TP 2

Exercice 1 : LGN, TCL.

Loi forte des Grands nombres

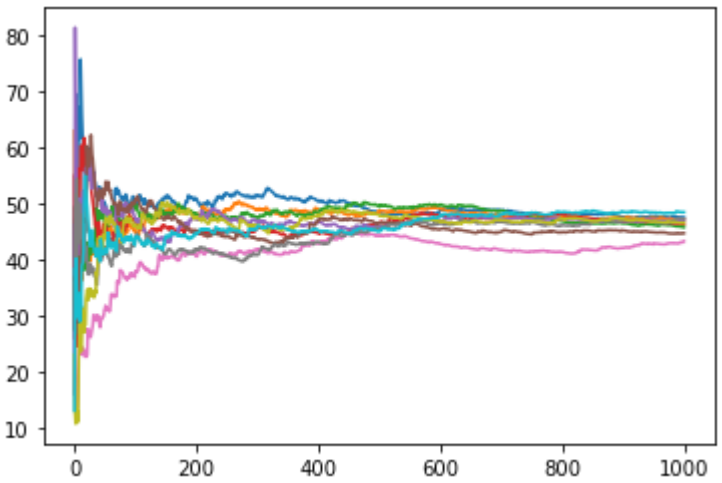
In [6]:
```
import matplotlib.pyplot as plt
import numpy as np
#import numpy.random as nr
```

Question 1 (a)

In [58]:
```
N= 1000
p = 1/45 #on a supposé que le numéro étudiant est 12027045
k= 10 #le nombre de trajectoire

x = np.random.geometric(p,size = (N,k)) #on fabrique une matrice d
e taille N*K de la loi géométrique
```

In [59]:
```
y = np.cumsum(x,0)/np.fromfunction(lambda  n,i:(n+1) , (N, k))
plt.figure()
for i in range(k):
    plt.plot(y[:,i])
```



Question 1 (b)

In [60]:
```
for i in range(k) :
   print(y[N-1,i])
```

```
47.534
47.205
45.8
46.347
47.104
44.729
43.291
47.257
46.542
48.487
```

Question 1 (c)

On a affiché 10 courbes indépendantes. On observe, dans notre cas que les courbes convergent vers 45 ce qui est conforme à la loi des grands nombres. On sait qu'une loi géométrique de paramètre p admet $1/p$ comme moyenne. Autrement dit, on voit que pour (presque tout) $\omega \in \Omega$, on a bien la convergence de $\bar{X}_n(\omega)$ vers la valeur limite $\mathbb{E}[X] = 45$

Théorème central limit

Question 2 (a)

Grâce à wikipédia on trouve que si X suit un loi $\mathrm{Géométrique}(p)$ alors

$$E(X) = \frac{1}{p}$$

et

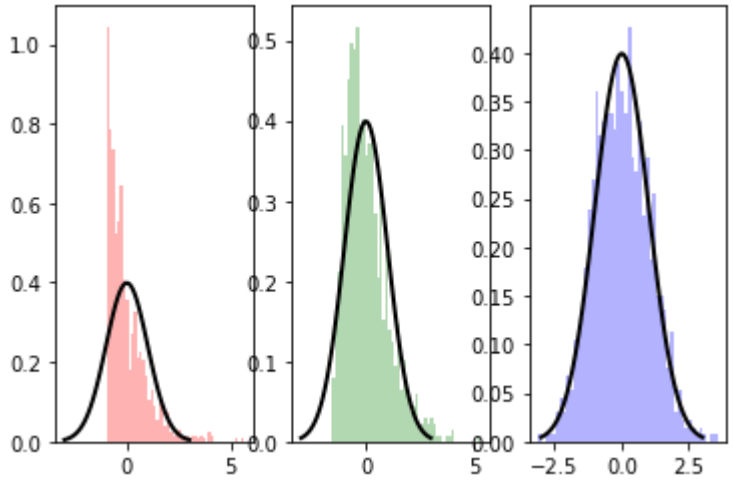
$$Var(X) = \frac{q}{p^2}$$

Dans notre cas où $p = 1/45$ on a $E(X) = 45$ et $Var(X) = 45 * 44 = 1980$.

