

NUMERIQUE ET SCIENCES INFORMATIQUES

⚠ A lire attentivement

- Le sujet comporte trois exercices indépendants
- La durée de l'épreuve est de 3h30
- L'usage de la calculatrice n'est pas autorisée
- Le sujet comporte 4 pages, assurez-vous qu'il est complet avant de commencer

□ Exercice 1 : Analyse et écriture de programmes récurrents

2023 Sujet zéro 🎓

1. a) Expliquer en quelques mots ce qu'est une fonction récursive
- b) On considère la fonction Python suivante :

```
1 def compte_rebours(n):
2     """ n est un entier positif ou nul """
3     if n >= 0:
4         print(n)
5         compte_rebours(n-1)
```

L'appel `compte_rebours(3)` affiche successivement les nombres 3, 2, 1 et 0. Expliquer pourquoi le programme s'arrête après l'affichage du nombre 0.

2. En mathématiques, la factorielle d'un entier naturel n est le produit des nombres entiers strictement positifs inférieurs ou égaux à n . Par convention, la factorielle de 0 est 1. Par exemple :

- la factorielle de 1 est 1
- la factorielle de 2 est $2 \times 1 = 2$
- la factorielle de 3 est $3 \times 2 \times 1 = 6$
- la factorielle de 4 est $4 \times 3 \times 2 \times 1 = 24 \dots$

Recopier et compléter sur votre copie le programme donné ci-dessous afin que la fonction récursive `fact` renvoie la factorielle de l'entier passé en paramètre de cette fonction.

Exemple : `fact(4)` renvoie 24.

```
1 def fact(n):
2     """ Renvoie le produit des nombres entiers strictement positifs inférieurs
3         à n """
4     if n == 0:
5         return .....
6     else:
7         return .....
```

3. La fonction `somme_entiers_rec` ci-dessous permet de calculer la somme des entiers, de 0 à l'entier naturel n passé en paramètre.

Par exemple :


- pour $n = 0$, la fonction renvoie la valeur 0
- pour $n = 1$, la fonction renvoie la valeur $0 + 1 = 1$.
- ...
- pour $n = 4$, la fonction renvoie la valeur $0 + 1 + 2 + 3 + 4 = 10$.

```
1 def somme_entiers_rec(n):
2     """ Permet de calculer la somme des entiers, de 0 à l'entier naturel n """
3     if n == 0:
4         return 0
5     else:
6         print(n) #pour vérification
7         return n + somme_entiers_rec(n - 1)
```

L'instruction `print(n)` de la ligne 6 dans le code précédent a été insérée afin de mettre en évidence le mécanisme en oeuvre au niveau des appels récursifs.

- a) Ecrire ce qui sera affiché dans la console après l'exécution de la ligne suivante :
- ```
res = somme_entiers_rec(3)
```
- b) Quelle valeur sera alors affectée à la variable `res` ?
4. Ecrire en Python une fonction `somme_entiers` non récursive : cette fonction devra prendre en argument un entier naturel  $n$  et renvoyer la somme des entiers de 0 à  $n$  compris. Elle devra donc renvoyer le même résultat que la fonction `somme_entiers_rec` définie à la question 3.
- Exemple : `somme_entiers(4)` renvoie 10.

### □ Exercice 2 : Programmation objet en langage Python

2022 Asie-Pacifique 

Un fabricant de brioches décide d'informatiser sa gestion des stocks. Il écrit pour cela un programme en langage Python. Une partie de son travail consiste à développer une classe `Stock` dont la première version est la suivante :

```
1 class Stock:
2 def __init__(self):
3 self.qt_farine = 0 # quantité de farine initialisée à 0 g
4 self.nb_oeufs = 0 # quantité de d'oeufs (0 à l'initialisation)
5 self.qt_beurre = 0 # quantité de beurre initialisée à 0 g
```

1. Ecrire une méthode `ajouter_beurre(self, qt)` qui ajoute la quantité `qt` de beurre un objet de la classe `Stock`

On admet que l'on a écrit deux autres méthodes `ajouter_farine` et `ajouter_oeufs` qui ont des fonctionnements analogues.

