# DS 7

## Durée 3h10

- Les calculatrices sont interdites durant les cours, TD et *a fortiori* durant les DS de mathématiques.
- Si vous pensez avoir découvert une erreur, indiquez-le clairement sur la copie et justifiez les initiatives que vous êtes amenés à prendre.
- Une grande attention sera apportée à la clarté de la rédaction et à la présentations des solutions. (Inscrivez clairement en titre le numéro de l'exercice, vous pouvez aussi encadrer les réponses finales.)
- Vérifiez vos résultats.
- Le résultat d'une question peut être admis et utilisé pour traiter les questions suivantes en le signalant explicitement sur la copie.

**Exercice 1.** Soit $f$ la fonction définie par

$$f(x) = \frac{(x-1)\sin(x)}{x\ln(x)}$$

1. Déterminer l'ensemble de définition de $f$.
2. Déterminer les limites de $f$ au bord de cet ensemble.
3. $f$ est-elle prolongeable par continuité en $0$ ?

**Exercice 2.** 1. Montrer que $\displaystyle\int_0^1 \ln(x+1)dx = 2\ln(2) - 1$

2. Soit $P_n = \dfrac{1}{\sqrt{n}}\left(\displaystyle\prod_{k=1}^{n}(n+k)\right)^{\frac{1}{2n}}$. Montrer que pour otut $n \geq 1$ on a :

$$\ln(P_n) = \frac{1}{2n}\sum_{k=1}^{n}\ln\left(1+\frac{k}{n}\right)$$

3. En déduire la valeur de la limite de $P_n$ quand $n \to +\infty$.

**Exercice 3.** On suppose que $f$ est une fonction définie sur $[0,1]$ à valeurs dans $[0,1]$ et qu'il existe $k \in ]0,1[$ tel que

$$\forall (x,y) \in [0,1]^2,\ |f(x) - f(y)| \leq k|x-y|.$$

Une telle fonction s'appelle une fonction $k$-contractante.

1. Montrer que $f$ est continue sur $[0,1]$.
2. On appelle point fixe de $f$, un réel $x \in [0,1]$ tel que $f(x) = x$
   Déduire de la question précédente que $f$ admet au moins un point fixe.
3. Montrer par l'absurde que ce point fixe est unique. On le note $c$ dans le reste de l'énoncé.
4. On considère alors une suite $(c_n)_{n\in\mathbb{N}}$ définie par son premier terme $c_0 \in [0,1]$ et par la relation de récurrence : $\forall n \in \mathbb{N},\ c_{n+1} = f(c_n)$.
   (a) Montrer que pour tout $n \in \mathbb{N}$, $|c_n - c| \leq k^n|c_0 - c|$. (On rappelle que $c$ désigne l'unique point fixe de $f$)
   (b) En déduire la limite de la suite $(c_n)_{n\in\mathbb{N}}$.

**Exercice 4.** Le petit Tchoupi fait des nuits de longueur différentes. Il dort moins de 6 heures avec probabilité $p \in ]0,1[$ et plus de 6 heures avec une probabilité $1-p$. En fonction de la durée de sommeil du petit Tchoupi, M G. fait parfois des erreurs de signes dans ses calculs... A chaque nouveau calcul, il a une probabilité $\frac{1}{10}$ de faire une faute si il a bien dormi et $\frac{1}{2}$ si la nuit a été courte... A une journée fixée, les calculs sont supposés indépendants entre eux. On note $E_k$ l'événement $\{$ M G. fait une erreur au calcul $k\}$.

1. Calculer, en fonction de $p$, $P(E_1)$.
2. M. G fait une faute à son premier calcul. Quelle est la probabilité qu'il ait bien dormi. (On exprimera le résultat en fonction de $p$)
3. Soit $T$ l'événement $\{$ Tchoupi a bien dormi $\}$. A-t-on $P_T(E_1 \cap E_2) = P_T(E_1)P_T(E_2)$ ? A-t-on $P(E_1 \cap E_2) = P(E_1)P(E_2)$ ? (Attention le fait d'être indépendant dépend fortement de la probabilité considérée, cette question doit être réfléchie...)
4. M. G fait une faute à son premier calcul. Quelle est la probabilité qu'il fasse une faute au deuxième calcul.

**Exercice 5.** Roudoudou le hamster vit une vie paisible de hamster. Il a deux activités : manger et dormir... On va voir Roudoudou à 00h00 ($n = 0$). Il est en train de dormir.