Question 2 (b)

In [68]:
```
k = 1000
Nval = [1,3,100]
z = np.array([ [np.sqrt(n)*(np.mean(np.random.geometric(p,size =
n))-45)/np.sqrt(1980)
                for i in range(k)]   for n in Nval ])
```

In [69]:
```
from scipy.stats import norm
fig,ax = plt.subplots(1,3)
color = ["red","green","blue"]
for i in range(z.shape[0]):
    ax[i].hist(z[i,:],bins = 50,color =color[i],alpha = 0.3,densit
y = True)
    ax[i].plot(np.linspace(-3,3,100),norm.pdf(np.linspace(-3,3,10
0),loc = 0,scale = 1),color ="black",linewidth = 2)
```



Question 2 (c)

On observe bien que la forme des histogramme s'approche avec N qui augmente, de la densité de la loi normale, centrée réduite.

Il est certain que pour le premier graphique, celui en rose, on oberve simplement une loi gémotrique recentrée, rien à voir avec une loi normale.

Théorème central limit, suite

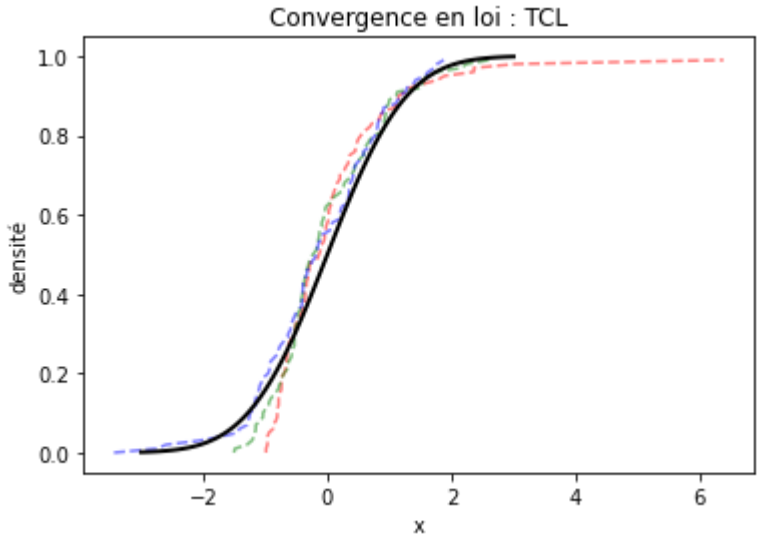
Question 3 (a)

In [105]:
```
k = 100
from scipy.stats import norm
Nval = [1,3,10000]
z = np.array([ [np.sqrt(n)*(np.mean(np.random.geometric(1/45,size
=n))-45)/(np.sqrt(1980))
                for i in range(k)]   for n in Nval ])

plt.figure()
color = ["red","green","blue"]
for i in range(z.shape[0]):
    plt.plot(np.sort(z[i,:]),np.array(range(k))/float(k) ,color =c
olor[i],linestyle  = "--",alpha = 0.5)
plt.plot(np.linspace(-3,3,100),norm.cdf(np.linspace(-3,3,100),loc
= 0,scale = 1),color ="black",linewidth = 2)

plt.title("Convergence en loi : TCL")
plt.xlabel("x")
plt.ylabel("densité")
```

Out[105]:
```
Text(0, 0.5, 'densité')
```



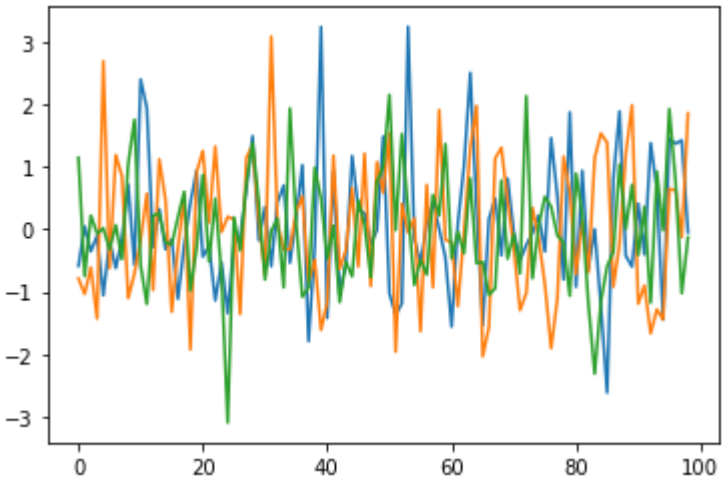
Question 3 (b)

Ce graphique illustre la convergece en loi par la fonction de répartition. On observe bien donc la même chose, la convergence des fonctions de répartitions.

