Planche 1 :

```
import matplotlib.pyplot as plt
import numpy as np
from mpl_toolkits.mplot3d import Axes3D
import random as rd
from math import sqrt, pi

# Q1  :

def creation (n)  :
    return [rd.randint(1,n+1) for i in range(n+1)]

# principe des tiroirs.

# Q2  :

# naif  :
def indice (L)  :
    n = len(L)
    for i in range(1,n)  :
        for j in range(i)  :
            if L[j] == L[i]  :
                return i
    return(0)

# recursif

def indice_rec(L)  :
    if L ==[]  :
        return 0
    else  :
        ind = indice_rec (L[ :-1])
        if ind ==0  :
            if L[-1] in L[ :-1]  : return (len(L)-1)
            else  : return 0
        else  : return ind

# Q3  :

def moyenne (m,n)  :
    somme = 0
    for j in range(m)  :
        L = creation(n)
        somme += indice(L)
    return somme/m

m = 1000
X = [n for n in range(10,101)]
Y = [moyenne(m,n) for n in X]
Z = [sqrt(n) for n in X]
```

```
plt.clf()
plt.plot(X,Y,label="moyenne de l'indice en fonction de n")
plt.plot(X,Z,label="racine carrée de n")
plt.legend()
plt.savefig("Q3.png")

# Conjecture  : la moyenne est égale à sqrt(n)*constante légérement
# supérieure à 1

# Q4

# P(X_n = 1) = 1/n
# P(X_n > 1) = 1-P(X_n=1) = 1- 1/n
# P(X_n > 2) = P(X_n > 2 | X_n > 1)*P(X_n > 1)=(1 - 2/n)*(1 - 1/n)
# P(X_n > j+1) = P(X_n > j+1 | X_n > j)*P(X_n > j) = (1-j/n)*P(X_n > j)
# et on récurre.

# Q5.a

# pour l'inégalité, simple étude de fonction
# (ou alors on utilise la série entière)
# P(X_n>j) <= produit(exp(-i/n))=exp((-1/n)*somme i)=exp(-(j(j+1))/n)
# <= exp(-j**2/n)

# Q5.b

# E(X_n) = somme de i=1..n ( i*P(X_n=i)) = somme(i*(P(X_n>i-1) - P(X_n>i)),i=1..n)
# = somme((i-1)*P(X_n>i-1)-i*P(X_n>i) + P(X_n>i-1),i=1..n)
# = 0 -nP(X_n>n) + somme(P(X_n>i),i=0..n-1) = somme(P(X_n>i), i=0..n) car P(X_n>n)=0

# Q5.c  : Comparaison série / intégrale
"""
for i in range(10,301,20)  :
    print(moyenne(100,i), sqrt(i*(pi/2)))
"""
```

Planche 2 :

```
import matplotlib.pyplot as plt
import numpy as np
import numpy.linalg as lin
from math import sqrt

# Q1  :

A = np.array([[-9,-21,-20],[-9,3,20],[-9,-9,-18]])
d = lin.det(A)

# d n'est pas nul, donc A est inversible.
```

```
# Q2  :

poca = np.poly(A)

# poca = X**3+24X**2-108X-3888

# poca’ = 3X**2+48X-108, a pour discriminant 3600, donc -18 et 2.
# on remarque que -18 est aussi racine de poca, donc c’est une racine double
# avec lin.eigvals(A), on voit que l’autre est 12.
# donc on peut poser P=(X-12)(X+18)

P = np.array([1,6,-12*18])

def P_matrix(P,A)  :
    return sum([P[i]*lin.matrix_power(A,len(P)-i) for i in range(len(P))])

ct = lambda t  : A-t*P_matrix(P,A)

lin.eigvals(ct(3))

# On observe qu’il y a toujours les 3 mêmes valeurs propres  : -18 (double) et 12.

# Q4  : on reprend Q=poca.

# Q5  :

# norme euclidienne

def N(A)  :
    return sqrt(sum([sum([A[i,j]**2 for i in range(len(A))])\
                 for j in range(len(A[0]))]))
```

