*r*h
# Volume V2 par excès d'une demi-boule
V2=0
for j in range(0,n):
    r=sqrt(R*R-j*j*h*h)
    V2=V2+pi*r*r*h
# Volume de la boule complète
print("Le volume de cette boule est compris entre ",2*V1," et ",2*V2)
print("Volume de la boule complète : ",V1+V2)
```

## 4. Intégration approchée par la méthode des rectangles

### 4.3 Un programme pour calculer l'intégrale d'une fonction continue

```
# Intégration par la méthode des rectangles
from math import*
def f(x):
    return (3**x)*exp(x)
a=eval(input("Valeur de a ?"))
b=eval(input("Valeur de b ?"))
n=eval(input("Valeur de l'entier n ?"))
long=(b-a)/n
s=0
M=1E-30
h=0.001
for i in range(1,n+1):
    x=a+(i-1)*long
    y=f(x)
    d=(f(x+h)-f(x-h))/(2*h)
    if abs(d)>M:
        M=abs(d)
    s=s+long*y
ds=(m1*(b-a)*(b-a))/(2*n)
print("intégrale I=",s)
print("erreur <=",ds)
```

### 4.4 Le cas des fonctions monotones

```
# Intégration. Méthode des rectangles pour une fonction continue monotone
from math import*
def f(x):
    return x*x
a=eval(input("Valeur de a ?"))
b=eval(input("Valeur de b ?"))
n=eval(input("Valeur de l'entier n ?"))
long=(b-a)/n
s1=0
s2=0
h=0.001
for i in range(1,n+1):
    x=a+(i-1)*long
    y=f(x)
    s1=s1+long*y
for i in range(2,n+2):
```

```
x=a+i*long
y=f(x)
s2=s2+long*y
s=(s1+s2)/2
ds=abs((s1-s2)/2)
print("intégrale=",s)
print("erreur <=",ds)
```

## 5. Intégration approchée par la méthode des trapèzes

### 5.3 Un programme pour calculer l'intégrale d'une fonction continue

```

# Intégration par la méthode des trapèzes
from math import*
def f(x):
    return (3**x)*exp(x)
a=eval(input("Valeur de a ?"))
b=eval(input("Valeur de b ?"))
n=eval(input("Valeur de l'entier n ?"))
long=(b-a)/n
s=0
M=1E-30
h=0.001
for i in range(1,n+1):
    x1=a+(i-1)*long
    x2=x1+long
    y1=f(x1)
    y2=f(x2)
    d=(f(x2+2*h)+f(x2-2*h)-2*f(x2)) # dérivée seconde de f
    if abs(d)>M:
        M=abs(d)
    s=s+(y1+y2)*long/2
ds=(M*(b-a)**3)/(12*n*n)
print("intégrale I=",s)
print("erreur <=",ds)

```

## 6. Intégration approchée par la méthode de Simpson

### 6.2 La méthode de Simpson

```
# Intégration par la méthode de Simpson
from math import*
def f(x):
    return (3**x)*exp(x)
a=eval(input("Valeur de a ?"))
b=eval(input("Valeur de b ?"))
n=eval(input("Valeur de l'entier n ?"))
long=(b-a)/2/n
s=0
for i in range(1,2*n,2):
    x1=a+(i-1)*long
    x2=x1+long
    x3=x2+long
    s=s+f(x1)+4*f(x2)+f(x3)
s=s*long/3
print("Valeur approchée de l'intégrale=",s)
```

## 7. Intégration approchée par la méthode de Gauss

### 7.3 Un programme

```
# Intégration par la méthode de Gauss
from math import*
def f(x):
    return (3**x)*exp(x)
a=eval(input("Valeur de a ?"))
b=eval(input("Valeur de b ?"))
x1,x2,x3=-0.932469514,-0.661209386,-0.238619186
listedesx=[x1,x2,x3,-x3,-x2,-x1]
k1,k2,k3=0.171324492,0.360761573,0.467913915
listedesk=[k1,k2,k3,k3,k2,k1]
s=0
for j in range(0,6):
    x=(a+b)/2+listedesx[j]*(b-a)/2
    y=listedesk[j]*f(x)
    ** s=s+y
s=(b-a)*s/2
print("Intégrale cherchée=",s)
```

# Chapitre 7

## Les nombres complexes

### 1. Les nombres complexes dans Python

#### 1.6 Forme trigonométrique d'un nombre complexe

```
# Conversion de la forme algébrique vers la forme trigonométrique  
from cmath import*  
from math import*  
j=complex(0,1)  
print("Entrez z sous la forme x+j*y")  
z=eval(input())  
print("forme trigonométrique de z : ",polar(z))
```

```
# Conversion de la forme trigonométrique vers la forme algébrique  
from cmath import*  
from math import*  
j=complex(0,1)  
r=eval(input("Quel est le module de z ? "))  
t=eval(input("Quel est l'argument de z (en radians) ?"))  
z=rect(r,t)  
print("forme algébrique de z : ",rect(r,t))
```

## 2. Résolution dans $\mathbb{C}$ des équations du second degré

### 2.1 Cas d'une équation à coefficients réels

```
# Résolution dans C d'une équation du second degré
# à coefficients réels
from math import*
# Coefficients de l'équation
a=eval(input("Quelle est la valeur de a ? "))
b=eval(input("Quelle est la valeur de b ? "))
c=eval(input("Quelle est la valeur de c ? "))
# Calcul du discriminant
delta=b*b-4*a*c
print("delta=", delta)
# Résultat si delta > 0
if delta > 0:
    x1=(-b+sqrt(delta))/2/a
    x2=(-b-sqrt(delta))/2/a
    print("1ère solution=", x1)
    print("2ème solution=", x2)
# Résultat si delta = 0
if delta == 0:
    x=-b/2/a
    print("Une solution double x=", x)
# Résultat si delta < 0
if delta < 0:
    delta=-delta
    d=sqrt(delta)
    x1=complex(-b/(2*a),d/(2*a))
    x2=complex(-b/(2*a),-d/(2*a))
    print("1ère solution=", x1)
    print("2ème solution=", x2)
```

### 2.2 Cas d'une équation du second degré à coefficients complexes

```
# Résolution dans C d'une équation du second degré à coefficients complexes
from math import*
from cmath import*
j=complex(0,1)
# Coefficients de l'équation
print("On entre les coefficients sous la forme x+j*y.")
a=eval(input("Quelle est la valeur de a ? "))
b=eval(input("Quelle est la valeur de b ? "))
c=eval(input("Quelle est la valeur de c ? "))
# Calcul du discriminant
```

```
delta=b*b-4*a*c
print("delta=", delta)
d=sqrt(delta)
# Résultats
x1=-b/(2*a)+d/(2*a)
x2=-b/(2*a)-d/(2*a)
print("1ère solution=", x1)
print("2ème solution=", x2)
```

### 3. Les suites de nombres complexes

#### 3.3 Convergence d'une suite

```
from math import*
a=1
b=1
for i in range(1,21):
    aa=(a+sqrt(a*a+b*b))/3
    bb=b/3
    print(aa," ",bb)
    a=aa
    b=bb
```

```
from math import*
a=1
b=1
for i in range(1,1000):
    aa=(a+sqrt(a*a+b*b))/3
    bb=b/3
    a=aa
    b=bb
print(aa," ",bb)
```

#### 3.4 Une suite géométrique

1°)

```
from math import*
from cmath import*
j=complex(0,1)
z=complex(2,0)
print(z)
for i in range(1,21):
    zz=(1+j)*z/2
    print("z",i,"=",zz)
    z=zz
```

2°)

```
from math import*
from cmath import*
j=complex(0,1)
q=(1+j)/2
```

```
z0=2
n=eval(input("Valeur de n ? "))
z=(q**n)*z0
print("z",n,"=",z)
```

### 3.5 Représentation graphique d'une suite

1°)

```
from math import*
x=-3
y=4
print("n=",0,"x=-3","y=4","r=",sqrt((-3)*(-3)+4*4))
for k in range(1,10):**
    xx=0.8*x-0.6*y
    yy=0.6*x+0.8*y
    r=sqrt(x*x+y*y)
    print("n=",k,"x=",int(100*xx)/100,"y=",int(100*yy)/100," r=",int(100*r)/100)
    x=xx
    y=yy
```

2°) :

```
from math import*
from cmath import*
j=complex(0,1)
for k in range(1,11):
    z=exp(j*pi/4)+exp(j*pi*k/4)/k
    print("z",k,"=",z)
```

## 4. Aperçu sur les fonctions d'une variable complexe

### 4.2 La fonction $z \rightarrow z+a$

```
# Fonction  $z \rightarrow z+a$ 
from math import*
xA=eval(input("Valeur de xA ? "))
yA=eval(input("Valeur de yA ? "))
xM=eval(input("Valeur de xM ? "))
yM=eval(input("Valeur de yM ? "))
xN=eval(input("Valeur de xN ? "))
yN=eval(input("Valeur de yN ? "))
xP=eval(input("Valeur de xP ? "))
yP=eval(input("Valeur de yP ? "))
xxM,yyM=xM+xA,yM+yA
xxN,yyN=xN+xA,yN+yA
xxP,yyP=xP+xA,yP+yA
print("xM'=",xxM," yM'=",yyM)
print("xN'=",xxN," yN'=",yyN)
print("xP'=",xxP," yP'=",yyP)
```

### 4.5 Les fonctions homographiques complexes

```
# Fonction homographique complexe
from math import*
from cmath import*
j=complex(0,1)
xa=eval(input("partie réelle de a ? "))
ya=eval(input("partie imaginaire de a ? "))
a=complex(xa,ya)
xb=eval(input("partie réelle de b ? "))
yb=eval(input("partie imaginaire de b ? "))
b=complex(xb,yb)
xc=eval(input("partie réelle de c ? "))
yc=eval(input("partie imaginaire de c ? "))
c=complex(xc,yc)
xd=eval(input("partie réelle de d ? "))
yd=eval(input("partie imaginaire de d ? "))
d=complex(xd,yd)
xz=eval(input("partie réelle de z ? "))
yz=eval(input("partie imaginaire de z ? "))
z=complex(xz,yz)
# Calcul de  $z'=f(z)$ 
zz=complex((a*z+b)/(c*z+d))
print("f(z)=",zz)
```

## Chapitre 8

### Éléments de statistiques

#### 1. Les paramètres d'une série statistique

##### 1.1.

