

NUMÉRIQUE ET SCIENCES INFORMATIQUES

Durée de l'épreuve : 3 heures 30

L'usage de la calculatrice n'est pas autorisé

*Le candidat traite au choix 3 exercices parmi les 4 exercices
proposés*

Chaque exercice devra être traité sur une copie séparée

Exercice 1

arbres et programmation orientée objet

Une agence immobilière développe un programme pour gérer les biens immobiliers qu'elle propose à la vente.

Dans ce programme, pour modéliser les données de biens immobiliers, on définit une classe **Bim** avec les attributs suivants :

- **nt** de type **str** représente la nature du bien (appartement, maison, bureau, commerces, *et cætera*);
- **sf** de type **float** est la surface du bien;
- **pm** de type **float** est le prix moyen par m² du bien qui dépend de son emplacement.

La classe **Bim** possède une méthode **estim_prix** qui renvoie une estimation du prix du bien. Le code (incomplet) de la classe **Bim** est donné ci-dessous :

Code Python

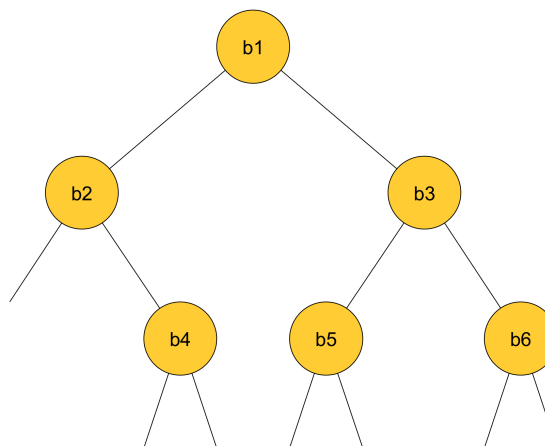
```
class Bim:
    def __init__(self, nature, surface, prix_moy):
        ...

    def estim_prix(self):
        return self.sf * self.pm
```

1. Recopier et compléter le code du constructeur de la classe **Bim**.
2. On exécute l'instruction suivante :
`b1 = Bim('maison', 70.0, 2000.0)`
Que renvoie l'instruction `b1.estim_prix()`? Préciser le type de la valeur renvoyée.
3. On souhaite affiner l'estimation du prix d'un bien en prenant en compte sa nature :
 - pour un bien dont l'attribut **nt** est **'maison'** la nouvelle estimation du prix est le produit de sa surface par le prix moyen par m² multiplié par 1,1;
 - pour un bien dont l'attribut **nt** est **'bureau'** la nouvelle estimation du prix est le produit de sa surface par le prix moyen par m² multiplié par 0,8;
 - pour les autres biens, l'estimation du prix ne change pas.

Modifier le code de la méthode **estim_prix** afin de prendre en compte ces changements.

4. Écrire le code Python d'une fonction `nb_maison` qui
- prend en argument une liste `lst` d'objets de la classe `Bim`;
 - renvoie le nombre d'objets de nature `'maison'` contenus dans `lst`.
5. Pour une recherche efficace des biens immobiliers selon le critère de leur surface, on stocke les objets de la classe `Bim` dans un arbre binaire de recherche, nommé `abr`.
Pour tout nœud de cet arbre :
- tous les objets de son sous-arbre gauche ont une surface inférieure ou égale à la surface de l'objet contenue dans ce nœud;
 - tous les objets de son sous-arbre droit ont une surface strictement supérieure à la surface de l'objet contenue dans ce nœud.
- `abr` est lui-même un objet et dispose des **méthodes** suivantes :
- `abr.est_vide` ne prend aucun paramètre d'entrée, renvoie **True** si `abr` est vide et **False** sinon.
 - `abr.get_v` ne prend aucun paramètre d'entrée, renvoie l'élément situé à la racine de `abr` si `abr` n'est pas vide et **None** sinon.
 - `abr.get_g` : ne prend aucun paramètre d'entrée, renvoie le sous-arbre gauche de `abr` si `abr` n'est pas vide et **None** sinon.
 - `abr.get_f` : ne prend aucun paramètre d'entrée, renvoie le sous-arbre droit de `abr` si `abr` n'est pas vide et **None** sinon.
- a. Dans cette question, on suppose que l'arbre binaire `abr` a la forme ci-dessous :



Donner la liste des biens triée dans l'ordre croissant de leur surface en expliquant la méthode choisie.

