# Centres étrangers - mai 2022 - sujet 1

**Exercice 1 (Dictionnaires et listes - 4 points)**
On dispose de la liste `jours` suivante et du dictionnaire `mois` suivant :

```
jours = ["dimanche", "lundi", "mardi", "mercredi", "jeudi", "vendredi", "samedi"]
mois = {1 :("janvier",31) , 2 :("février",28) , 3 :("mars",31) , 4 :("avril",30) ,
        5 :("mai",31) , 6 :("juin",30) , 7 :("juillet",31) , 8 :("aout",31) ,
        9 :("septembre",30) , 10 :("octobre",31) , 11 :("novembre",30) ,
        12 :("décembre",31)}
```

1. (a) A partir de la liste jours, comment obtenir l'élément `"lundi"` ?
   (b) On rappelle que l'opérateur « modulo » `%` renvoie le reste de la division entière (division euclidienne).
   Exemple : `7%3` renvoie `1` qui est le reste de la division de `7` par `3` :

$$\begin{array}{r|l} 7 & 3 \\ \cline{2-2} -6 & 2 \\ \cline{1-1} 1 & \end{array}$$

   Que renvoie l'instruction `jours[18%7]` ?
2. On rappelle que `jours.index[element]` renvoie l'indice de `element` dans la liste `jours`.
   Par exemple, `jours.index["mercredi"]` renvoie `3`.
   Le nom du jour actuel est stocké dans une variable `j` (par exemple : `j = "mardi"`).
   Recopier et compléter l'instruction suivante permettant d'obtenir le numéro du jour de la semaine `n` jours plus tard :

```
numero_jour =(jours.index( ... ) + ... )% ...
```

3. (a) A partir du dictionnaire `mois`, comment obtenir le nombre de jours du mois de mars ?
   (b) Le numéro du mois actuel est stocké dans une variable `numero_mois`.
   Ecrire le code permettant d'obtenir le nom du mois qu'il sera `x` mois plus tard à partir du dictionnaire `mois`. Par exemple :
   - si `numero_mois = 4` et `x = 5`, on doit obtenir `"septembre"` ;
   - si `numero_mois = 10` et `x = 3`, on doit obtenir `"janvier"`.
4. On définit une date comme un tuple : `(nom_jour,numero_jour, numero_mois,annee)`.
   (a) Sachant que `date = ("samedi", 21, 10, 1995)`, que renvoie `mois[date[2]][1]` ?
   (b) Ecrire une fonction `jour_suivant` qui prend en paramètre une date `date` sous forme de tuple et qui renvoie un tuple désignant la date du lendemain. Par exemple :
   - `jour_suivant(("samedi", 21, 10, 1995))` renvoie `("dimanche", 22, 10, 1995)` ;
   - `jour_suivant(("mardi", 31, 10, 1995))` renvoie `("mercredi", 1, 11, 1995)`.

   On ne tient pas compte des années bissextiles et on considère que le mois de février comporte toujours 28 jours.

**Exercice 2 (Files et POO - 4 points)**
Un supermarché met en place un système de passage automatique en caisse. Un client scanne les articles à l'aide d'un scanner de code-barres au fur et à mesure qu'il les ajoute dans son panier. Les articles s'enregistrent alors dans une structure de données.

La structure de données utilisée est une file définie par la classe `Panier`, avec les primitives habituelles sur la structure de file. Pour faciliter la lecture, le code de la classe `Panier` n'est pas écrit.

```
class Panier():
    def __init__(self):
        """Initialise la file comme une file vide."""

    def est_vide(self):
        """Renvoie True si la file est vide, False sinon."""

    def enfiler(self, e):
        """Ajoute l'élément e en dernière position de la file, ne renvoie rien."""

    def defiler(self):
        """Retire le premier élément de la file et le renvoie."""
```

Le panier d'un client sera représenté par une file contenant les articles scannés. Les articles sont représentés par des tuples `(code_barre, designation, prix, horaire_scan)` où

- `code_barre` est un nombre entier identifiant l'article ;
- `designation` est une chaine de caractères qui pourra être affichée sur le ticket de caisse ;
- `prix` est un nombre décimal donnant le prix d'une unité de cet article ;
- `horaire_scan` est un nombre entier de secondes permettant de connaitre l'heure où l'article a été scanné.