Planche 3 :

```
import matplotlib.pyplot as plt
import numpy as np
from mpl_toolkits.mplot3d import Axes3D
import random as rd

# Q1  :

def theta(n,x)  :
    p = 1
    ex = 1
    for k in range(n+1)  :
        p *= (1-x**(ex))
        ex *= 2
    return p

def dessin(n)  :
    plt.clf()
```

```
    X = [-1+2*k/1000 for k in range(1001)]
    for p in range(n+1)  :
        Y = [theta(p,x) for x in X]
        plt.plot(X,Y,label="theta_"+str(p))
    plt.legend()
    plt.savefig("theta.png")


# Q2  : Conjectures  :
# theta_n n'est ni paire ni impaire, elle est croissante puis décroissante
# (sauf pour n=0).
# La suite converge simplement.
# Pour la monotonie  : on passe au log  : ln theta_n = ln_theta_{n-1} + ln(1-x**2**k)
# si alpha_{n-1} est le point d'annulation de theta_{n-1}', alors de -1 à
# alpha_{n-1}, les deux memebres de ln(theta_n) sont croissants. Et ils sont
# décroissants de 0 à 1. En alpha_{n-1}, (ln theta_n)'=(ln(1-x**2**k)'>0 et en
# 0, c'est <0. Donc avec le TVI, il y a un pt d'annulation entre les 2.
# Pourquoi un seul  ? Je bloque.

#convergence simple  : en -1 et 1, ça vaut tuojours 0.
# soit x\in ]-1,1[. ln(theta_n(x))-ln(theta_{n-1})(x) = ln(1-x**2**n)<0 donc la
# suite est décroissante, donc elle a une limite, finie ou -infini, donc
# theta_n(x) converge.

# si a\in]0,1[, sur ]-a,a[, et n<p, theta_p(x)-theta_n(x)=theta_n(x)*
# (produit((1-x**2**k),k=n+1..p)-1). theta_n est bornée, et
# |produit en x-1|\leq |produi en a -1|, qui tend vers 0, donc theta_n converge
# uniformément. Donc theta est continue.

# Par décalage d'indice, theta_n(x^2)=produit de k=1 à n+1 de (1-x**2**k),
# donc theta_{n+1}(x)=(1-x)theta_n(x^2) et on passe à la limite.

# Q4  : par récurrence, f(x)=theta_n(x)f(x**2**(k+1)) donc par passage à la limite,
# f(x) = f(0)theta(x). Donc les seules fonctions vérifiant cela sont les ctes*theta.

# Q5  : simple développement de (1-x)theta(x**2).
```

Planche 4 :

```
import matplotlib.pyplot as plt
import numpy as np
from mpl_toolkits.mplot3d import Axes3D
import random as rd

# Q1  : les X_n ne prennent p.s. que deux valeurs  : 1 et 2, donc S_n div grossiérement.

def X (p)  :
    x = rd.random()
    if x < p  : return 1
```

```
    else  : return 2

def Y (k,p)  :
    S = 0
    n = 0
    while S < k  :
        S += X(p)
        n += 1
    return n

# Q2  :

def esp (k,p)  :
    N = 1000
    S = 0
    for n in range(N)  :
        S += Y(k,p)
    return S/N

P = [0.1,0.3,0.5,0.7,0.9]

def Q2 ()  :
    plt.clf()
    X = [i for i in range(1,100)]
    for p in P  :
        Y = [esp(k,p) for k in X]
        plt.plot(X,Y,label="p="+str(p))
    plt.legend()
    plt.savefig("Q2.png")

# Q3  :

# P(Yk=n)=P(Yk=n|Y(k-1)=n-1).P(Y(k-1)=n-1) + P(Yk=n|Y(k-2)=n-1).P(Y(k-1)=n-2)
#           +somme pour i < k-2 P(Yk=n|Y(k-i)<n-1).P(Y(k-i)=n-2)
# = P(Xk=1).P(Y(k-1)=n-1) + P(Xk=2).P(Y(k-2)=n-1) + 0

# Q4  :

# E(Yk) = somme P(Yk=n).n = somme pP(Y(k-1)=n-1).n + (1-p)P(Y(k-2)=n-1).n
# = somme pP(Y(k-1)=n-1).(n-1) + (1-p)P(Y(k-2)=n-1).(n-1) + pP(Y(k-1)=n-1)
# + (1-p)P(Y(k-2)=n-1) = pE(Y(k-1)) + (1-p)E(Y(k-2)) + p + (1-p)

# Q5  :

# On remarque que (k/(2-p)) est une suite solution de cette relation de récurrence.
# Équation homogène  : poca X**2-pX-(1-p), discriminant p**2+4(1-p)=(2-p)**2
# racines 1 et (p-1), solutions de la forme a + b(p-1)**n + n/(2-p)
# ce qui est équivalent à n/(2-p) car (p-1) \in ]-1,0[.

P2=[0.1,0.5,0.9]
```

```
def Q5 ()  :
    plt.clf()
    X = [i for i in range(1000,2000,100)]
    for p in P2  :
        cp = 1/(2-p)
        Y = [esp(k,p)-k*(cp) for k in X]
        plt.plot(X,Y,label="p="+str(p))
    plt.legend()
    plt.savefig("Q5.png")
```