TP10 - Révisions

Dans tout le TP, on importe les modules suivants :

```
import numpy as np
import numpy.random as rd
import matplotlib.pyplot as plt
```

Le but de ce TP est, plutôt que d'illustrer un concept mathématiques, de profiter de la période de révisions pour revoir les méthodes algorithmiques et les outils élémentaires en `Python` susceptibles de tomber dans un exercice écrit aux concours.

1 Algorithmique

Exercice 1 ⋆

Soit (u_n) la suite définie par : $\begin{cases} u_0 = 2 \\ \forall n \in \mathbb{N},\ u_{n+1} = \sin(u_n) \end{cases}$. Écrire une fonction qui prend comme paramètre un entier n et calcule la valeur de u_n.

On doit avoir $u_{100} \approx 0,16926$.

Exercice 2 ⋆

Même exercice que précedemment avec $\begin{cases} u_0 = 1 \\ \forall n \in \mathbb{N},\ u_{n+1} = u_n + \frac{(-1)^n}{n+1} \end{cases}$.

On doit avoir $u_{10} \approx 0,1645$.

Exercice 3 ⋆⋆

Écrire une fonction prenant en paramètre un entier naturel n et qui renvoie $u_n = \sum_{k=1}^{n} \frac{1}{k} - \ln(n)$.
Vérifier numériquement que (u_n) converge et donner une valeur approchée de la limite.

Exercice 4 ⋆⋆⋆

La suite de Fibonacci est définie par : $\begin{cases} u_0 = 1 \\ u_1 = 1 \\ \forall n \in \mathbb{N},\ u_{n+2} = u_{n+1} + u_n \end{cases}$. Écrire une fonction qui prend un paramètre n et renvoie la valeur de u_n. On pourra chercher à compléter le code suivant :

```
def fibonacci(n):
    v = 1
    u = 1
    for i in range(2,n+1):
        a = ...
        u = ...
        v = a
    return u
```

Exercice 5 ⋆⋆⋆⋆

On considère à présent la suite définie par $u_1 = 2$ et pour tout $n \geq 1$, $u_{n+1} = \frac{\sqrt{2}u_n}{\sqrt{1+\sqrt{1-\left(\frac{u_n}{2^n}\right)^2}}}$.

1. Écrire une fonction qui calcule u_n. *On vérifiera que la fonction donne bien $u_1 = 2$ et $u_2 = 2\sqrt{2}$.*
2. Essayer cette fonction sur de grandes valeurs de n. Reconnaître la limite de la suite.
3. On admet que pour tout $n \in \mathbb{N}^\star$, $|u_n - \pi| \leq \frac{\pi^3}{6\times 4^n}$. Écrire une fonction qui prend en paramètre un nombre ϵ et retourne une valeur approchée de π à ϵ près.

Exercice 6 ⋆⋆⋆

L'algorithme d'Euclide permet de calculer le plus grand diviseur commun (PGCD) de deux nombres entiers. Pour déterminer le PGCD de $a, b \in \mathbb{N}^\star$, l'algorithme fonctionne de la manière suivante :

- On calcule le résultat de la division euclidienne de a par b. On note : $a = bq + r$.
- Si $r = 0$ alors le PGCD est b.
- Si $r \neq 0$, le PGCD est le même que celui de b et r. On recommence alors l'algorithme avec $a' = b$ et $b' = r$.

Compléter le code suivant pour qu'il calcule le PGCD de `a` et `b` donnés en paramètres :

```
def pgcd(a,b):
  q = ...
  r = ...
  if ... :
    return ...
  else:
    return pgcd(..., ...)
```

Exercice 7 - Triangle de Pascal ⋆⋆⋆

1. Créer un matrice de taille 21×21 ne contenant que des 0 à l'exception de la premère colonne qui ne doit contenir que des 1.
2. En utilisant la formule de Pascal, écrire un programme remplissant les coefficients de la matrices pour qu'elle forme le triangle de Pascal. Que vaut $\binom{20}{16}$?

Exercice 8 ⋆⋆

Écrire une fonction prenant comme paramètre un nombre réel A et qui retourne la valeur du plus petit entier n tel que $\sum_{k=1}^{n} \frac{1}{k} > A$. *On pourra vérifier que pour $A = 10$, $n = 12\ 367$.*

Quel est le plus entier n tel que $\sum_{k=1}^{n} \frac{1}{k} > 15$?

2 Probabilités et simulations

Exercice 9 ⋆

L'indice de masse corporelle (IMC) est défini par $I = \frac{P}{t^2}$ où P est le poids en kilogramme et t la taille en mètres.

1. Écrire une fonction `def IMC(P,t):` prenant deux paramèrtres : le poids et la taille et retourne l'IMC.
2. On considère que la taille (en mètres) et le poids (en kilogramme) sont dzeux variables aléatoires indépendantes suivant respectivement des lois normales $\mathcal{N}(1.78, 0.0016)$ et $\mathcal{N}(72, 36)$.
   Créer deux tableaux `numpy` P et t contenant des simulations de poids et de tailles pour 10 000 personnes.
3. Utiliser la fonction `IMC` pour calculer l'IMC de ces personnes en enregistrant les résultats dans un nouveau tableau `numpy`. Donner la valeur moyenne de l'IMC.
4. Une personne est considérée en surpoids si son IMC dépasse 25 kg.m^{-2}. Déterminer la proportion de personnes en surpoids.

Exercice 10 ⋆⋆

En se rappelant que la loi binomiale compte le nombre de succès d'une répétition d'épreuves de Bernoulli, écrire une fonction `def binomiale(n,p):` qui simule une réalisation de la loi $\mathcal{B}(n,p)$ en utilisant `rd.random`.

