TP 1

Exercice 1

1.

On commence par importer les packages, et simuler un échantillon

In [41]:
```
#imports
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats

n = 100
lam = 10

x = np.random.poisson(lam,n)
print(x)
```

```
[13  6  9  7 12 10 14  6  6  8 11  7 12  7 10  7  7 11  9  7 16  9 1
1  6
 12  9  7  3  8  6 15 17  7 11 17  7  9  9  7  9  4 14  7  8  8  8
8  5
  6  7  4  9  9  7 13  5  9 10  9 13  5  8 11 10  6 12  6 10  8  5
9 12
 11 11 12  5  9  7 10 14  5  9 12 12  6  9  4  9 12 14  5 13 13 11 1
0 11
 13 11 12  7]
```

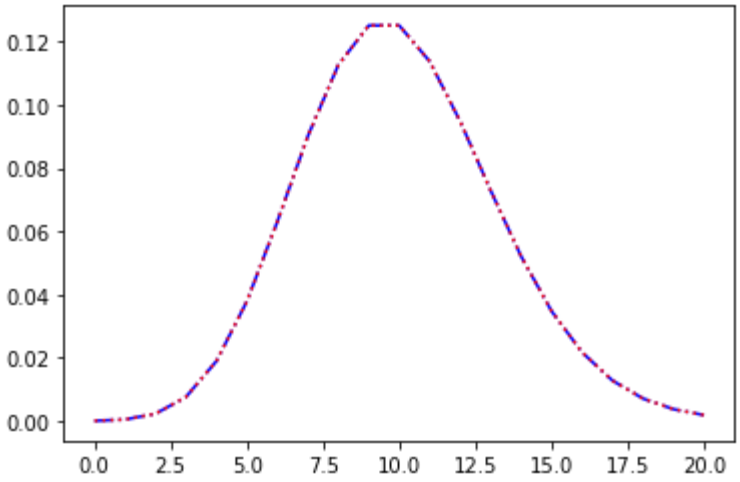
2.

D'après le cours, on a

$$\mathbb{P}(X = k) = \frac{\lambda^k}{k!} e^{-\lambda}.$$

In [42]:
```
from math import factorial
kvec = range(21)
pk = [lam**k/factorial(k)*np.exp(-lam) for k in kvec]
print(pk[:2])
print(stats.poisson.pmf(kvec,lam)[:2])#pour comparaison
%matplotlib inline
plt.figure()
plt.plot(kvec,pk,color = 'blue',linestyle  = "-.")
plt.plot(kvec,stats.poisson.pmf(kvec,lam),color = "red",linestyle =
":")
print(pk)
```

```
[4.5399929762484854e-05, 0.00045399929762484856]
[4.53999298e-05 4.53999298e-04]
[4.5399929762484854e-05, 0.00045399929762484856, 0.0022699964881242427, 0.007566654960414142, 0.018916637401035358, 0.037833274802070715, 0.06305545800345119, 0.09007922571921598, 0.11259903214901998, 0.1251100357211333, 0.1251100357211333, 0.1137363961101212, 0.09478033009176765, 0.07290794622443666, 0.05207710444602618, 0.03471806963068413, 0.021698793519177577, 0.012763996187751515, 0.007091108993195287, 0.003732162627997519 7, 0.0018660813139987598]
```

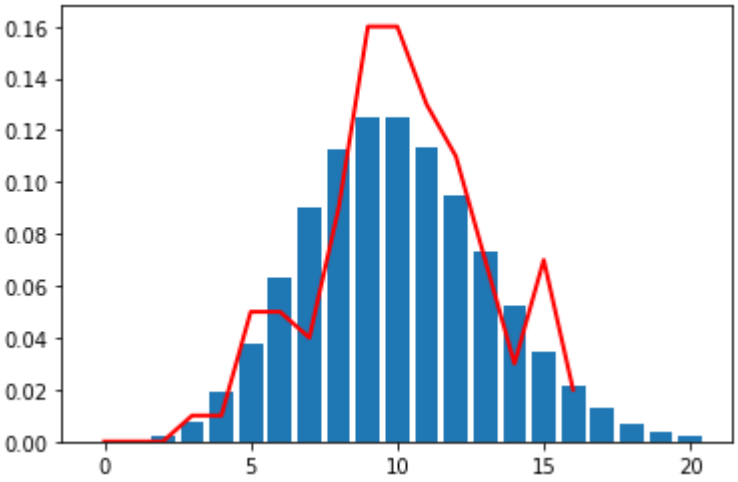


3.