```
# Rangement des notes de la série dans un ordre croissant
série=[11,8,17,12,11,6,11,10,12,15,8,8,9,8,11,15,5,5,9,12,13,11,8, 7,5,12,13,13]
N=len(série)
sérieclassée=sorted(série)
print(sérieclassée)
# Note la plus basse, note la plus élevée et étendue de la série
print("note la plus basse :",sérieclassée[0])
print("note la plus haute :", sérieclassée[N-1])
print("étendue de la série :", sérieclassée[N-1]-sérieclassée[0])
```

##### 1.2 Construction du tableau des effectifs

```
# Liste des notes dans un ordre croissant
liste=[5,5,5,6,7,8,8,8,8,8,9,9,10,11,11,11,11,11,12,12,12,12,13,13,13,15,15,17]
print("Liste des notes : ",liste)
# liste des rangs des éléments de liste des notes
liste2=[]
for i in range(0,len(liste)):
    y=liste[i]
    n=liste.count(y)
    liste2=liste2+[n]
# Suppression des doublons dans la liste des notes
copie=[]
for i in range(0,len(liste)):
    if liste[i] not in copie:
        copie.append(liste[i])
print("Les notes suivantes ont été données : ", copie)
# Nouvelle liste montrant les différentes notes et leurs effectifs
for i in range(0,len(copie)):
    y=copie[i]
    for j in range(0,len(liste)):
        rang=liste.index(y)
        print("note=",y," effectif=",liste2[rang])
```

##### 1.3 Calcul de la médiane

```
# Calcul de la médiane de la série
```

```

liste=[5, 5, 5, 6, 7, 8, 8, 8, 8, 8, 9, 9, 10, 11, 11, 11, 11, 11, 12, 12, 12, 12, 13, 13,
13, 15, 15, 17]
N=len(liste)
y=sorted(liste)
print(y)
if N%2==1 :
    m=y[(N-1)//2]
else :
    m=(y[(N//2-1)]+y[N//2])/2
print("médiane : ",m)

```

#### 1.4 Calcul des quartiles Q1 et Q3 et de l'écart interquartile Q3-Q1

```

# Liste des notes
liste=[5,5,5,6,7,8,8,8,8,8,9,9,10,11,11,11,11,11,12,12,12,12,13,13,13,15,15,17]
N=len(liste)
# Recherche des rangs de Q1 et de Q3
if N%4==0:
    i=N//4
    j=3*N//4
else :
    i=1+int(N/4)
    j=1+int(3*N//4)
# Calcul de Q1 et de Q3
Q1=liste[i-1]
print("Q1=",Q1)
Q3=liste[j-1]
print("Q3=",Q3)
print("écart interquartile=",Q3-Q1)

```

#### 1.5 . Calcul de la moyenne

```

# Calcul de la moyenne
liste=[5, 5, 5, 6, 7, 8, 8, 8, 8, 8, 9, 9, 10, 11, 11, 11, 11, 11, 12, 12, 12, 12, 13, 13,
13, 15, 15, 17]
sommeliste=0
for i in liste :
    sommeliste=sommeliste+i
moyenne=round(sommeliste/len(liste),2)
print("moyenne=",moyenne)

```

#### 1.6 Calcul de la variance et de l'écart-type

```

from math import*
liste=[5,5,5,6,7,8,8,8,8,8,9,9,10,11,11,11,11,11,12,12,12,12,13,13,13,15,15,17]
# Calcul de la moyenne
sommeliste=0

```

```
for i in liste :
    sommeliste=sommeliste+i
    m=round(sommeliste/len(liste),2)
# Calcul de la variance V et de l'ecart-type s
a=0
for i in liste :
    a=a+(m-i)**2
    V=a/len(liste)
    s=round(sqrt(V),2)
print("variance de la série :", V)
print("écart-type de la liste des notes=", s)
```

## 2. Covariance et coefficient de corrélation

### 2.3 Un programme de calcul de $\text{cov}(x,y)$ et de $r$

```

# Covariance de 2 séries statistiques
from math import*
n=eval(input("Quel est l'effectif de chaque série ? "))
listex=[]
listey=[]
# Calcul des 2 moyennes
sommex=0
sommey=0
for i in range (1,n+1):
    x=eval(input("Valeur de x ? "))
    listex=listex+[x]
    sommex=sommex+x
for i in range (1,n+1):
    y=eval(input("Valeur de y ? "))
    listey=listey+[y]
    sommey=sommey+y
mx,my=sommex/n, my=sommey/n
# Calcul des 2 variances
a,b=0,0
for i in range(0,n):
    a=a+(mx-listex[i])**2
for i in range(0,n):
    b=b+(my-listey[i])**2
Vx=a/n
Vy=b/n
# Calcul de la covariance
S=0
for i in range(0,n):
    S=S+(listex[i]-mx)*(listey[i]-my)
covxy=S/n
print("cov(x,y)=",covxy)
# Calcul du coefficient de corrélation
r=covxy/(sqrt(Vx)*sqrt(Vy))
print("Coefficient de corrélation : ",r)

```

### 3. Ajustements linéaires et autres

#### 3.2 Ajustement linéaire

```

# Ajustement linéaire
from math import*
n=eval(input("Quel est l'effectif de chaque série ? " ))
listex,listey=[],[]
# Calcul des moyennes et des variances
sommex,sommey=0,0
for i in range (1,n+1):
    x=eval(input("Valeur de x ? "))
    listex=listex+[x]
    sommex=sommex+x
for i in range (1,n+1):
    y=eval(input("Valeur de y ? "))
    listey=listey+[y]
    sommey=sommey+y
mx,my=sommex/n,sommey/n
v,w=0,0
for i in range(0,n):
    v=v+(mx-listex[i])**2
    w=w+(my-listey[i])**2
Vx,Vy=v/n,w/n
# Calcul de la covariance
p=0
for i in range(0,n):
    p=p+(listex[i]-mx)*(listey[i]-my)
cov=p/n
# Calcul des coefficients de l'ajustement
a=cov/Vx
b=my-a*mx
print("a=",a,"    b=",b)

```

#### 3.3 Ajustement par une exponentielle

```

# Ajustement par une exponentielle
from math import*
n=eval(input("Quel est l'effectif de chaque série ? " ))
listex,listey=[],[]
# Calcul des moyennes et des variances
sommex,sommey=0,0
for i in range (1,n+1):
    x=eval(input("Valeur de x ? "))
    listex=listex+[x]
    sommex=sommex+x

```

```

for i in range (1,n+1):
    y=eval(input("Valeur de y ? "))
    y=log(y)
    listey=listey+[y]
    sommey=sommey+y
mx,my=sommex/n,sommey/n
v,w=0,0
for i in range(0,n):
    v=v+(mx-listex[i])**2
    w=w+(my-listey[i])**2
Vx,Vy=v/n,w/n
# Calcul de la covariance
p=0
for i in range(0,n):
    p=p+(listex[i]-mx)*(listey[i]-my)
cov=p/n
# Calcul des coefficients de l'ajustement
a=cov/Vx
b=my-a*mx
b=exp(b)
print("a=",a,"    b=",b)

```

### 3.4 Ajustement par une fonction puissance

```

# Ajustement par une fonction puissance
from math import*
n=eval(input("Quel est l'effectif de chaque série ? " ))
listex,listey=[],[]
# Calcul des moyennes et des variances
sommex,sommey=0,0
for i in range (1,n+1):
    x=eval(input("Valeur de x ? "))
    x=log(x)
    listex=listex+[x]
    sommex=sommex+x
for i in range (1,n+1):
    y=eval(input("Valeur de y ? "))
    y=log(y)
    listey=listey+[y]
    sommey=sommey+y
mx,my=sommex/n,sommey/n
v,w=0,0
for i in range(0,n):
    v=v+(mx-listex[i])**2
    w=w+(my-listey[i])**2
Vx,Vy=v/n,w/n

```

```
# Calcul de la covariance
p=0
for i in range(0,n):
    p=p+(listex[i]-mx)*(listey[i]-my)
cov=p/n
# Calcul des coefficients de l'ajustement
a=cov/Vx
b=my-a*mx
b=exp(b)
print("a=",a,"    b=",b)
```

## Chapitre 9

# Combinatoire et échantillonnage

### Factorielles et combinaisons

#### 1.2 L'invention des factorielles

```
# Calcul d'une factorielle
from math import*
n=eval(input("Choisir l'entier n : "))
F=factorial(n)
print(n,"!=" ,F)
```

#### 1.4 Calcul du nombre $\binom{n}{p}$

```
# Calcul du nombre p parmi n
from math import*
n=eval(input("Choisir l'entier n ; "))
p=eval(input("Choisir l'entier p :"))
if p<=n:
    c=factorial(n)/(factorial(p)*factorial(n-p))
    print("Combinaison de ",p," parmi ",n,"=" ,c)
else:
    print("Calcul impossible. Revoir la valeur de p.")
```

```
# Calcul récursif de la combinaison p parmi n
def combi(n,p):
    if p==1 or p==0:
        return 1
    if p==n+1:
        return 1
    if p>1 and p<=n:
        return combi(n-1,p)+combi(n-1,p-1)
n=int(input("Valeur de n ?"))
p=int(input("Valeur de p ?"))
C=combi(n,p)
print(C)
```

```
# Affichage des premières lignes du triangle de Pascal
def N(L,C):
    if C==1:
```

```
    return 1
    if C==L+1:
        return 1
    if C>1 and C<=L:
        return N(L-1,C)+N(L-1,C-1)
print("Combien de lignes voulez-vous afficher ?")
ligne=int(input())
for i in range(0,ligne+1):
    L=[]
    for j in range(1,i+2):
        L=L+[N(i,j)]
print(L)
```

## 2. Échantillonnage

### 2. . Fabrication expérimentale d'un échantillon

```
# Fabrication d'un échantillon de 3 boules
from random import*
indicateur=[]
boulestirées=0
liste=[]
n=9
k=3
# Création de la liste des indicateurs
for i in range(0,15):
    indiqueur=indiqueur+[0]
# Tirage au sort des boules
j=0
while boulestirées!=k:
    j=j+1
    x=randint(0,n-1)
    if indiqueur[x]==0:
        liste=liste+[x+1]
        indiqueur[x]=1
        boulestirées= boulestirées+1
# Composition de l'échantillon
couleurs=[]
for i in range(0,k):
    if liste[i]<4:
        couleurs=couleurs+["R"]
    if liste[i]>6:
        couleurs=couleurs+["V"]
    if liste[i]>3 and liste[i]<7
        couleurs=couleurs+["B"]
print("Composition de l'échantillon :",couleurs)
```

### 3. Echantillonnage et fréquences

#### 3.2 Intervalle de fluctuation de la fréquence d'un échantillon

```
# Bornes d'un intervalle de fluctuation
from math import*
dr=0
p=eval(input("Valeur connue de la fréquence p ? "))
n=eval(input("Taille n de l'échantillon ? "))
if p<0.2 or p>0.8:
    dr=1
if n<25:
    dr=1
if dr==1:
    print("Les conditions du calcul ne sont pas vérifiées.")
else:
    a=p-1/sqrt(n)
    b=p+1/sqrt(n)
    print("Bornes de l'intervalle de fluctuation de f : ",a," ; ",b)
```

#### 3.3 Estimation de la fréquence d'un caractère dans une population

```
# Bornes d'un intervalle de confiance
from math import*
dr=0
n=eval(input("Taille n de l'échantillon ? "))
f=eval(input("valeur de la fréquence f dans l'échantillon ? "))
if f<0.2 or f>0.8:
    dr=1
if n<25:
    dr=1
if dr==1:
    print("Les conditions du calcul ne sont pas vérifiées.")
else:
    a=f-1/sqrt(n)
    b=f+1/sqrt(n)
    print("Bornes de l'intervalle de confiance : ",a," ; ",b)
```

# Chapitre 10

## Les probabilités

### Les probabilités conditionnelles

#### 1.1 Une simulation pour conjecturer

```
# Fréquence de l'événement B
from random import*
n=0
for i in range(1,10001):
    dé1=randint(1,6)
    dé2=randint(1,6)
    s=dé1+dé2
    if s>7:
        n=n+1
f=n/10000
print("Fréquence de l'événement B=",f)
```