Question 3 (c)

In [106]:
```
k = 3
from scipy.stats import norm
Nval = range(1,100)
z = np.array([ [np.sqrt(n)*(np.mean(np.random.geometric(1/45,size
=n))-45)/(np.sqrt(1980))
                for i in range(k)]   for n in Nval ])
```

In [107]:
```
plt.figure()
for i in range(k):
    plt.plot(z[:,i])
```



Bon, on voit qu'il ne se passe rien, plus exactement, on observe qu'il ne semble pas y avoir de convergence presque sûre.

Processus de Galton-Watson

Question 1 (a)

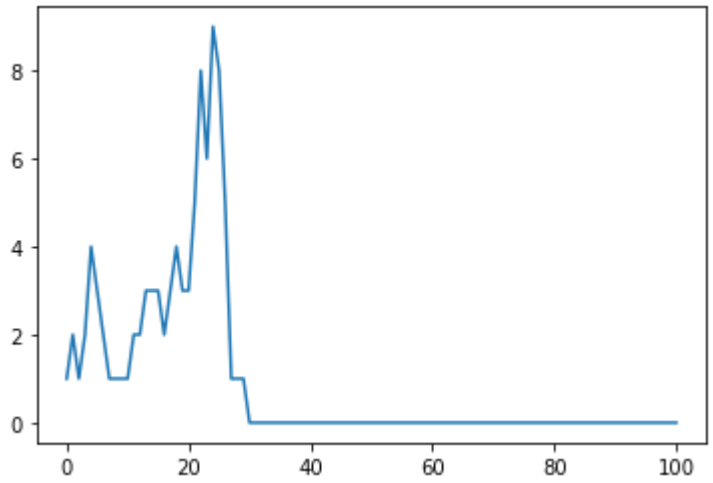
In [115]:
```
import numpy as np
import matplotlib.pyplot as plt


mu=[0.25,0.5,0.25]
T=100 #nombre de générations simulées
Z=np.zeros(T+1,int)
Z[0]=1 # population initiale
for t in range(T):
    u=np.random.rand(Z[t])
    xi = np.sum( np.array([ u> p for p in np.cumsum(mu) ]),0  )
    Z[t+1] = np.sum(xi)
```

Question 1 (b)

In [117]:
```
%matplotlib inline
plt.plot(Z)
```

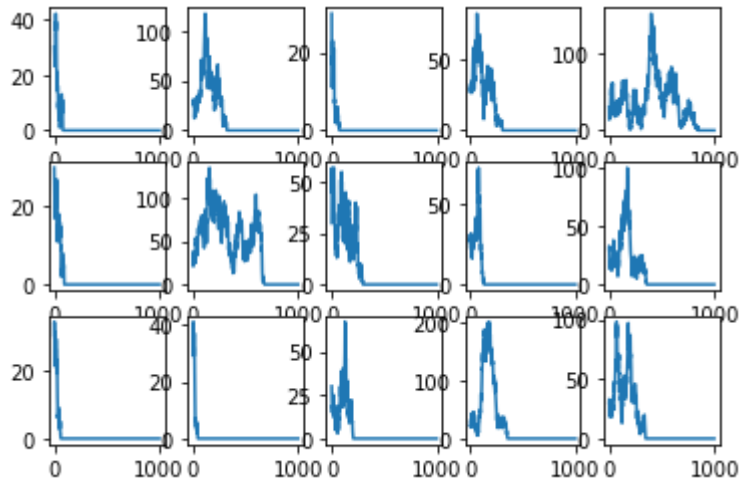
Out[117]:
```
[<matplotlib.lines.Line2D at 0x7f8c041c9d60>]
```



On remarque que le comportement varie, parfois le processus tombe à zéro très rapidement. Plusieurs simulations semblent montrer l'extinction à chaque fois. On peut illustrer cela en représentant plusieurs réalisations côte à côte.

