Probabilités, expériences répétées, cours, terminale spécialité Mathématiques

1 Dénombrements

1.1 k-uplets

Définitions :

Soit E un ensemble.

- Soit k un entier naturel non nul et a_1, a_2,...,a_k k éléments de E. la suite ordonnée et finie d'éléments $(a_1; a_2; ...; a_n)$ est appelé *k-uplet*. En particulier, si $k = 2$, $(a_1; a_2)$ est appelé ..... d'éléments et si $k = 3$, $(a_1; a_2; a_3)$ est appelé ..... .
- L'ensemble des k-uplets est appelé *produit cartésien* des k ensembles E et est noté ..... .

Propriété :

Soit E un ensemble de n éléments et k un entier naturel non nul. Le produit cartésien E^k est constitué de ..... k-uplets.

1.2 Combinaisons

Définition :

Soit n un entier supérieur ou égal à 1. On appelle *factorielle n* le nombre noté $n!$ défini par $n!$ =.....

Par convention $0! = 1$.

Exemple :

$5!$ =.....

Définition :

n et p désignent des entiers tels que $0 \leq p \leq n$ et E un ensemble à n éléments. Une combinaison de p éléments de E est ....

Exemple :

Si $E = \{1; 2; 3; 4; 5\}$, alors $F = \{3; 5\}$ et $G = \{1; 3\}$ sont des combinaisons de 2 éléments de E.

Propriété :

Soient n et p deux entiers naturels tels que $0 \leq p \leq n$ et E un ensemble à n éléments. Le nombre de combinaisons de p éléments de E, noté ..... (lire p parmi n) est donné par

.....

Preuve :

Admise

Exemple :

Nombre de combinaisons de 4 éléments parmi 7 c'est à dire de sous ensembles de 4 éléments dans un ensemble de 7 éléments :

$\binom{7}{4}$ =.....

Utilisation de la calculatrice :

Casio : Touche nCr. Taper sur OPTN puis PROB puis nCr.
Syntaxe : 7 nCr 4

TI : Touche Combinaison. Taper sur math puis aller dans le menu PROB puis choisir Combinaison.
Syntaxe : 7 Combinaison 4

Numworks : menu boîte à outils puis Dénombrement puis binomial(n,k) (Attention : nom mal choisi, il s'agit de coefficient binomial pas d'une loi binomiale).

1.3 Propriétés des combinaisons

Propriété :

Pour tous les entiers n et p tels que $0 \leq p \leq n$,

$$\binom{n}{n-p} = ...$$

Preuve :

Soit E un ensemble à n élément. Á toute partie de E on associe de manière unique sa partie complémentaire $\bar{A}$. A contient p éléments si et seulement si $\bar{A}$ contient $n-p$ éléments. Il y a donc autant de parties à p éléments dans E que de parties à $n-p$ éléments.

Propriété :

Pour tous les entiers n et p tels que $1 \leq p \leq n-1$,

$$\binom{n-1}{p-1} + \binom{n-1}{p} = ...$$

Preuve :

Soit E un ensemble à n éléments. Soit a un élément de E. Les parties de E à p éléments se répartissent en deux catégories :

- les parties F à p éléments qui ne contiennent pas l'élément a : il y en a donc autant que de parties à p éléments dans un ensemble à $n-1$ éléments, c'est à dire $\binom{n-1}{p}$
- les parties G à p éléments qui contiennent l'élément a : il en a autant que de parties à $p-1$ éléments dans un ensemble à $n-1$ éléments, c'est à dire $\binom{n-1}{p-1}$.

Comme il y a au total $\binom{n}{p}$ parties à p éléments dans E, on obtient le résultat.

Triangle de Pascal :

Le triangle de Pascal, du nom de Blaise Pascal, mathématicien français du XVII^{e} siècle qui le« redécouvra » en Occident (car il était connu avant en Orient et au Moyen-Orient) est une disposition permettant de visualiser et de calculer les coefficients binomiaux et qui s'appuie sur la formule précédente.

$\binom{0}{0} = 1$					
$\binom{1}{0} = 1$	$\binom{1}{1} = 1$				
$\binom{2}{0} = 1$	$\binom{2}{1} = 2$	....			
$\binom{3}{0} = 1$	$\binom{3}{1} = 3$	....	....		
1	....	....	....	1	
1	....	....	....	....	1

Explication de la construction : le nombre de la ligne n et de la colonne k est le coefficient binomial $\binom{n-1}{k-1}$. Il est obtenu en ajoutant le nombre situé au dessus (ligne $n-1$ et colonne k) au nombre de la colonne et de la ligne précédente (ligne $n-1$ et colonne $k-1$).
Par exemple, $\binom{3}{1} = 3$ est la somme de $\binom{2}{1} = 2$ et de $\binom{2}{0} = 1$.

