Exercices Python

August 29, 2021

1 Entraînement en Python

Pensez à toujours tester vos fonctions sur des exemples. Appelez-moi si un exercice vous pose problème.

Pour écrire une réponse à un exercice : cliquer sur l'exercice (sortez du mode édition avec Échap si besoin) et appuyer sur la touche B pour faire apparaître une case de code pour votre réponse.

2 Petits exercices

1. Écrire une fonction pour renvoyer le produit des éléments d'une liste.
2. Écrire une fonction permettant de savoir si une liste est triée par ordre croissant.
3. Écrire une fonction pour inverser une liste.
4. Déduire des deux questions précédentes une fonction pour savoir si une liste est triée par ordre décroissant.
5. Écrire une fonction vérifiant que tous les éléments d'une liste sont pairs.

3 Arithmétique

3.1 Divisibilité

Soit a et b deux entiers. On appelle q et r le quotient et le reste de la division euclidienne de a par b.
Par définition, q et r sont les uniques entiers vérifiants :

$$a = bq + r$$

$$0 \leq r < b$$

1. Rappeler comment obtenir le quotient et le reste en Python. 2. Donner une condition nécessaire et suffisante sur r pour que b divise a. 3. En déduire une fonction `divise` telle que `divise(a, b)` vaut `True` si `b` divise `a`, `False` sinon. 4. Écrire une fonction `diviseurs` renvoyant la liste des diviseurs positifs d'un entier n.
Par exemple, `diviseurs(6)` peut renvoyer `[1, 2, 3, 6]`.

3.2 Algorithme d'Euclide

Implémenter l'algorithme d'Euclide qui permet d'obtenir le PGCD de deux entiers a et b (c'est-à-dire le plus grand entier divisant a et b) :

```
Tant que b > 0 :
    r = reste de la division de a par b
    a = b
    b = r
Renvoyer a
```

Vérifier par exemple que PGCD(30, 18) = 6.

3.3 Primalité

5. On rappelle qu'un entier n est premier s'il possède 2 diviseurs positifs : 1 et n.
   Écrire une fonction `premier` déterminant si un entier n est premier.
6. Donner un ordre de grandeur du nombre d'opérations de `premier(n)` en fonction de `n`.
   Pouvez-vous obtenir de l'ordre de $\sqrt{n}$ opérations (éventuellement en modifiant votre fonction) ?

3.4 Crible d'Ératosthène

7. Écrire une fonction `tous_premiers` telle que `tous_premiers(n)` renvoie la liste des entiers premiers entre 2 et n.
   Par exemple, `tous_premiers(12)` peut renvoyer `[2, 3, 5, 7, 11]`.
8. Implémenter en Python l'algorithme suivant (crible d'Ératosthène), qui est une méthode plus efficace :

```
Initialiser une liste L de taille n+1 remplie de True (à la fin, L[i] sera True si l'entier i e

Pour chaque entier k de 2 à n:
    Mettre à False tous les multiples de k dans L (ils ne peuvent pas être premiers)
```

Crible d'Eratosthene

4 Récursivité

4.1 Somme

Écrire une fonction récursive pour calculer la somme des n premiers entiers $(1 + 2 + ... + n)$.

4.2 Recherche par dichotomie récursif

Réécrire l'algorithme de recherche par dichotomie en récursif. Voici un squelette qu'on pourra compléter (pour tester votre fonction, vous appelerez `appartient_dicho(e, L, 0, len(L)-1)`) :

```python
def appartient_dicho(e, L, i, j):  # détermine si e apparaît dans L entre les indices i et j
    m = #...
    if L[m] == e:
        # ...
    elif L[m] < e:
        # appel récursif sur la partie droite
    else:
        # appel récursif sur la partie gauche
    # ...
```

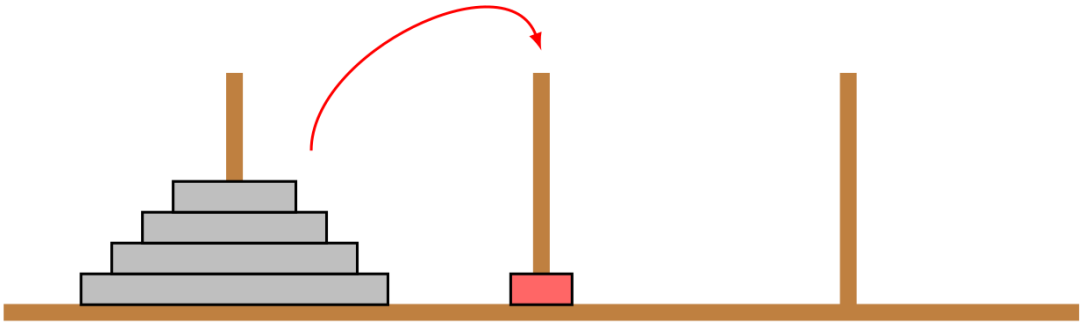
4.3 Tours de Hanoï

n disques sont posés sur la tige à gauche. L'objectif est de déplacer tous les disques sur la tige à droite :

Règles du jeu :

- On ne peut déplacer qu'un disque à la fois (celui tout en haut), sur une autre tige.
- Il est interdit de poser un disque sur un autre plus petit.

Exemple de premier déplacement valide :



On souhaite écrire une fonction récursive `hanoi` telle que `hanoi(n, tige1, tige2)` affiche une suite de déplacements (avec des `print`) permettant de déplacer n disques depuis `tige1` vers `tige2`. On supposera que les tiges sont numérotées 0, 1, 2 (de gauche à droite).

1. Supposons que vous sachiez déplacer $n-1$ disques d'une tige à une autre. Comment déplacer n disques d'une tige à une autre ?
2. Écrire `hanoi`.
3. Essayer d'écrire `hanoi` en itératif (sans fonction récursive, seulement avec des boucles). Échouez et concluez que la solution récursive est beaucoup plus simple.

4. (Seulement si vous avez du temps à perdre) Afficher graphiquement les déplacements avec `matplotlib`. (utiliser `!pip install matplotlib numpy` pour avoir les bibilothèques sur Binder)