1. On souhaite ajouter un article dont le tuple est le suivant `(31002, "café noir", 1.50, 50525)`.
   Ecrire le code utilisant une des quatre méthodes ci-dessus permettant d'ajouter l'article à l'objet de classe `Panier` appelé `panier1`.
2. On souhaite définir une méthode `remplir(panier_temp)` dans la classe `Panier` permettant de remplir la file avec tout le contenu d'un autre panier `panier_temp` qui est un objet de type `Panier`.
   Recopier et compléter le code de la méthode `remplir` en remplaçant chaque `.....` par la primitive de file qui convient.

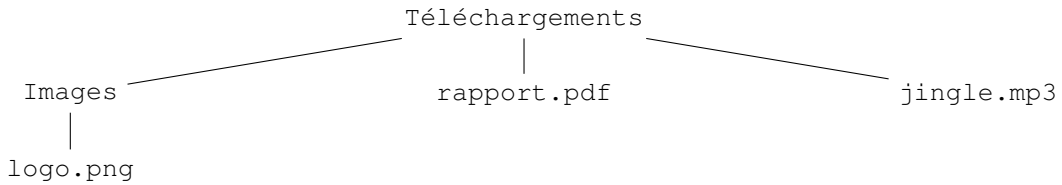
```
def chiffrer(mot, alpha):
    mc = ""
    for l in mot:
        mc = mc + alpha[l]
    return mc
```

3. Pour que le client puisse connaitre à tout moment le montant de son panier, on souhaite ajouter une méthode `prix_total` à la classe `Panier` qui renvoie la somme des prix de tous les articles présents dans le panier.
   Ecrire le code de la méthode `prix_total`. Attention, après l'appel de cette méthode, le panier devra toujours contenir ses articles.
4. Le magasin souhaite connaitre pour chaque client la durée des achats. Cette durée sera obtenue en faisant la différence entre le champ `horaire_scan` du dernier article scanné et le champ `horaire_scan` du premier article scanné dans le panier du client. Un panier vide renverra une durée égale à zéro. On pourra accepter que le panier soit vide après l'appel de cette méthode.
   Ecrire une méthode `duree_courses` de la classe `Panier` qui renvoie cette durée.

**Exercice 3 (Dictionnaires - 4 points)**
Afin d'organiser les dossiers et les fichiers sur un disque dur, une structure arborescente est utilisée. Les fichiers sont dans des dossiers qui sont eux-mêmes dans d'autres dossiers, etc.

Dans une arborescence, chaque dossier peut contenir des fichiers et des dossiers, qui sont identifiés par leur nom. Le contenu d'un dossier est modélisé par la structure de données dictionnaire. Les clés de ce dictionnaire sont des chaînes de caractères donnant le nom des fichiers et des dossiers contenus.

Illustration par un exemple : le dossier appelé `Téléchargements` contient deux fichiers `rapport.pdf` et `jingle.mp3`, et un dossier `Images` contenant simplement le fichier `logo.png`. Il est représenté ci-dessous.




Ce dossier `Téléchargements` est modélisé en Python par le dictionnaire suivant :

```
{"Images": {"logo.png": 36}, "rapport.pdf": 450, "jingle.mp3": 4800}
```

Les valeurs numériques sont exprimées en ko (kilo-octets). Ainsi, `"logo.png": 36` signifie que le fichier `logo.png` occupe un espace mémoire de 36ko sur le disque dur.

On rappelle ci-dessous quelques commandes sur l'utilisation d'un dictionnaire :

- `dico = {}` crée un dictionnaire vide appelé `dico`;
- `dico[cle] = contenu` met la valeur `contenu` pour la clé `cle` dans le dictionnaire `dico`;
- `dico[cle]` renvoie la valeur associée à la clé `cle` dans le dictionnaire `dico`;
- `cle in dico` renvoie un booléen indiquant si la clé `cle` est présente dans le dictionnaire `dico`;
- `del dico[cle]` supprime la clé `cle` et sa valeur associée du dictionnaire `dico`;
- `dico.keys()` renvoie la liste des clés du dictionnaire `dico`.