Exercice 11 ⋆⋆

Une urne contient initialement des boules numérotées de 2 à n. On effectue un tirage dans cette urne, et on enlève de l'urne toutes les boules portant un numéro supérieur ou égal à celui de la boule tirée. On ajoute alors la boule numéro 1 dans l'urne, et on effectue un nouveau tirage, et on note X le numéro de la boule obtenue.
Écrire un programme qui simule la variable aléatoire X.

Exercice 12 ⋆⋆⋆

On lance n fois une pièce équilibrée, et on note X le nombre de fois où l'on obtient deux résultats identiques consécutifs. Écrire un programme qui simule la variable aléatoire X.
Utiliser ensuite ce programme pour estimer la valeur de $E(X)$.

Exercice 13 ⋆⋆⋆

Si $(X_1,\dots,X_n)$ sont des variables aléatoires indépendantes suivant la loi $\mathcal{N}(0,1)$, la loi suivie par la variable aléatoire $\sum_{i=1}^{n} X_i^2$ est appelée *loi du* χ^2 (« loi du ki-deux ») de paramètre n.

1. Écrire une fonction `def chi_deux(n):` qui prend en paramètre un entier $n \geq 1$ et qui simule une variable suivant la loi $\chi^2(n)$.
2. Écrire une fonction qui simule la variable T_p, où $T_p = \max(Y_1,\dots,Y_p)$ où $Y_1,\dots,Y_p$ sont p variables aléatoires indépendantes suivante la loi $\chi^2(n)$.
3. Proposer une méthode pour obtenir une valeur approchée de $E(T_p)$.

Exercice 14 ⋆⋆⋆⋆

Pour $n \in \mathbb{N}^\star$, on considère deux variables aléatoires X et Y indépendantes et suivant la même loi $\mathcal{U}(\llbracket 1,n \rrbracket)$.
On pose alors $u_n = P(PGCD(X,Y) = 1)$. u_n est donc la probabilité que X et Y soient deux nombres entiers premiers entre eux.
Nous allons chercher à estimer u_n. On pose $N \in \mathbb{N}^\star$ un entier naturel.

1. En utilisant `rd.randint`, créer deux tableaux `X` et `Y` contenant N simulations des variables $X, Y \hookrightarrow \mathcal{U}(\llbracket 1,n \rrbracket)$. Puis en utilisant la fonction `pgcd` de l'exercice 6, créer un nouveau tableau contenant les PGCD des simulations de X et Y.
2. Utiliser le code tapé à la question précédente pour écrire une fonction calculant une estimation de u_n.
3. En prenant des valeurs de plus en plus grandes de n et N, vérifier que u_n converge numériquement. Comparer la limite estimée à $\frac{6}{\pi^2}$.

3 Sujet de concours

Exercice 15 - EDHEC ECS 2021 ⋆⋆

La suite (u_n) est définie par $\left\{ \begin{array}{rcl} u_0 & = & 1 \\ \forall n \in \mathbb{N},\ u_{n+1} & = & \sqrt{u_n^2 + u_n} \end{array} \right.$. On pose également pour $n \in \mathbb{N}^\star$: $S_n = \sum_{k=0}^{n-1} u_k$.

1. Compléter le script `Python` suivant afin qu'il permette de déterminer et d'afficher la plus petite valeur de n pour laquelle $S_n > 1000$.

```
n = 1
u = 1
S = 1 # S1 = u0 = 1
while S <= 1000:
  u = ...
  S = ...
  n = n+1
print(...)
```

2. Montrer que $\forall n \in \mathbb{N}^\star,\ S_n = u_n^2 - 1$. En déduire un nouveau script `Python` faisant le même travail que le précédent mais sans calculer S_n :

```
n = 1
u = 1 # u0 = 1
while u <= ... :
  u = ...
  n = n+1
print(...)
```

Exercice 16 - ECRICOME 2016 ⋆⋆⋆⋆

Soient a et b deux entiers strictement positifs. Une urne contient initialement a boules rouges et b boules blanches. On effectue une succession d'épreuves, chaque éreuve étant constituée des trois étapes suivantes :

- on pioche une boule au hasard dans l'urne ;
- on replace la boule tirée dans l'urne ;
- on rajoute dans l'urne une boule de la même couleur que celle qui vient d'être piochée.

Après n épreuves, l'urne contient donc $a+b+n$ boules. Pour tout $n \in \mathbb{N}^\star$, on note X_n le nombre de boules rouges qui ont été **ajoutées** dans l'urne (par rapport à la composition initiale) à l'issue des n premières épreuves. Pour tout $n \in \mathbb{N}^\star$, on notera R_n l'événement « on pioche une boule rouge au $n^{\text{ème}}$ tirage ».

1. Compléter la fonction suivante, qui simule le tirage d'une boule dans une urne contenant x boules rouges et y boules blanches et qui retourne la valeur 0 si la boule est rouge et 1 si elle est blanche.

```
def tirage(x,y):
  r = rd.random()
  if ... :
    return 0
  else
    return 1
```

2. Compléter la fonction suivante, qui simule n tirages successifs dans une urne contenant initialement a boules rouges et b boules blanches (selon le protocole décrit ci-dessus) et qui retourne la valeur de X_n :

```
def experience(a,b,n):
  x = a
  y = b
  for k in range(n):
    r = tirage(x,y)
    if r == 0:
      x = ...
    else:
      ...
  return ...
```

3. Écrire une fonction d'en-tête **`def simulation(ab,n,m):`** qui fait appel m fois à la fonction **`experience`** pour estimer la loi de X_n et renvoie un tableau `numpy` contenant les approximations de $P(X_n = 0)$, ..., $P(X_n = n)$.