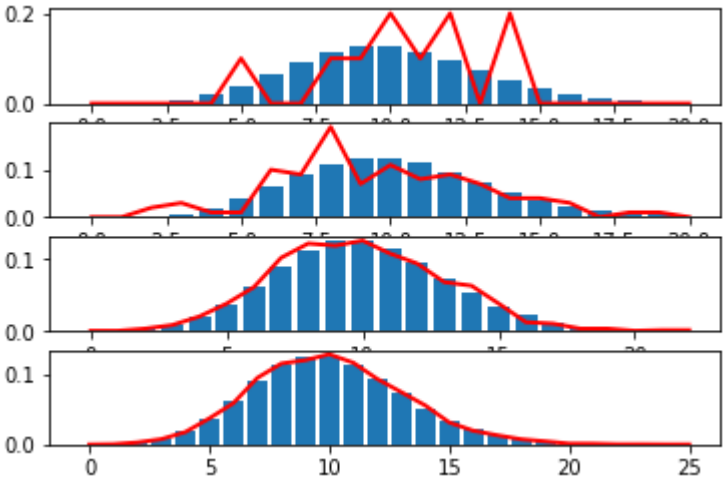
D'après le cours, on a, par la loi forte des grands nombres

$$\hat{p}_k \xrightarrow{p.s.} \mathbb{E}\left[\mathbb{1}_{X_i=k}\right] = p_k.$$

In [43]:
```
n =100
lam = 10
x = np.random.poisson(lam,n)
counts = np.bincount(x)
freq = counts/n
plt.figure()
plt.bar(range(21),stats.poisson.pmf(range(21),lam))
plt.plot(freq,color = "red",linewidth = 2)
plt.show()
```



In [44]:
```
def compare_theo_and_empirical(n,ax,lam = 10):
    x = np.random.poisson(lam,n)
    counts = np.bincount(x,minlength = 21)
    freq = counts/n
    ax.bar(range(20),stats.poisson.pmf(range(20),lam))
    ax.plot(freq,color = "red",linewidth = 2)


fig,axes = plt.subplots(4)
for i,n in enumerate([10,100,1000,10000]):
    compare_theo_and_empirical(n,axes[i],lam = 10)

plt.show()
```



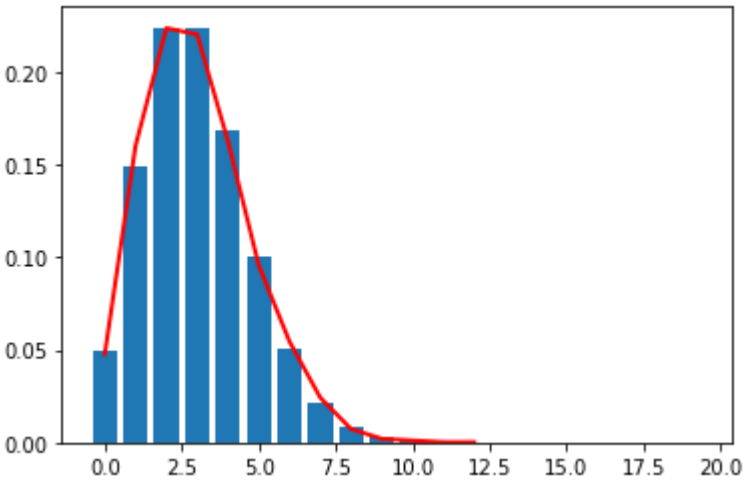
On observe effectivement la convergence.

4.

In [45]:
```
lam = 1.4
mu = 1.6
n = 10000

x= np.random.poisson(lam,n)
y= np.random.poisson(mu,n)
z = x+y

print(np.mean(z==4),stats.poisson.pmf(4,lam+mu),sep = " vs ")
```

```
0.1628 vs 0.16803135574154085
```

In [46]:
```
freq = np.bincount(z)/n
plt.figure()
plt.bar(range(20),stats.poisson.pmf(range(20),lam+mu))
plt.plot(freq,color = "red",linewidth = 2)
plt.show()
```



Exercice 2

1.