- b. Recopier et compléter le code de la fonction récursive `contient` donnée ci-dessous, qui

- prend en arguments un nombre **surface** de type **float** et un arbre binaire de recherche **abr** contenant des éléments de la classe **Bim** ordonnés selon leur attribut de surface **sf**;
- renvoie **True** s'il existe un bien dans **abr** d'une surface supérieure ou égale à surface et **False** sinon.

Code Python

```
def contient(surface, abr):
    if abr.est_vide():
        return False
    elif abr.get_v().sf >= ..... :
        return True
    else:
        return contient( surface , ..... )
```

Exercice 2

bases de données, langage SQL

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

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

Pour la gestion des réservations clients, on dispose d'une base de données nommée **gare** dont le schéma relationnel est le suivant :

Train (numT, provenance, destination, horaireArrivee, horaireDepart)

Reservation (numR, nomClient, prenomClient, prix, numT)

Les attributs soulignés en trait plein sont des clés primaires. L'attribut souligné en pointillés est une clé étrangère : **Reservation.numT** fait référence à la clé primaire **Train.numT**.

Les attributs **horaireDepart** et **horaireArrivee** sont de type **TIME** et s'écrivent selon le format **hh:mm**, où **hh** représente les heures et **mm** les minutes.

1. Quel nom générique donne-t-on aux logiciels qui assurent, entre autres, la persistance des données, l'efficacité de traitement des requêtes et la sécurisation des accès pour les bases de données ?

2. a. On considère les requêtes SQL suivantes :

Code SQL

```
DELETE FROM Train WHERE numT = 1241 ;  
DELETE FROM Reservation WHERE numT = 1241 ;
```

Sachant que le train n°1241 a été enregistré dans la table **Train** et que des réservations pour ce train ont été enregistrées dans la table **Reservation**, expliquer pourquoi cette suite d'instructions renvoie une erreur.

- b. Citer un cas pour lequel l'insertion d'un enregistrement dans la table **Reservation** n'est pas possible.
3. Écrire des requêtes SQL correspondant à chacune des instructions suivantes :
- a. Donner tous les numéros des trains dont la destination est « Lyon ».
 - b. Ajouter une réservation n°1307 de 33 € pour M. Alan Turing dans le train n°654.
 - c. Suite à un changement, l'horaire d'arrivée du train n°7869 est programmé à 08 : 11. Mettre à jour la base de données en conséquence.
 - d. Produire la table des noms et prénoms de tous les clients qui ont effectué une réservation pour un train partant de Paris et allant à Marseille.

Exercice 3

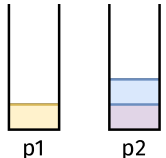
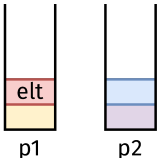
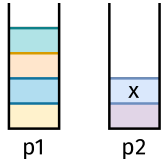
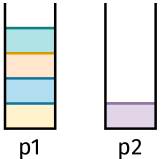
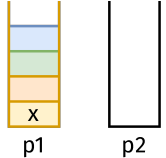
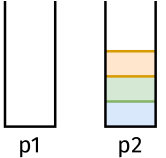
structures de données linéaires

Une méthode simple pour gérer l'ordonnancement des processus est d'exécuter les processus en une seule fois et dans leur ordre d'arrivée.