1					
1	1				
1	....	1			
1	....	....	1		
1	....	....	....	1	
1	....	....	....	....	1

Exemple de programmation du triangle de Pascal en langage python :

```
def trianglePascal(n):
        L=[1]
        L=L+[0 for k in range(n)]
        t=[L]
        for k in range(1,n+1):
                L=[1]
                for l in range(1,k+1):
                        L.append(t[k-1][l]+t[k-1][l-1])
                L=L+[0 for l in range(k,n)]
                t=t+[L]
        return t
```

Propriété :

Soit E un ensemble à n éléments où $\in \mathbb{N}$. Alors E est constitué de ..... parties.

Preuve :

Admise

2 Schéma et loi de Bernoulli

Définition :

On considère une expérience aléatoire ne comportant que deux issues ; l'une appelée « succès » et l'autre appelée « échec ». On note p la probabilité de succès avec $p \in [0;1]$ et q la probabilité d'échec ($q = 1-p$). Soit X la variable aléatoire qui prend pour valeurs 0 et 1 avec pour probabilités respectives p et $q = 1-p$. On dit que X suit ....

Définition :
Deux expériences sont dites ..... si le résultat de l'une n'influe pas sur le résultat de l'autre.

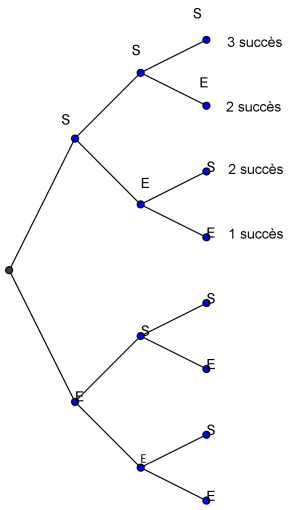
Exemple :
il y a ..... lorsqu'on lance deux fois de suite une pièce de monnaie.

Définition :
On considère une expérience de ..... de probabilité de succès p. La répétition de cette expérience n fois de manière ..... constitue .... de paramètres n et p.

Propriété :
Dans un schéma de Bernoulli, la probabilité d'une liste de résultats est le ..... des probabilités de chaque résultat

Exemple :

On contrôle la qualité d'un produit sur une chaîne de production. On prélève 3 produits au hasard. On suppose que les prélèvements sont indépendants. Statistiquement, chaque produit a une probabilité $0,05$ d'être défectueux.



Sur l'arbre ci-dessus représentant un schéma de Bernoulli de paramètres $n =$ ..... et $p =$ ....., la probabilité d'avoir les deux premières expériences qui donnent un succès et la dernière qui donne un échec est $P(SS\bar{S}) =$.....
soit ..... de chances d'avoir deux produits défectueux sur les trois prélevés.

Propriété :

Si X est une variable aléatoire qui suit une loi de Bernoulli de paramètre$p \in [0;1]$, alors :

- $E(X) =$.....;
- $V(X) =$.....;
- $\sigma(X) =$..... .

Preuve :

- $E(X) =$.....
- $V(X) =$.....

  .....
- $\sigma(X) =$.....

Algorithmique :

Algorithme de simulation d'une épreuve de Bernoulli de paramètre p.

Données : p : nombre décimal entre 0 et 1 ;
Début traitement
  t prend une valeur aléatoire décimale entre 0 inclus et 1 exclu ;
  si $t < p$ **alors**
    Renvoyer "Succès" ;
  fin
  sinon
    Renvoyer "Échec" ;
  fin
Fin

Exemple de programmation en langage python :

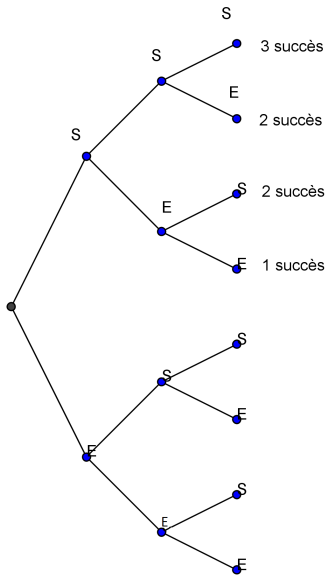
```
import random
def bernoulli(p):
        t = ........
        if  ....  :
                return 1
        else:
                return 0
```

3 Loi binomiale

Propriété :

Soient n et k deux entiers naturels avec $k \leq n$. Le nombre de manières d'obtenir k succès et $n-k$ échecs pour n répétitions indépendantes de la même expérience de Bernoulli est .....