2. Ecrire une méthode `afficher(self)` qui affiche la quantité de farine, d'oeufs et de beurre d'un objet de type `Stock`. L'exemple ci-dessous illustre l'exécution de cette méthode dans la console :

```
1 >>> mon_stock = Stock()
2 >>> mon_stock.afficher()
3 farine: 0
4 oeuf: 0
5 beurre: 0
6 >>> mon_stock.ajouter_beurre(560)
7 >>> mon_stock.afficher()
8 farine: 0
9 oeuf: 0
10 beurre: 560
```

3. Pour faire une brioche, il faut 350 g de farine, 175 g de beurre et 4 oeufs. Ecrire une méthode `stock_suffisant_brioche(self)` qui renvoie un booléen : `True` s'il y a assez d'ingrédients dans le stock pour faire une brioche et `False` sinon.
4. On considère la méthode supplémentaire `produire(self)` de la classe `Stock` donnée par le code suivant :

```
1 def produire(self)
2 res = 0
3 while self.stock_suffisant_brioche():
4 self.qt_beurre = self.qt_beurre - 175
5 self.qt_farine = self.qt_farine - 350
6 self.nb_oeufs = self.nb_oeufs - 4
7 res = res + 1
8 return res
```

On considère une stock défini par les instructions suivantes :

```
1 >>> mon_stock = Stock()
2 >>> mon_stock.ajouter_beurre(1000)
3 >>> mon_stock.ajouter_farine(1000)
4 >>> mon_stock.ajouter_oeufs(10)
```

- a) On exécute ensuite l'instruction

```
1 >>> mon_stock.produire()
```

Quelle valeur s'affiche dans la console ? Que représente cette valeur ?


b) On exécute ensuite l'instruction

```
1 >>> mon_stock.afficher()
```

Que s'affiche-t-il dans la console ?

5. L'industriel possède  $n$  lieux de production distincts et donc  $n$  stocks distincts. On suppose que ces stocks sont dans une liste dont chaque élément est un objet de type `Stock`. Ecrire une fonction Python `nb_brioche` prenant pour seul paramètre une liste de d'objets de type `Stock` et renvoie le nombre total de brioches produites.

❑ **Exercice 3 : Structure de données (piles) – Base de données**

Etranger 2021 

Cette exercice se compose de deux parties **indépendantes**.

**Partie A**

Dans cet exercice, on considère une pile d'entiers positifs. On suppose que les autres fonctions suivantes ont été programmées préalablement en Python :

- `empiler(P,e)` : ajoute l'élément `e` sur la pile `P`;
- `depiler(P)` : enlève le sommet de la pile `P` et renvoie la valeur de ce sommet
- `est_vide(P)` : renvoie `True` si la pile est vide et `False` sinon ;
- `creer_pile()` : retourne une pile vide.

Dans cet exercice, seule l'utilisation de ces quatre fonctions sur la structure de données pile est autorisée

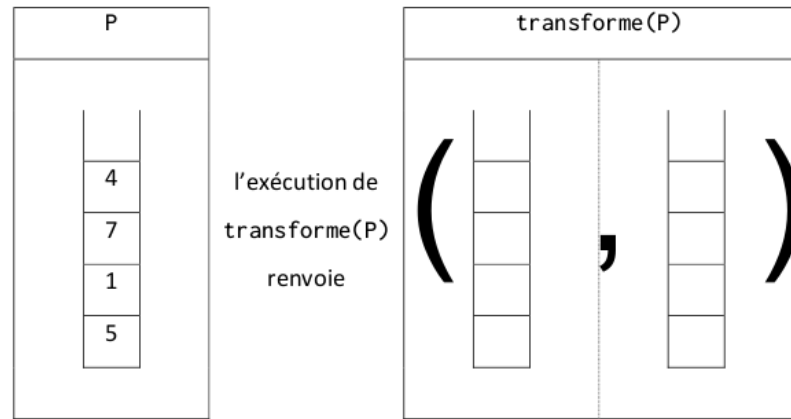
1. **Recopier** le schéma ci-dessous et le compléter sur votre copie en exécutant les appels de fonctions donnés. On écrira ce que renvoie la fonction utilisée dans chaque cas, et on indiquera `None` si la fonction ne renvoie aucune valeur :

