Révisions_Générales

Exercice 1 : Réseaux

Principaux thèmes abordés : protocoles de communication, architecture d'un réseau et protocoles de routage.

Les parties A et B sont indépendantes.

Partie A : Réseau

1. Parmi les termes ci-dessous, préciser celui qui désigne l'ensemble des règles de communication utilisées pour réaliser un service particulier sur le réseau ?

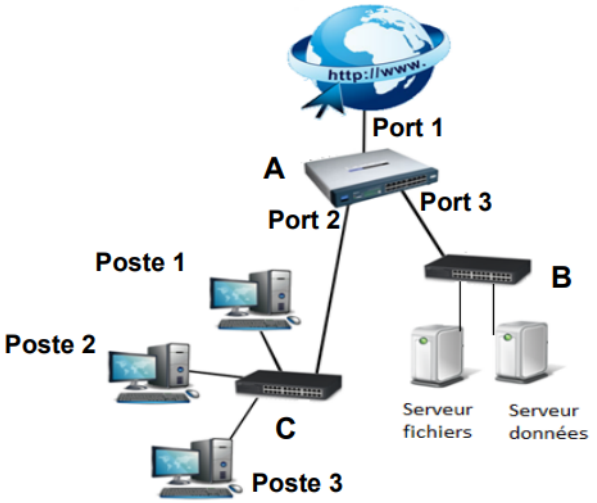
Architecture	Protocole	Paquet

2. On considère le schéma réseau de l'entreprise Lambda :

Parmi les quatre propositions suivantes (Routeur, Commutateur (Switch), Contrôleur WIFI et Serveur), préciser celle qui correspond à :

a) L'élément A

b) L'élément B



3. En reprenant le schéma de la question **2.** et le tableau d'adressage du réseau de l'entreprise Lambda, recopier sur votre copie et compléter la ligne du tableau du poste 3 :

Matériel	Adresse IP	Masque	Passerelle
Routeur Port 1	172.16.0.1	255.255.0.0	
Routeur Port 2	192.168.11.1	255.255.255.0	
Routeur Port 3	192.168.11.254	255.255.255.0	
Serveur fichiers	192.168.11.10	255.255.255.0	192.168.11.1
Serveur données	192.168.11.11	255.255.255.0	192.168.11.1
Poste 1	192.168.11.20	255.255.255.0	192.168.11.1
Poste 2	192.168.11.21	255.255.255.0	192.168.11.1
Poste 3			

Partie B : Routage réseaux

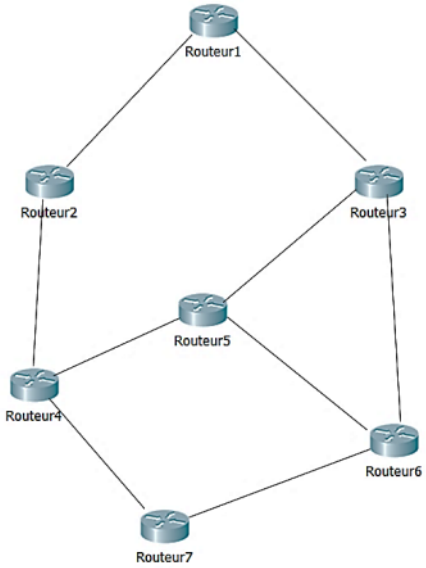
L'extrait de la table de routage d'un routeur R1 est donné ci-dessous :

Réseau IP destination		Passerelle	Interface Machine ou Port	Métrique (distance)
Réseau IP	Masque			
10.0.0.0	255.0.0.0	10.0.0.1	10.0.0.1	0
172.16.0.0	255.255.0.0	172.16.0.1	172.16.0.1	0
192.168.0.0	255.255.255.0	192.168.0.1	192.168.0.1	0
11.0.0.0	255.0.0.0	192.168.0.2	192.168.0.1	1
172.17.0.0	255.255.0.0	192.168.0.2	192.168.0.1	1
172.18.0.0	255.255.0.0	172.15.0.2	172.15.0.1	1
192.168.1.0	255.255.255.0	192.168.0.2	192.168.0.1	1
192.168.2.0	255.255.255.0	172.15.0.2	172.15.0.1	1

1. Indiquer sur votre copie les adresses IP du(des) réseau(x) directement connectés à ce routeur.
2. Indiquer sur votre copie l'interface utilisée pour transférer les paquets contenant les adresses IP destination suivantes :

Adresse IP destination	Interface Machine ou Port
192.168.1.55	
172.18.10.10	

3. On considère un réseau selon le schéma ci-contre.



Recopier sur votre copie et compléter la table de routage simplifiée du Routeur1 (R1) (ci-dessous) en prenant comme métrique le nombre de routeurs à « traverser » avant d'atteindre le réseau de la machine destinataire.

