⋆ Planche d'exercices d'oraux Python ⋆

Exercice 1

En utilisant `rd.random()`, mais sans `rd.binomial`, écrire une foncton d'en-tête `def binomiale(n,p)` qui simule une réalisation d'une loi $\mathscr{B}(n,p)$.

Exercice 2

1. Compléter la fonction suivante pour qu'elle simule une réalisation d'une loi géométrique de paramètre p.

```python
def geom(p):
    y = 1
    while rd.random() ----- :
        y = -----
    return ( ----- )
```

2. On dit qu'une variable suit la loi binomiale négative de paramètres n et p si elle a la même loi que $\sum_{i=1}^{n} X_i$ où les X_i sont des variables i.i.d. suivant la loi $\mathscr{G}(p)$.
   Écrire un programme d'entête `def bin_neg(n,p)` qui simule une réalisation d'une loi binomiale négative de paramètres n et p.

Exercice 3 - *Questions de cours*

1. Définition et représentation graphique de la fonction partie entière.
2. Donner un programme `Python` permettant de représenter la fonction partie entière sur $[-5/2, 5/2]$.

Exercice 4

Soit a un réel strictement positif.

1. Étudier la nature de la série $\sum_{n\in\mathbb{N}}(\arctan(n+a) - \arctan(n))$.
2. Proposer un programme `Python` permettant d'en donner une valeur approchée à 10^{-3} près quand $a = \frac{1}{2}$.
   On rappelle que la fonction `arctan(x)` sous `numpy` renvoie la valeur de $\arctan(x)$.

Exercice 5

1. Indiquer l'allure du graphe de la fonction de répartition e la loi normale centrée réduite.
2. En `Python`, la fonction `norm.cdf` (respectivement `norm.ppf`) de la librairie `scipy.stats` permet de déterminer les valeurs de la fonction de répartition d'une loi normale (respectivement de sa bijection réciproque). Voici deux exemples d'utilisation :

```
In [1]:   import scipy.stats as ss
          ss.norm.cdf(1.96,0,1)

Out[1]:   0.9750021048517795
```

```
In [2]:   ss.norm.ppf(0.975,0,1)

Out[2]:   1.959963984540054
```

   (a) Indiquer les sorties `Python` consécutives aux 3 entrées suivantes :

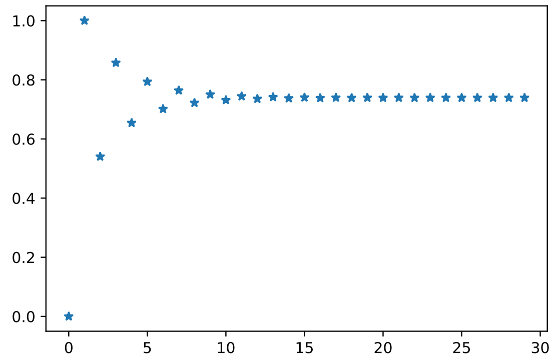
```
In [1]:   ss.norm.cdf(0,0,1)

In [2]:   ss.norm.ppf(0.5,0,1)

In [3]:   ss.norm.ppf(0.025,0,1)
```

   (b) Expliquer le script suivant et indiquer une estimation de la valeur affectée à `p` à l'issue de l'exécution du script.

```python
import numpy as np
import numpy.random as rd
n = 1000
X = np.zeros(n)
for i in range(n) :
    X[i] = rd.normal(0,1)*rd.binomial(1,0.5)
p = np.sum(X < 1.96)/n
```

Exercice 6 On exécute le programme `Python` suivant qui retourne la courbe ci-dessous :

```python
import numpy as np
import matplotlib.pyplot as plt
nmax = 30
U = np.zeros(nmax)
U[0] = 0
for i in range(nmax-1) :
    U[i+1] = np.cos(U[i])
plt.plot(range(nmax), U, '*')
```



Justifier le résultat obtenu.

Exercice 7

Le script `Python` suivant permet de simuler une épreuve aléatoire consistant à effectuer des tirages successifs dans une urne dont le contenu est lui-même aléatoire.

```python
import numpy.random as rd
p = 1/2
U = [1] # contenu initial de l'urne
N = 1
while rd.random() > p :
   U.append(0)
   N += 1
C = U[rd.randint(0,N)] # couleur de la boule tiree
```

1. Détailler l'épreuve simulée.
2. Quelles sont les lois des variables aléatoires simulées par N et C ?

Exercice 8

On considère un circuit électronique avec 3 composants C_1, C_2 et C_3. Ce circuit ne fonctionne qui si C_1 fonctionne ainsi que C_2 ou C_3. Sachant que les durées de vie de chaque composant, supposées mutuellement indépendantes, suivent une loi exponentielle de paramètre $\lambda > 0$, déterminer la loi de la durée de vie du circuit complet.
Proposer un programme `Python` permettant de vérifier le résultat obtenu.