```
# Fréquence de l'événement A inter B
from random import*
n=0
for i in range(1,10001):
    dé1=randint(1,6)
    dé2=randint(1,6)
    s=dé1+dé2
    if s>7 and s%2==1:
        n=n+1
f=n/10000
print("Fréquence de l'événement A inter B=",f)
```

```
# Fréquence de l'événement B sachant A
from random import*
n=0
compteur=0
for I in range(1,10001):
    dé1=randint(1,6)
    dé2=randint(1,6)
    s=dé1+dé2
    if s%2=1:
        n=n+1
    if s>7:
        compteur=compteur+1
f=compteur/n
print("fréquence de l'événement B sachant A=",f)
```

## 2. La formule de Bayes

### 2.3 Une première simulation

```

from random import*
compteur=0
for n in range(1,1000):
    blanche=0
    # Choix de l'urne
    numéro=randint(1,3)
    if numéro==1:
        x=randint(1,3)
        if x<3:
            blanche=1
            compteur=compteur+1
    if numéro==2:
        x=randint(1,4)
        if x<4:
            blanche=1
            compteur=compteur+1
    if numéro==3:
        x=randint(1,4)
        if x<3:
            blanche=1
            compteur=compteur+1
print("Une blanche a été sortie dans ",compteur, " cas sur 1000.")

```

### 2.4 Une deuxième simulation

```

from random import*
compteur1=0
compteur2=0
for n in range(1,1000):
    blanche=0
    # Choix de l'urne
    numéro=randint(1,3)
    if numéro==1:
        x=randint(1,40)
        if x<31:
            blanche=1
            compteur1=compteur1+1
    if numéro==2:
        x=randint(1,40)
        if x<21:
            blanche=1

```

```
compteur2=compteur2+1
f=compteur1/(compteur1+compteur2)
print("compteur1=",compteur1)
print("total des boules blanches sorties= ",compteur1+compteur2)
print("probabilité que la boule blanche vienne de l'urne n°1 :",f)
```

### 3. Espérance et écart-type d'une variable aléatoire discrète

#### 3.5 Un programme pour calculer $E(X)$ , $V(X)$ et $\sigma(X)$

```
# Espérance et écart-type d'une variable aléatoire .
from math import*
n=eval(input("Combien de valeurs pour X ?"))
E=0
V=0
for k in range(1,n+1):
    print("Valeur numéro ",k)
    x=eval(input())
    p=eval(input("Probabilité de cette valeur ?"))
    E=E+p*x
    V=V+p*x*x
V=V-E*E
S=sqrt(V)
print("espérance=",E)
print("écart(type=",S)
```

## 4. La loi binomiale

### 4.4 Un programme pour calculer $P(X=k)$

```
# Loi binomiale. Calcul de  $P(X=k)$ 
from math import*
def combi(k,n):
    c=factorial(n)/(factorial(k)*factorial(n-k))
    return c
n=eval(input("Valeur de n ?"))
p=eval(input("Valeur de p ?"))
q=1-p
k=eval(input("Valeur de k ?"))
c=combi(k,n)
proba=c*(p**k)*(q**(n-k))
print("P(X=",k,")=",proba)
```

### 4.5 Un programme pour calculer $P(X \leq k)$

```
# Loi binomiale. Calcul de  $P(X \leq k)$ 
from math import*
def combi(k,n):
    c=factorial(n)/(factorial(k)*factorial(n-k))
    return c
n=eval(input("Valeur de n ?"))
p=eval(input("Valeur de p ?"))
q=1-p
k=eval(input("Valeur de k ?"))
proba=0
for j in range(1,k+1):
    c=combi(k,n)
    proba=proba+c*(p**k)*(q**(n-k))
print("P(X<=",k,")=",proba)
```

## 5. La loi de Poisson

### 5.2 Expression de la loi de Poisson

```
# Loi de Poisson. Calcul des probabilités P(X=k)
m=eval(input("Valeur de m ?"))
k=eval(input("Valeur de k ?"))
from math import*
F=factorial(k)
proba=(m**k)*exp(-m)/F
print("P(X)=" ,proba)
```

```
# Loi de Poisson. Calcul des probabilités P(X<=k)
m=eval(input("Valeur de m ?"))
k=eval(input("Valeur de k ?"))
from math import*
proba=0
for j in range(1,k+1):
    F=factorial(j)
    p=(m**j)*exp(-m)/F
    proba=proba+p
print("P(X)<=" ,proba)
```

## 6. Les variables aléatoires continues

### 6.2 Qu'est-ce qu'une variable aléatoire continue ?

```
from random import*
a=0
b=20
compteur=0
n=10000
for i in range(1,n+1):
    x=uniform(a,b)
    if x<=5:
        compteur=compteur+1
print("Le compteur marque ",compteur)
```

## 7. La loi exponentielle

### 7.4 Calcul de la probabilité $P(\alpha < X < \beta)$

```
# Loi exponentielle. Calcul de  $P(a < X < b)$ 
from math import *
L=eval(input("Valeur du paramètre lambda ?"))
a=eval(input("Valeur de a ?"))
b=eval(input("Valeur de b ?"))
proba=exp(-L*a)-exp(-L*b)
print("probabilité demandée=",proba)
```

## 8. La loi normale

### 8.3 Calcul de $P(X < a)$

```
# Loi normale. Calcul de  $P(X \leq a)$ 
from math import*
m=eval(input("Valeur de la moyenne m ?"))
s=eval(input("Valeur de l'écart-type s ?"))
a=eval(input("Valeur du nombre a ?"))
z=(a-m)/s
if z<-4:
    p=0
if z>4:
    p=1
if z>-4 and z<4:
    n=25
    c=z
    t=z
    for k in range(1,n+1):
        c=-c*(2*k-1)*z*z/(2*k*(2*k+1))
        t=t+c
    p=0.5+t/sqrt(2*pi)
p=int(100000*p)/100000
print("P(X<,"a,")=",p)
```

### 8.4 Calcul inverse

```
# Loi normale. Calcul de a pour avoir  $P(X < a)=p$ , p étant connu
from math import*
m=eval(input("Valeur de la moyenne m ?"))
s=eval(input("Valeur de l'écart-type s ?"))
p=eval(input("Valeur de la probabilité p ?"))
h=-4
b=4
n=25
for i in range(1,101):
    z=(h+b)/2
    c=z
    t=z
    for k in range(1,n+1):
        c=-c*(2*k-1)*z*z/(2*k*(2*k+1))
        t=t+c
    q=0.5+t/sqrt(2*pi)
    if abs(p-q)<=0.000001:
        a=z*s+m
    if q>p:
```

```
b=z  
if q<p:  
    h=z  
a=z*s+m  
print("a=",a)
```

## 9. Loi normale et jugements statistiques

### 9.1 Intervalle de fluctuation d'une moyenne

```
# Intervalle de fluctuation des moyennes d'un échantillon
from math import*
n=eval(input("Taille de la population ? "))
m=eval(input(" Moyenne m de la population ? "))
s=eval(input(" Ecart-type s de la population ? "))
a=m-1.96*s/sqrt(n)
b=m+1.96*s/sqrt(n)
print("intervalle de fluctuation de la moyenne de l'échantillon ",a, " ", b)
```

### 9.2 Intervalle de fluctuation d'une fréquence

```
# Intervalle de fluctuation d'une fréquence f
from math import*
n=eval(input("Taille de l'échantillon ? "))
p=eval(input("Fréquence dans la population de la propriété étudiée ? "))
m=p-1.96*sqrt(p*(1-p))/sqrt(n)
M=p+1.96*sqrt(p*(1-p))/sqrt(n)
print("intervalle de fluctuation de f [" ,m," ; ",M,"])
```

### 9.3 Intervalle de confiance d'une moyenne

```
# Intervalle de confiance d'une moyenne m
from math import*
n=eval(input("Quelle est la taille de l'échantillon ? "))
m=eval(input("Quelle est la moyenne de l'échantillon ? "))
s=eval(input("Quel est l'écart-type de l'échantillon ? "))
mu1=m-1.96*s/sqrt(n)
mu2=m+1.96*s/sqrt(n)
print("Bornes de l'intervalle de confiance de m : ",mu1," et ",mu2)
```

### 9.4 Intervalle de confiance d'une fréquence

```
# Intervalle de confiance d'une fréquence p
from math import*
n=eval(input("Taille de l'échantillon ? "))
f=eval(input("Fréquence dans l'échantillon de la propriété étudiée ? "))
m=p-1.96*sqrt(p*(1-p))/sqrt(n)
M=p+1.96*sqrt(p*(1-p))/sqrt(n)
print("Bornes de l'intervalle de confiance de p : ",m," et ",M)
```

# Chapitre 11

## Arithmétique et cryptographie

### 1. La division euclidienne des entiers

```
# Quotient et reste euclidiens de 2 entiers naturels
a=eval(input("Valeur de a ? " ))
b=eval(input("Valeur de b ? " ))
q=a//b
r=a%b
print("quotient euclidien : ",q)
print("reste euclidien : ",r)
```

### 1.2 La division euclidienne des entiers relatifs

```
# Division euclidienne dans Z
a=eval(input("Choisir a : "))
b=eval(input("Choisir b : "))
if a>=0 and b>0:
    q=a//b
    r=a%b
if a>=0 and b<0:
    q=-(a//(-b))
    r=a%(-b)
if a<=0 and b>0:
    qq=(-a)//b
    rr=(-a)%b
    if rr==0:
        q=-qq
        r=0
    else:
        q=-qq-1
        r=b-rr
if a<=0 and b<0:
    qq=(-a)//(-b)
    rr=(-a)%(-b)
    if rr==0:
        q=qq
        r=0
    else:
        q=qq+1
        r=-b-rr
print("q=",q)
print("r=",r)
```

```
print("Vérification :",b*q+r)
```

## 2. Diviseurs d'un entier naturel

### 2.1 Recherche des diviseurs d'un entier naturel

```
# Liste des diviseurs d'un entier naturel n
from math import *
n=eval(input("Choisissez un entier plus grand que 1 : "))
nombrediviseurs=2
d=2      # d=2 est le premier diviseur de n possible
dr=0     # dr est un drapeau qui vaut 0 tant que d ne divise pas n;
print("Liste des diviseurs :")
print ("1 et ", n)
while d<=sqrt(n):
    r=n%d      # On cherche le reste de la division euclidienne de n par d
    if r==0 and d!=sqrt(n) :
        print(d, " et ", n//d)
        dr=1      # dr=1 quand d divise n
        nombrediviseurs=nombrediviseurs+2
    elif r==0 and d==sqrt(n) : # Si n est un carré, sqrt(n) est un diviseur de n ;
        print(d)
        nombrediviseurs=nombrediviseurs+1
        dr=1
    d=d+1
if dr==1 :
    print (n, " possède ", nombrediviseurs, " diviseurs.")
if dr==0 :
    print (n, " est un nombre premier.")
```

```
# Liste des diviseurs d'un entier naturel n
from math import *
n=eval(input("Choisissez un entier n : "))
Ldiv=[]
Ldiv1=[1]
Ldiv2=[n]
d=2
dr=0      # dr est un drapeau qui vaut 0 tant qu'on n'a
          # pas encore trouvé de diviseurs propres de n
while d<=sqrt(n):
    r=n%d
    if r==0 and d!=sqrt(n) :
        Ldiv1=Ldiv1+[d]
        Ldiv2=[n//d]+Ldiv2
        dr=1      # dr=1 si d divise n
    elif r==0 and d==sqrt(n) : # Si n est un carré, d est un diviseur de n
        Ldiv1=Ldiv1+[d]
        dr=1
```

```
d=d+1
Ldiv=Ldiv1+Ldiv2
print(Ldiv)
```

## 2.2 Somme des diviseurs propres d'un entier

```
# Somme des diviseurs propres d'un entier
n=eval(input("Choisissez un entier : "))
somme=1
fin=n//2
for i in range(2,fin+1) :
    if N%i==0 :
        print(i)
        somme=somme+i
print("Somme des diviseurs de ", n," = ",somme)
```

```
def somdiviseurs(n) :
    somme=1
    fin=n//2
    for i in range(2,fin+1) :
        if n%i==0 :
            somme=somme+i
    return somme
n=eval(input(« Choisissez un entier : »))
s=somdiviseurs(n)
print("Somme des diviseurs propres de ",n," = ",s)
```

## 2.3. Les nombres parfaits

```
# Liste des nombres parfaits compris entre 1 et 10000
for j in range(1,10001):
    def somdiviseurs(n) :
        somme=1
        fin=n//2
        for i in range(2,fin+1) :
            if N%i==0 :
                somme=somme+i
        return somme
    s=somdiviseurs(j)
    if s==j :
        print(j)
```

### 3. Les nombres premiers

#### 3.3 Comment savoir si un entier donné est premier ?

