# Nom Prénom :

# Programmation récursive :

## Exercice 1 :

On considère la fonction`mystere(n)` définit par le code :

```
def mystere(n):
    if n == 0:
        return 1
    else:
        return n * mystere(n-1)
```

1. Quel est le résultat renvoyé par `mystere(4)` ? .....

2. Décrire, en français, ce que fait cette fonction :

   .....

   .....

   .....

3. Proposer une version itérative documentée pour cette fonction :

```
def mystere(n :    ) ->     :
    '''



    '''



    return
```

Exercice 2 :

On considère la fonction`sigma(n)` définit par le code :

```
def sigma(n):
    resultat = 0
    for i in range(n+1) :
        resultat = resultat + i
    return resultat
```

1. Quel est le résultat renvoyé par `sigma(0)` ? .....
2. Quel est le résultat renvoyé par `sigma(4)` ? .....
3. Décrire, en français, ce que fait cette fonction :

   .....

   .....

   .....

   .....
4. Proposer une version récursive documentée pour cette fonction :

```
def sigma(n :    ) ->     :
    '''



    '''
```

## EXERCICE 3

*Cet exercice porte sur les bases de données relationnelles et le langage SQL.*

L'énoncé de cet exercice utilise les mots clefs du langage SQL suivants : `SELECT`, `FROM`, `WHERE`, `JOIN…ON`, `UPDATE…SET`, `DELETE`, `INSERT INTO…VALUES`, `ORDER BY`.

- La clause `ORDER BY` suivie d'un attribut permet de trier les résultats par ordre croissant des valeurs de l'attribut ;

Radio France souhaite créer une base de données relationnelle contenant les podcasts des émissions de radio. Pour cela elle utilise le langage SQL. Elle crée :

- une relation (ou table) **`podcast`** qui contient le thème et l'année de diffusion.
- une relation **`emission`** qui contient les émissions (**`id_emission, nom)`**, la radio de diffusion et l'animateur.
- une relation **`description`** qui contient un résumé et la durée du podcast en minutes.

- Relation **`podcast`**

| **`id_podcast`** | **`theme`** | **`annee`** | **`id_emission`** |
|---|---|---|---|
| 1 | Le système d'enseignement supérieur français est-il juste et efficace ? | 2022 | 10081 |
| 2 | Trois innovations pour la croissance future (1/3) : La révolution blockchain. | 2021 | 10081 |
| 3 | Travailleurs de plateformes : vers un nouveau prolétariat ? | 2021 | 10175 |
| 4 | Le poids de la souveraineté numérique française | 2019 | 10183 |
| 40 | Le poids de la souveraineté numérique française | 2019 | 10183 |
| 5 | Dans le cloud en Islande, terre des data center | 2019 | 10212 |

- Relation **`emission`**

| **`id_emission`** | **`nom`** | **`radio`** | **`animateur`** |
|---|---|---|---|
| 10081 | Entendez-vous l'éco ? | France culture | Tiphaine De R. |
| 10175 | Le Temps du débat | France culture | Léa S. |
| 10183 | Soft power | France culture | Frédéric M. |
| 10212 | La tête au carré | France inter | Mathieu V. |

**`id_podcast`** de la relation **`podcast`** et **`id_emission`** de la relation **`emission`** sont des clés primaires.

L'attribut **id_emission** de la relation **podcast** fait directement référence à la clé primaire de la relation **emission.**

- Relation **description**

| id_description | resume | duree | id_emission |
|---|---|---|---|
| 101 | Autrefois réservé à une élite, l'enseignement supérieur français s'est profondément démocratisé : donne-t-il pour autant les mêmes chances à chacun ? | 4 | 10081 |
| 102 | Quelles sont leurs conditions de travail et quels sont leurs moyens de contestation ? | 58 | 10175 |
| 103 | La promesse de la blockchain, c'est la suppression des intermédiaires et la confiance à grande échelle. | 4 | 10081 |

1.

