Spring 2026

Programmation 2

TP 9 — *Lisez attentivement l'ensemble du document.* — *Nabil Mustafa*

Consignes

(a) *L'objectif est d'apprendre la programmation en Python.* ***Inutile de se précipiter !*** *Prenez votre temps et terminez correctement tous les TP précédents avant d'aborder celui-ci.*

(b) *Pour réussir,* ***lisez le texte attentivement et lentement****. Ne parcourez pas les consignes en diagonale. Lisez chaque phrase une fois, puis une deuxième. Si un passage reste flou après plusieurs lectures, c'est le moment idéal pour poser une question au professeur. C'est ainsi que l'on apprend efficacement.*

(c) *Avant de commencer,* ***mettez votre téléphone en mode « ne pas déranger » et rangez-le dans votre sac****. Chaque notification consultée brise votre concentration et vous oblige à reprendre le fil de votre raisonnement, ce qui allonge considérablement le temps nécessaire pour terminer l'exercice et favorise les erreurs.*

(d) *Adoptez une approche progressive : lisez une instruction, puis exécutez-la immédiatement. Évitez de lire tout le TP d'un coup.* ***Avancez étape par étape pour rester concentré sur l'action immédiate et ne pas vous sentir submergé par un long texte.***

(e) *Souvenez-vous : les étudiants qui réussissent le mieux ne sont pas ceux qui « finissent en premier », mais ceux qui* ***lisent avec soin, posent des questions lorsqu'ils bloquent et restent concentrés*** *jusqu'au bout.*

Votre objectif aujourd'hui n'est pas d'aller vite, mais de comprendre profondément.

Tâche 1: Exercises

(a) Connectez-vous à `LINUX` avec votre compte utilisateur.

(b) Téléchargez le Jupyter Notebook fichier `Lec7-iterables1.ipynb`, disponible sur le site web du module.

(c) Pour le lire, ouvrez un `terminal` dans le *même dossier* que le fichier `Lec7-iterables1.ipynb`, et exécuter le command :

```
jupyter-notebook Lec7-iterables1.ipynb
```

(d) Pour exécuter un code Python dans le Jupyter Notebook, cliquez sur le code et appuyez sur "SHIFT + ENTER".

(e) **Lisez toutes les informations** dans cette Jupyter Notebook.

(f) **Faites tous les exercices** dans cette Jupyter Notebook.

Diogenes's Game

Il y a n personnes dans une village, appelée

`Personne` 0, `Personne` 1, ..., `Personne` $n-1$.

> Chaque personne est soit **honnête**, soit **malhonnête**.
>
> —Une personne **honnête** dira *toujours* la vérité.
>
> —Une personne **malhonnête** *parfois* dira la vérité, et *parfois* mentira. À chaque fois, vous ne savez pas si cette personne ment ou dit la vérité.

Le nombre de personnes honnêtes est strictement supérieur à $\frac{n}{2}$.

Objectif : votre mission est d'identifier avec certitude **une personne honnête** parmi le groupe.

Pour réaliser cette tâche, vous pouvez poser autant de questions que vous le souhaitez, du type :

Choisissez deux personnes distinctes i et j, et demandez à la Personne i : « Selon vous, la Personne j est-elle honnête ? »

- Si `Personne i` est **honnête** : Elle répond toujours **correctement** à la question posée. Autrement dit, sa réponse reflète fidèlement la véritable nature de `Personne j`.
- Si `Personne i` est **malhonnête** : Son comportement est **imprévisible**. Elle peut choisir de répondre correctement ou incorrectement concernant `Personne j`. Vous n'avez aucun moyen de savoir si sa réponse est vraie ou fausse, ni de prédire son comportement.

Notez que :

- Vous ne connaissez *a priori* la nature (honnête ou malhonnête) d'aucune personne.
- Les réponses des personnes malhonnêtes ne sont pas nécessairement cohérentes dans le temps.