L'adresse d'un fichier ou d'un dossier correspond au nom de tous les dossiers à parcourir depuis la racine afin d'accéder au fichier ou au dossier. Cette adresse est modélisée en Python par la liste des noms de dossier à parcourir pour y accéder. Par exemple, l'adresse du dossier `/home/pierre/Documents/` est modélisée par la liste `["home", "pierre", "Documents"]`.

1. Dessiner l'arbre donné par le dictionnaire suivant, qui correspond au dossier `Documents`.

```
Documents = {"Administratif": {"certificat JDC.pdf ": 1500,
                               "attestation recensement.pdf ": 850
                              },
             "Cours": {"NSI": {"TP.html ": 60,
                               "dm.odt": 34
                              },
                       "Philo": {"Tractatus logico-philosophicus.epub": 2600}
                      },
             "liste de courses.txt ": 24
            }
```

2. (a) On donne la fonction `parcourir` suivante qui prend en paramètres un dossier racine et une liste représentant une adresse, et qui renvoie le contenu du dossier cible correspondant à l'adresse.
Exemple : si la variable `Documents` contient le dictionnaire de l'exemple de la question 1 alors `parcourir(Documents, ["Cours", "Philo"])` renvoie le dictionnaire `{"Tractatus logicophilosophicus.epub": 2600}`.
Recopier et compléter la ligne 4.

```
1 SELECT auteur FROM Evaluations
```

   (b) Soit la fonction suivante :

```
SELECT Nom_evaluation, Date FROM Evaluations WHERE auteur= "Peltier"
```

Qu'affiche l'instruction `afficher(Documents, ["Cours", "NSI"], "TP.html")` sachant que la variable `Document` contient le dictionnaire de la question 1 ?

3. (a) La fonction `ajouter_fichier(racine, adr, nom_fichier, taille)` suivante ajoute au dictionnaire `racine`, à l'adresse `adr`, la clé `nom_fichier` associé à la valeur `taille`.
Une ligne de la fonction donnée ci-dessous contient une erreur. Laquelle ? Proposer une correction.

```
def ajouter_fichier(racine, adr, nom_fichier, taille):
    dossier = parcourir(racine, adr)
    taille = dossier[nom_fichier]
```

   (b) Ecrire une fonction `ajouter_dossier(racine, adr, nom_dossier)` pour créer un dictionnaire représentant un dossier vide appelé `nom_dossier` dans le dictionnaire `racine` à l'adresse `adr`.

4. Ecrire une fonction `taille` qui prend en paramètre un dictionnaire `dossier` modélisant le contenu du répertoire `dossier` et qui renvoie le total de l'espace mémoire occupé par les fichiers contenus dans le dossier. On considère que le répertoire `dossier` ne contient que des fichiers et aucun sous-dossier.

**Exercice 4 (Bases de données et SQL - 4 points)**

**Un rappel sur la syntaxe de quelques fonctions SQL est donné en annexe 1 en fin de sujet.**

Dans le cadre d'une étude sur le réchauffement climatique, un centre météorologique rassemble des données. On considère que la base de données contient deux relations (tables) :

- la relation `Centres` qui contient l'identifiant des centres météorologiques, la ville, la latitude, la longitude et l'altitude du centre ;
- la relation`Mesures` qui contient l'identifiant de la mesure, l'identifiant du centre, la date de la mesure, la température, la pression et la pluviométrie mesurées.

