TD 23 correction
Variables aléatoires

Exercice 9

1. Direct par définition :

```
def Bernoulli(p):
    X = [0, 1]
    P = [1-p, p]
    return (X, P)
```

2. Aussi (de nombreuses façons de l'écrire) :

```
def uniforme(a, b):
    n = b - a + 1
    P = [1/n] * n
    X = [x for x in range(a, a+n)]
    return (X, P)
```

3. Espérance : calcul de moyenne pondérée.

```
def espérance(X, P):
    n = len(X)
    E = 0
    for i in range(n):
        E = E + P[i] * X[i]
    return E
```

Variance : König-Huygens, on calcule dans une seule boucle l'espérance et l'espérance des carrés (c.f. TP de statistiques).

```
def variance(X, P):
    n = len(X)
    E = 0
    EE = 0
    for i in range(n):
        E = E + P[i] * X[i]
        EE = EE + P[i] * X[i]**2
    return EE - E**2
```

Puis on prend la racine carrée :

```
from math import sqrt
def écarttype(X, P):
    v = variance(X, P)
    return sqrt(v)
```

4. C'est la somme des `P[i]` tels que `X[i]` soit dans l'intervalle $[a, b]$. Peu importe l'ordre dans lequel sont rangés les valeurs et les probabilités !

```
def proba_intervalle(X, P, a, b):
    n = len(X)
    p = 0
    for i in range(n):
        if a <= X[i] and X[i] <= b:
            p = p + P[i]
    return p
```

5. (a) À savoir faire (c.f. TP de statistiques), c'est la liste des cumuls.

```
def cumul(X, P):
    n = len(X)
    C = [0] * n
    C[0] = P[0]
    for i in range(1, n):
        C[i] = C[i-1] + P[i]
    return C
```

(b) On utilise alors la liste des cumuls. C'est la même idée que « simuler une variable aléatoire en écrivant le tableau des probabilités cumulées » généralisée avec des listes quelconques !

```
from random import random
def simule(X, P):
    n = len(X)
    C = cumul(X, P)
    p = random()
    i = 0
    while C[i] <= p:
        i = i + 1
    # ici i est le plus petit indice pour lequel C[i] > p
    # p est donc dans l'intervalle C[i-1] <= p < C[i]
    # dans lequel on veut renvoyer la valeur d'indice i
    return X[i]
```

6. La somme vaut `X[i] + X[j]` avec la probabilité `P[i] * P[j]` :

```
def somme(X1, P1, X2, P2):
    n = len(X1)
    m = len(X2)
    X = []
    P = []
    for i in range(n):
        for j in range(m):
            X.append(X1[i] + X2[j])
            P.append(P1[i] * P2[j])
    return (X, P)
```

C'est complètement analogue pour le produit :

```
def produit(X1, P1, X2, P2):
    n = len(X1)
    m = len(X2)
    X = []
    P = []
    for i in range(n):
        for j in range(m):
            X.append(X1[i] * X2[j])
            P.append(P1[i] * P2[j])
    return (X, P)
```

7. On applique la définition comme somme de variables de Bernoulli :

```
def binomiale(n, p):
    X = [0]
    P = [1]
    for i in range(n):
        (Y, Q) = Bernoulli(p)
        (X, P) = somme(X, P, Y, Q)
    return (X, P)
```