Écrire le schéma relationnel de la relation `description`, en précisant les attributs et leurs types probables, la clé primaire et la ou les clé(s) étrangère(s) éventuelle(s)

2.

**a.** Écrire ce qu'affiche la requête suivante appliquée aux extraits précédents :

```
SELECT theme, annee FROM podcast WHERE id_emission = 10081
```

**b.** Écrire une requête SQL permettant d'afficher les thèmes des podcasts de l'année 2019.

**c.** Écrire une requête SQL affichant la liste des thèmes et des années de diffusion des podcasts dans l'ordre chronologique des années.

3.

**a.** Décrire simplement le résultat obtenu avec cette requête SQL.

```
SELECT DISTINCT theme FROM podcast
```

**b.** Écrire une requête SQL supprimant la ligne contenant l'`id_podcast = 40` de la relation `podcast`.

4.

**a.** Une erreur de saisie a été faite dans la relation `emission`. Écrire une requête SQL permettant de changer le nom de l'animateur de l'émission "Le Temps du débat" en "Emmanuel L.".

**b.** Écrire une requête SQL permettant d'ajouter l'émission "Hashtag" sur la radio "France inter" avec "Mathieu V.". On lui donnera un `id_emission` égal à 12850.

5.

Écrire une requête permettant de lister les thèmes, le nom des émissions et le résumé des podcasts pour lesquels la durée est strictement inférieure à 5 minutes.

## Exercice 4

*Cet exercice traite du thème base de données, et principalement du modèle relationnel et du langage SQL.*

L'énoncé de cet exercice peut utiliser les mots du langage SQL suivants :

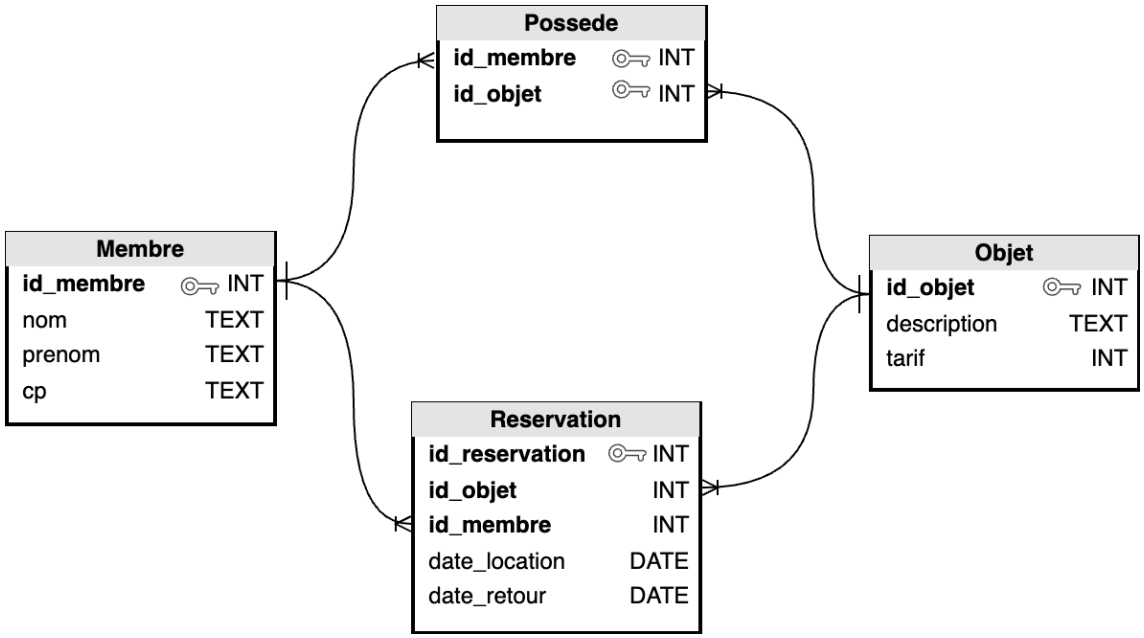
`SELECT`, `FROM`, `WHERE`, `JOIN ON`, `INSERT INTO`, `VALUES`, `UPDATE`, `SET`, `DELETE`, `COUNT`, `AND`, `OR`

Un site permet à ses membres de proposer à la location du matériel et de louer du matériel. Ceci permet de mutualiser du matériel entre membres et au propriétaire de rentabiliser cet achat. Le temps d'utilisation du matériel s'en trouve ainsi augmenté et le nombre d'appareils diminué.