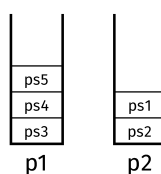
1. Parmi les propositions suivantes, quelle est la structure de données la plus appropriée pour mettre en œuvre le mode FIFO (First In First Out)?
 - a. liste
 - b. dictionnaire
 - c. pile
 - d. file
2. On choisit de stocker les données des processus en attente à l'aide d'une liste Python `lst`. On dispose déjà d'une fonction `retirer(lst)` qui renvoie l'élément `lst[0]` puis le supprime de la liste `lst`.

Écrire en Python le code d'une fonction `ajouter(lst, proc)` qui ajoute à la fin de la liste `lst` le nouveau processus en attente `proc`.

3. On choisit maintenant d'implémenter une file **file** à l'aide d'un couple (**p1**,**p2**) où **p1** et **p2** sont des piles. Ainsi **file[0]** et **file[1]** sont respectivement les piles **p1** et **p2**.
- Pour enfiler un nouvel élément **elt** dans file, on l'empile dans **p1**.
 - Pour défiler file, deux cas se présentent :
 - La pile **p2** n'est pas vide : on dépile **p2**.
 - La pile **p2** est vide : on dépile les éléments de **p1** en les empilant dans **p2** jusqu'à ce que **p1** soit vide, puis on dépile **p2**.

	État initial de la file	État final de la file
enfiler(file, elt)	 <p>p1 p2</p>	 <p>p1 p2</p>
defiler(file) cas où p2 n'est pas vide	 <p>p1 p2</p>	 <p>p1 p2</p>
defiler(file) cas où p2 est vide	 <p>p1 p2</p>	 <p>p1 p2</p>

On considère la situation représentée ci-dessous.



On exécute la séquence d'instructions suivante :

```

enfiler(file, ps6)
defiler(file)
defiler(file)
defiler(file)
enfiler(file, ps7)

```

Représenter le contenu final des deux piles à la suite de ces instructions.

4. On dispose des fonctions :

- **empiler(p,elt)** qui empile l'élément **elt** dans la pile **p**;

- `depiler(p)` qui renvoie le sommet de la pile `p` si `p` n'est pas vide et le supprime ;
 - `pile_vide(p)` qui renvoie **True** si la pile `p` est vide, **False** si la pile `p` n'est pas vide.
- a. Écrire en Python une fonction `est_vide(f)` qui prend en argument un *couple de piles* `f` et qui renvoie **True** si la file représentée par `f` est vide, **False** sinon.
 - b. Écrire en Python une fonction `enfiler(f,elt)` qui prend en arguments un **couple de piles** `f` et un élément `elt` et qui ajoute `elt` en queue de la file représentée par `f`.
 - c. Écrire en Python une fonction `defiler(f)` qui prend en argument un **couple de piles** `f` et qui renvoie l'élément en tête de la file représentée par `f` en le retirant. On supposera que la file `f` n'est pas vide.

Exercice 4

gestion des processus et ressources

Les parties A et B peuvent être traitées indépendamment.

Partie A

Dans un bureau d'architectes, on dispose de certaines ressources qui ne peuvent être utilisées simultanément par plus d'un processus, comme l'imprimante, la table traçante, le modem.

Chaque programme, lorsqu'il s'exécute, demande l'allocation des ressources qui lui sont nécessaires. Lorsqu'il a fini de s'exécuter, il libère ses ressources.

Programme 1	Programme 2	Programme 3
demander table traçante	demander modem	demander imprimante
demander modem	demander imprimante	demander table traçante
exécution	exécution	exécution
libérer modem	libérer imprimante	libérer table traçante
libérer table traçante	libérer modem	libérer imprimante

On appelle P1, P2 et P3 les processus respectivement associés aux programmes 1, 2 et 3. Ces processus s'effectuent de manière concurrente.

