Corrige DS - Moyennes

*Note finale = 0.4 * Note de Cours + 0.6 * Note de Problème.*

1 Exercices

c.f cours pour le corrigé.

1. Questions de cours coefficients 2
2. Questions algo coefficients 3

2 Problèmes

2.1 Moyennes

Question 2.1.

Coefficient : 1

Attendus :
- Définition d'une fonction (3/9)
  - Syntaxe (1/9)
  - Arguments (2/9)
- Retour du résultat (2/9)
- Expression arithmétique et comparaison(3/9)
- Accès aux coordonnées (1/9)

```
def plus_faible(n1,n2) :
    return n1[0] * n1[1] < n2[0] * n2[1]
```

Question 2.2.

Coefficient : 3

Attendus :
- Faire un parcours d'éléments de la liste (3/9)
- Faire une comparaison en utilisant `plus_faible` (2/9)
- Initialisation de la recherche (2/9)
- Algorithme de recherche de maximum correct (2/9)

```python
def meilleur(l) :
    nmax = l[0]
    for n in l[1:] :
        if plus_faible(nmax,n) :
            nmax = n
    return nmax
```

Question 2.3.

Coefficient : 3

Attendus :
- Faire un parcours d'éléments de la liste (2/9)
- Initialisation des sommes (1/9)
- Calcul de la somme des coefficients (2/9)
- Calcul de la somme des notes (2/9)
- Retour de la division (2/9)

```python
def moyenne_ponderee_couples(l) :
    s = 0
    s_coeff = 0
    for nc in l :
        s = s + nc[0]*nc[1]
        s_coeff = s_coeff + nc[1]
    return s / s_coeff
```

Question 2.4.

Coefficient : 4

Attendus :
- Faire un parcours d'indice en parallèle (5/9)
  - Parcours d'indice correct (1/9)
  - Récupération correcte des coefficients et des notes (4/9)
- Initialisation (1/9)
- Calcul de la somme des coefficients (1/9)
- Calcul de la somme des notes (1/9)
- Retour de la division (1/9)

```python
def moyenne_ponderee_listes(l,c) :
    s = 0
    s_coeff = 0
    for i in range(len(c)) :
        s = s + l[i]*c[i]
        s_coeff = s_coeff + c[i]
    return s / s_coeff
```

2.2 Distance

Question 2.5. *Distance*

Coefficient : 1

Attendus :
- Définition d'une fonction (3/9)
  - Syntaxe (1/9)
  - Arguments (2/9)
- Retour du résultat (2/9)
- Expression arithmétique avec puissance (3/9)
- Accès aux coordonnées (1/9)

```python
def distance(a,b) :
    return  ( (a[0] - b[0]) ** 2 +  (a[1] - b[1]) ** 2 ) ** (0.5)
```

Question 2.6. *Barycentre*

Coefficient : 3

Attendus :
- Faire un parcours d'éléments de la liste (2/9)
- Initialisation des sommes (1/9)
- Récupération des coordonnées (1/9)
- Calcul de la somme des abscisses (1/9)
- Calcul de la somme des ordonnées (1/9)
- Retour couple correct (3/9)

```python
def barycentre(l) :
    n = len(l)
    mx = 0
    my = 0
    for p in l :
        mx = mx + p[0]
        my = my + p[1]
    return ( mx / n, my / n )
```

Question 2.7. *Plus proches*

Coefficient : 6

Attendus :

- Vérifier tous les couples de points différents possibles par une boucle imbriquée (4/9)
  - Vérifier tous les couples (2/9)
  - Ne pas traiter les couples `(p,p)` (2/9)
- Initialisation (1/9)
- Calcul du minimum correct (1/9)
- Traitement si plusieurs couples de distance minimum (1/9)
- Retour couple correct (1/9)

```python
def plus_proche(l) :
    i_a = 0
    i_b = 1
    dmin = distance(l[i_a],l[i_b])
    for i in range(len(l)) :
        for j in range(i+1,len(l)) :
            d = distance(l[i],l[j])
            if d == dmin and j >= i_b :
                i_a = i
                i_b = j
            if d < dmin :
                i_a = i
                i_b = j
                dmin = d
    return (a,b)
```