```
# L'entier n est-il premier ?
from math import*
n=eval(input("Choisissez un entier plus grand que 1 :"))
d=2
dr=0
while d<=sqrt(n):
    r=n%d
    if r==0 :
        dr=1
        d=d+1
if dr==0 :
    print(n, " est un nombre premier.")
if dr==1 :
    print(n, "n'est pas premier")
```

#### 3.4 Des listes de nombres premiers

```
# Liste des nombres premiers compris entre a et b
from math import*
a=eval(input("Valeur de a ? "))
b=eval(input("Valeur de b ? "))
liste=[]
s=int(sqrt(b)+0.5)
while a!=b+1:
    d=1
    compteur=0
    while d!=a+1:
        if a%d==0:
            compteur=compteur+1
        d=d+1
    if compteur==2:
        liste=liste+[a]
    a=a+1
print(liste)
print("Il y a ",len(liste)," nombres premiers dans cet intervalle.")
```

#### 3.5 La conjecture des nombres premiers jumeaux

```
# Recherche des nombres premiers jumeaux dans un intervalle [a;b]
from math import*
a=eval(input("Valeur de a ? "))
b=eval(input("Valeur de b ? "))
liste=[]
```

```

s=int(sqrt(b)+0.5)
while a!=b+1:
    d=1
    compteur=0
    while d!=a+1:
        if a%d==0:
            compteur=compteur+1
        d=d+1
    if compteur==2:
        liste=liste+[a]
    a=a+1
n=len(liste)
for i in range(0,n-1):
    if liste[i+1]-liste[i]==2:
        print(liste[i+1],"et", liste[i])

```

### 3.7 La conjecture de Goldbach

```

# Conjecture de Goldbach
from math import*
a=eval(input("Valeur de a ? "))
aa=a
b=eval(input("Valeur de b ? "))
liste=[]
s=int(sqrt(b)+0.5)
while a!=b:
    d=1
    compteur=0
    while d!=a+1:
        if a%d==0:
            compteur=compteur+1
        d=d+1
    if compteur==2:
        liste=liste+[a]
    a=a+1
long=len(liste)
for n in range(aa,b+1):
    for i in range(0,long-1):
        for j in range(0,long-1):
            if i<=j and n==liste[i]+liste[j]:
                print(n,"=",liste[i], "+",liste[j])

```

## 4. PGCD de deux entiers

### 4.1. Historique

```
# Calcul d'un pgcd par différences successives
a=eval(input("Valeur de a ? "))
b=eval(input("Valeur de b ? "))
while a!=b:
    d=abs(b-a)
    b=a
    a=d
print("pgcd=",d)
if d==1:
    print("Les deux entiers sont premiers entre eux.")
```

### 2. La méthode des divisions successives

```
# Calcul d'un pgcd par divisions successives
a=eval(input("Choisissez l'entier a :"))
b=eval(input("Choisissez l'entier b :"))
r=a%b
while r !=0 :
    a=b
    b=r
    r=a%b
print("Le pgcd cherché est ",b)
if b==1:
    print("Les deux entiers sont premiers entre eux.")
```

```
# Calcul récursif par divisions successives du pgcd de 2 entiers
# Définition de la fonction pgcd
def pgcd(a,b):
    r=a%b
    if r==0 :
        return b
    else :
        return pgcd(b,r)
# Entrée de a et de b et calcul du pgcd
x=eval(input("Choisissez l'entier a :"))
y=eval(input("Choisissez l'entier b ;"))
print("pgcd des 2 nombres =", pgcd(a,b))
if pgcd(a,b)==1:
    print("Les deux entiers sont premiers entre eux.")
```

#### 4.4 La fonction pgcd dans Python

```
from math import*
a=eval(input("valeur de a ? " ))
b=eval(input("valeur de b ? " ))
print("pgcd des 2 nombres=",gcd(a,b))
if gcd(a,b)==1:
    print("Les deux entiers sont premiers entre eux.")
```

## 5. Factorisations d'un entier naturel

### 5.1 Produits de facteurs premiers

```
# Décomposition de l'entier n en un produit de facteurs premiers entre eux
n=eval(input("Entrez l'entier n : "))
d=2
while n>1:
    while n%d==0:
        n=n//d
        print(d)
    d=d+1
```

### 5.2 Décomposition en facteurs premiers et recherche d'un pgcd

```
# m est décomposé en un produit de facteurs 1ers
m=eval(input("Entrez l'entier m : "))
d=2
liste1=[]
while m>1:
    while m%d==0:
        m=m//d
        liste1=liste1+[d]
    d=d+1
# n est décomposé en un produit de facteurs 1ers
n=eval(input("Entrez l'entier n : "))
d=2
liste2=[]
while n>1:
    while n%d==0:
        n=n//d
        liste2=liste2+[d]
    d=d+1
# Résultats
print(liste1)
print(liste2)
```

### 5.3 Une autre méthode de factorisation

```
# Plus grande puissance de 2 qui divise un entier n
n=eval(input("Entrez n : "))
liste=[]
while n%2==0:
    n=n//2
    liste=liste+[2]
```

```
liste=liste+[n]
print(liste)
```

```
# Recherche de x et de y tels que  $n=x^2-y^2$  pour n impair non premier
n=eval(input("Entrez un entier n impair n : "))
r=n%2
dr=0
if r==1:
    p=int((n+1)/2)
    for x in range(0,p):
        for y in range(0,x):
            m=x*x-y*y
            if m==n:
                dr=1
                print(n,"=",x,"au carré -",y,"au carré.")
if dr==0:
    print("ce nombre est premier.")
```

```
# Factorisation d'un entier n
n=eval(input("Entrez n : "))
# Recherche de la plus grande puissance de 2 qui divise n
dr=0
liste=[]
while n%2==0:
    n=n/2
    liste=liste+[2]
liste=liste+[n]
print(liste)
# Factorisation du facteur de n qui est impair
r=n%2
if r==1:
    p=int((n+1)/2)
    for x in range(0,p):
        for y in range(0,x):
            m=x*x-y*y
            if m==n:
                print(n,"=",x+y,"x",x-y)
                dr=1
if dr==0:
    print(n)
```

#### 5.4 La méthode de Fermat

```
# Factorisation par la méthode de Fermat
from math import*
n=eval(input("Choisir un entier impair n :"))
```

```
if int(sqrt(n))*int(sqrt(n))==n:          # Cas où n serait un carré parfait
    print(n,"=",int(sqrt(n)), "x", int(sqrt(n)))
else:
    r=n%2
    a=int(sqrt(n))+1
    if r==0:                             # Elimination du cas où n serait pair
        print("Impossible car ",n," est pair.")
    else:
        x=a
        y=sqrt(a*a-n)
        while y-int(y)>0:
            x=x+1
            y=sqrt(x*x-n)
        print(n,"=",int(x+y),"x",int(x-y))
```

## 6. Le théorème de Bezout

### 6.3 Recherche des coefficients de Bezout avec Python

```
# pgcd de 2 entiers et calcul des coefficients de Bezout
a=eval(input("Valeur de a ? "))
b=eval(input("Valeur de b ? "))
# initialisation
e=a
f=b
r=e%f
u=1
v=0
uu=0
vv=1
# calculs et résultats
while r>0:
    q=e//f
    g=e-q*f
    uuu= u-q*uu
    vvv=v-q*vv
    e=f
    f=g
    r=e%f
    u=uu
    v=vv
    uu=uuu
    vv=vvv
print("pgcd=",f)
print("Coefficients de Bezout :", uu," et ",vv)
```

## 7. Introduction aux équations diophantiennes

### 7.2 Un exemple d'équation diophantienne

```
# Résolution de l'équation diophantienne  $11x+8y=79$ 
for x in range(0,42):
    for y in range(0,42):
        if 11*x+8*y==79:
            print("x=",x,"y=",y)
```

### 7.3 Un autre exemple

```
for k in range(-5,6):
    x=4-5*k
    y=3*k-2
    print("x=",x," y=",y)
```

### 7.4 Un programme pour résoudre l'équation $ax+by=c$

```
# Résolution d'une équation diophantienne de degré 1
# Deux fonctions
def pgcd(a,b):
    r=a%b
    if r==0:
        d=b
    else:
        d=pgcd(b,r)
    return d
def bezout(a,b):
    if b==0:
        return (1,0)
    else:
        q=a//b
        r=a%b
        u,v=bezout(b,r)
        return (v,u-q*v)
# Entrée des données
a=eval(input("Valeur de a :"))
b=eval(input("Valeur de b :"))
c=eval(input("Valeur de c :"))
# Résolution de l'équation
d=pgcd(a,b)
r=c%d
if r!=0:
    print("Il n'y a pas de solution.")
else:
```

```

s,t=bezout(a,b)
x,y=s*c//d,t*c//d
print("Solution particulière=(,x,",",y,")")
print("solution générale : ", "x=",x,"+k fois ",b//d," y=",y,"-k fois
",a//d)

```

### 7.5 Une équation diophantienne du second degré

```

# Recherche de quelques solutions de l'équation  $x^2-8y^2=1$ 
for x in range(1,1000):
    for y in range(1,1000):
        s=x*x-8*y*y
        n=(x-1)/2
        T=n*(n+1)/2
        if s==1:
            print("n=",int(n), "x=",x,"y=",y,"T=",int(T))

```

## 8. La congruence des entiers relatifs

### 8.2 Calcul des restes modulo $n$

```
# Reste modulo n d'un entier relatif z
z=eval(input("Choisissez l'entier relatif z :"))
n=eval(input("Choisissez l'entier naturel n :"))
if z>=0:
    r=z%n
else:
    r=abs(z)%n
    r=n-r
print("Le reste est ",r)
```