Le schéma relationnel de la relation `Centres` est le suivant :

`Centres(id_centre: INT, nom_ville: VARCHAR, latitude: FLOAT, longitude: FLOAT, altitude: FLOAT)`

Le schéma relationnel de la relation `Mesures` est le suivant :

`Mesures(id_mesure: INT, id_centre: INT, date: DATE, temperature: FLOAT, pression: INT, pluviometrie: FLOAT)`

Relation `Centres`

| id_centre | nom_ville | latitude | longitude | altitude |
|---|---|---|---|---|
| 213 | Amiens | 49.894 | 2.293 | 60 |
| 138 | Grenoble | 45.185 | 5.723 | 550 |
| 263 | Brest | 48.388 | -4.49 | 52 |
| 185 | Tignes | 45.469 | 6.909 | 2594 |
| 459 | Nice | 43.706 | 7.262 | 260 |
| 126 | Le Puy-en-Velay | 45.042 | 3.888 | 744 |
| 317 | Gérardmer | 48.073 | 6.879 | 855 |

Relation `Mesures`

| id_mesure | id_centre | date | temperature | pression | pluviometrie |
|---|---|---|---|---|---|
| 1566 | 138 | 2021-10-29 | 8.0 | 1015 | 3 |
| 1568 | 213 | 2021-10-29 | 15.1 | 1011 | 0 |
| 2174 | 126 | 2021-10-30 | 18.2 | 1023 | 0 |
| 2200 | 185 | 2021-10-30 | 5.6 | 989 | 20 |
| 2232 | 459 | 2021-10-31 | 25.0 | 1035 | 0 |
| 2514 | 213 | 2021-10-31 | 17.4 | 1020 | 0 |
| 2563 | 126 | 2021-11-01 | 10.1 | 1005 | 15 |
| 2592 | 459 | 2021-11-01 | 23.3 | 1028 | 2 |
| 3425 | 317 | 2021-11-02 | 9.0 | 1012 | 13 |
| 3430 | 138 | 2021-11-02 | 7.5 | 996 | 16 |
| 3611 | 263 | 2021-11-03 | 13.9 | 1005 | 8 |
| 3625 | 126 | 2021-11-03 | 10.8 | 1008 | 8 |

1. (a) Proposer une clé primaire pour la relation `Mesures`. Justifier votre choix.
   (b) Avec quel attribut peut-on faire une jointure entre la relation `Centres` et la relation `Mesures` ?
2. (a) Qu'affiche la requête suivante ?

```
SELECT * FROM Centres WHERE altitude>500
```

   (b) On souhaite récupérer le nom de la ville des centres météorologiques situés à une altitude comprise entre 700m et 1200m. Ecrire la requête SQL correspondante.

   (c) On souhaite récupérer la liste des longitudes et des noms des villes des centres météorologiques dont la longitude est supérieure à 5. La liste devra être triée par ordre alphabétique des noms de ville. Ecrire la requête SQL correspondante.

3. (a) Qu'affiche la requête suivante ?

```
SELECT * FROM Mesures WHERE date="2021-10-30"
```

(b) Ecrire une requête SQL permettant d'ajouter une mesure prise le 8 novembre 2021 dans le centre numéro 138, où la température était de 11◦C, la pression de 1013 hPa et la pluviométrie de 0 mm. La donnée dont l'attribut est `id_mesure` aura pour valeur 3650.

4. (a) Expliquer ce que renvoie la requête SQL suivante ?

```
SELECT * FROM Centres WHERE latitude = (SELECT MIN(latitude) FROM Centres)
```

5. (b) Ecrire une requête SQL donnant la liste des villes dans lesquelles on a enregistré une température inférieure à 10◦C en octobre 2021. On utilisera le mot clé `DISTINCT` afin d'éviter d'avoir des doublons. On rappelle que l'on peut utiliser les opérateurs de comparaison avec les dates.