(a)

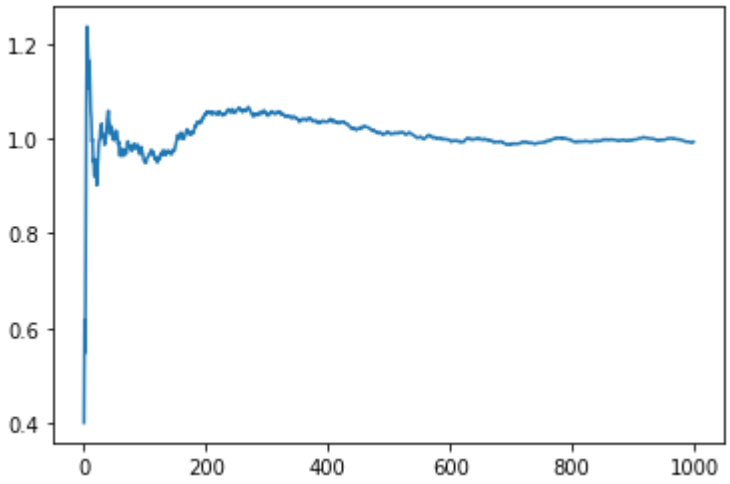
In [47]:
```
#pour les plots interactif dans notebook

N = 10**3
x = np.random.uniform(0,2,N)
```

(b)

In [48]:
```
y = np.cumsum(x)/range(1,N+1)
```

(c)

In [49]:
```
%matplotlib inline
plt.plot(y)
plt.show()
```



Il semble y avoir convergence (p.s.)

2.

(a)

On peut reconnaître la densité de la loi exponentielle, d'où

$$I = \mathbb{E}[X^5],$$

où $X \sim \mathcal{E}(1)$.

(b)

In [50]:
```
n = 10000000
x=  np.random.exponential(scale = 1,size = n)
print(np.mean(x**5))
from math import gamma
gamma(6)
```

```
120.15422976222577
```

Out[50]:
```
120.0
```

On remarque que l'on fait une erreur sur l'évaluation de cette quantité, mais cela peut-être utile pour calculer des intégrales pour lesquelles on n'a pas de valeur explicite.

3.

On reconnaît de nouveau des lois, et on identifie

$$I_2 = \mathbb{E}\left[\cos\left(Z(X^2+Y)\right)\right],$$

avec $X \sim \mathcal{E}(1)$, $Y \sim \mathcal{N}(0,1)$, $Z \sim \mathcal{U}([0,1])$, indépendantes.

La technique précédente donne :

In [51]:
```
n = 100000
x = np.random.exponential(size = n)
y = np.random.normal(size =n)
z = np.random.uniform(size = n)

print(np.mean( np.cos(  z*(x**2+y) )))
```

```
0.6217801107020385
```

Exercice 3

1.

On simule l'échantillon. On fera bien attention à la paramétrisation des fonctions, par le "scale", i.e. $s = \frac{1}{\lambda}$ selon comment a été définie la loi exponentielle. La densité considérée ici est donc

$$f_s(x) = \frac{1}{s} e^{-\frac{x}{s}}$$

In [52]:
```
lam = 2
scale = 1/lam
n = 100
x = np.random.exponential(scale = scale,size = n)
print(x[:4])
print(np.mean(x))
```

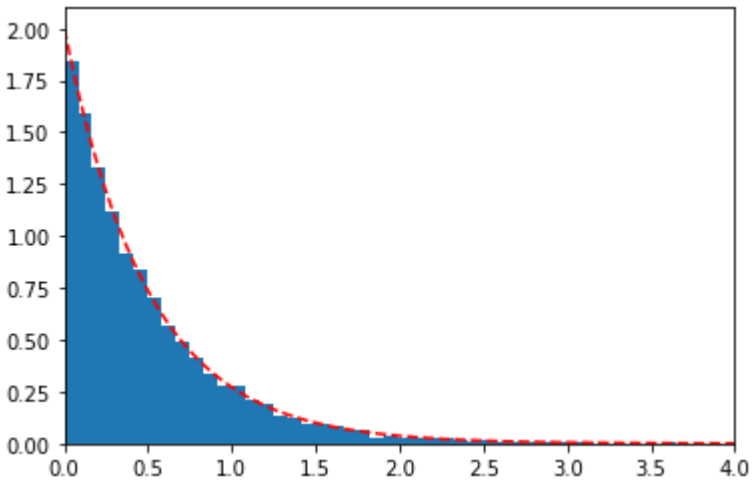
```
[0.05086701 0.34368684 0.19570821 0.26910347]
0.5013063897815807
```

2.