Table de routage simplifiée du Routeur1		
Routeur destination	***Métrique***	***Route***
R2 : Routeur2	*0*	*R1 – R2*
…		

Exercice 2 : fichiers CSV

Principaux thèmes abordés : Traitement de données en tables (CSV) et langages et programmation (spécification).

Afin d'améliorer l'ergonomie d'un logiciel de traitement des inscriptions dans une université, un programmeur souhaite exploiter l'intelligence artificielle pour renseigner certains champs par auto-complétion. Il s'intéresse au descripteur « genre » (masculin, féminin ou indéterminé). Pour cela il propose d'exploiter les dernières lettres du prénom pour proposer automatiquement le genre.

Pour vérifier son hypothèse, il récupère un fichier CSV associant plus de 60 000 prénoms du monde entier au genre de la personne portant ce prénom. En utilisant seulement la dernière lettre, le taux de réussite de sa démarche est de 72,9% avec la fonction définie ci-dessous :

```
def genre(prenom):
    liste_M = ['f', 'd', 'c', 'b', 'o', 'n', 'm', 'l', 'k',
               'j', 'é', 'h', 'w', 'v', 'u', 't', 's', 'r',
               'q', 'p', 'i', 'þ', 'z', 'x', 'ç', 'ö', 'ã',
               'â', 'ï', 'g']
    liste_F = ['e', 'a', 'ä', 'ü', 'y', 'ë']

    if prenom[len(prenom)-1].lower() in liste_M :
        return "M"
    elif prenom[len(prenom)-1].lower() in liste_F :
        return "F"
    else :
        return "I"


# Pour rappel, C.lower() convertit le caractère C en minuscule.
```

1. Appropriation

a) Expliquer ce qu'est un fichier CSV.

b) Donner le type de l'argument **`prenom`** de la fonction **`genre`**, et le type de la réponse renvoyée.

2. Développement

Pour effectuer son étude sur les prénoms à partir du fichier CSV, le programmeur souhaite utiliser la bibliothèque csv.

a) La bibliothèque csv est un module natif du moteur python.

Donner, dans ce cas, l'instruction d'importation.

b) Au cours du développement de son projet, le programmeur souhaite insérer une assertion sur l'argument donné à la fonction.

Proposer une assertion sur le type de l'argument qui corrige une erreur lorsque le type ne correspond pas à celui attendu.

(Voir le document à la fin de l'exercice).

c) Avant le déploiement de sa solution, le programmeur décide de rendre sa fonction plus robuste.

Pour cela il veut remplacer l'assertion proposée dans la question **2.b)** par une gestion de l'argument pour éviter toutes erreurs empêchant la poursuite du programme.

Proposer alors une ou plusieurs instructions en Python utilisant l'argument afin de s'assurer que la fonction se termine quel que soit le type de l'argument.

3. Améliorations

En prenant en compte les deux dernières lettres du prénom, il parvient à augmenter son taux de réussite à 74,4%. Pour cela, son étude du fichier CSV lui permet de créer deux listes : **`liste_M2`** pour les terminaisons de deux lettres associées aux prénoms masculins et **`liste_F2`** pour les prénoms féminins.
Sur votre copie, recopier et modifier la structure conditionnelle (lignes 8 à 13) de la fonction **`genre`** afin de prendre en compte les terminaisons de deux lettres des listes **`liste_M2`** et **`liste_F2`**.

Document :

La fonction **isinstance()** prend deux paramètres:

- **object** – objet à vérifier
- **type** – classe, type ou tuple de classes et de types

Valeur de retour:

La fonction **isinstance()** renvoie True si l'objet est une instance ou une sous-classe d'une classe, ou tout élément du tuple, sinon renvoie False.