**Exercice 5 (Architecture matérielle, réseaux et protocoles - 4 points)**

Un nano-ordinateur est un ordinateur possédant une taille inférieure à un microordinateur. Les nano-ordinateurs (sans l'alimentation, le clavier, la souris et l'écran) tiennent dans la paume de la main. Le Soc (System on a cheap), littéralement un système sur une puce, est un système complet embarqué sur une seule puce (circuit intégré) pouvant comporter de la mémoire, un ou plusieurs microprocesseurs, des périphériques d'interface, ou tout autre composant. On souhaite comparer les performances de deux nano-ordinateurs contenant chacun un SOC différent dont les caractéristiques sont détaillées ci-dessous :

| | SOC de deux nano-ordinateurs | |
|---|---|---|
| Processeur | Broadcom BCM271 | Broadcom BCM2835 |
| Architecture | ARMv8-A (64-bit) | ARMv6Z (32-bit) |
| Microarchitecture | Cortex-A72 | ARM11 |
| Famille du processeur0 | BCM | BCM |
| Coeur | 4 | 1 |
| Fréquence de base | 1,5 GHz | 700 MHz |
| Fréquence turbo | - | 1,0 GHz |
| Mémoire cache | 1 MB | 128 KB |
| Capacité mémoire maxi | 8 GB | 512 MB |
| Types de mémoire | LPDDR4-3200 SDRAM | SDRAM |
| GPU (processeur graphique) integer | Broadcom VideoCore VI | Aucun |
| GPU, unités d'exécution | 4 | - |
| GPU, unités shader | 64 | |
| GPU, cadence | 500 MHz | |
| GPU, flottant FP32 | 32 GFLOPS | |
| Drystone MIPS | 22 740 DMIPS | 1 190 DMIPS |
| Résol affichage max | 4K@60fps | 1080p@30fps |
| Décodage vidéo | H.265 4K@60fps, H.264 1080p@60fps | H.264 1080p@30fps |
| Encodage vidéo | H.264 1080p@30fps | H.264 1080p@30fps |
| Interface réseau | 10/100/1000M Gigabit Ethernet | - |
| Connectivité | USB 2.0, USB 3.0, HDMI 2.0 | USB 2.0, HDMI 1.3 |
| Wifi | 2.4GHz/5GHz 802.11 b/g/n/ac | - |
| Bluetooth | Bluetooth 4.2 | - |
| Audio | I2S | I2S |

1. (a) Expliquer ce qui différencie un SOC d'un nano-ordinateur d'un microprocesseur classique ?
   (b) Lequel de ces SOC peut être connecté à un réseau filaire. Justifier la réponse.
   (c) Citer deux caractéristiques permettant de comparer la puissance de calcul de ces deux SOC.
2. Un réseau local est relié à internet à l'aide d'une box faisant office de routeur. Un utilisateur connecte un nouveau nano-ordinateur à ce réseau et veut tester son fonctionnement. Il utilise en premier la commande `ifconfig` qui correspond à `ipconfig` sous environnement Windows. Cela lui donne le résultat suivant.

```
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
        inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
        inet6 fe80::761a:3e85:cc97:6491 prefixlen 64 scopeid 0x20<link>
        ether 08:00:27:8b:c3:91 txqueuelen 1000 (Ethernet)
        RX packets 136 bytes 13703 (13.3 KiB)
        RX errors 0 dropped 0 overruns 0 frame 0
        TX packets 180 bytes 17472 (17.0 KiB)
        TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
        device interrupt 9 base 0xd020
```

   (a) L'indication `ether 08:00:27:8b:c3:91` correspond à une adresse MAC. Que représente-t-elle ?
   (b) On s'intéresse ensuite à l'indication `inet 10.0.2.15`. Que représente `10.0.2.15` ?
   (c) Pour connaitre la passerelle, l'utilisateur fait alors un `traceroute` dont la première ligne sortie est la ligne suivante :

```
1 _gateway (10.0.2.2) 0.328 ms 0.275 ms 0.267 ms
```

A quel type de matériel correspond l'adresse `10.0.2.2` ?

