Éléments de correction sujet 05 (2023)

Exercice 1

1.
   a. id_equipe doit être unique (clé primaire). La requête essaye d'ajouter une équipe avec un id_equipe égal à 11 alors qu'il existe déjà un id_équipe égal à 11 dans la table.
   b. un numéro de téléphone est composé de chiffres et d'espaces (chaine de caractères)
   c. On obtient le résultat suivant :

Lyon	451 cours d'Emile Zola, 69100 Villeurbanne	04 05 06 07 08

   d.
   On obtient le nombre d'équipe de la table, c'est à dire 12
   e.
```
SELECT nom
FROM Equipe
ORDER BY nom
```
   f.
```
UPDATE Equipe
SET nom = 'Tarbes'
WHERE id_equipe = 4
```
2.
   a.
   l'attribut id_equipe de la table Joueuse permet de lier la table Joueuse à la table Equipe, c'est donc une clé étrangère.
   b. La suppression d'une équipe de la table Equipe doit obligatoirement s'accompagner d'une modification de la table joueuse afin qu'aucun élément de l'attribut id_equipe de la table Joueuse ne pointe vers une équipe qui n'existe plus.
   c.
```
SELECT Joueuse.nom, prenom
FROM Joueuse
JOIN Equipe ON Equipe.id_equipe = Joueuse.id_equipe
WHERE Equipe.nom = 'Angers'
ORDER BY Joueuse.nom
```
3.
   a.
   Match (id_match : INT, date : DATE, #id_eq_dom : INT, #id_eq_dep : INT, score_eq_dom : INT, score_eq_dep : INT)
   Les clés étrangères font référence à l'attribut id_equipe de la table Equipe
   b.
```
INSERT INTO Match
VALUES
(10, 23/10/2021, 3, 6, 73, 78)
```
4.
   a.
   Stat (id_stat : INT, #id_joueuse : INT, #id_match : INT, points : INT, rebonds : INT, passes_dec : INT)

b.

```
SELECT Equipe.nom, Joueuse.nom, prenom, points, rebonds,
passes_dec
FROM Stat
JOIN Joueuse ON Joueuse.id_joueuse = Stat.id_joueuse
JOIN Equipe ON Equipe.id_equipe = Joueuse.id_equipe
WHERE Stat.id_match = 53
```

Exercice 2

1.
   a.
   11, 20, 32, 11, **20, 32, 11, 32, 11**
   b.
   11, 11, 20, 20, 32, 32, 11, 11, 32
2.
   a.

```
liste_attente = [Processus(11, 4), Processus(20, 2),
Processus(32, 3)]
```

   b.

```
def execute_un_cycle(self):
    self.reste_a_faire = self.reste_a_faire - 1

def change_etat(self, nouvel_etat):
    self.etat = nouvel_etat

def est_termine(self):
    return self.reste_a_faire <=  0
```

   c.

```
def tourniquet(liste_attente, quantum):
    ordre_execution = []
    while liste_attente != []:
        processus = liste_attente.pop(0)
        processus.change_etat("En cours d'exécution")
        compteur_tourniquet = 0
        while compteur_tourniquet < quantum and not
processus.est_termine():
            ordre_execution.append(processus.pid)
            processus.execute_un_cycle()
            compteur_tourniquet = compteur_tourniquet + 1
        if not processus.est_termine():
            processus.change_etat("Suspendu")
            liste_attente.append(processus)
        else:
            processus.change_etat("Terminé")
    return ordre_execution
```

Exercice 3

1.
   a. Cette ligne permet d'importer la fonction sqrt de la bibliothèque math
   b. Cette expression renvoie False à cause du problème d'arrondi des flottants (impossible de coder 0.1 en mémoire avec exactitude)
   c.
   point_A est un tulpe et la ligne `point_A[0] = 2` a pour but de modifier ce tuple, alors qu'un tuple n'est pas modifiable.
2.
   a.
```
from math import sqrt
class Segment:
    def __init__(self, point1, point2):
        self.p1 = point1
        self.p2 = point2
        self.longueur = sqrt((self.p1[0]-self.p2[0])**2 +
(self.p1[1]-self.p2[1])**2)
```
   b.
```
def liste_segments(liste_points):
    n = len(liste_points)
    segments = []
    for i in range(n-1):
        for j in range(i+1, n):
            seg = Segment(liste_points[i], liste_points[j])
            segments.append(seg)
    return segments
```
   c.
   (n - 1) + (n - 2) + (n - 3) + … + 1
   d.
   La complexité est en $O(n^2)$ (boucles imbriquées)
3.
   a.
```
def plus_court_segment(tab_seg):
    if len(tab_seg) == 1 :
        return tab_seg[0]
    else :
        seg_g = plus_court_segment(moitie_gauche(tab_seg))
        seg_d = plus_court_segment(moitie_droite(tab_seg))
        if seg_g.longueur < seg_d.longueur :
            return seg_g
        else :
            return seg_d
```
4.
   a.
```
nuage_points = liste_segments([(3, 4), (2, 3), (-3, -1)])
```
   b.
```
seg = plus_court_segment(nuage_points)
print("point 1 : ("+str(seg.p1[0])+","+str(seg.p1[1])+")")
print("point 2 : ("+str(seg.p2[0])+","+str(seg.p2[1])+")")
```