Exemple 1:

```
1.  nbrs = [1, 2, 3, 4]
2.
3.  res = isinstance(nbrs, list)
4.  print(nbrs,'instance de List?', res)
5.
6.  res = isinstance(nbrs, dict)
7.  print(nbrs,'instance de Dictionary?', res)
8.
9.  res = isinstance(nbrs, (dict, list))
10. print(nbrs,'instance de List ou Dictionary?', res)
```

Sortie:

```
[1, 2, 3, 4] instance de List? True
[1, 2, 3, 4] instance de Dictionary? False
[1, 2, 3, 4] instance de List ou Dictionary? True
```

Exercice 3 : SGBD

Gestion d'un club de handball

Thèmes abordés : bases de données

Vous trouverez, en annexe 1, des rappels sur le langage SQL

Un club de handball souhaite regrouper efficacement toutes ses informations. Il utilise pour cela des bases de données relationnelles afin d'avoir accès aux informations classiques sur les licenciés du club ainsi que sur les matchs du championnat. Le langage SQL a été retenu.

On suppose dans l'exercice que tous les joueurs d'une équipe jouent à chaque match de l'équipe.

La structure de la base de données est composée des deux tables (ou relations) suivantes:

Table licencies	
Attributs	**Types**
id_licencie	INT
prenom	VARCHAR
nom	VARCHAR
annee_naissance	INT
equipe	VARCHAR

Table matchs	
Attributs	**Types**
id_matchs	INT
equipe	VARCHAR
adversaire	VARCHAR
lieu	VARCHAR
date	DATE

Ci-dessous un exemple de ce que l'on peut trouver dans la base de données :

Exemple **non exhaustif** d'entrées de la table `licencies`

id_licencie	prenom	nom	annee_naissance	equipe
63	Jean-Pierre	Masclef	1965	Vétérans
102	Eva	Cujon	1992	Femmes 1
125	Emile	Alinio	2000	Hommes 2
247	Ulysse	Trentain	2008	-12 ans

Exemple **non exhaustif** d'entrées de la table `matchs`

id_match	equipe	adversaire	lieu	date
746	-16 ans	PHC	Domicile	2021-06-19
780	Vétérans	PHC	Exterieur	2021-06-26
936	Hommes 3	LSC	Exterieur	2021-06-20
1032	-19 ans	LOH	Exterieur	2021-05-22
1485	Femmes 2	CHM	Domicile	2021-05-02
1512	Vétérans	ATC	Domicile	2021-04-12

1.

1.a. L'attribut `nom` de la table `licencies` pourrait-il servir de clé primaire ? **Justifier**.

1.b. **Citer** un autre attribut de cette table qui pourrait servir de clé primaire.

2.

2.a. **Expliquer** ce que renvoie la requête SQL suivante :

```
SELECT prenom,nom FROM licencies WHERE equipe ="-12ans"
```

2.b. **Que renvoie** la requête précédente si `prenom,nom` est remplacé par une étoile (*) ?

2.c. **Ecrire** la requête qui permet l'affichage des dates de tous les matchs joués à domicile de l'équipe *Vétérans*.

3. **Ecrire** la requête qui permet d'inscrire dans la table `licencies`, *Jean Lavenu* né en *2001* de l'équipe *Hommes 2* et qui aura comme numéro de licence *287* dans ce club.

4. On souhaite mettre à jour les données de la table `licencies` du joueur *Joseph Cuviller*, déjà inscrit. Il était en équipe *Hommes 2* et il est maintenant en équipe *Vétérans*. Afin de modifier la table dans ce sens, **proposer** la requête adéquate.

5. Pour obtenir le nom de tous les licenciés qui jouent contre le LSC le 19 juin 2021, **recopier et compléter** la requête suivante :

```
SELECT nom FROM licencies
JOIN Matchs ON licencies.equipe = matchs.equipe
WHERE …… ;
```

Exercice 4 : POO, piles

Principaux thèmes abordés : structure de données (programmation objet) et langages et programmation (spécification).

Une entreprise fabrique des yaourts qui peuvent être soit nature (sans arôme), soit aromatisés (fraise, abricot ou vanille).

Pour pouvoir traiter informatiquement les spécificités de ce produit, on va donc créer une classe *Yaourt* qui possèdera un certain nombre d'attributs :

- Son genre : nature ou aromatisé
- Son arôme : fraise, abricot, vanille ou aucun
- Sa date de durabilité minimale (DDM) exprimée par un entier compris entre 1 et 365 (on ne gère pas les années bissextiles). Par exemple, si la DDM est égale à 15, la date de durabilité minimale est le 15 janvier.

