


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 autorisé.
- Le sujet comporte 4 pages, assurez-vous qu'il est complet avant de commencer.

Exercice 1 : Langages et programmation (récursivité)2022 Centres étrangers 

1. Voici une fonction codée en Python :

```

1 def f(n):
2     if f(n) == 0:
3         print("Partez !")
4     else:
5         print(n)
6         f(n-1)

```

- a) Qu'affiche la commande `f(5)` ?
- b) Pourquoi dit-on de cette fonction qu'elle est récursive ?
2. On rappelle qu'en python l'opérateur `+` a le comportement suivant sur les chaînes de caractères :

```

1 >>> S = 'a' + 'bc'
2 >>> S
3 'abc'

```

Et le comportement suivant sur les listes :

```

1 >>> L = ['a'] + ['b', 'c']
2 >>> L
3 ['a', 'b', 'c']

```

On a besoin pour les questions suivantes de pouvoir ajouter une chaîne de caractères `s` en préfixe à chaque chaîne de caractères de la liste `liste`. On appellera cette fonction `ajouter`.

Par exemple, `ajouter("a", ["b", "c"])` doit renvoyer `["ac", "ac"]`.

a) Recopier le code suivant et compléter les pointillées :

```

1 def ajouter(s, liste):
2     res = []
3     for m in liste:
4         res .....
5     return res

```

- b) Que renvoie la commande `ajouter("b", ["a", "b", "c"])` ?
- c) Que renvoie la commande `ajouter("a", [])` ?
3. On s'intéresse ici à la fonction suivante écrite en Python où `s` est une chaîne de caractères et `n` un entier naturel.

```

1 def produit(s, n):
2     if n == 0:
3         return []
4     else:
5         res = []
6         for i in range(len(s)):
7             res = res + ajouter(s[i], produit(s, n-1))
8         return res

```

- a) Que renvoie la commande `produit("ab", 0)` ? Le résultat est-il une liste vide ?
- b) Que renvoie la commande `produit("ab", 1)` ?
- c) Que renvoie la commande `produit("ab", 2)` ?

□ Exercice 2 : Programmation objet en langage Python

2022 Nouvelle Calédonie 🎓

Des élus d'un canton français décident de mettre en place une monnaie locale complémentaire dans leur circonscription, appelée le « sou ». L'objectif est d'encourager la population à acheter auprès de vendeurs et producteurs locaux.

La plus petite valeur du sou est le billet de 1 sou, il ne peut donc pas être séparé en dessous de ce montant. La sou a un taux de change à parité avec l'euro (1 sou = 1 euro), de façon à ce que le prix d'un article puisse être intégralement payé en sous. Si le prix d'un article n'est pas entier, la différence peut être payée en euros. Par exemple, un article coûtant 3,50 € peut être payé avec 3 sous et 50 centimes d'euros.

Le canton a créé une association chargée de gérer les comptes de ses futurs adhérents au moyen d'une application en langage Python. On a commencé à créer une classe **Compte** dont les propriétés sont : le numéro de compte, le nom de l'adhérent, son adresse et le solde du compte. Lors de la création d'un compte, il faudra renseigner toutes les propriétés sauf le solde qui sera mis à 0. Les méthodes de cette classe sont les suivantes :

Méthodes de la classe Compte	Description
<code>__init__(self, numero, adherent, adresse)</code>	Crée un nouveau compte et met le solde à 0.
<code>crediter(self, montant)</code>	Ajoute montant au solde
<code>debiter(self, montant)</code>	Enlève montant au solde
<code>est_positif(self)</code>	Renvoie True si le solde du compte est positif ou nul

Partie 1 : Création de la classe Compte

On a commencé à écrire la classe **Compte**.

```

1 class Compte:
2     def __init__(self, numero, adherent, adresse):
3         self.numero = numero
4         self.adherent = adherent
5         self.adresse = adresse
6         self.solde = 0

```

1. Ecrire sur la copie la méthode **crediter**.
2. Ecrire sur la copie la méthode **débiter**.
3. Ecrire sur la copie la méthode **est_positif**.

Partie 2 : Utilisation de la classe Compte

Monsieur Martin souhaite rejoindre la communauté des utilisateurs du sou et déposer 200 € sur son compte en sou.

1. Ecrire la ligne de code en langage Python permettant de créer le compte **cpt_0123456A** dont le numéro est "0123456A", le titulaire est "MARTIN Dominique" qui habite à l'adresse "12 rue des sports".
2. Ecrire la ligne de code en langage Python permettant de déposer 200 € sur le compte de monsieur Martin.
3. Monsieur Martin souhaite transférer 50 sous de son compte **cpt_0123456A** vers un autres compte **cpt_2468794E**. On a besoin pour cela d'ajouter une méthode à la classe **Compte**.
Ecrire la méthode **transférer(self, autre_compte, montant)** qui transfère le montant passé en paramètre vers un autre compte. On suppose que le compte courant est suffisamment approvisionné. On utilisera les autres méthodes de la classe **Compte** sans en modifier directement les attributs.