Le modèle relationnel est donné par le schéma ci-dessous.

La table `Membre` contient les informations de chaque utilisateur du site (nom, prénom et code postal). La table `Objet` décrit le type d'objet à la location ainsi que son tarif de location journalier.

La table `Reservation` répertorie toutes les réservation effectuées par les membres du site avec notamment leur date de début et de fin de location. La table `Possede` permet de lier les tables `Membre` et `Objet`.




Convention utilisées pour le schéma :

- Les clés primaires et étrangères sont mises en gras ;
- un symbole 🔑 identifie une clé primaire ;
- un symbole ⊢⟶⊣ entre deux attributs indique qu'ils doivent partager les mêmes valeurs et qu'ils sont reliés de la manière suivante : le côté ⊢ indique la clé primaire et le côté ⊣ indique la clé étrangère.

On donne ci-dessous le contenu de ces tables à un instant donné :

**Membre**

| id_membre | nom | prenom | cp |
|---|---|---|---|
| 1 | "Ali" | "Mohamed" | "69110" |
| 2 | "Alonso" | "Fernando" | "69005" |
| 3 | "Dupont" | "Antoine" | "69003" |
| 4 | "Ferrand" | "Pauline" | "69160" |
| 5 | "Kane" | Harry" | "69003" |

**Possede**

| id_membre | id_objet |
|---|---|
| 1 | 4 |
| 1 | 6 |
| 2 | 4 |
| 3 | 3 |
| 3 | 5 |
| 4 | 1 |
| 4 | 2 |

**Objet**

| id_objet | description | tarif |
|---|---|---|
| 1 | "Nettoyeur haute pression" | 20 |
| 2 | "Taille-haie" | 15 |
| 3 | "Perforatrice" | 15 |
| 4 | "Appareil à raclette" | 10 |
| 5 | "Scie circulaire" | 15 |
| 6 | "Appareil à gauffre" | 10 |

**Reservation**

| id_reservation | id_objet | id_membre | date_location | date_retour |
|---|---|---|---|---|
| 1 | 4 | 5 | 2022-02-18 | 2022-02-19 |
| 2 | 1 | 2 | 2022-05-05 | 2022-05-06 |
| 3 | 3 | 1 | 2022-07-10 | 2022-07-12 |
| 4 | 3 | 1 | 2022-08-12 | 2022-08-14 |
| 5 | 2 | 2 | 2022-10-20 | 2022-10-22 |
| 6 | 2 | 2 | 2022-10-20 | 2022-10-22 |

**1.** Dans cette partie, on ne demande pas de requête SQL. En étudiant le contenu des tables ci-dessus :

**a)** Indiquer quels sont les prénoms et noms du ou des membres du site qui proposent la location d'un appareil à raclette ;

**b)** Donner le prénom et le nom du membre qui ne propose pas d'objet à la location.

**2. a)** Donner le résultat de la requête suivante :

```
SELECT nom, prenom FROM Membre WHERE cp = "69003" ;
```

**b)** Écrire une requête permettant de connaître le tarif de location d'une scie circulaire.

**c)** Écrire une requête permettant de modifier le tarif de location d'un nettoyeur à haute pression pour le passer à 15 € par jour au lieu de 20 € par jour.

**d)** Écrire une requête SQL permettant d'ajouter Wendie Renard habitant à Villeurbanne (code postal 69100) dans la table `Membre`, avec un `id_membre` de 6.

**3. a)** Expliquer la limitation importante d'utilisation du service offert par le site si l'on utilisait le couple de clés étrangères (`id_objet`, `id_membre`) en tant que clé primaire de la relation `Reservation`.

