corrigé



# Probabilités

```
import math
import numpy as np
import matplotlib.pyplot as plt
import numpy.random as rd
```

**Exercice 1**

```
def T(N, p):
    n = 1
    nb = 1
    while n < N:
        if rd.rand() < p:
            n += 1
        else:
            n = 1
        nb += 1
    return nb
```

```
def moyenne(N, p):
    s = 0
    for _ in range(10000):
        s += T(N, p)
    return s / 10000
```

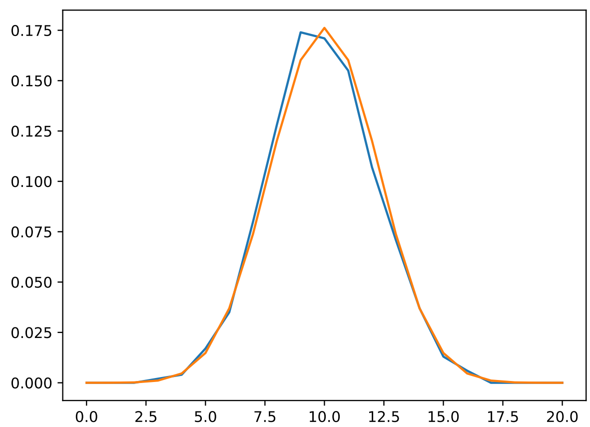
```
>>> moyenne(3, 0.7)
4.4533
>>> moyenne(6, 0.5)
63.7758
```

**Exercice 2**

```
def experience(n, p=1/2):
    s = 0
    for k in range(n):
        if rd.rand() < p:
            s += 1
    return s
```

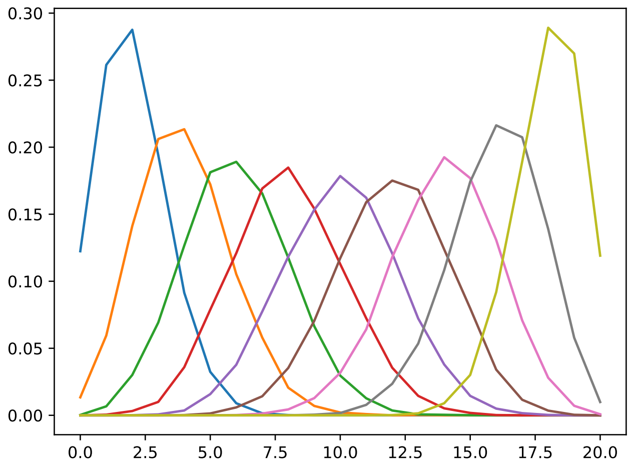
```
def simulation(n, N, p=1/2):
    t = [0 for _ in range(n+1)]
    for _ in range(N):
        t[experience(n, p)] += 1
    return [x / N for x in t]
```

```
>>> plt.plot(simulation(20, 1000))
>>> plt.plot([math.comb(20, k) / 2**20 for k in range(21)])
>>> plt.show()
```



On superpose les différentes courbes en faisant varier $p$ entre $0,1$ et $0,9$ :

```
>>> for p in range(1, 10):
...     plt.plot(simulation(20, 10000, p / 10))
>>> plt.show()
```




**Exercice 3**

```
def lancer(n, p):
    x = rd.rand() < p
    for _ in range(n - 1):
        y = rd.rand() < p
        if x and y:
            return False
        x = y
    return True
```

```
def proba(n, p):
    if n == 1:
        return 1
    if n == 2:
        return 1 - p * p
    x, y = 1, 1 - p * p
    for _ in range(n - 1):
        x, y = y, (1 - p) * y + p * (1 - p) * x
    return y
```