```
# Reste modulo n de la somme de 2 entiers relatifs
z1=eval(input("Choisissez l'entier relatif z1 :"))
z2=eval(input("Choisissez l'entier relatif z2 :"))
n=eval(input("Choisissez l'entier naturel n :"))
s=z1+z2
a=abs(s)
r=a%n
if s>=0:
    print("Le reste modulo ",n," de leur somme est ",r)
else :
    print("Le reste modulo ",n," de leur somme est ",n-r)
```

```
# Reste modulo n du produit de 2 entiers relatifs
z1=eval(input("Choisissez l'entier relatif z1 :"))
z2=eval(input("Choisissez l'entier relatif z2 :"))
n=eval(input("Choisissez l'entier naturel n :"))
p=z1*z2
a=abs(p)
r=a%n
if p>=0:
    print("Le reste modulo ",n," de leur produit est ",r)
else :
    print("Le reste modulo ",n," de leur produit est ",n-r)
```

### 8.3 Calculs modulo $n$ et calculs dans l'anneau $\mathbb{Z}/n\mathbb{Z}$

```
# Recherche de l'opposé d'un relatif z modulo n
n=eval(input("Choisissez un entier naturel n :"))
z=eval(input("Choisissez l'entier relatif z : "))
r=abs(z)%n
if z>=0:
    print("L'opposé de ",z," est ",n-r)
else:
```

```
print("L'opposé de ",z," est ",r)
```

```
# Recherche de l'inverse d'un entier relatif z non nul modulo n
```

```
n=eval(input("Choisissez un entier naturel n : "))
```

```
z=eval(input("Choisissez un entier relatif z : "))
```

```
dr=0
```

```
for j in range(0,n):
```

```
    r=(j*z)%n*
```

```
    if r==1:
```

```
        dr=1
```

```
        inverse=j
```

```
if dr==1:
```

```
    print("L'inverse de ",z, "modulo" , n,"existe. C'est ",inverse)
```

```
if dr==0:
```

```
    print(z, " n'a pas d'inverse modulo ",n)
```

```
# Puissance d'un élément de  $\mathbb{Z}/n\mathbb{Z}$ 
```

```
n=eval(input("Choisissez l'entier naturel n : "))
```

```
a=eval(input("Choisissez un élément a dans l'anneau  $\mathbb{Z}/n\mathbb{Z}$  :"))
```

```
k=eval(input("Choisissez un entier naturel k comme exposant : "))
```

```
p=a**k
```

```
e=p%n
```

```
print(a, " puissance ",k,"=",e)
```

## 9. Le code secret de Jules Cesar

### 9.2 Les instructions ord() et chr() de Python

```
print("Choisissez une lettre quelconque :")
lettre=input()
code=ord(lettre)
print("Code de la lettre choisie=",code)
```

```
for i in range(65,91) :
    print(chr(i))
```

### 9.3 Un programme pour coder un texte

```
# Coder un texte avec la méthode de Jules Cesar
x=input("Quelle phrase voulez-vous coder ?")
n=len(x)
q=eval(input("Quel sera le décalage des lettres ? "))
nouveautexte="" # création de la chaîne qui recevra le texte codé
for i in range(0,n):
    y=ord(x[i])
    y=65+((y+q)-65)%26 # codage de la chaîne x ; % donne le reste
                        # euclidien par 26
    nouveautexte=nouveautexte+chr(y)
print("Voici le texte codé :", nouveautexte)
```

### 9.5 Décodage avec une analyse des fréquences des lettres

```
# Décodage d'un texte codé par la méthode de Jules Cesar
x=" SVUNALTWZQLTLZBPZJVBJOLKLIVUULOLBYL "
n=len(x)
for j in range(1,26) :
    nouveautexte=""
    for i in range(0,n):
        y=ord(x[i])
        y=65+((y+j)-65)%26
        nouveautexte=nouveautexte+chr(y)
    print(nouveautexte)
```

## 10. Le chiffre de Vigenère

### 10.3 Un programme de chiffrement et de déchiffrement

```
# Codage d'un texte par le chiffrement de Vigenère
# Entrée du texte à coder et choix de la clé
texte=input("Entrer le texte à coder en lettres capitales: ")
cle=input("Choisir une clef en lettres capitales : ")
longtexte=len(texte)
longcle=len(cle)
# Ajustement de la clé au texte à coder
while longcle<longtexte:
    cle=cle+cle
    longcle=len(cle)
# Codage du texte
textecode=""
for i in range(longtexte):
    lettre=texte[i]
    if ord(lettre)>64 and ord(lettre)<91:
        decalage=ord(cle[i])-65
        rang=(ord(lettre)-65+decalage)%26
        code=chr(rang+65)
        textecode=textecode+code
# Edition du texte chiffré
print("Texte codé : ",textecode)
```

```
# Décodage d'un texte par le chiffrement de Vigenère
# Entrée du texte codé et choix de la clé
texte=input("Entrer le texte à décoder en lettres capitales: ")
clef=input("Quelle est la clé : ")
longtexte=len(texte)
longclef=len(clef)
# Ajustement de la clé au texte à décoder
while longclef<longtexte:
    clef=clef+clef
    longclef=len(clef)
# Décodage du texte
texte=""
for k in range(longtexte):
    lettre=texte[k]
    if ord(lettre)>64 and ord(lettre)<91:
        decalage=65-ord(clef[k])
        numero=(ord(lettre)-65+decalage)%26
        code=chr(numero+65)
        texte=texte+code
# Edition du texte décodé
```

```
print("Texte décodé : ",textecode
```

## 11. Les codages affines

### 11.2 Un programme pour coder un texte

```
# Codage affine d'un texte
print("Quelle phrase voulez-vous coder ?")
x=input()
n=len(x)
a=eval(input("Choisissez un entier a :"))
b=eval(input("Choisissez un entier b :"))
nouveautexte=""
for i in range(0,n):
    y=ord(x[i])
    y=65+((a*y+b)-65)%26
    nouveautexte=nouveautexte+chr(y)
print("Voici le texte codé :", nouveautexte)
```

### 11.6 Un programme général de décodage

```
# Programme pour le décodage affine
print("Entrez le texte à décoder :")
x=input()
n=len(x)
listedes_a=[1,3,5,7,9,11,15,17,19,21,23,25]
for a in listedes_a:
    for b in range(0,25):
        nouveautexte=""
        for j in range(0,n):
            y=ord(x[j])
            y=65+((a*y+b)-65)%26
            nouveautexte=nouveautexte+chr(y)
        print("a=",a,"b=",b," ",nouveautexte)
```

## 12. Le chiffrement de Hill

### 12.4 Un programme pour coder

Rappelons que le texte à coder doit être écrit en lettres capitales non accentuées et sans espaces.

```
# Entrée du texte à coder
textecod=""
print("Entrez le texte à coder :")
x=input()
longx=len(x)
# Choix de la matrice de codage
a=eval(input("Entrez a :"))
b=eval(input("Entrez b :"))
c=eval(input("Entrez c :"))
d=eval(input("Entrez d :"))
det=a*d-b*c
if det==1:
    for i in range(0,longx-1,2):
        x1=(ord(x[i])-65)%26 # 2 lignes pour calculer les rangs des
        x2=(ord(x[i+1])-65)%26 # lettres du texte à coder
        y1=(a*x1+b*x2)%26 # calcul des rangs des lettres codées
        y2=(c*x1+d*x2)%26
        y1=65+y1 # calcul des codes ASCII des lettres codées
        y2=65+y2
        lettre1=chr(y1) # détermination des lettres codées
        lettre2=chr(y2)
        textecod=textecod+lettre1+lettre2
    print("Texte codé : ",textecod)
else:
    print("Le déterminant n'est pas égal à 1. Choisissez une autre matrice.")
```

### 12.5 Un programme pour décoder

```
# Entrée des coefficients de la matrice de codage
print("Entrez les coefficients de la matrice de codage :")
a=eval(input("Entrez a :"))
b=eval(input("Entrez b :"))
c=eval(input("Entrez c :"))
d=eval(input("Entrez d :"))
det=a*d-b*c
if det!=1:
    print("Le déterminant n'est pas égal à 2. Changez la matrice.")
else:
    aa,bb=d//det,-b//det # calcul des coefficients
    cc,dd=-c//det,a//det # de la matrice inverse
```

```

textedecod=""          # entrée du texte à décoder
print("Entrez le texte à décoder")
x=input()
longx=len(x)
for i in range(0,longx-1,2):
    x1=(ord(x[i])-65)%26      # 2 lignes pour calculer les rangs des
    x2=(ord(x[i+1])-65)%26    # lettres du texte à coder
    y1=(aa*x1+bb*x2)%26      # calcul des rangs des lettres codées
    y2=(cc*x1+dd*x2)%26
    y1=65+y1                # calcul des codes ASCII des lettres codées
    y2=65+y2
    lettre1=chr(y1)          # détermination des lettres codées
    lettre2=chr(y2)
    textedecod=textedecod+lettre1+lettre2
print("Texte décodé : ",textedecod)

```

```

# Entrée du texte à coder
textecod=""
print("Entrez le texte à coder :")
x=input()
longx=len(x)
# Choix de la matrice de codage
a=9
b=4
c=5
d=7
for i in range(0,longx-1,2):
    x1=(ord(x[i])-65)%26      # 2 lignes pour calculer les rangs des
    x2=(ord(x[i+1])-65)%26    # lettres du texte à coder
    y1=(a*x1+b*x2)%26        # calcul des rangs des lettres codées
    y2=(c*x1+d*x2)%26
    y1=65+y1                # calcul des codes ASCII des lettres codées
    y2=65+y2
    lettre1=chr(y1)          # détermination des lettres codées
    lettre2=chr(y2)
    textecod=textecod+lettre1+lettre2
print("Texte codé : ",textecod)

```

```

# Entrée du texte à décoder
textecod=""
print("Entrez le texte à décoder :")
x=input()
longx=len(x)
# Choix de la matrice de décodage
a=5
b=12

```

```
c=15
d=25
for i in range(0,longx-1,2):
    x1=(ord(x[i])-65)%26
    x2=(ord(x[i+1])-65)%26
    y1=(a*x1+b*x2)%26
    y2=(c*x1+d*x2)%26
    y1=65+y1
    y2=65+y2
    lettre1=chr(y1)
    lettre2=chr(y2)
    textecod=textecod+lettre1+lettre2
print("Texte codé : ",textecod)
```

En entrant RBDLLD, on obtient le texte décrypté TURING.

## Chapitre 12

### Matrices 2×2 et matrices 3×3

#### 1. Matrices carrées et applications linéaires

##### 1.3 Représentation d'une matrice avec Python

```
M=[[1,2],[3,4]]
print("Lignes de la matrice : ")
print("1ère ligne : ", M[0])
print("2ème ligne : ", M[1])
print("Coefficients de la matrice : ")
for i in range (0,2):
    for j in range (0,2):
        print("ligne ",i+1,"colonne ",j+1," : ",M[i][j])
```

##### 1.4 Image d'un vecteur par une matrice carrée 2×2 ou 3×3

```
# Coordonnées de l'image d'un vecteur V du plan par une matrice 2x2
from math import*
# Une fonction pour calculer les coordonnées X et Y de l'image de V
def X(x,y):
    return a*x+b*y
def Y(x,y):
    return c*x+d*y
# Entrée des données
a,b=eval(input("Entrez les coefficients a11 et a12 de la matrice : "))
c,d=eval(input("Entrez les coefficients a21 et a22 de la matrice : "))
x,y=eval(input("Entrez les coordonnées x et y du vecteur V : "))
# Calculs et résultats
X,Y=X(x,y),Y(x,y)
print("X=",X)
print("Y=",Y)
```