**b)** Mohamed Ali décide de ne plus être membre du site. Il faut donc le supprimer de la table `Membre` à l'aide de la requête :

```
DELETE FROM Membre
WHERE nom = "Ali" AND prenom = "Mohamed" ;
```

Expliquer pourquoi cette requête produit une erreur.

**c)** Proposer une suite de requêtes utilisant le mot clé `DELETE`, précédant la requête ci-dessus pour supprimer correctement Mohamed Ali, dont l'`id_membre` est 1, de la base de donnée.

Rappel : la relation `Objet` décrit le type d'objet à la location. Il n'y a pas d'objet à supprimer dans cette table lors du départ d'un membre.

**4.** Dans cette partie, les requêtes utilisent des jointures entre tables. On supposera donc les numéros `id_membre` et `id_objet` non connus.

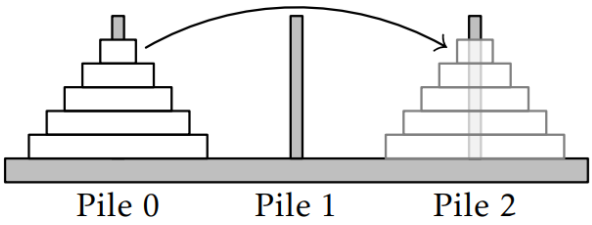
**a)** Écrire une requête permettant de compter le nombre de réservations réalisées par Fernando Alonso.

**b)** Écrire une requête permettant de connaître les noms et prénoms des membres possédant un appareil à raclette.

# Exercice 5
## Les tours de Hanoï

On appelle **tours de Hanoï** un jeu, inventé par le mathématicien Édouard Lucas, composé de disques de bois s'empilant sur 3 tiges. L'objectif est de déplacer vers la droite la pile de disques se trouvant à gauche, et cela en un minimum de coups. Pour cela il faut respecter les règles suivantes :

- On ne peut déplacer qu'un disque à la fois.
- On ne peut déplacer qu'un disque sur un autre disque plus grand que lui ou sur un emplacement vide.
- Dans la position initiale, l'empilement de disques respecte la règle précédente.

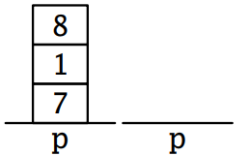



**Question 1 :** On part avec une pile représentée ci-contre. Représenter l'état de la pile après les instructions suivantes :

```
>>> depiler(p)
>>> depiler(p)
>>> empiler(3, p)
```

```
>>> empiler(5, p)
>>> depiler(p)
>>> empiler(0, p)
```

```
>>> depiler(p)
>>> empiler(4, p)
>>> empiler(2, p)
```

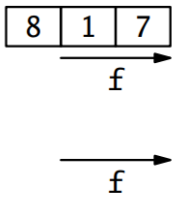



**Question 2 :** On part avec une file représentée ci-contre. Représenter l'état de la file après les instructions suivantes :

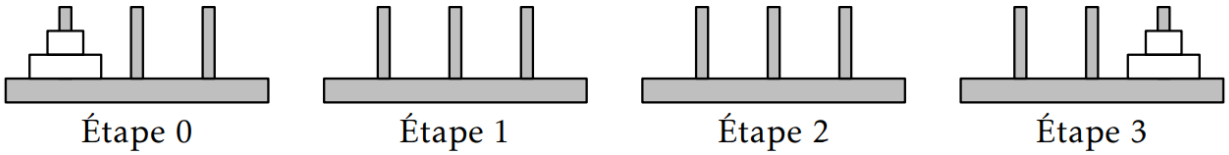
```
>>> retirer(f)
>>> retirer(f)
>>> ajouter(3, f)
```

```
>>> ajouter(5, f)
>>> retirer(f)
>>> ajouter(0, f)
```