Exemple :



$\binom{3}{3}$ =..... : il y a une seule manière d'obtenir 3 succès lors de la répétition de 3 épreuves identiques indépendantes.

$\binom{3}{2}$ =..... : il y a trois manières d'obtenir 2 succès et un échec lors de la répétition de 3 épreuves identiques indépendantes (.....; .....; .....).

Définition :

On considère une épreuve de Bernoulli de paramètre $p \in [0;1]$. On répète n fois ($n \geq 1$) cette expérience indépendamment et on note X la variable aléatoire qui compte le nombre de succès. On dit alors que la variable aléatoire X suit une loi .... et on note $X \sim$..... .

Propriété :

Si X est une variable aléatoire qui suit une loi binomiale de paramètres n et p, alors la probabilité d'obtenir k succès avec $k \in \{0;1;2;...;n\}$ est

.....

Preuve :

Il y a $\binom{n}{k}$ manières d'obtenir k succès dans n répétitions d'expériences identiques et indépendantes. La probabilité de chacune de ces événements qui sont évidemment incompatibles est ..... d'où le résultat.

Exemple :

On considère le problème précédent de test des produits d'une chaîne de production. Les prélèvements étant supposés indépendants les uns des autres, l'expérience constitue un schéma de Bernoulli de paramètres $n = 3$ et $p = 0,05$. La variable aléatoire X qui compte le nombre de succès suit la loi binomiale de paramètres $n = 3$ et $p = 0,05$

On a $P(X = 2) =$.....

car trois chemins permettent d'obtenir deux succès c'est à dire deux objets défectueux.

D'où $P(X = 2) =$.....

donc $P(X = 2) =$.....

soit une probabilité très faible de ..... d'avoir deux produits défectueux.

Calcul pratique de $P(X = k)$ et $P(X \leq k)$:

Soit X une variable aléatoire de paramètres n et p. Pour k allant de 0 à n, pour calculer $P(X = k)$ ou $P(X \leq k)$, on utilise une calculatrice :

- Sur Texas instrument : aller dans le menu [2nd] [distrib], choisir [binomFdp] et taper [n] [,] [p] [,] [k] [)] pour calculer $P(X = k)$ et choisir [binomFRép] et taper [n] [,] [p] [,] [k] [)] pour calculer $P(X \leq k)$.
- Sur Casio : aller dans le menu [STAT] puis [DIST] puis [BINM]. Sélectionner alors [Bpd] puis [Var] pour variable, puis entrer alors k dans la ligne « x », n dans la ligne « numtrial » et p dans la ligne « p » puis aller sur « execute » pour valider et calculer ainsi $P(X = k)$. Pour le calcul de $P(X \leq k)$, on utilisera [Bcd] au lieu de [Bpd].

Exemple :

On considère une variable aléatoire X suivant la loi binomiale de paramètres $n = 4$ et $p = 0,4$.

Sur TI, la probabilité $P(X \leq 3)$ est donnée par `binomFrép(4,0.4,3)`.

Remarques :

Pour tout entier naturel k non nul :

- On a $P(X < k) =$.....
- Pour calculer $P(X > k)$, on calcule ..... .
- Pour calculer $P(X \geq k)$, on calcule ..... .
- Pour calculer $P(a \leq X \leq b)$ avec a et b entiers naturels tels que $a \leq b$, on calcule ..... .

Algorithmique :

Algorithme de simulation d'une loi binomiale de paramètres n et p.

Données : p : nombre décimal entre 0 et 1 ; n : entier naturel ;
Début traitement
  $c \leftarrow 0$;
  pour k *de 1 jusque* n **faire**
    t prend une valeur aléatoire décimale entre 0 inclus et 1 exclu ;
    si $t < p$ **alors**
      $c \leftarrow c + 1$;
    fin
  fin
Fin

Exemple de programmation en langage python :

```
import random
def simulBinomiale(n,p):
        c=0
        for k in range(1,n+1):
                t=random.random()
                if t<p:
                c=c+1
        return c
```