Critère d'évaluation :

L'efficacité de votre stratégie est mesurée par le **nombre de questions posées**. Vous devez concevoir un algorithme qui garantit de trouver une personne honnête tout en minimisant ce nombre, dans le **pire des cas** (en considérant que les personnes malhonnêtes peuvent répondre de la manière la plus défavorable à votre stratégie).

Tâche 2: Exécuter Python

(a) Connectez-vous à `LINUX` avec votre compte utilisateur.

(b) **Téléchargez** le fichier `DiogenesgameFAST.py`.

(c) **Éditer** ce fichier avec l'éditeur de votre choix. Par exemple, ouvrez un `terminal` dans le même dossier que le fichier `DiogenesgameFAST.py`, et exécuter le command:

```
kate DiogenesgameFAST.py
```

(d) **Exécuter** le code avec `Python`: ouvrez un `terminal` dans le *même dossier* que le fichier `DiogenesgameFAST.py`, et exécuter le command :

```
python3 DiogenesgameFAST.py
```

Tâche 3: Recursive algorithm avec $O(n \log n)$ questions—TD 11.1.

Votre objectif est d'écrire la fonction

`computeHonestPerson_MEDIUM_RECURSIVE(A, isHonest)`

qui prend une liste `A` de n personnes et renvoie une personne parmi `A` qui est honnête.

Elle prend également un paramètre **isHonest**, **qui est une fonction**.

> Pour poser la question à la `personne A[i]` : 'La `personne A[j]` est-elle honnête ?', appelez
>
> `isHonest(A[i], A[j])`.
>
> Si la `personne A[i]` pense que la `personne A[j]` est **honnête**, la fonction renvoie **True**.
>
> Si la `personne A[i]` pense que la `personne A[j]` est **malhonnête**, la fonction renvoie **False**.

Contrainte 1: Votre algorithme doit trouver la bonne réponse avec au plus $n \log_2 n$ **questions.**

Contrainte 2: Votre algorithme doit utiliser la **récursion** plutôt que des boucles imbriquées.

Vous pouvez utiliser au plus **une seule boucle** (par exemple, parcourir une liste une fois).

Exemple : Supposons que `A = [56, 22, 90]`, où la personne `A[0] = 56` est honnête, la personne `A[1] = 22` est malhonnête, et la personne `A[2] = 90` est honnête. Alors :

- `isHonest(A[0], A[2])` renvoie **True** (personne 56 honnête dit que personne 90 est honnête).
- `isHonest(A[1], A[2])` peut renvoyer **True** ou **False** (personne 22 malhonnête répond arbitrairement).
- `isHonest(A[2], A[1])` renvoie **False** (personne 90 honnête dit que personne 22 est malhonnête).

```
########ci-dessous: tache 3.########
def computeHonestPerson_MEDIUM_RECURSIVE(A, isHonest):
  '''
  vous pouvez utiliser la fonction isHonest(A[i],A[j])
  pour avoir avis sur person A[j] donne par person A[i].
  '''

  #ECRIVEZ VOTRE CODE ICI

  return random.choice(A) #<--- il faut changer, pour le moment c'est aleatoire.
########ci-dessus: tache 3.########
```

Le code que je vous ai donné analysera votre réponse et vous indiquera si elle est correcte.

Pour le moment, la fonction renvoie une personne aléatoire. Vous devez la remplacer par la bonne algorithme.

Tâche 4: Iterative algorithm avec $O(n)$ questions—TD 11.2.

Votre objectif est d'écrire la fonction

`computeHonestPerson_FAST_CHAIN(A, isHonest)`

qui prend une liste `A` de n personnes et renvoie une personne parmi `A` qui est honnête.

Elle prend également un paramètre **isHonest**, **qui est une fonction**.

Pour poser la question à la `personne A[i]` : 'La `personne A[j]` est-elle honnête ?', appelez

`isHonest(A[i], A[j])`.

Si la `personne A[i]` pense que la `personne A[j]` est **honnête**, la fonction renvoie **True**.

Si la `personne A[i]` pense que la `personne A[j]` est **malhonnête**, la fonction renvoie **False**.