3. Cinq routeurs R1, R2, R3, R4, R5 sont connectés dans un réseau avec les caractéristiques suivantes :

| Routeur 1 (R1) | | | |
|---|---|---|---|
| Destination | Direction | Saut | Débit (Mbits/s) |
| R2 | R2 | 1 | 10 |
| R3 | R3 | 1 | 100 |
| R4 | R2 | 2 | |
| R5 | R5 | 1 | 10 |

| Routeur 2 (R2) | | | |
|---|---|---|---|
| Destination | Direction | Saut | Débit (Mbits/s) |
| R1 | R1 | 1 | 10 |
| R3 | R3 | 1 | 100 |
| R4 | R4 | 1 | 10 |
| R5 | R1 | 2 | |

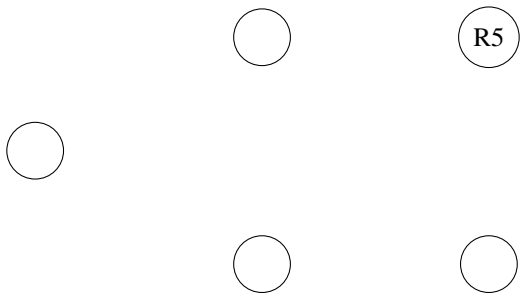
| Routeur 3 (R3) | | | |
|---|---|---|---|
| Destination | Direction | Saut | Débit (Mbits/s) |
| R1 | R1 | 1 | 100 |
| R2 | R2 | 1 | 100 |
| R4 | R2 | 2 | |
| R5 | R1 | 2 | |

| Routeur 4 (R4) | | | |
|---|---|---|---|
| Destination | Direction | Saut | Débit (Mbits/s) |
| R1 | R2 | 2 | |
| R2 | R2 | 1 | 10 |
| R3 | R2 | 2 | |
| R5 | R2 | 3 | |

| Routeur 5 (R5) | | | |
|---|---|---|---|
| Destination | Direction | Saut | Débit (Mbits/s) |
| R1 | R1 | 1 | 10 |
| R2 | R1 | 2 | |
| R3 | R1 | 2 | |
| R4 | R1 | 3 | |

Dans cette question, on utilise le protocole de routage RIP, qui cherche à minimiser le nombre de sauts.

(a) Recopier et compléter le schéma ci-dessous qui représente le réseau : indiquer le nom des routeurs dans les cercles et tracer les connexions entre eux.




(b) Quelle route faut-il prendre pour aller de R4 à R5 ?

4. Les cinq routeurs précédents sont connectés dans la même configuration que précédemment. Toutefois le protocole de routage appliqué est désormais le protocole OSPF qui prend en compte le débit (Mbits/s) pour minimiser le coût total de la transmission. Le coût pour passer d'un routeur à un autre est donné par la formule :

$$C = \frac{100}{\text{débit}}.$$

Quelle route faut-il prendre pour aller de R4 à R5 en respectant le protocole OSPF ?

**Annexe 1 – Langage SQL**

**(à ne pas rendre avec la copie)**

⋆ **Types de données :**

| | |
|---|---|
| `CHAR(t)` | Texte fixe de `t` caractères |
| `VARCHAR(t)` | Texte de `t` caractères variables |
| `TEXT` | Texte de $65\,535$ caractères maximum |
| `INT` | Nombre entier de $-2^{31}$ à $2^{31}-1$ (signé) ou de $0$ à $2^{32}-1$ (non signé) |
| `FLOAT` | Réel à virgule flottante |
| `DATE` | Date format `AAAA-MM-JJ` |

⋆ **Quelques exemples de syntaxe SQL :**

- Insérer des enregistrements :
```
INSERT INTO Table (attribut1, attribut2) VALUES(valeur1 , valeur2)
```

- Modifier des enregistrements :
```
UPDATE Table SET attribut1=valeur1, attribut2=valeur2 WHERE Selecteur
```

- Supprimer des enregistrements :
```
DELETE FROM Table WHERE Selecteur
```

- Sélectionner des enregistrements :
```
SELECT attributs FROM Table WHERE Selecteur
```

- Sélectionner des enregistrements dans un ordre ascendent :
```
SELECT attributs FROM Table WHERE Selecteur ORDER BY attribut ASC
```

- Sélectionner des enregistrements sans doublon :
```
SELECT DISTINCT attributs FROM Table WHERE Selecteur
```

- Effectuer une jointure :
```
SELECT attributs FROM TableA JOIN TableB ON TableA.cle1=TableB.cle2 WHERE Selecteur
```