```
# Coordonnées de l'image d'un vecteur V de l'espace par une matrice 3x3
from math import*
# Une fonction pour calculer les coordonnées X, Y et Z de l'image de V
def X(x,y,z):
    return a*x+b*y+c*z
def Y(x,y,z):
    return d*x+e*y+f*y
def Z(x,y,z):
    return g*x+h*y+i*z
# Entrée des données
```

```
a,b,c=eval(input("Entrez les coefficients a11, a12 et a13 : "))
d,e,f=eval(input("Entrez les coefficients a21, a22 et a23 : "))
g,h,i=eval(input("Entrez les coefficients a31, a32 et a33: "))
x,y,z=eval(input("Entrez les coordonnées x,y,z du vecteur V : "))
# Calculs et résultats
X,Y,Z=X(x,y,z),Y(x,y,z),Z(x,y,z)
print("X=",X)
print("Y=",Y)
print("Z=",Z)
```

Voici un exemple :

```
Entrez les coefficients a11, a12 et a13 : 1,0,0
Entrez les coefficients a21, a22 et a23 : 0,1,0
Entrez les coefficients a31, a32 et a33: 0,0,1
Entrez les coordonnées x, y et z du vecteur V : 4,5,6
X= 4 Y= 5 Z= 6
```

## 2. Opérations sur les matrices

### 2.1 Addition, soustraction et multiplication par un réel

```
# Produit d'une matrice 2x2 par un réel k
A=[[1,2],[3,4]]
k=3
P=[[0,0],[0,0]]
for i in range(0,2):      # i prend donc successivement les valeurs 0 et 1
    for j in range(0,2):  # j prend successivement les valeurs 0 et 1
        P[i][j]=k*A[i][j]
print("P=",k,"A=",P)
```

### 2.4 Un programme pour multiplier des matrices 2x2

```
# Produit d'une matrice-ligne par une matrice-colonne
def lignecol(L,C):
    return L[0]*C[0]+L[1]*C[1]
# Entrées des coefficients
a,b=eval(input("Entrez les coefficients de la matrice-ligne L : "))
c,d=eval(input("Entrez les coefficients de la matrice-colonne C : "))
L=[a,b]
C=[c,d]
# Calculs et résultats
print("X=",lignecol(L,C))
```

```
# Produit de 2 matrices A et B de taille 2
def lignecol(liste1,liste2):
    return liste1[0]*liste2[0]+liste1[1]*liste2[1]
# Entrées des coefficients
a,b=eval(input("Coefficients de la ligne 1 de la matrice A ? "))
c,d=eval(input("Coefficients de la ligne 2 de la matrice A ? "))
e,f=eval(input("Coefficients de la ligne 1 de la matrice B ? "))
g,h=eval(input("Coefficients de la ligne 2 de la matrice B ? "))
# Calculs et résultats
L1=[a,b]
L2=[c,d]
C1=[e,g]
C2=[f,h]
print("p11=",lignecol(L1,C1))
print("p12=",lignecol(L1,C2))
print("p21=",lignecol(L2,C1))
print("p22=",lignecol(L2,C2))
```

```
# Une fonction pour multiplier 2 matrices A et B de taille 2
def prodmat2x2(a,b,c,d,e,f,g,h):
```

```

        return[a*e+b*g,a*f+b*h],[c*e+d*g,c*f+d*h]
# Entrées des coefficients des 2 matrices
a,b=eval(input("Entrez les coefficients de la ligne 1 de la matrice A : "))
c,d=eval(input("Entrez les coefficients de la ligne 2 de la matrice A : "))
e,f=eval(input("Entrez les coefficients de la ligne 1 de la matrice B : "))
g,h=eval(input("Entrez les coefficients de la ligne 2 de la matrice B : "))
# Calculs et résultats
print("Résultat : ",produit(a,b,c,d,e,f,g,h))

```

## 2.5 Un programme pour multiplier des matrices 3×3

```

# Produit de 2 matrices A et B de taille 3
def lignecol(liste1,liste2):
    return liste1[0]*liste2[0]+liste1[1]*liste2[1]+liste1[2]*liste2[2]
# Entrées des coefficients
a,b,c=eval(input("Entrez les coefficients de la ligne 1 de la matrice A : "))
e,f,g=eval(input("Entrez les coefficients de la ligne 2 de la matrice A : "))
h,i,j=eval(input("Entrez les coefficients de la ligne 3 de la matrice A : "))
aa,bb,cc=eval(input("Entrez les coefficients de la ligne 1 de la matrice B : "))
dd,ee,ff=eval(input("Entrez les coefficients de la ligne 2 de la matrice B : "))
gg,hh,ii=eval(input("Entrez les coefficients de la ligne 3 de la matrice B : "))
# Calculs et résultats
L1=[a,b,c]
L2=[e,f,g]
L3=[h,i,j]
C1=[aa,dd,gg]
C2=[bb,ee,hh]
C3=[cc,ff,ii]
print([lignecol(L1,C1),lignecol(L1,C2),lignecol(L1,C3)])
print([lignecol(L2,C1),lignecol(L2,C2),lignecol(L2,C3)])
print([lignecol(L3,C1),lignecol(L3,C2),lignecol(L3,C3)])

```

### 3. Déterminant d'une matrice carrée 2×2 ou 3×3

#### 3.1 Déterminant d'une matrice 2×2

```
# Calcul du déterminant d'une matrice de taille 2x2
from math import*
def det(a,b,c,d):
    return a*d-b*c
# Entrée des coefficients de la matrice A
a,b=eval(input("Entrez a11 et a12 : "))
c,d=eval(input("Entrez a21 et a22 : "))
# Calculs et résultats
D=det(a,b,c,d)
print("Le déterminant de la matrice est égal à ",D)
```

#### 3.2 Déterminant d'une matrice 3×3

```
# Calcul du déterminant d'une matrice 3x3
from math import*
# Fonction pour calculer un déterminant d'ordre 2
def det(a,b,c,d):
    return a*d-b*c
# Entrée des données
a,b,c=eval(input("Entrez les coefficients a11, a12 et a13 : "))
d,e,f=eval(input("Entrez les coefficients a21, a22 et a23 : "))
g,h,i=eval(input("Entrez les coefficients a31, a32 et a33 : "))
# Calculs et résultats
D=a*det(e,f,h,i)-d*det(b,c,h,i)+g*det(b,c,e,f)
print("Le déterminant de la matrice est égal à ",D)
```

## 4. Inversion des matrices carrées 2×2 et 3×3

### 4.2 Inverse d'une matrice carrée 2×2

```
# Inversion d'une matrice carrée A de taille 2
from math import*
# Entrée des coefficients de A
a,b=eval(input("Valeur de a11 et de a12 ? "))
c,d=eval(input("Valeur de a21 et de a22 ? "))
# Calculs et résultats
det=a*d-b*c
if abs(det)<0.001:
    print("Matrice singulière.")
else:
    L1=[d/det,-b/det]
    L2=[-c/det,a/det]
    print(L1,L2)
```

### 4.3 Inverse d'une matrice carrée 3×3

```
# Inversion d'une matrice carrée A de taille 3
from math import*
# Entrée des coefficients de la matrice A
a,b,c=eval(input("Entrez a11,a12 et a13 : "))
d,e,f=eval(input("Entrez a21,a22 et a23 : "))
g,h,i=eval(input("Entrez a31,a32 et a33 : "))
# Calculs et résultats
det=a*e*i+c*d*h+b*f*g-a*f*h-b*d*i-c*e*g
if abs(det)<0.001:
    print("Matrice singulière.")
else:
    L1=[(e*i-f*h)/det,(c*h-b*i)/det,(b*f-c*e)/det]
    L2=[(f*g-d*i)/det,(a*i-c*g)/det,(c*d-a*f)/det]
    L3=[(d*h-e*g)/det,(b*g-a*h)/det,(a*e-b*d)/det]
    print(L1,L2,L3)
```

## 5. Résolution d'un système linéaire d'équations

### 5.3 Un programme pour résoudre les systèmes de 2 équations à 2 inconnues

```
# Résolution d'un système linéaire de 2 équations à 2 inconnues
from math import*
# Entrée des coefficients de la matrice A
a11, a12=eval(input("Valeur de a11 et de a12 ? "))
a21,a22=eval(input("Valeur de a21 et de a 22 ? "))
# Entrée des coefficients b1 et b2
b1,b2=eval(input("Valeur de b1 et de b2 ? "))
# Calcul de la matrice inverse de A et calcul de x et de y
det=a11*a22-a12*a21
if abs(det)>=0.001:
    a,b=a22/det,-a12/det
    c,d=-a21/det,a11/det
    x,y=a*b1+b*b2,c*b1+d*b2
    print("x=",x," y=",y)
n,else:
    print("Système singulier")
```

### 5.4 Un programme pour résoudre les systèmes de 3 équations à 3 inconnues

```
# Résolution d'un système linéaire de 3 équations à 3 inconnues
from math import*
# Entrée des coefficients de la matrice A
a,b,c =eval(input("Entrez a11, a12 et a13 : "))
d,e,f=eval(input("Entrez a21, a22 et a23 : "))
g,h,i=eval(input("Entrez a31, a32 et a33 : "))
# Entrée des coefficients b1,b2 et b3
b1,b2,b3=eval(input("Entrez b1, b2 et b3 : "))
# Calcul du déterminant de la matrice A et des coefficients
# de la matrice inverse A-1
dr=1
det=(a*e*i)+(c*d*h)+(b*f*g)-(a*f*h)-(b*d*i)-(c*e*g)
if abs(det)<0.001:
    dr=0
else:
    aa,bb,cc=(e*i-f*h)/det,(c*h-b*i)/det,(b*f-c*e)/det
    dd,ee,ff=(f*g-d*i)/det,(a*i-c*g)/det,(c*d-a*f)/det
    gg,hh,ii=(d*h-e*g)/det,(b*g-a*h)/det,(a*e-b*d)/det
# Calcul de x, de y et de z
if dr==1:
    x,y,z=aa*b1+bb*b2+cc*b3,dd*b1+ee*b2+ff*b3,gg*b1+hh*b2+ii*b3
    print("x=",x)
    print("y=",y)
```

```
    print("z=",z)
else:
    print("Système singulier")
```

## 6. Puissances d'une matrice 2×2 ou 3×3

### 6.1 Puissance d'une matrice 2×2

```
# Puissance d'une matrice 2x2
from math import*
# Entrée des données
a,b=eval(input("Coefficients de la 1ère ligne : "))
c,d=eval(input("Coefficients de la 2ème ligne : "))
n=eval(input("Valeur de l'exposant ?"))
# Calculs et résultats
A1,B1,C1,D1=1,0,0,1
for i in range(0,n):
    A2=A1*a+B1*c
    B2=A1*b+B1*d
    C2=C1*a+D1*c
    D2=C1*b+D1*d
    A1,B1,C1,D1=A2,B2,C2,D2
M=[[A1,B1],[C1,D1]]
print("Résultat du calcul :", M)
```

### 6.2 Puissance d'une matrice 3×3

