Sts-Ciel 2 Loi binomiale- Python

Une entreprise fabrique en très grande série une pièce technique. Le processus de fabrication n'est pas parfait et statistiquement, on a constaté que 5% des pièces avaient un défaut.

Pour vérifier que ce pourcentage ne s'aggrave pas, on constitue quotidiennement un échantillon de n pièces, en sortie de production (par tirage au sort).

On répète ainsi n fois un tirage au sort. Dans cette expérience aléatoire, la probabilité p de prélever une pièce défectueuse est $p = 0.05$. Une variable aléatoire X repère le nombre de pièces défectueuses dans l'échantillon. Pour calculer la probabilité $p(X = k)$ d'avoir k pièces défectueuses dans ce lot, on utilise les propriétés liées à une loi Binomiale $\mathcal{B}(n, p)$:

$$p(X = k) = \binom{n}{k} \; p^k \; (1-p)^{n-k}$$

Le coefficient binomial peut être calculé avec la relation :

$$\binom{n}{k} = \frac{n!}{k! \; (n-k)!}$$

L'objectif de ce Tp est de coder en python afin de pouvoir calculer $p(X = k)$.

⇨ Ouvrir un nouveau fichier sur Pyzo et l'enregistrer sous le nom *loiBinomiale_monNom.py* .

1. : FONCTION FACT() :

La fonction *fact()* donnée ci-contre est incomplète. Elle a comme paramètre un nombre entier n et renvoie le nombre $n! = n \times (n-1) \times (n-2) \times \ldots \times 2 \times 1$

```
def fact(n) :



    return
```

⇨ Ecrire ce code en le complétant, dans votre fichier *loiBinomiale _monNom.py*.

⇨ Vérifier le bon fonctionnement de votre script en exécutant dans la console les tests suivants :

Tests simples :

```
>>> fact(3)
6
```

```
>>> fact(8)
40320
```

Tests avec des valeurs limites :

```
>>> fact(0)
1
```

```
>>> fact(1)
1
```

```
>>> fact(20)
2432902008176640000
```

2. : FONCTION COEFBIN() :

La fonction *coefBin()* donnée ci-contre est incomplète. Elle a comme paramètre 2 nombres entiers n et k . Elle renvoie le nombre

$$\binom{n}{k} = \frac{n!}{k!\ (n-k)!}$$

```
def coefBin(n,k) :


    return
```

⇨ Ecrire ce code en le complétant, dans votre fichier *loiBinomiale _monNom.py*. Utiliser bien sur la fonction *fact()* mises au point précédemment0

⇨ Vérifier le bon fonctionnement de votre script en exécutant dans la console les tests suivants :

Tests simples :

```
>>> coefBin(8,4)
70
```

```
>>> coefBin(2,1)
2
```

Tests avec des valeurs limites

```
>>> coefBin(1,1)
1
```

```
>>> coefBin(100,100)
1
```

```
>>> coefBin(100,1)
100
```

```
>>> coefBin(100,50)
100891344545564202071714955264
```

3. : FONCTION LOIBIN() :

La fonction *loiBin()* donnée ci-contre est incomplète. Elle a comme paramètre 2 nombres entiers n , k et un nombre réel p. Elle renvoie le nombre

```
def loiBin(n,p,k) :


    return
```

$$p(X = k) = \binom{n}{k}\ p^k\ (1-p)^{n-k}$$

⇨ Ecrire ce code en le complétant, dans votre fichier *loiBinomiale _monNom.py*. Utiliser bien sur les fonctions *fact()* et *coefBin()* mises au point précédemment. Pour rappel, l'opération puissance en python est repérée par ** . par exemple $0.05^{15} = 0.05 ** 15$

⇨ Vérifier le bon fonctionnement de votre script en exécutant dans la console les tests suivants :

Test simples: Pour calculer $p(X = 1)$ sur la loi Binomiale $\mathcal{B}(5,0.05)$ on exécute :

```
>>> loiBin(5,0.05,1)
0.2036265624999997
```

On a ainsi $p(X = 1) \approx 0{,}204$

⇨ Utiliser votre code pour calculer les probabilités suivantes :

x_i	0	1	2	3	4	5
	$p(X=0)$	$p(X=1)$	$p(X=2)$	$p(X=3)$	$p(X=4)$	$p(X=5)$

4. : FONCTION LOIBINCUMUL() :

La fonction *loiBinCumul()* donnée ci-contre est incomplète. Elle a comme paramètre 2 nombres entiers n , k et un nombre réel p. Elle renvoie le nombre

```
def loiBinCumule(n,p,k) :
    cumul = 0


    return cumul
```

$$p(X \leq k) = p(X = 0) + p(X = 1) + \cdots . + p(X = k)$$

⇨ Ecrire ce code en le complétant, dans votre fichier *loiBinomiale _monNom.py*. Utiliser bien sur la fonction *loiBin()* mise au point précédemment.