In [120]:
```
%matplotlib inline
mu=[0.25,0.5,0.25]
def simul_branch(mu,T = 1000,z0 = 1):
    Z=np.zeros(T+1,int)
    Z[0]=z0 # population initiale
    for t in range(T):
        u=np.random.rand(Z[t])
        xi = np.sum( np.array([ u> p for p in np.cumsum(mu) ]),0
)
        Z[t+1] = np.sum(xi)
    return(Z)

f,ax=  plt.subplots(3,5)
for a in ax.reshape(-1):
    a.plot(simul_branch(mu,z0 = 30))
```



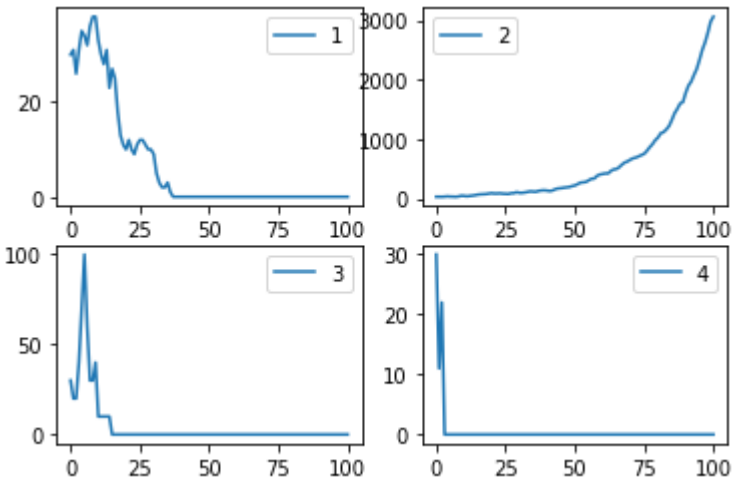
Question 1 (c)

La théorie est que la probabilité d'extinction est le plus petit point fixe de la fonction génératrice de la loi de branchement sur $[0,1]$. On est ici dans le cas $\mathbb{E}[\xi_{i,n}] = 1$, donc la théorie prédit extinction presque sûre, et c'est ce qu'on a observé.

Question 2 (a), (b), (c) et (d).

In [121]:
```
mulist = [[0.25,0.55,0.2],  [0.25,0.45,0.3],  [0.9]+[0]*9+[0.1],
[0.9]+[0]*10+[0.1]  ]

f,ax=  plt.subplots(2,2)
for i,a in enumerate(ax.reshape(-1)):
    a.plot(simul_branch(mulist[i],T = 100,z0 = 30))
    a.legend(str(1+i))
```



On calcule les espérances :

(a) $\mathbb{E}_{\mu_1}[\xi] = 0.55 + 2 * 0.2 = 0.95$: Extinction presque-sûre.

(b) $\mathbb{E}_{\mu_2}[\xi] = 0.45 + 2 * 0.3 = 1.05$: Possibilité de croissance exponentielle.

(c) $\mathbb{E}_{\mu_3}[\xi] = 10 * 0.1 = 1$: Extinction presque-sûre.

(d) $\mathbb{E}_{\mu_4}[\xi] = 11 * 0.1 = 1.1$: Possibilité de croissance exponentielle.

Question 3 (a)

La théorie prédit que le probabilité d'extinction sera donnée par la plus petite solution dans $[0,1]$ à l'équation

$$\phi(s) = s,$$

où $\phi(s) = \mathbb{E}[s^{\xi}]$.

Ici, pour la loi du $2.\ (b)$, on a $\phi(s) = 0.25 + 0.45s + 0.3s^2$.

On peut donc résoudre $\phi(s) = s$

$$\begin{aligned} \phi(s) &= s \\ 0 &= 0.25 - 0.55s + 0.3s^2 \\ s &= \frac{0.55 - \sqrt{0.55^2 - 4 * 0.25 * 0.3}}{0.6} \\ s &= \frac{5}{6} \end{aligned}$$

(l'autre solution étant égale à 1).

In [125]:
```
s = (0.55-np.sqrt(0.55**2-0.3))/0.6
print(s)
```

```
0.8333333333333325
```

Question 3 (b)

In [126]:
```
prop_extinc = np.mean([simul_branch(mulist[1],T = 100,z0 = 1)[-1]==0 for i in range(1000)])
prop_extinc
##Un peu long ! mais ok
```

Out[126]:
```
0.827
```

In [ ]:
```

```