Exercice 9

Soit n un nombre entier supérieur ou égal à 2. On dit qu'une matrice carrée $M \in \mathcal{M}(\mathbb{R})$ est *nilpotente* s'il existe un entier $k \in \mathbb{N}^*$ tel que M^k est la matrice nulle.

1. (a) Donner un exemple de matrice nilpotente non nulle de $\mathcal{M}_2(\mathbb{R})$.
   (b) Démontrer que la seule matrice nilpotente et diagonalisable de $\mathcal{M}_n(\mathbb{R})$ est la matrice nulle.
2. On appelle *indice de nilpotence* d'une matrice nilpotente M le plus petit entier strictement positif k tel que M^k est la matrice nulle.
   La fonction `Python` suivante, dont le code est incomplet, permet de calculer l'indice de nilpotence d'une matrice nilpotente.

```python
import numpy as np
????
def indnilp(A,n) :
    """n est le nombre de lignes et de colonnes de A"""
    k=1
    B=A
    while np.sum(np.abs(B))>0 :
        k = ????
        B = ????
    return k
```

   (a) Expliquer en détail la ligne de code `while np.sum(abs(B))>0`.
   (b) Compléter le code de la fonction `indnilp`.

Exercice 10 Soit le script `Python` suivant :

```
def X(p) :
    y = 0
    u = 1
    while u > p :
        y = y+1
        u = rd.random()
    return y
p = int(input('p='))
q = int(input('q='))
Y = X(p)
Z = X(q)
M = np.array([[1,2],[Y,Z]])
print(M)
```

1. Expliquer le script.
2. On exécute le script. Quelle est alors la probabilité que M soit inversible ?

Bonus. Proposer une méthode et un script permettant de vérifier cela empiriquement.

Exercice 11 Soient a, b deux réels. Pour tout $n \in \mathbb{N}^*$, on pose :

$$u_n = \ln(n) + a\ln(n+1) + b\ln(n+2).$$

Déterminer des conditions nécessaires et suffisantes sur a et b pour la série de terme général u_n converge.
Écrire un script `Python` indiquant si la série converge et permettant, en cas de divergence, de déterminer le plus petit entier n tel que $\left|\sum_{k=1}^{n} u_k\right| > 100$.
En cas de convergence, calculer la somme.

Exercice 12 Soit le script `Python` suivant :

```
plt.plot([-1,0.01],[0,0])
x = np.linspace(0,3,300)
y = x**2/2
y = 1 - np.exp(-y)
plt.plot(x,y)
```

L'exécution de ce script permet d'obtenir la représentation graphique sur $[-1, 3]$ d'une fonction F. On suppose que l'expression de F proposée sur $[-1, 0]$ est en fait valable sur $\mathbb{R}_-$ et que celle proposée sur $]0, 3]$ est valable sur $\mathbb{R}_+^*$.

1. Tracer la courbe représentative de F. On donne $\exp\left(-\frac{1}{2}\right) \approx 0,6$.
2. F est-elle la fonction de répartition d'une variable aléatoire X ? Si oui, quelle est l'espérance de X si elle existe ?
3. On suppose connue une fonction `Python` nommée `simul` permettant de simuler X. Écrire un script donnant une valeur approchée de π.

Exercice 13 On considère la fonction `Python` suivante :

```
def simul(n,a) :
    Y = rd.poisson(a,n)
    T = np.zeros(n)
    S = 0
    for k in range(n):
        S = S+1/(1+Y[k])
        T[k] = S/k
    plt.plot(T)
```

À quoi peut-on s'attendre lors de l'appel de cette fonction quand $a \in \mathbb{R}_+^*$ et n est un entier plus grand que 1000 ?

Exercice 14 Deux urnes A et B contiennent initialement chacune deux boules numérotées 0 et 1. Leur contenu évolue lors d'une expérience aléatoire et est simulé par un vecteur ligne `Python`. On considère la fonction `Python` suivante qui permet de simuler une variable aléatoire X :

```
def simulX():
    A, B = [0,1]*2
    Y = rd.randint(0,1,(4,2))
    k = 0
    while (sum(A)>0) and (k<4):
        k = k+1
        i = Y[k-1,0]
        j = Y[k-1,1]
        c = A[i]
        A[i] = B[j]
        B[j] = c
    return k
```

Expliquer le protocole ainsi simulé et donner la loi de X.