1. Justifier qu'une situation d'interblocage peut se produire.
2. Modifier l'ordre des instructions du programme 3 pour qu'une telle situation ne puisse pas se produire (aucune justification n'est attendue).
3. Supposons que le processus P1 demande la table traçante alors qu'elle est en cours d'utilisation par le processus P3.
Parmi les états suivants, quel sera l'état du processus P1 tant que la table traçante n'est pas disponible ?

a. élu

b. bloqué

c. prêt

d. terminé

Partie B

Avec une ligne de commande dans un terminal sous Linux, on obtient l'affichage suivant :

```

UID      PID  PPID  C  STIME TTY      TIME CMD
...
pi      6211   831   8  09:07 ?        00:01:16 /usr/lib/chromium-browser/chromium-browser-v7 --disable-quic --enable-tcp-fast-open --c
pi      6252  6211   0  09:07 ?        00:00:00 /usr/lib/chromium-browser/chromium-browser-v7 --type=zygote --ppapi-flash-path=/usr/lib
pi      6254  6252   0  09:07 ?        00:00:00 /usr/lib/chromium-browser/chromium-browser-v7 --type=zygote --ppapi-flash-path=/usr/lib
pi      6294  6211   4  09:07 ?        00:00:40 /usr/lib/chromium-browser/chromium-browser-v7 --type=gpu-process --field-trial-handle=1
pi      6300  6211   1  09:07 ?        00:00:16 /usr/lib/chromium-browser/chromium-browser-v7 --type=utility --field-trial-handle=1075
pi      6467  6254   1  09:07 ?        00:00:11 /usr/lib/chromium-browser/chromium-browser-v7 --type=renderer --field-trial-handle=1075
pi     11267  6254   2  09:12 ?        00:00:15 /usr/lib/chromium-browser/chromium-browser-v7 --type=renderer --field-trial-handle=1075
pi     12035   836   0  09:13 ?        00:00:00 /usr/lib/libreoffice/program/oosplash --writer file:///home/pi/Desktop/mon_fichier.odt
pi     12073 12035   2  09:13 ?        00:00:15 /usr/lib/libreoffice/program/soffice.bin --writer file:///home/pi/Desktop/mon_fichier.c
pi     12253   831   1  09:13 ?        00:00:07 /usr/bin/python3 /usr/bin/sense_emu_gui
pi     20010  6211   1  09:21 ?        00:00:00 /usr/lib/chromium-browser/chromium-browser-v7 --type=utility --field-trial-handle=1075
pi     20029  6254  56  09:21 ?        00:00:28 /usr/lib/chromium-browser/chromium-browser-v7 --type=renderer --field-trial-handle=1075
pi     20339  6254   4  09:21 ?        00:00:01 /usr/lib/chromium-browser/chromium-browser-v7 --type=renderer --field-trial-handle=1075
pi     20343  6254   2  09:21 ?        00:00:00 /usr/lib/chromium-browser/chromium-browser-v7 --type=renderer --field-trial-handle=1075
pi     20464  6211  17  09:22 ?        00:00:00 /proc/self/exe --type=utility --field-trial-handle=1075063133478094917,6306120996223181
pi     20480  6254  14  09:22 ?        00:00:00 /usr/lib/chromium-browser/chromium-browser-v7 --type=renderer --field-trial-handle=1075
pi     20519   676   0  09:22 pts/0    00:00:00 ps -ef

```

La documentation Linux donne la signification des différents champs :

- UID : identifiant utilisateur effectif;
- PID : identifiant de processus;
- PPID : PID du processus parent;
- C : partie entière du pourcentage d'utilisation du processeur par rapport au temps de vie des processus;
- STIME : l'heure de lancement du processus;
- TTY : terminal de contrôle;
- TIME : temps d'exécution;
- CMD : nom de la commande du processus.

1. Parmi les quatre commandes suivantes, laquelle a permis cet affichage ?

a. `ls -l`c. `cd ..`b. `ps -ef`d. `chmod 741 processus.txt`

2. Quel est l'identifiant du processus parent à l'origine de toutes les processus concernant le navigateur Web (chromium-browser)?

3. Quel est l'identifiant du processus dont le temps d'exécution est le plus long?