Correction

NSI - 2021 Métropole Septembre (21-NSIJ2ME3)

Exercice 1 - Réseau

Partie A - Réseau

1. Protocole
2.
   a. L'élément A est un routeur.
   b. Les éléments B et C sont des commutateurs (switch).
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	192.168.11.22	255.255.255.0	192.168.11.1

Partie B - Routage réseaux

1. On repère ici que les routeurs directement connectés ont un coût de 0.
   Les trois réseaux directement connectés ont pour IP : **10.**0.0.0 **172.16.**0.0 et **192.168.0.**0
2.

Adresse IP destination	Interface Machine ou Port
192.168.1.55	192.168.0.1
172.18.10.10	172.15.0.1

3.

Table de routage simplifiée du Routeur1 (R1)		
Routeur destination	**Métrique**	**Route**
R2 : Routeur2	0	R1 – R2
R3 : Routeur3	0	R1 – R3
R4 : Routeur4	1	R1 – R2 – R4
R5 : Routeur5	1	R1 – R3 – R5
R6 : Routeur6	1	R1 – R3 – R6
R7 : Routeur7	2	R1 – R2 – R4 – R7 ou R1 – R3 – R6 – R7

Correction

NSI - 2021 Métropole Septembre (21-NSIJ2ME3)

Exercice 2 - Les aventuriers du rail

1. La liste n'est pas valide car il n'y a pas de liaison directe entre Luchon et Muret (via St-Gaudens).
2. a.

```
liaisonsJoueur2 = [["Toulouse","Castres"],
                   ["Toulouse","Castelnaudary"],
                   ["Castres","Mazamet"],
                   ["Castelnaudary","Carcasonne"]]
```

2. b.

```
DictJoueur2 = {"Toulouse":["Castres","Castelnaudary"],
               "Castres":["Toulouse","Mazamet"],
               "Castelnaudary":["Toulouse","Carcasonne"],
               "Mazamet":["Castres"],
               "Carcasonne":["Castelnaudary"]}
```

3. a.

```
assert len(listeLiaisons)
```

3. b. Le dictionnaire obtenu est :

```
{"Toulouse": ["Muret", "Montauban"],
 "Gaillac": ["St Sulpice"],
 "Muret": ["Pamiers"]}
```

La fonction ne répond que partiellement car elle ne prend en compte que la première ville du couple [villeA, villeB]. Pour être complète, la réponse devrait être :

```
{"Toulouse": ["Muret", "Montauban"],
 "Muret": ["Toulouse", "Pamiers"] ,
 "Montauban" :[ "Toulouse"],
 "Gaillac": ["St Sulpice"],
 "St Sulpice" :["Gaillac"],
 "Pamiers" :["Muret"]}
```

3. c.

Il faut ajouter les lignes suivants entre les lignes 15 et 16 :

```
        if not villeB in Dict.keys():
            Dict[villeB] = [villeA]
        else:
            destinationsB = Dict[villeB]
            if not villeA in destinationsB:
                destinationsB.append(villeA)
```

Correction

NSI - 2021 Métropole Septembre (21-NSIJ2ME3)

Exercice 3 - BD Tableau Mendeleïev

1. SQL (Structured Query Language)

2. a. Attributs et leurs types de domaine

Table Atomes	
Attribut	**Type**
Z	INTEGER ou INT
nom	VARCHAR(15)
Sym	CHAR(2)
L	INTEGER
C	INTEGER
masse_atom	FLOAT

Table Valence	
Attribut	**Type**
Col	INTEGER
couche	CHAR(1)

2. b. Pour la table **Atomes**, les attributs pouvant servir de clé primaire sont **Z**, **nom** et **Sym**. L'attribut **C** peut avoir le rôle de clé étrangère en pointant vers l'attribut **Col** de la table **Valence**.

2. c. Schémas relationnels

Atomes(Z, nom, Sym, L, #C, masse_atom)
Valence(Col, couche)

3. a. aluminium, argon, chlore, magnesium, sodium, phosphore, soufre, silicium

3. b. 1, 2, 13, 14, 15, 16, 17, 18, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12

4. a.
```
SELECT nom, masse_atom FROM Atomes
```

4. b.
```
SELECT A.Sym FROM Atomes AS A JOIN Valence AS V ON A.C = V.Col
    WHERE V.couche = 's'
```

5.
```
    UPDATE Atomes SET masse_atom = 39.948 WHERE Z = 18
ou UPDATE Atomes SET masse_atom = 39.948 WHERE nom = 'argon'
ou UPDATE Atomes SET masse_atom = 39.948 WHERE Sym = 'Ar'
```

Correction

NSI - 2021 Métropole Septembre (21-NSIJ2ME3)

Exercice 4 - Objet Yaourt

1. a.

```
assert arome in ['fraise', 'abricot', 'vanille', 'aucun'], "Arôme inconnu"
assert 0 < duree < 366, "Date de durabilité inacceptable"
```

Remarque : Ici, on ne peut pas mettre `self.__arome` ou `self.__duree` car ces attributs ne sont pas encore définis à l'endroit où il est demandé d'insérer les assertions.

1. b. La valeur de l'attribut genre est `'aromatise'`.
1. c.

```
def GetArome(self):
    return self.__arome
```

2.

Solution 1	Solution 2

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

3. a.

```
def empiler(p, Yaourt):
    p.append(Yaourt)
    return p
```

3. b.

```
def depiler(p):
    return p.pop()
```

Remarque : Par défaut, la méthode pop prend pour argument -1, ce qui revient à extraire le dernier élément de la liste.

3. c.

Solution 1	Solution 2	Solution 3

`def estVide(p):`
`  if p == []:`
`  return True`
`  else:`
`  return False`
	`def estVide(p):`
	`  return len(p) == 0`
		`def estVide(p):`
		`  return not p`

3. d. Le bloc de commande affiche :

```
24
False
```

Correction

NSI - 2021 Métropole Septembre (21-NSIJ2ME3)

Exercice 5 - Genre d'un prénom

1. a. Le CSV pour Comma-Separated Values est un format texte représentant ligne par ligne des enregistrements de données ; chaque donnée est séparée par une virgule.

1. b. L'argument de la fonction et ce qu'elle renvoie sont tout deux de type chaîne de caractère (str).

2. a. Pour importer le module csv :

```
import csv   ou   from csv import *
```

2. b.

```
assert type(prenom) is str   ou   assert type(prenom) == type('')
```

2. c.

Solution 1	Solution 2
`if type(prenom) is not str :` `    return None`	`prenom = str(prenom)`

3. Ces lignes doivent remplacer les lignes 8 à 13 du code original.

```
let1 = prenom[-1].lower()
let2 = prenom[-2:].lower()
if let1 in liste_M or let2 in liste_M2 :
    return "M"
elif let1 in liste_F or let2 in liste_F2 :
    return "F"
else :
    return "I"
```