— Quand Roudoudou dort à l'heure $n$, il y a 7 chances sur 10 qu'il dorme à l'heure suivante.

— Quand Roudoudou mange à l'heure $n$, il y a 2 chances sur 10 qu'il dorme à l'heure suivante.

On note $D_n$ l'événement 'Roudoudou dort à l'heure $n$' et $M_n$ 'Roudoudou mange à l'heure $n$'. On note $d_n = P(D_n)$ et $m_n = P(M_n)$ les probabilités respectives.

1. Justifier que $d_n + m_n = 1$.
2. Montrer rigoureusement que
$$d_{n+1} = 0,7d_n + 0,2m_n$$
3. Exprimer de manière similaire $m_{n+1}$ en fonction de $d_n$ et $m_n$.
4. Soit $A$ la matrice
$$A = \frac{1}{10}\begin{pmatrix} 7 & 2 \\ 3 & 8 \end{pmatrix}.$$
et $X_n = \begin{pmatrix} d_n \\ m_n \end{pmatrix}$. Exprimer $X_{n+1}$ en fonction de $A$ et $X_n$. Puis en déduire - et la prouver- une expression de $X_n$ en fonction de $n, X_0$ et $A$.
On admet que $\forall n \in \mathbb{N},\ A^n = \frac{1}{5}\begin{pmatrix} 3\,(1/2)^n + 2 & -2\,(1/2)^n + 2 \\ -3\,(1/2)^n + 3 & 2\,(1/2)^n + 3 \end{pmatrix}$.
5. En déduire la valeur de $d_n$ en fonction de $n$.

**Exercice 6.** On modélise une carte d'un jeu de 52 cartes par une liste de deux éléments le premier sera sa couleur (on dispose d'une liste `couleur=['Pique', 'Coeur', 'Carreau' , 'Trefle']` que l'on pourra utiliser dans les fonctions) et la deuxième valeur son numéro. Pour simplifier le valet sera 11, la dame 12 et le roi 13. Ainsi la dame de carreau sera modélisé par `['Carreau', 12]` et le 6 de coeur par `['Coeur', 6]`.

On modélise le jeu de carte par une liste contenant toutes les cartes.

Afin de répondre aux différentes questions, on pourra utiliser les fonctions des questions précédentes même si elles n'ont pas été codées.

1. Ecrire une fonction Python `paquet` qui retourne une liste correspondant à la modélisation décrite d'un jeu de cartes.
2. Remplir la ligne suivante afin que la liste `L` soit affectée à la même liste mais sans le terme d'indice `i`.

```
L=L[...:...] + L[...:...]
```

3. Ecrire une fonction Python `tirage_sans_remise` qui prend en argument un entier `n` qui retourne une liste de `n` cartes tirées aux hasard et sans remise du paquet. (On n'utilisera pas la fonction `.pop`)
4. Dans la question précédente quelle condition doit on imposer sur `n` ? Que faudrait il ajouter pour vérifier cette condition ?
5. Que fait la fonction suivante :

```
def mystere(n):
  J=tirage_sans_remise(n)
  C=[c[0] for c in J]
  return(C)
```

6. Ecrire une fonction Python `check_couleur` qui prend en argument un entier `n`, selectionne `n` cartes sans remises et retourne `True` si on obtient `n` cartes de la même couleur et `False` sinon.
7. On se propose d'écrire une fonction qui permet de vérifier si les cartes tirées forment une suite. Pour cela il faut trier la liste des valeurs des cartes. Compléter la fonction suivante qui prend en argument un entier `n` et retourne la liste des cartes triées dans l'ordre croissant

```
def tri_carte(n):
   J=tirage_sans_remise(n)
   T= [J[0]]  #on place la premiere carte dans une liste
   for i in .......... : # on regarde toutes les cartes de J
      val_carte= .... # on regarde la valeur de la carte
      k=..... # On initialise le compteur
      while  k<len(T) and val_carte<T[k][1] : #on compare avec
                                         #les cartes deja tries
        k=..... #on augmente le compteur
      T = T[... : ...] + [J[i]] +T[ ... : ...]  #on  place la carte
                                         #au bon endroit dans T
   return(T)
```

8. Ecrire une fonction Python `check_suite` qui prend en argument un entier `n`, selectionne `n` cartes sans remises et retourne `True` si on obtient `n` cartes dont les valeurs se suivent et `False` sinon.