On peut représenter la densité théorique de la loi, ainsi qu'un histogramme

In [53]:
```
n = 10000
x = np.random.exponential(scale,n)
plt.figure()
plt.hist(x,bins = 50,density = True)
xseq = np.linspace(0,10,1000)
plt.plot(xseq,stats.expon.pdf(xseq,scale = scale),color = "red",lin
estyle ="--")
plt.xlim(0,4)
```

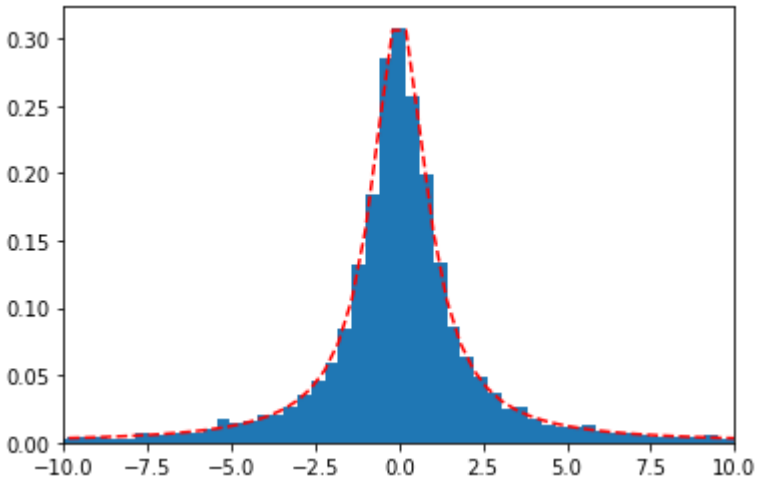
Out[53]: (0, 4)



3.

On peut comparer (au choix) la densité ou la fonction de répartition.

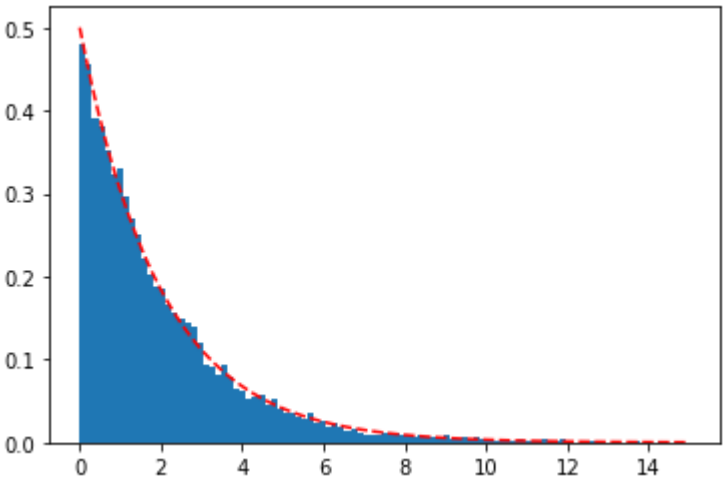
In [54]:
```
n = 10000## assez pour pouvoir illustrer les propriétés
x = np.random.normal(size= n)
y = np.random.normal(size = n)
z = x/y
xseq= np.linspace(-20,20,100)
xseq[0] = -10000
xseq[99] = 10000##juste pour avoir la bonne normalisation de l'hist
ogramme, on m'avait fait la remarque
plt.figure()
plt.plot(xseq,stats.cauchy.pdf(xseq),color = "red",linestyle ="--")
plt.xlim(-10,10)
plt.hist(z, bins = xseq,density = True)
plt.show()
```



4.

On peut faire de même pour cette propriété.

In [55]:
```
n = 10000## assez pour pouvoir illustrer les propriétés
x = np.random.normal(size= n)
y = np.random.normal(size = n)
z = x**2+y**2
plt.figure()
xseq= np.linspace(0,15,100)
plt.hist(z, bins = xseq,density = True)
plt.plot(xseq,stats.expon.pdf(xseq,scale = 2),color = "red",linesty
le ="--")
plt.show()
```

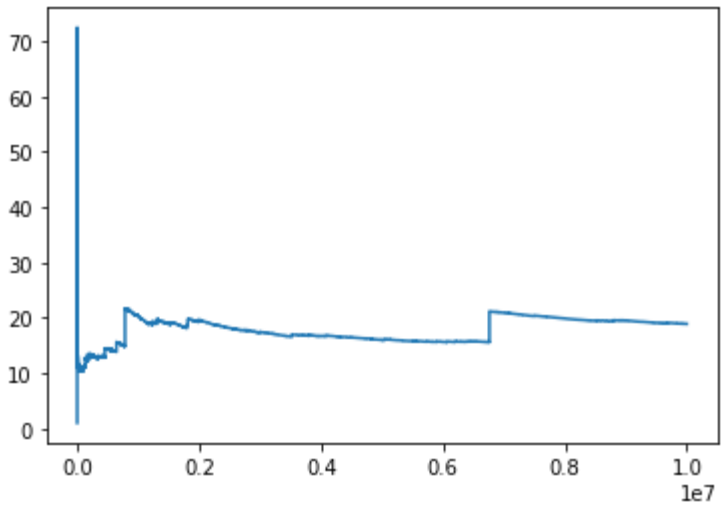


Exercice 4

1.