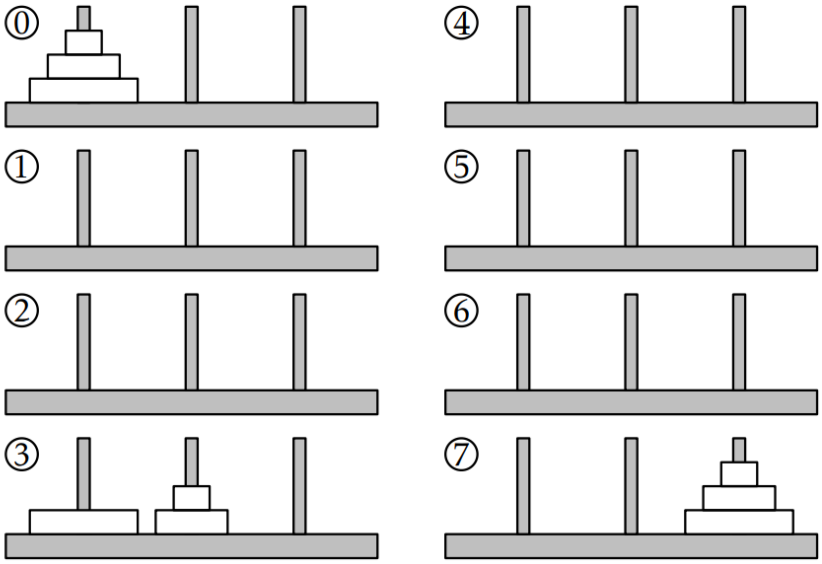
```
>>> retirer(f)
>>> ajouter(4, f)
>>> ajouter(2, f)
```




**Question 3 :** Il faut 3 étapes pour déplacer une pile de 2 disques. Compléter les étapes intermédiaires sur le schéma ci-dessous :




**Question 4 :** Il faut 7 étapes pour déplacer une pile de 3 disques. Compléter les étapes manquantes sur le schéma ci-contre :



## Programmation en Python

Tous les exercices de cette partie ont pour but de programmer le jeu et de permettre de le résoudre automatiquement. Pour cela on dispose de deux interfaces pour les piles : une avec une structure non spécifiée et des fonctions, et une autre avec une approche objet. Pour les exercices, vous pouvez choisir d'utiliser l'approche de votre choix.

| Fonction | Méthode de POO | Description |
|---|---|---|
| `creer_pile()` | `creer_pile()` | Renvoie une pile vide. |
| `empiler(pile, element)` | `pile.empiler(element)` | Empile au sommet. |
| `depiler(pile)` | `pile.depiler()` | Renvoie et enlève la valeur au sommet. |
| `est_vide(pile)` | `pile.est_vide()` | Renvoie un booléen. |

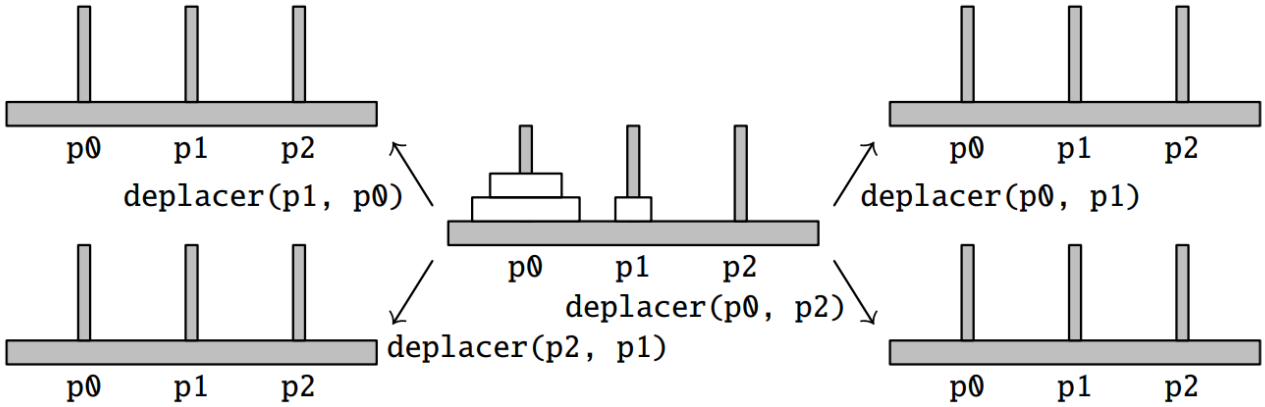
Dans tous les cas, les piles sont modélisées par des objets mutables. Cela veut dire que toute modification faite lors de l'exécution d'une fonction perdurera après. Il n'est donc pas nécessaire d'utiliser de **`return`** pour renvoyer le nouvel état de la pile.
**Vous pouvez utiliser la fonction d'un exercice dans les exercices suivants.**
Dans la suite, on notera `p0`, `p1` et `p2` les piles sur les 3 tiges, de gauche à droite.

