Correction du DS 0

Informatique pour tous, deuxième année

Julien Reichert

Exercice 1

```
SELECT DISTINCT(Identite) FROM Resolution JOIN Concurrent ON Concur = Id_concurrent
```

Exercice 1bis

```
def no_quitters():
    identifiants = {}
    for reso_pas_reso in resolution:
        identifiants[reso_pas_reso["Concur"]] = 42
    reponse = []
    for conc in concurrent:
        if conc["Id_concurrent"] in identifiants:
        reponse.append(conc["Identite"])
    return reponse
```

Exercice 2

```
SELECT Concur, Exo FROM Resolution JOIN Exercice ON Exo = Id_exercice
WHERE Date_resolution > Date + 86400
```

Exercice 2bis

```
def les_dates():
    reponse = {}
    for exo in exercice:
        reponse[exo["Id_exercice"]] = exo["Date"]
    return reponse

def timeout():
    dates = les_dates()
    reponse = []
    for reso_pas_reso in resolution:
        if reso_pas_reso["Date_resolution"] > dates[reso_pas_reso["Exo"]] + 86400:
        reponse.append(reso_pas_reso)
    return reponse
```

Exercice 3

```
SELECT Classe, COUNT(*) FROM Concurrent JOIN Resolution ON Id_concurrent = Concur
WHERE (Exo, Date_resolution) IN (SELECT Exo, MIN(Date_resolution) FROM Resolution GROUP BY Exo)
GROUP BY Classe
```

Exercice 4

```
def la_date_et_le_delai(id_exo):
    for exo in exercice:
        if exo["Id_exercice"] == id_exo:
            return exo["Date"], exo["Delai"]
    raise ValueError("Introuvable")

def points_arbitre(id_exo):
    date, _ = la_date_et_le_delai(id_exo)
    points = {}
    premier = None
    min_date = None
    for reso_pas_reso in resolution:
        if reso_pas_reso["Exo"] == id_exo:
            if min_date == None or reso_pas_reso["Date_resolution"] < min_date:
                min_date = reso_pas_reso["Date_resolution"]
                premier = reso_pas_reso["Concur"]
            if reso_pas_reso["Date_resolution"] <= date + 86400:
                points[reso_pas_reso["Concur"]] = 10
    if premier != None:
        if premier in points:
            points[premier] += 10
        else:
            points[premier] = 10
    return points
```

Exercice 5

La fonction de cet exercice peut tout à fait capturer la fonction précédente, en témoigne le copier-coller. En l'occurrence la fonction précédente n'a été demandée que pour amener à écrire celle-ci plus facilement. Il ne s'agit donc pas d'un copier-coller mais d'une ébauche améliorée. En particulier, dans l'exercice 6, on n'utilisera que la fonction points avec un identifiant 0 pour le posteur, dont on admet qu'il n'est pas attribué (au pire, une vérification de la liste des identifiants permet de récupérer une valeur non attribuée, par exemple le minimum moins un).

En prévision de l'exercice 6, il aurait d'ailleurs été envisageable que le tableau des scores actuel soit un argument de la fonction et qu'il soit mis à jour, pour ne pas avoir à faire une boucle et une addition (fort heureusement, l'accès à un élément d'un dictionnaire est en temps assimilé à constant). C'est un bricolage imaginable, mais il faut faire attention à ne pas oublier de remplacer un = par un += après vérification de l'existence d'une valeur (mieux : initialiser toutes les valeurs à 0 pour les éléments de `no_quitters()`). À ce stade, le tableau des scores peut même être une variable globale.

On notera qu'il y avait des simplifications possibles en filtrant et triant les éléments de `resolution`, mais la complexité est supérieure, comme on le verra cette année en cours. Quant à espérer que les éléments sont triés, c'est certes très probable mais il faut jouer le jeu...

```
def points(id_exo, posteur):
    date, delai = la_date_et_le_delai(id_exo)
    points = {posteur : len(no_quitters())-1}
    premier = None
    min_date = None
    for reso_pas_reso in resolution:
        if reso_pas_reso["Exo"] == id_exo:
            if min_date == None or reso_pas_reso["Date_resolution"] < min_date:
                min_date = reso_pas_reso["Date_resolution"]
                premier = reso_pas_reso["Concur"]
            if reso_pas_reso["Date_resolution"] <= date + 86400:
                points[reso_pas_reso["Concur"]] = 10
                points[posteur] -= 1
    if premier != None:
        if premier in points:
            points[premier] += 10
        else:
            points[premier] = 10
        if min_date > date + delai * 3600:
            points[posteur] += 10
    return points, premier # ajout d'une valeur de retour en prévision de la suite
```

Exercice 6

```
def liste_exercices():
    reponse = []
    for exo in exercice:
        reponse.append(exo["Id_exercice"])
    return sorted(reponse) # tri au cas où,
# mais ici un algorithme de vérification en amont
# ou un tri rapide sur les données presque triées serait bienvenu

def scores():
    posteur = 0 # arbitre
    points_totaux = {}
    for un_exo in liste_exercices():
        pts, posteur = points(exo, posteur)
        for marqueur in pts:
            if marqueur != 0:
                if marqueur in points_totaux:
                    points_totaux[marqueur] += pts[marqueur]
                else:
                    points_totaux[marqueur] = pts(marqueur]
        if posteur == None:
            posteur = 0
    return points_totaux

def scores_jolis():
    points_totaux = scores()
    reponse = {}
    for concur in concurrent:
        if concur["Id_concurrent"] in points_totaux:
            reponse[concur["Identite"]] = points_totaux[concur["Id_concurrent"]]
    return reponse # et il n'y a plus qu'à trier pour constater que Willy a gagné !
```