In [56]:
```
%matplotlib notebook
N = 10**7
x = np.random.uniform(-1,1,N)
y = np.cumsum(1/np.abs(x))/range(1,N+1)
%matplotlib inline
plt.plot(y)
```

Out[56]:
```
[<matplotlib.lines.Line2D at 0x7f3bc7c052d0>]
```



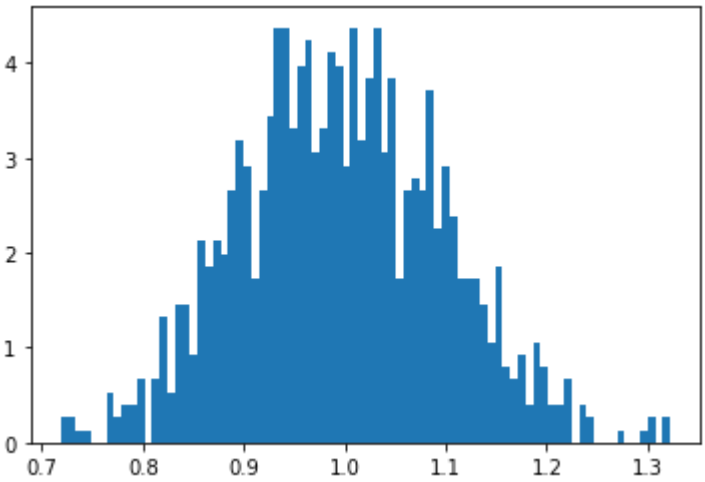
On n'observe pas de convergence. Cela signifie a priori que l'on a pas $\mathbb{E}\left[\frac{1}{|U|}\right] < +\infty$. On peut le démontrer aisément, puisque

$$\mathbb{E}\left[\frac{1}{|U|}\right] = \int_{-1}^{1} \frac{\mathrm{d}u}{2u} = +\infty$$

2.

On peut simuler plusieurs réalisations de la moyenne de 100 réalisations i.i.d. de loi exponentielles de paramètre 1.

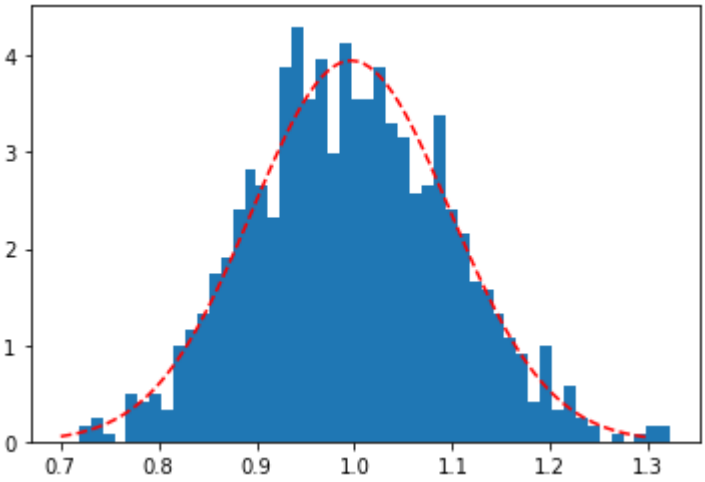
In [57]:
```
x = [ np.mean( np.random.exponential(size = 100)  )   for i in rang
e(1000)]
plt.figure()
plt.hist(x,bins = 80,density = True);
```



Cela ressemble à une loi normale, ce que l'on peut illuster

In [58]:
```
m = np.mean(x)
v = np.var(x)
plt.figure()
plt.hist(x,bins = 50,density = True)
xseq = np.linspace(0.7,1.3,100)
plt.plot(xseq,stats.norm.pdf(xseq,loc =m,scale = np.sqrt(v)),color
= "red",linestyle ="--")
```

Out[58]:
```
[<matplotlib.lines.Line2D at 0x7f3bc7259f50>]
```



In [ ]:

In [ ]: