TD 4 (programmation) - Fonctions.

Exercice 1 (Variables locales et globales)

Que se passe-t-il lors de l'exécution des programmes suivants ?

```python
def f1(n):
    n += 1

if __name__ == "__main__":
    x = 3
    f1(x)
    print(x)
    print(n)
```

```python
def f2(n):
    n += 1
    return n

if __name__ == "__main__":
    n = 3
    x = f2(n)
    print(x)
    print(n)
```

```python
def f3(n):
    x = 4
    return n + x

if __name__ == "__main__":
    y = f3(3)
    print(y)
    print(x)
```

```python
def f4(n):
    x = 2
    return n + x

if __name__ == "__main__":
    x = 3
    n = f4(4)
    print(x)
    print(n)
```

Exercice 2 (Fonction mystère)

Que calcule la fonction `f` ? Essayez à la main sur `f('abracdabra abracadabra')`.

```python
def f(chaine):
    i = 0
    for c in chaine:
        if c == 'a':
            i = i+1
        if c == ' ':
            break
    return i
```

Exercice 3 (Étoiles)

Écrivez une fonction `etoiles` qui prend un entier `n` et affiche le message correspondant:

si `n` vaut 1:
```
*
```

si `n` vaut 2:
```
*
**
```

si `n` vaut 3:
```
*
**
***
```

si `n` vaut 4:
```
*
**
***
****
```

. . . et ainsi de suite.

Exercice 4 (Nombres rationnels)

On souhaite réaliser une série de fonctions pour gérer les nombres rationnels, c'est-à-dire

les $\frac{p}{q}$ avec $p \in \mathbb{Z}$ et $q \in \mathbb{N}^*$. Un tel rationnel est représenté par une liste à deux éléments `[p, q]`. Vous pouvez utiliser la fonction `gcd(a, b)` du module `math`, qui renvoie le plus grand commun diviseur de a et b.

(a) Écrivez une fonction `simplifier(R)` qui prend un rationnel `R` au format sus-mentionné en entrée et qui renvoie la fraction simplifiée : $\frac{-2}{4}$ se simplifie en $\frac{-1}{2}$.

(b) Écrivez des fonctions pour calculer la somme, le produit, la différence et le quotient de deux rationnels.

(c) Écrivez une fonction qui teste l'égalité de deux rationnels, et une autre fonction pour tester si le premier est plus petit que le second.

Exercice 5 (Sans doublons)

Écrivez une fonction `singletons` qui prend une liste d'entiers et renvoie une liste contenant les mêmes nombres mais sans doublons et dans le même ordre. Par exemple, `singletons([2, 1, 2, 1, 3, 2, 1, 4])` renverra la liste `[2, 1, 3, 4]`.

Exercice 6 (Combien?)

(a) Écrivez une fonction prenant en paramètres une liste `L` et un nombre `v`, et renvoyant le nombre d'éléments de `L` supérieurs à `v`.

(b) Modifiez votre solution pour qu'elle ne plante pas en tentant de comparer des éléments de types incomparables.

Exercice 7 (Liste triée)

(a) Écrivez une fonction qui vérifie qu'une liste passée en paramètre est triée dans l'ordre croissant (on renvoie `True` si elle est triée, `False` sinon).

Indices

Quelle propriété doivent satisfaire les éléments pour que la liste soit triée?

(b) Écrivez une fonction qui vérifie qu'une liste passée en paramètre est triée dans l'ordre croissant ou décroissant (on renvoie `False` si elle est "désordonnée", `True` sinon).

Si vous avez fini, créez un module par séance de TD précédente (`td01prog.py`, `td02prog.py`, ...). Pour chacun des exercices qui s'y prête, réécrivez votre réponse sous la forme d'une fonction et placez-la dans le module de la séance correspondante. Par exemple, au lieu d'écrire un programme qui calcule et affiche le nombre de Fibonacci F_n, vous écrirez une fonction `fibonacci(n)` qui renverra sa valeur.