```
# Puissance d'une matrice 3x3
from math import*
# Entrée des données
a,b,c=eval(input("Entrer les 3 coefficients de la 1ère ligne : "))
d,e,f=eval(input("Entrer les 3 coefficients de la 2ème ligne : "))
g,h,i=eval(input("Entrer les 3 Coefficients de la 3ème ligne : "))
n=eval(input("Quel est l'exposant n ? "))
# Calculs et résultats
A1,B1,C1=1,0,0
D1,E1,F1=0,1,0
G1,H1,I1=0,0,1
for j in range(0,n):
    A2,B2,C2=A1*a+B1*d+C1*g,A1*b+B1*e+C1*h,A1*c+B1*f+C1*i
    D2,E2,F2=D1*a+E1*d+F1*g,D1*b+E1*e+F1*h,D1*c+E1*f+F1*i
    G2,H2,I2=G1*a+H1*d+I1*g,G1*b+H1*e+I1*h,G1*c+H1*f+I1*i
    A1,B1,C1=A2,B2,C2
    D1,E1,F1=D2,E2,F2
    G1,H1,I1=G2,H2,I2
M=[[A1,B1,C1],[D1,E1,F1],[G1,H1,I1]]
print("Résultat du calcul :", M)
```

## 7. Diagonalisation d'une matrice 2×2

### 7.4. Un programme pour calculer les valeurs propres d'une matrice 2×2

```
# Calcul des valeurs propres d'une matrice 2x2
a,b=eval(input("Valeurs des coefficients de la 1ère ligne ? "))
c,d=eval(input("Valeurs des coefficients de la 2ème ligne ? "))
detA=a*d-b*c
TrA=a+d
# Résolution de l'équation  $P(\lambda) = \lambda^2 - \text{Tr}(A)\lambda + \text{det}(A) = 0$ 
from math import*
delta=TrA*TrA-4*detA
if delta>0:
    x1=(-TrA+sqrt(delta))/2
    x2=(-TrA-sqrt(delta))/2
    print("1ère valeur propre : ",x1)
    print("2ème valeur propre : ",x2)
else :
    print("Les valeurs propres n'existent pas ou ne sont pas distinctes.")
```

## 8. Matrices et suites récurrentes

### 8. 2. Calcul des nombres de Fibonacci à l'aide d'une matrice 2×2

```

from math import*
# Coefficients de la matrice M
a,b,c,d=1,1,1,0
liste=[]
n=eval(input("Valeur de n ? "))
# Calcul de M^n et affichage des résultats
A1,B1,C1,D1=1,0,0,1
for i in range(0,n):
    A2=A1*a+B1*c
    B2=A1*b+B1*d
    C2=C1*a+D1*c
    D2=C1*b+D1*d
    A1,B1,C1,D1=A2,B2,C2,D2
    liste=liste+[C1]
print(liste)

```

### 8. 4 Les relations de Binet

```

from math import*
a=(1+sqrt(5))/2
b=(1-sqrt(5))/(1+sqrt(5))
n=eval(input("Valeur de n ? "))
for i in range(0,n):
    c=a**(i+1)/sqrt(5)
    d=1-b**(i+1)
    F=c*d
    print(F)

```

```

Valeur de n ? 7
0.9999999999999999 ; 0.9999999999999999 ; 1.9999999999999998 ;
3.0000000000000004 ; 5.0 ; 8.0000000000000002 ; 13.000000000000000

```

# Chapitre 13

## Géométrie analytique

### 1. Equation réduite d'une droite dans le plan

#### 1.2 Détermination de l'équation réduite d'une droite

```
# Détermination de l'équation d'une droite connaissant 2 de ses points
from math import*
x1=eval(input("abscisse du 1er point ? "))
y1=eval(input("ordonnée du 1er point ? "))
x2=eval(input("abscisse du 2ème point ? "))
y2=eval(input("ordonnée du 2ème point ? "))
if x2==x1:
    print("la droite a pour équation x=",x2)
else:
    a=(y2-y1)/(x2-x1)
    b=y2-a*x2
    print("la droite a pour équation y=",a,"x+",b)
```

```
# Détermination de l'équation d'une droite connaissant 1 point et son coefficient
# directeur
from math import*
xM=eval(input("abscisse du point M ? "))
yM=eval(input("ordonnée du point M ? "))
a=eval(input("coefficient directeur de la droite ? "))
b=yM-a*xM
if b>0:
    print("la droite a pour équation y=",a,"x +",b)
else:
    print("la droite a pour équation y=",a,"x",b)
```

```
abscisse du point M ? 2
ordonnée du point M ? 3
coefficient directeur de la droite ? 2
la droite a pour équation y= 2 x -1
```

#### 1.3 Intersection de 2 droites

```
# Intersection de 2 droites dont les équations réduites sont connues
a1=eval(input("Valeur de a1 ? "))
b1=eval(input("valeur de b1 ? "))
a2=eval(input("Valeur de a2 ? "))
b2=eval(input("valeur de b2 ? "))
```

```

                                if a1==a2:
print("Les 2 droites sont parallèles.")
else:
    x=(b1-b2)/(a2-a1)
    y=a1*x+b1
    print("x=",x," et y=",y

```

#### 1.4 Distance d'un point à une droite

```

# Calcul de la distance d'un point à une droite
from math import*
# Données
dr=0
xM=eval(input("abscisse du point M ? "))
yM=eval(input("ordonnée du point M ? "))
a1=eval(input("coefficient directeur a de la droite ? "))
b1=eval(input("valeur de b ? "))
if a1==0:
    dr=1
    d=sqrt((b1-yM)**2)
# Equation de la perpendiculaire
if dr==0:
    a2=-1/a1
    b2=yM-a2*xM
# Intersection des 2 droites
    x=(b1-b2)/(a2-a1)
    y=a1*x+b1
# Distance
    d=sqrt((yM-y)**2+(xM-x)**2)
print("distance=",d)

```

## 2. Équation cartésienne d'une droite dans le plan

### 2.2 Recherche de l'équation cartésienne d'une droite dont on connaît 2 points

```
# Equation cartésienne d'une droite dont on connaît 2 points
from math import*
x1,y1=eval(input("Valeurs des coordonnées de A ? "))
x2,y2=eval(input("Valeurs des coordonnées de B ? "))
# Résultats
a,b=y1-y2,x2-x1
c=x1*(y2-y1)-y1*(x2-x1)
if a==0 and b==0:
    print("Il n'y a qu'un seul point.")
else :
    print("Coefficients de l'équation cartésienne de la droite (AB) :")
    print("a=",a," b=",b," c=",c)
```

### 2. 3. Recherche de l'équation cartésienne d'une droite dont on connaît un vecteur directeur et un point

```
# Equation cartésienne d'une droite connaissant un point et un vecteur directeur
from math import*
xA,yA=eval(input("Coordonnées du point A ? "))
vx,vy=eval(input("Coordonnées du vecteur directeur v ? "))
# Résultats
a=vy
b=-vx
c=vx*yA-vy*xA
if vx==0 and vy==0:
    print("Impossible.")
else:
    print("Coefficients de l'équation cartésienne de la droite :")
    print("a=",a," b=",b," c=",c)
```

### 2.4 Intersection de 2 droites

```
# Intersection de 2 droites dont les équations cartésiennes sont connues
from math import*
a,b,c=eval(input("Droite n°1. Valeur de a, de b et de c ? "))
aa,bb,cc=eval(input("Droite n°2. Valeur de a', de b' et de c' ? "))
# Condition à respecter
d=a*bb-aa*b
# Résultats
if d==0:
    print("Les deux droites sont parallèles.")
```

```
else :
    x=(b*cc-bb*c)/d
    y=(aa*c-a*cc)/d
    print("Coordonnées du point de rencontre : x=",x," et y=",y)
```

## 2.5 Droites parallèles

```
# Equation cartésienne de la parallèle à une droite d'équation connue
# passant par un point connu
from math import*
xA,yA=eval(input("abscisse et ordonnée du point A ? "))
a,b,c=eval(input("Valeur de a, de b et de c ? "))
# Résultats
cc=-a*xA-b*yA
print("L'équation cherchée est ax+by+c'=0 avec a=",a," b=",b," et c'=",cc)
```

## 2.7 Droites perpendiculaires

```
# Perpendiculaire à une droite d'équation connue passant par un point connu
from math import*
xA,yA=eval(input("abscisse et ordonnée du point A ? "))
a,b,c=eval(input("valeur de a, de b et de c ? "))
# Résultats
cc=-b*xA+a*yA
print("L'équation cherchée est de la forme ax+by+c=0 avec a=",b," b=",a," c=",cc)
```

### 3. Les droites dans l'espace

#### 3.2 Points alignés

```
# Les points A, B et C sont-ils alignés ?
from math import*
xA,yA,zA=eval(input("Coordonnées du point A ? "))
xB,yB,zB=eval(input("Coordonnées du point B ? "))
xC,yC,zC=eval(input("coordonnées du point C ? "))
# Calculs et résultats
a,b,c=xB-xA,yB-yA,zB-zA
aa,bb,cc=xC-xA,yC-yA,zC-zA
k1,k2,k3=a*bb-b*aa, b*cc-c*bb,a*cc-c*aa
if k1==0 and k2==0 and k3==0:
    print("Les points sont alignés.")
else :
    print("Les points ne sont pas alignés. ")
```

#### 3.3 Représentation paramétrique d'une droite

```
# Représentation paramétrique d'une droite définie par 2 points distincts
from math import*
xA,yA,zA=eval(input("Coordonnées du point A ? "))
xB,yB,zB=eval(input("Coordonnées du point B ? "))
# Calculs et résultats
a,b,c=xB-xA,yB-yA,zB-zA
print("Voici la représentation paramétrique de la droite (AB)")
print("x=",xA,"+",a,"t")
print("y=",yA,"+",b,"t")
print("z=",zA,"+",c,"t")
```

#### 3. 4. Comment reconnaître qu'un point appartient à une droite ?

