AGR. Bases. Exercices

Exercice 1 : Vrai / Faux

	VRAI	FAUX
En Python, une boucle *for* unique se termine toujours après un nombre fini d'étapes.		
S'il faut un temps moyen *t* pour trouver un élément dans une liste non triée de *n* éléments, alors un temps moyen de *2t* pour une liste contenant *2n* éléments.		
Le nombre moyen d'étapes pour trouver une valeur d'une liste de *n* éléments est n/2.		
Le premier algorithme non trivial décrit est celui d'Euclide.		
Pour trouver un élément d'une liste triée, la recherche dichotomique est **toujours** plus rapide qu'une recherche séquentielle.		
Avec une liste triée de *n* éléments, le nombre d'étapes pour trouver une valeur est de l'ordre du nombre de chiffres de l'écriture en binaire du nombre *n*.		

Exercice 2 : QCM

Pour chaque question, une seule réponse est correcte parmi les quatre proposées.

Question 1 : L'algorithme d'Euclide date de combien d'années ?

1. Plus de 2000 ans.
2. Environ 1500 ans.
3. Environ 1000 ans.
4. Environ 500 ans.

Question 2 : Combien de fois la fonction *print* est-elle appelée ?

```
for i in range(5) :
    for j in range(i+1,5) :
        print(i+j)
```

1. 10 fois.
2. 11 fois.
3. 15 fois.
4. 20 fois.

Question 3 : On considère le code qui suit où la variable *n* est un entier naturel non nul. On s'intéresse au coût de cet algorithme en fonction de *n*. Parmi les affirmations suivantes, laquelle est vraie ?

```
n = int(input("Entrer un entier naturel non nul : "))
x = 1
while x < n :
    x = 2*x
```

```
Entrer un entier naturel non nul : 100
```

1. Le coût est semblable à celui d'une recherche dichotomique.
2. Le coût est de l'ordre de *n* (coût linéaire).
3. Le coût est de l'ordre de n^2 (coût quadratique).
4. Il est impossible de quantifier ce coût.

Question 4 : Qu'affiche le programme suivant ?

```
liste1 = ['a','c','d','c','f','g','t','b','c']

compteur = 0
for lettre in liste1 :
    if lettre == 'c' :
        compteur = compteur + 1
print(compteur)
```

1. 'c'
2. 1
3. 2
4. 3

Question 5 : On exécute le script suivant :

```
for i in range(n) :
    for j in range(i) :
        print("coucou")
```

Combien de fois le mot « coucou » est-il écrit ?

1. n^2
2. $(n + 1)^2$
3. 1 + 2 + 3 + … + n-2 + n-1 + n
4. 1 + 2 + 3 + … + n-2 + n-1

Exercice 3 : Moyenne pondérée

On considère deux listes :
notes = [17,12,11,18]
coefficients = [1,2,1,1]
La note « 17 » a un coefficient de « 1 », la note « 12 » a un coefficient de « 2 » etc.

1/ **Calculer** la moyenne des notes (en tenant compte des coefficients) issues des tableaux précédents.
2/ **Ecrire** un programme permettant de calculer la moyenne pondérée avec comme données une liste *notes* et une liste *coefficients*.
On vérifiera l'exactitude du programme à l'aide de l'exemple de la question précédente.
3/ Pour plus de lisibilité, on utilise une liste de *tuples (note, coef).*
On a alors évaluations = [(17, 1) , (12 , 2) , (11 , 1) , (18 , 1)]
Modifier alors le programme précédent en considérant la variable *évaluations*.
4/ Aurait-on pu utiliser un dictionnaire pour la variable *évaluations* ? **Justifier**.

Exercice 4 : Détermination d'une valeur approchée de $\sqrt{3}$.

En s'aidant du cours, écrire un programme donnant une valeur approchée de $\sqrt{3}$.
On veillera à proposer une **fonction *f* adéquate (polynome)**, un **intervalle adapté** et à démontrer que la **fonction *f*** est **strictement monotone**.

Exercice 5 : **Recherche d'extremum**

On considère la fonction *f* définie par $f(x) = -x^2 + 4x - 3$.
A l'aide d'un programme en Python, **déterminer** les coordonnées du maximum de la courbe représentative de la fonction *f* sur l'intervalle [1,3[à 0,0001 près.
Aide : utiliser le programme présenté en cours et le modifier.