On va créer également des méthodes permettant d'interagir avec l'objet `Yaourt` pour attribuer un arôme ou récupérer un genre par exemple. On peut représenter cette classe par le tableau de spécifications ci-dessous :

Yaourt	
ATTRIBUTS: - genre - arome - duree	**METHODES**: - Construire(arome,duree) - Obtenir_Arome() - Obtenir_Genre() - Obtenir_Duree() - Attribuer_Arome(arome) - Attribuer_Duree(duree) - Attribuer_Genre(arome)

1. La classe `Yaourt` est déclarée en Python à l'aide du mot-clé class :

```
class Yaourt:
    """ Classe définissant un yaourt caractérisé par :
    - son arome
    - son genre
    - sa durée de durabilité minimale"""
```

L'annexe 1 de l'exercice 4 donne le code existant et l'endroit des codes à produire dans les questions suivantes.

a) Quelles sont les assertions à prévoir pour vérifier que l'arôme et la durée correspondent bien à des valeurs acceptables. Il faudra aussi expliciter les commentaires qui seront renvoyés.

Pour rappel :

- L'arôme doit prendre comme valeur 'fraise', 'abricot', 'vanille' ou 'aucun'.
- Sa date de durabilité minimale (DDM) est une valeur positive.

b) Pour créer un yaourt, on exécutera la commande suivante :

```
Mon_Yaourt = Yaourt('fraise',24)
```

Quelle valeur sera affectée à l'attribut genre associé à Mon_Yaourt ?

c) Écrire en python une fonction `GetArome(self)`, renvoyant l'arôme du yaourt créé.

2. On appelle mutateur une méthode permettant de modifier un ou plusieurs attributs d'un objet.

Sur votre copie, écrire en Python le mutateur `SetArome(self,arome)` permettant de modifier l'arôme du yaourt.

On veillera à garder une cohérence entre l'arôme et le genre.

3. On veut créer une pile contenant le stock de yaourts. Pour cela il faut tout d'abord créer une pile vide :

```
def creer_pile():
    pile = [ ]
    return pile
```

a) Créer une fonction `empiler(p,Yaourt)` qui renvoie la pile `p` après avoir ajouté un objet de type `Yaourt` à la fin.

b) Créer une fonction `depiler(p)` qui renvoie l'objet à dépiler.

c) Créer une fonction `estVide(p)` qui renvoie `True` si la pile est vide et `False` sinon.

d) Qu'affiche le bloc de commandes suivantes ci-dessous ?

```
mon_Yaourt1 = Yaourt('aucun',18)
mon_Yaourt2 = Yaourt('fraise',24)
ma_pile = creer_pile()
empiler(ma_pile, mon_Yaourt1)
empiler(ma_pile, mon_Yaourt 2)
print(depiler(ma_pile).GetDuree())
print(estVide(ma_pile))
```

Exercice 5 : Pile, tri

Thème abordé : structures de données : les piles

On cherche à obtenir un mélange d'une liste comportant un nombre **pair** d'éléments.
Dans cet exercice, on notera N le nombre d'éléments de la liste à mélanger.

La méthode de mélange utilisée dans cette partie est inspirée d'un mélange de jeux de cartes :

- On sépare la liste en deux piles :
  - ⇨ à gauche, la première pile contient les N/2 premiers éléments de la liste ;
  - ⇨ à droite, la deuxième pile contient les N/2 derniers éléments de la liste.
- On crée une liste vide.
- On prend alors le sommet de la pile de gauche et on le met en début de liste.
- On prend ensuite le sommet de la pile de droite que l'on ajoute à la liste et ainsi de suite jusqu'à ce que les piles soient vides.

Par exemple, si on applique cette méthode de mélange à la liste `['V','D','R','3','7','10']`, on obtient pour le partage de la liste en 2 piles :

Pile gauche
'R'
'D'
'V'

Pile droite
'10'
'7'
'3'

La nouvelle liste à la fin du mélange sera donc `['R','10','D','7','V','3']`.