```
# Le point P appartient-il à la droite D ?
from math import*
xA,yA,zA=eval(input("Coordonnées du point A ? "))
a,b,c=eval(input("Valeur de a , de b et de c ? "))
xP,yP,zP=eval(input("Valeur de x ? "))
# Calculs et résultats
k1=a*(yP-yA)-b*(xP-xA)
k2=b*(zP-zA)-c*(yP-yA)
k3=a*(zP-zA)-c*(xP-xA)
if k1==0 and k2==0 and k3==0:
    print("P appartient à la droite.")
else:
    print("P n'appartient pas à la droite.")
```

### 3.5 Droites coplanaires, intersection de 2 droites

```
# Position relatives de 2 droites de l'espace
from math import*
dr=0
xA,yA,zA=eval(input("Coordonnées du point A ? "))
a,b,c=eval(input("Valeur de a, de b et de c ? "))
xAA=eval(input("Coordonnées du point A' ? "))
aa=eval(input("Valeur de a', de b' et de c' ? "))
# Calculs auxiliaires
A1,A2,A3=a*bb-aa*b,a*cc-aa*c,b*cc-bb*c
B1=bb*(xAA-xA)-aa*(yAA-yA)
B2=cc*(xAA-xA)-aa*(zAA-zA)
B3=cc*(yAA-yA)-bb*(zAA-zA)
# Résultats
if A1==0 and A2==0 and A3==0:
    print("Les 2 droites sont parallèles.")
    dr=1
else:
    if A1!=0:
        t1=B1/A1
    if A2!=0:
        t2=B2/A2
    if A3!=0:
        t3=B3/A3
    if t1==t2 and t2==t3:
        print("Les 2 droites ont un point commun.")
        print("x=",xA+a*t1," y=",yA+b*t1," z=",zA+c*t1)
        dr=1
if dr==0:
    print("Les 2 droites ne sont ni parallèles ni sécantes.")
```

## 4. Equations paramétriques d'un plan

### 4.2 Détermination de l'équation paramétrique d'un plan

```
# Représentation paramétrique d'un plan défini par 3 points
from math import*
# Entrée des données
xA,yA,zA=eval(input("Valeurs des coordonnées de A ? "))
xB,yB,zB=eval(input("Valeurs des coordonnées de B ? "))
xC,yC,zC=eval(input("Valeurs des coordonnées de C ? "))
# Les points A, B et C sont-ils alignés ?
a,b,c=xB-xA,yB-yA,zB-zA
aa,bb,cc=xC-xA,yC-yA,zC-zA
k1,k2,k3=a*bb-b*aa,b*cc-c*bb,a*cc-c*aa
# Résultats si A, B et C ne sont pas alignés
if k1==0 and k2==0 and k3==0:
    print("Le plan n'existe pas car A, B et C sont alignés.")
else:
    print("Représentation paramétrique du plan (ABC) :")
    print("x=",xA,"+",a,"t+",aa,"t")
    print("y=",yA,"+",b,"t+",bb,"t")
    print("z=",zA,"+",c,"t+",cc,"t")
```

### 4.3 Comment reconnaître qu'un point appartient à un plan ?

```
# Le point M appartient-il au plan (P) ?
from math import*
xM,yM,zM=eval(input("Coordonnées du point M ? "))
print("Entrée des équations du plan (P).")
xA,yA,zA=eval(input("Coordonnées du point A ? "))
a,b,c=eval(input("Coordonnées du vecteur directeur n°1 ? "))
aa,bb,cc=eval(input("Coordonnées du vecteur directeur n°2 ? "))
dr=1
# Calculs
e,f,g=xM-xA,yM-yA,zM-zA
D=aa*b-a*bb
if D==0:
    dr=0
if dr==1:
    t=(aa*f-e*bb)/D
    tt=(e*b-a*f)/D
    E=c*t+cc*tt
# Résultats
if dr==1 and E==g:
    print("M appartient au plan (P).")
```

```

print("On a t=",t," et t'=",tt)
else:
print("M n'appartient pas au plan (P).")

```

#### 4.4 Comment reconnaître que 4 points sont coplanaires ?

```

# Les points A, B, C et D sont-ils coplanaires ?
from math import*
xA,yA,zA=eval(input("Coordonnées de A ? "))
xB,yB,zB=eval(input("Coordonnées de B ? "))
xC,yC,zC=eval(input("Coordonnées de C ? "))
xD,yD,zD=eval(input("Coordonnées de D ? "))
dr1,dr2,dr3=1,1,1
# Les points A, B et C sont-ils alignés ?
a,b,c=xB-xA,yB-yA,zB-zA
aa,bb,cc=xC-xA,yC-yA,zC-zA
k1,k2,k3=a*bb-b*aa, b*cc-c*bb,a*cc-c*aa
if k1==0 and k2==0 and k3==0:
print("A, B et C sont alignés.")
dr1=0
# Recherche des 2 nombres alpha et beta
e,f,g=xD-xA,yD-yA,zD-zA
D=a*bb-aa*b
if D!=0:
beta=(a*f-b*e)/D
alpha=(e*bb-aa*f)/D
else:
dr2=0
K=alpha*c+beta*cc-g
if K!=0:
dr3=0
if dr1==1 and dr2==1 and dr3==1:
print("Les 4 points sont coplanaires. On a alpha=",alpha," et beta=",beta)
else:
print("Les 4 points ne sont pas coplanaires.")

```

#### 4.5 Intersection d'un plan et d'une droite

```

# Intersection d'un plan et d'une droite
def det(a,b,c,d):
return a*d-b*c
# Entrée des équations paramétriques
print("Entrée des paramètres du plan : ")
xA,yA,zA=eval(input("Valeurs de xA, de yA et de zA : "))
a,b,c=eval(input("Valeurs des nombres a, b et c : "))
d,e,f=eval(input("Valeurs des nombres d, e et f : "))

```

```

print("Entrée des paramètres de la droite : ")
xAA,yAA,zAA=eval(input("Valeurs de xA', de yA' et de zA' : "))
g,h,i=eval(input("Valeurs des nombres a', b' et c' : "))
# Calcul des coefficients du système à résoudre
a1,b1,c1,d1=a,d,-g,xAA-xA
a2,b2,c2,d2=b,e,-h,yAA-yA
a3,b3,c3,d3=c,f,-i,zAA-zA
# Calcul des déterminants
Detsys=a1*det(b2,c2,b3,c3)-a2*det(b1,c1,b3,c3)+a3*det(b1,c1,b2,c2)
DetT1=d1*det(b2,c2,b3,c3)-d2*det(b1,c1,b3,c3)+d3*det(b1,c1,b2,c2)
DetT2=a1*det(d2,c2,d3,c3)-a2*det(d1,c1,d3,c3)+a3*det(d1,c1,d2,c2)
DetT3=a1*det(b2,d2,b3,d3)-a2*det(b1,d1,b3,d3)+a3*det(b1,d1,b2,d2)
# Résolution du système et résultats
if Detsys!=0:
    print("Valeurs des paramètres t1, t2 et t3 : ")
    T1=DetT1/Detsys
    T2=DetT2/Detsys
    T3=DetT3/Detsys
    print("t1=",T1,"t2=",T2,"t3=",T3)
    print("Coordonnées du point commun : ")
    print("x=",xA+a*T1+d*T2, "y=",yA+b*T1+e*T2, "z=",zA+c*T1+f*T2)
    print("Vérification :")
    print("x=",xAA+g*T3, "y=",yAA+h*T3, "z=",zAA+i*T3)
else:
    print("L'intersection est vide.")

```

## 5. Equation cartésienne d'un plan

### 5.2 Equation d'un plan défini par un de ses points et par un vecteur normal

```
# Equation cartésienne d'un plan défini par 1 point et 1 vecteur normal
from math import*
xA,yA,zA=eval(input("Coordonnées du point A ? "))
a,b,c=eval(input("Coordonnées du vecteur normal au plan ? "))
d=-(a*xA+b*yA+c*zA)
print("Voici les coefficients de l'équation cartésienne du plan :")
print("a=",a," b=",b," c=",c," d=",d)
```

### 5.3 Equation d'un plan défini par 3 points non alignés

```
# Equation cartésienne d'un plan défini par 3 points
from math import*
# Entrée des données
xA,yA,zA=eval(input("Valeurs des coordonnées de A ? "))
xB,yB,zB=eval(input("Valeurs des coordonnées de B ? "))
xC,yC,zC=eval(input("Valeurs des coordonnées de C ? "))
# Les points sont-ils alignés ?
k1=(xB-xA)*(yC-yA)-(yB-yA)*(xC-xA)
k2=(yB-yA)*(zC-zA)-(zB-zA)*(yC-yA)
k3=(xB-xA)*(zC-zA)-(zB-zA)*(xC-xA)
if k1==0 and k2==0 and k3==0:
    print("A, B et C sont alignés.")
else:
    # Calcul de a, de b de c et de d
    a=(yB-yA)*(zC-zA)-(yC-yA)*(zB-zA)
    b=(zB-zA)*(xC-xA)-(zC-zA)*(xB-xA)
    c=(xB-xA)*(yC-yA)-(xC-xA)*(yB-yA)
    d=-(a*xA+b*yA+c*zA)
    print("a=",a," b=",b," c=",c," d=",d)
```

### 5. 4. Intersection d'une droite et d'un plan

```
# Intersection droite/plan
from math import*
# Entrée des données
print("Equation cartésienne du plan :")
A,B,C,D=eval(input("Valeurs des coefficients A, B, C et D ? "))
print("Equation paramétrique de la droite :")
x0,y0,z0=eval(input("Valeurs des coordonnées x0, y0 et z0 ? "))
a,b,c=eval(input("Valeurs des coefficients a, b et c ? "))
# Résultats
p=A*a+B*b+C*c
```

```

if p!=0:
    t=-(A*x0+B*y0+C*z0+D)/p
    print("t=",t)
    print("Coordonnées de l'intersection :")
    print("x=",x0+a*t," y=",y0+b*t," z=",z0+c*t)
else:
    print("La droite et le plan sont parallèles.")

```

### 5.5 Distance d'un point à un plan

```

# Distance d'un point à un plan
from math import*
# Entrée des données
print("Equation cartésienne du plan (P) :")
A,B,C,D=eval(input("valeurs des coefficients A, B, C et D ? "))
x0,y0,z0=eval(input("valeurs des coordonnées du point M ? "))
# Coordonnées du projeté orthogonal de M sur le plan (P)
p=A*A+B*B+C*C
t=-(A*x0+B*y0+C*z0+D)/p
x,y,z=x0+A*t,y0+B*t,z0+C*t
print("Coordonnées de H : ", " x=",x," y=",y," z=",z)
# Distance de M au plan (P)
d=sqrt((x-x0)**2+(y-y0)**2+(z-z0)**2)
print("distance du point M au plan (P)=",d)

```

### 5.6 Intersection de 2 plans

```

# Intersection de 2 plans connus par leurs équations cartésiennes
from math import*
# Entrée des données
a1,b1,c1,d1=eval(input("Valeurs de a1, de b1, de c1 et de d1 ? "))
a2,b2,c2,d2=eval(input("Valeurs de a2, de b2, de c2 et de d2 ? "))
D1,D2,D3=a1*b2-a2*b1,b1*c2-b2*c1, a1*c2-a2*c1
print("D1=",D1," D2=",D2," D3=",D3)
# Les deux plans sont-ils parallèles ?
if D1==0 and D2==0 and D3==0:
    print("Les 2 plans sont parallèles.")
    dr=0
else:
    print("Les 2 plans ne sont pas parallèles.")
# Equations de la droite commune
print("Vous pouvez choisir x,y ou z comme paramètre.")
print("Si D1=0, choisissez 1 ou 2.")
print("Si D2=0, choisissez 2 ou 3.")
print("Si D3=0, choisissez 1 ou 3.")
print("Si aucun de ces nombres n'est nul, choisissez 1, 2 ou 3.")

```

```
R=eval(input("Votre choix ?"))
if R==1:
    print("x0=0","a=",1)
    print("y0=",-(c1*d2-c2*d1)/D2,"b=", -D3/D2)
    print("z0=",-(b1*d2-b2*d1)/D2,"c=",D1/D2)
if R==2:
    print("x0=",(c1*d2-c2*d1)/D3,"a=", -D2/D3)
    print("y0=0"," b=1")
    print("y0=",(a2*d1-a1*d2)/D3,"c=", -D1/D3)
if R==3:
    print("x0=",(b1*d2-b2*d1)/D1,"a=",D2/D1)
    print("y0=",-(a1*d2-a2*d1)/D1,"b=", -D3/D1)
    print("z0=0","c=",1)
```