Contrainte 1: Votre algorithme doit trouver la bonne réponse en utilisant au plus $\boldsymbol{n}$ **questions.**

Contrainte 2: Votre algorithme ne doit pas utiliser la récursion. Vous devez utiliser des boucles (y compris des boucles imbriquées) à la place. Vous pouvez utiliser autant de boucles que nécessaire (par exemple, des boucles `for` ou `while` imbriquées).

Exemple : Supposons que `A = [56, 22, 90]`, où la personne `A[0] = 56` est honnête, la personne `A[1] = 22` est malhonnête, et la personne `A[2] = 90` est honnête. Alors :

- `isHonest(A[0], A[2])` renvoie **True** (personne 56 honnête dit que personne 90 est honnête).
- `isHonest(A[1], A[2])` peut renvoyer **True** ou **False** (personne 22 malhonnête répond arbitrairement).
- `isHonest(A[2], A[1])` renvoie **False** (personne 90 honnête dit que personne 22 est malhonnête).

```
########ci-dessous: tache 4.########
def computeHonestPerson_FAST_CHAIN(A, isHonest):
'''
vous pouvez utiliser la fonction isHonest(A[i],A[j])
pour avoir avis sur person A[j] donne par person A[i].
'''

#ECRIVEZ VOTRE CODE ICI

return random.choice(A) #<--- il faut changer, pour le moment c'est aleatoire.
########ci-dessus: tache 4.########
```

Le code que je vous ai donné analysera votre réponse et vous indiquera si elle est correcte.

Pour le moment, la fonction renvoie une personne aléatoire. Vous devez la remplacer par la bonne algorithme.

(FACULTATIVE) Tâche 5: Recursive algorithm avec $O(n)$ questions—TD 11.3.

Votre objectif est d'écrire la fonction

`computeHonestPerson_FAST_RECURSIVE(A, isHonest)`

qui prend une liste `A` de n personnes et renvoie une personne parmi `A` qui est honnête.

Elle prend également un paramètre **isHonest**, **qui est une fonction**.

> Pour poser la question à la `personne A[i]` : 'La `personne A[j]` est-elle honnête ?', appelez
>
> `isHonest(A[i], A[j])`.
>
> Si la `personne A[i]` pense que la `personne A[j]` est **honnête**, la fonction renvoie **True**.
>
> Si la `personne A[i]` pense que la `personne A[j]` est **malhonnête**, la fonction renvoie **False**.

Contrainte 1: Votre algorithme doit trouver la bonne réponse en utilisant au plus ***n* questions.**

Contrainte 2: Votre algorithme doit utiliser la **récursion** plutôt que des boucles imbriquées.

Vous pouvez utiliser au plus **une seule boucle** (par exemple, parcourir une liste une fois).

Exemple : Supposons que `A = [56, 22, 90]`, où la personne `A[0] = 56` est honnête, la personne `A[1] = 22` est malhonnête, et la personne `A[2] = 90` est honnête. Alors :

- `isHonest(A[0], A[2])` renvoie **True** (personne 56 honnête dit que personne 90 est honnête).
- `isHonest(A[1], A[2])` peut renvoyer **True** ou **False** (personne 22 malhonnête répond arbitrairement).
- `isHonest(A[2], A[1])` renvoie **False** (personne 90 honnête dit que personne 22 est malhonnête).

```
########ci-dessous: tache 5.########
def computeHonestPerson_FAST_RECURSIVE(A, isHonest):
'''
vous pouvez utiliser la fonction isHonest(A[i],A[j])
pour avoir avis sur person A[j] donne par person A[i].
'''

#ECRIVEZ VOTRE CODE ICI

return random.choice(A) #<--- il faut changer, pour le moment c'est aleatoire.
########ci-dessus: tache 5.########
```

Le code que je vous ai donné analysera votre réponse et vous indiquera si elle est correcte.

Pour le moment, la fonction renvoie une personne aléatoire. Vous devez la remplacer par la bonne algorithme.