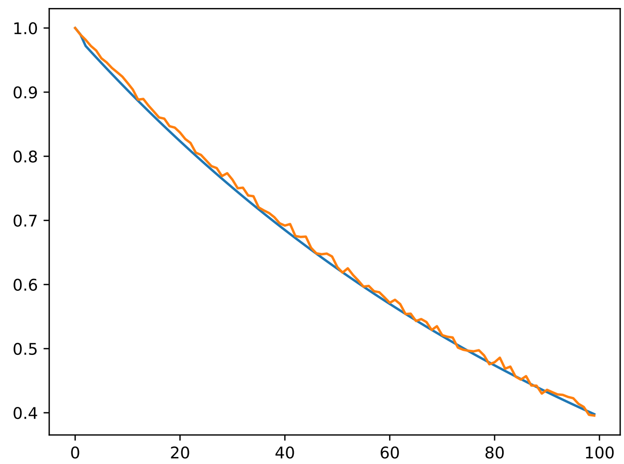
Pour estimer la probabilité on simule $\mathrm{N} = 10^4$ expériences puis on calcule le pourcentage de réussites :

```
def simul(n, p, N=10000):
    s = 0
    for _ in range(N):
        if lancer(n, p):
            s += 1
    return s / N
```

Pour comparer les valeurs théorique et pratique on choisit $p = 0,1$ et $n \in [\![1, 100]\!]$ et on supperpose les deux courbes obtenues :

```
X = [proba(n, 0.1) for n in range(1, 101)]
Y = [simul(n, 0.1) for n in range(1, 101)]

plt.plot(X)
plt.plot(Y)
plt.show()
```



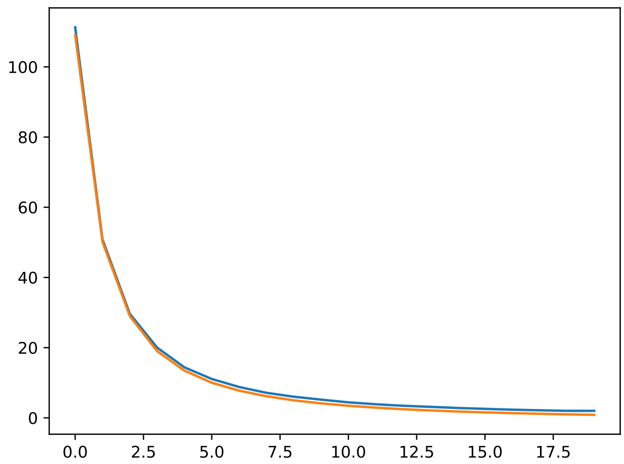

On définit d'abord une fonction qui simule la variable aléatoire T, puis une fonction qui estime l'espérance en calculant la moyenne de $10^4$ expériences.

```
def T(p):
    x = rd.rand() < p
    y = rd.rand() < p
    n = 2
    while not(x and y):
        x, y = y, rd.rand() < p
        n += 1
    return n
```

```
def E(p, N=10000):
    n = 0
    for _ in range(N):
        n += T(p)
    return n / N
```

On trace les deux courbes demandées, que l'on supperpose :

```
P = np.arange(0.1, 1.1, 0.05)
X = [E(p) for p in P]
Y = [(1+p*(1-p))/(p*p) for p in P]
plt.plot(X)
plt.plot(Y)
plt.show()
```



**Exercice 4**

```
def Y(lbda, n, p):
    N = rd.poisson(lbda)
    s = 0
    for _ in range(N):
        s += rd.binomial(n, p)
    return s
```

On estime espérance et écart type à partir de $10^4$ expériences (à l'aide de la formule de Koenig-Huyghens pour estimer la variance).

```
a, b = 0, 0
for _ in range(100000):
    y = Y(2, 10, 1 / 2)
    a += y
    b += y ** 2
esperance = a / 100000
ecartType = np.sqrt(b / 100000 - esperance ** 2)
```

```
>>> print(esperance, ecartType)
9.98216 7.403334501047483
```

C'est bien proche des valeurs théoriques $\mathbb{E}(Y) = 10$ et $\sigma(Y) = \sqrt{55} \approx 7,42$.