⇨ Vérifier le bon fonctionnement de votre script en exécutant dans la console les tests suivants :

Test simples: Pour calculer $p(X \leq 1)$ sur la loi Binomiale $\mathcal{B}(5,0.05)$ on exécute :

```
>>> loiBinCumule(5,0.05,1)
0.9774074999999998
```

On a ainsi $p(X \leq 1) \approx 0{,}977$

⇨ Utiliser votre code pour calculer les probabilités suivantes :

x_i	0	1	2	3	4	5
	$p(X \leq 0)$	$p(X \leq 1)$	$p(X \leq 2)$	$p(X \leq 3)$	$p(X \leq 4)$	$p(X \leq 5)$

5. : FONCTION DIAGRAMME() :

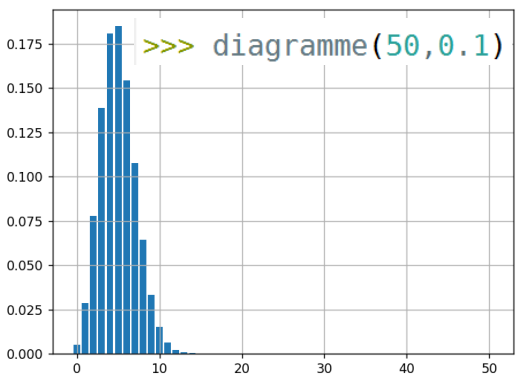
La fonction *diagramme()* donnée ci-contre est incomplète. Elle a comme paramètre un nombre entier n et un nombre réel p. Elle permet d'afficher le diagramme barre qui donne l'évolution des probabilités $p(X = k)$ en fonction de k .

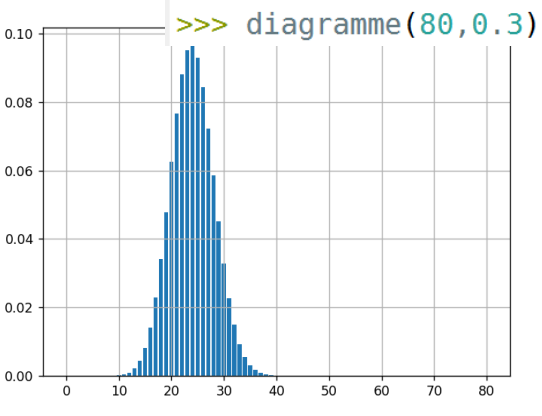
```
def diagramme(n,p) :
    import matplotlib.pyplot as plt


    plt.grid()
    plt.show()
```

⇨ Ecrire ce code en le complétant, dans votre fichier *loiBinomiale _monNom.py*. Utiliser bien sur la fonction *loiBin()* mise au point précédemment. (voir *Aide* ci-après)

⇨ Vérifier le bon fonctionnement de votre script en exécutant dans la console les tests suivants :

Tests simples :





AIDE : Pour le tracé d'un diagramme barre, on utilise la bibliothèque de Python *matplotlib*. Consulter la page https://sites.google.com/view/aide-python/graphiques/diagrammes-en-barres pour connaitre les commandes à utiliser. Python est un langage de programmation qui offre un ecosystème de bibliothèques extrêmement riche.

6. : FONCTION STAT() :

On voit dans le poly de cours sur la loi Binomiale, en page 9, la notion d'espérance $E(X)$ et celle d'écart-type $\sigma(X)$:

ESPERANCE : L'espérance $E(X)$ de la variable aléatoire X , est la valeur que prend en moyenne X. Pour une loi binomiale $\mathcal{B}(n,p)$, on peut montrer que :

$$E(X) = n\,p$$

ECART-TYPE : Pour chaque échantillon constitué, on obtient une valeur de X. La dispersion de ces valeurs autour de la moyenne $E(X)$ est quantifiée par l'écart-type $\sigma(X)$. Pour une loi binomiale $\mathcal{B}(n,p)$, on peut montrer que :

$$\sigma(X) = \sqrt{n\,p\,(1-p)}$$

La fonction *stat()* donnée ci-contre est incomplète. Elle a comme paramètre un nombre entier n et un nombre réel p. Elle permet d'afficher l'espérance $E(X)$ et celle de l'écart-type $\sigma(X)$ sous un format précis.

```
def stat(n,p) :
    from math import sqrt


    print(f"""
Pour une loi binomiale B({n} , {p}),


          """)
```

⇨ Ecrire ce code en le complétant, dans votre fichier *loiBinomiale _monNom.py*.

⇨ Vérifier le bon fonctionnement de votre script en exécutant dans la console les tests suivants :

Test simples: Avec une loi Binomiale $\mathcal{B}(50\,,0.1)$ on exécute :

```
Pour une loi binomiale B(50 , 0.1),
  - l'espérance E(X) est de 5.0
  - l'écart-type est de 2.1
```

Avec une loi Binomiale $\mathcal{B}(80\,,0.3)$ on exécute :

```
Pour une loi binomiale B(80 , 0.3),
  - l'espérance E(X) est de 24.0
  - l'écart-type est de 4.1
```

7. : FINALISATION :

⇨ Déposer votre fichier *loiBinomiale _monNom.py* sur le réseau pédagogique dans le répertoire *Devoir*.