Partie 3 : Gestion des comptes

Pour en faciliter la gestion, on crée une liste Python **liste_comptes** contenant tous les comptes des adhérents. Cette liste sera donc composée d'objets de type **Compte**.

Le gestionnaire de l'association souhaite relancer les adhérents dont le compte est débiteur, c'est à dire dont le solde est négatif. Il faut, pour cela, ajouter une autre fonction au programme en langage Python.


1. Ecrire la fonction **recherche_debiteurs** qui prend en argument **liste_comptes** et renvoie une liste de dictionnaires dont la première clé est le nom de l'adhérent et la seconde clé le solde de son compte en négatif.

Exemple de résultat :

```
[{'Nom' : 'DUPONT Thomas', 'solde': -60}, {'Nom' : 'CARNEIRO Sarah', 'solde': -75}]
```

2. Ecrire la fonction `urgent_debiteur` qui prend en argument la liste de dictionnaires et renvoie le nom de l'adhérent le plus endetté. On suppose qu'aucun adhérent n'a la même dette.

□ 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 autre 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 utilisé dans chaque cas, et on indiquera **None** si la fonction ne renvoie aucune valeur :

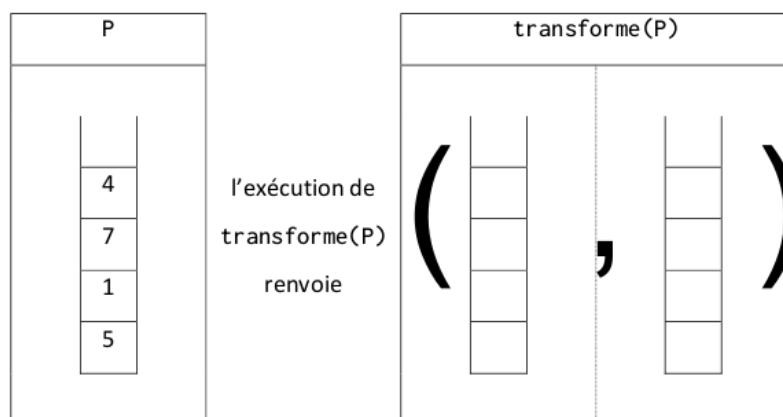
Etapas	Etape 0 Pile d'origine P	Etape 1 <code>empiler(P,8)</code>	Etape 2 <code>depiler(P)</code>	Etape 3 <code>est_vide(P)</code>
Pile	<div style="border: 1px solid black; padding: 5px; text-align: center;"> <div style="border: 1px solid black; height: 20px; width: 30px; margin: 0 auto;"></div> <div style="border: 1px solid black; height: 20px; width: 30px; margin: 0 auto;">4</div> <div style="border: 1px solid black; height: 20px; width: 30px; margin: 0 auto;">7</div> <div style="border: 1px solid black; height: 20px; width: 30px; margin: 0 auto;">1</div> <div style="border: 1px solid black; height: 20px; width: 30px; margin: 0 auto;">5</div> </div>	<div style="border: 1px solid black; padding: 5px; text-align: center;"> <div style="border: 1px solid black; height: 20px; width: 30px; margin: 0 auto;">...</div> <div style="border: 1px solid black; height: 20px; width: 30px; margin: 0 auto;">...</div> <div style="border: 1px solid black; height: 20px; width: 30px; margin: 0 auto;">...</div> <div style="border: 1px solid black; height: 20px; width: 30px; margin: 0 auto;">...</div> <div style="border: 1px solid black; height: 20px; width: 30px; margin: 0 auto;">...</div> </div>	<div style="border: 1px solid black; padding: 5px; text-align: center;"> <div style="border: 1px solid black; height: 20px; width: 30px; margin: 0 auto;">...</div> <div style="border: 1px solid black; height: 20px; width: 30px; margin: 0 auto;">...</div> <div style="border: 1px solid black; height: 20px; width: 30px; margin: 0 auto;">...</div> <div style="border: 1px solid black; height: 20px; width: 30px; margin: 0 auto;">...</div> <div style="border: 1px solid black; height: 20px; width: 30px; margin: 0 auto;">...</div> </div>	<div style="border: 1px solid black; padding: 5px; text-align: center;"> <div style="border: 1px solid black; height: 20px; width: 30px; margin: 0 auto;">...</div> <div style="border: 1px solid black; height: 20px; width: 30px; margin: 0 auto;">...</div> <div style="border: 1px solid black; height: 20px; width: 30px; margin: 0 auto;">...</div> <div style="border: 1px solid black; height: 20px; width: 30px; margin: 0 auto;">...</div> <div style="border: 1px solid black; height: 20px; width: 30px; margin: 0 auto;">...</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`
- `remove`
- `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) Ecrire une requête SQL permettant d'afficher les 5 processus ayant le plus long temps d'exécution.