Question `5 :` Écrire une fonction `sommet(pile)` qui renvoie la valeur au sommet de la pile. Vous pouvez dépiler, mais il faut bien rempiler la valeur avant la fin de l'exécution de la fonction. Si la pile est vide, il faut lever une exception avec **`raise`** `IndexError('pile vide')`.

```
def sommet(pile) :
```

**Question 6 :** On souhaite écrire une fonction `deplacer(origine, cible)` qui déplace la valeur au sommet de la pile `origine` vers le sommet de la pile `cible`. Si le déplacement n'est pas possible, parce qu'il ne respecte pas les règles du jeu, les piles ne sont pas modifiées.

1) Dans chacun des cas ci-dessous, en partant de l'état central, dessiner l'état des piles attendu après l'instruction proposée.




2) Écrire le code de la fonction `deplacer(origine, cible)` :

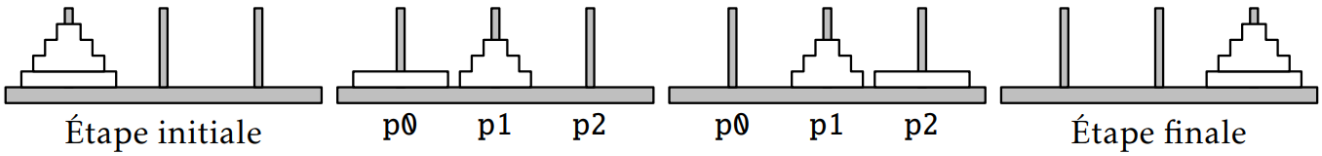
```
def deplacer(origine, cible) :
```

**Question 7 :** Donner la suite d'instructions nécessaires pour passer de l'étape 0 à l'étape 3 dans l'exercice 3, en utilisant la fonction `deplacer`.

**Question 8 :** On veut écrire une fonction récursive `resoudre(n, origine, cible, interm)` qui permet de déplacer les `n` premiers disques au sommet de la pile `origine` vers la pile `cible`, en utilisant éventuellement la pile `interm` comme pile intermédiaire pour les déplacements.

1) Indiquer la commande à effectuer si `n == 1`.

2) On considère maintenant une pile de `n` disques sur la pile `p0` qu'on souhaite amener en `p2`, avec `n` > 1. En utilisant `resoudre(n-1, ..., ..., ...)` pour déplacer les `n` – 1 premiers disques et `deplacer` pour le dernier disque, donner les instructions permettant d'effectuer les étapes ci-dessous :




```
```

**Question 9 :** On souhaite écrire une fonction récursive `nb_etapes(n)` qui renvoie le nombre d'étapes nécessaires pour déplacer une pile de `n` disques, avec `n` > 1. Vous pouvez vous inspirer de l'exercice précédent.

1) Combien vaut `nb_etapes(1)` ?

```
nb_etapes(1) = ...
```

2) Exprimer `nb_etapes(n)` en fonction de `nb_etapes(n-1)`. C'est à dire s'il faut `nb_etapes(n-1)` étapes pour déplacer une pile de `n-1` disques, combien en faut-il pour déplacer une pile de `n` disques ?

```
nb_etapes(n) = ...
```

3) Compléter le code de la fonction :

```
def nb_etapes(n) :
```