1. Que devient la liste `['7','8','9','10','V','D','R','A']` si on lui applique cette méthode de mélange ?

On considère que l'on dispose de la structure de données de type pile, munie des seules instructions suivantes :

`p = Pile()` : crée une pile vide nommée `p`
`p.est_vide()` : renvoie Vrai si la liste est vide, Faux sinon
`p.empiler(e)` : ajoute l'élément `e` dans la pile
`e = p.depiler()` : retire le dernier élément ajouté dans la pile et le retourne (et l'affecte à la variable `e`)
`p2 = p.copier()` : renvoie une copie de la pile `p` sans modifier la pile `p` et l'affecte à une nouvelle pile `p2`

2. Recopier et compléter le code de la fonction suivante qui transforme une liste en pile.

```
def liste_vers_pile(L):
    '''prend en paramètre une liste et renvoie une
pile'''
    N = len(L)
    p_temp = Pile()
    for i in range(N):
        ……..
    return …….
```

3. On considère la fonction suivante qui partage une liste en deux piles. Lors de sa mise au point et pour aider au débuggage, des appels à la fonction `affichage_pile` ont été insérés. La fonction `affichage_pile(p)` affiche la pile `p` à l'écran verticalement sous la forme suivante :

dernier élément empilé
...
...
premier élément empilé

```
def partage(L):
    N = len(L)
    p_gauche = Pile()
    p_droite = Pile()
    for i in range(N//2):
        p_gauche.empiler(L[i])
    for i in range(N//2,N):
        p_droite.empiler(L[i])
    affichage_pile(p_gauche)
    affichage_pile(p_droite)
    return p_gauche, p_droite
```

Quels affichages obtient-on à l'écran lors de l'exécution de l'instruction : `partage([1,2,3,4,5,6])` ?

4.

4.a Dans un cas général et en vous appuyant sur une séquence de schémas, **expliquer** en quelques lignes comment fusionner deux piles `p_gauche` et `p_droite` pour former une liste `L` en alternant un à un les éléments de la pile `p_gauche` et de la pile `p_droite`.

4.b. **Écrire** une fonction `fusion(p1,p2)` qui renvoie une liste construite à partir des deux piles `p1` et `p2`.

5. Compléter la dernière ligne du code de la fonction `affichage_pile` pour qu'elle fonctionne de manière récursive.

```
def affichage_pile(p):
  p_temp = p.copier()
  if p_temp.est_vide():
    print('____')
  else:
    elt = p_temp.depiler()
    print('| ', elt, ' |'))
    ...                          # ligne à compléter
```

Remarque : il y a une parenthèse en trop dans le second print !

Exercice 3 – Annexe 1

(à ne pas rendre avec la copie)

- **Types de données**

CHAR(t) VARCHAR(t) TEXT	Texte fixe de t caractères. Texte de t caractères variables. Texte de 65 535 caractères max.
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 DATETIME	Date format AAAA-MM-JJ Date et heure format AAAA-MM-JJHH:MI:SS

- **Quelques exemples de syntaxe SQL :**
  - Insérer des enregistrements :

```
INSERT INTO Table (attribut1, attribut2) VALUES(valeur1 , valeur2)

Exemple :
INSERT INTO client(id_client,nom,prenom) VALUES (112,'Dehnou','Danièle')
```

  - 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
```

  - Effectuer une jointure :

```
SELECT attributs FROM TableA JOIN TableB ON TableA.cle1=TableB.cle2 WHERE
Selecteur
```

Exercice 4 - Annexe 1

```
class Yaourt:

    def __init__(self,arome,duree):
    # **** Assertions : question 1.a. à compléter sur votre copie
        self.__arome = arome
        self.__duree = duree
        if arome == 'aucun':
            self.__genre = 'nature'
        else:
            self.__genre = 'aromatise'


    # **** Méthode GetArome(self) question 1.c. à compléter sur
    votre copie

    def GetDuree(self):
        return(self.__duree)

    def GetGenre(self):
        return ( self.__genre)

    def SetDuree(self,duree):
    # **** Mutateur de durée
        if duree > 0 :
            self.__duree = duree

    # **** Mutateur d'arôme SetArome(self,arome) - question 2. à
compléter sur votre copie

    def __SetGenre(self,arome):
        if arome == 'aucun':
            self.__genre = 'nature'
        else:
            self.__genre = 'aromatise'
```