| Etapes            | Etape 0<br>Pile d'origine P                                                                                                                      | Etape 1<br><code>empiler(P,8)</code>                                                                                                                       | Etape 2<br><code>depiler(P)</code>                                                                                                                         | Etape 3<br><code>est_vide(P)</code>                                                                                                                        |
|-------------------|--------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Pile              | <div style="border: 1px solid black; padding: 5px; text-align: center;"> <div> </div> <div>4</div> <div>7</div> <div>1</div> <div>5</div> </div> | <div style="border: 1px solid black; padding: 5px; text-align: center;"> <div>...</div> <div>...</div> <div>...</div> <div>...</div> <div>...</div> </div> | <div style="border: 1px solid black; padding: 5px; text-align: center;"> <div>...</div> <div>...</div> <div>...</div> <div>...</div> <div>...</div> </div> | <div style="border: 1px solid black; padding: 5px; text-align: center;"> <div>...</div> <div>...</div> <div>...</div> <div>...</div> <div>...</div> </div> |
| Valeurs renvoyées |                                                                                                                                                  | ...                                                                                                                                                        | ...                                                                                                                                                        | ...                                                                                                                                                        |

2. On propose la fonction ci-dessous, qui prend en argument une pile `P` et renvoie un couple de piles :

```
1 def transforme(P):
2 Q = creer_pile()
3 while not est_vide(P) :
4 v = depiler(P)
5 empiler(Q,v)
6 return (P,Q)
```

**Recopier et compléter** sur votre copie le document ci-dessous :



3. **Ecrire** une fonction en langage Python `maximum(P)` recevant une pile `P` comme argument et qui renvoie la valeur maximale de cette pile. On ne s'interdit pas qu'après exécution de la fonction, la pile soit vide.

### Partie B

On suppose qu'on dispose d'une base de données des processus lancés sur un ordinateur à un instant donné. Cette base de données est constituée d'une seule table appelée `processus` et contenant les champs suivants :

- `pid` : le PID du processus
- `ppid` : le PPID du processus
- `user` : le nom du propriétaire du processus
- `time` : le temps d'exécution du processus (en secondes)

1. A propos des processus

- a) Rappeler rapidement ce qu'est le `pid` d'un processus
- b) Parmi les commandes suivantes, indiquer sur votre copie laquelle permet de tuer un processus en cours d'exécution.

- `delete`
- `stop`
- `stop`
- `end`
- `kill`
- `interrupt`

- c) Expliquer pourquoi `pid` peut être utilisé comme clé primaire de cette table et pas `user`

2. Quelques requêtes

- a) Ecrire une requête SQL permettant d'afficher les champs `pid` et `user`.
- b) Ecrire une requête SQL permettant d'afficher les processus de l'utilisateur `root`.
- c) Ecrire une requête SQL permettant d'afficher tous les fils du processus de `pid 712`.
- d) Ecrire une requête SQL permettant d'afficher les `pid` des processus ayant un temps d'exécution supérieur à 50 secondes.
- e) Ecrire une requête SQL permettant d'afficher les processus classés dans l'ordre alphabétique du propriétaire du processus.
- f)
- g) Ecrire une requête SQL permettant d'afficher les 5 processus ayant le plus long temps d'exécution.

