TP11 - Récursivité

I Exponentiation rapide

Ouvrir le fichier `expo_rapide.py` et écrire en dessous de la fonction `trace`. Cette fonction simule la pile d'exécution.

Q1. Ecrire une fonction récursive `puissance(a, n)` qui a pour paramètre deux entiers `a` et `n`, avec `n` ≥ 0 et qui renvoie $\texttt{a}^{\texttt{n}}$.

```
@trace
def puissance(a, n):
    #A compléter
```

Tester la fonction avec l'instruction `puissance(5, 10)`.
On remarque qu'il y a ..... appels récursifs.

L'algorithme d'exponentiation rapide propose une solution plus efficace. L'algorithme est basé sur le fait que :

- $a^n = (a^{n//2})^2$ si n est pair
- $a^n = a \times (a^{n//2})^2$ si n est impair

Q2. Ecrire la fonction récursive `expoR(a, n)`
Tester la fonction avec l'instruction `expoR(5, 10)`.
On remarque qu'il y a ..... appels récursifs.

Q3. Comparer le nombre d'appels récursifs pour 2^{1024} dans les deux cas.

II Les tours de Hanoï

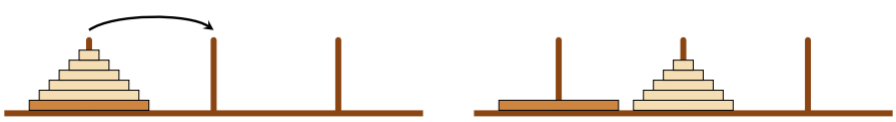
II.1 Description

Le problème des tours de Hanoï est un jeu consistant à déplacer une pile de disque d'un piquet de départ vers un piquet d'arrivée. Il n'est possible de déplacer qu'un disque à la fois, et on ne peut poser un disque que sur un disque de diamètre supérieur. La solution peut ne pas sembler évidente pour un grand nombre de disques, mais la formulation récursive de la solution est très élégante, et facile à mettre en place.

II.2 Résolution

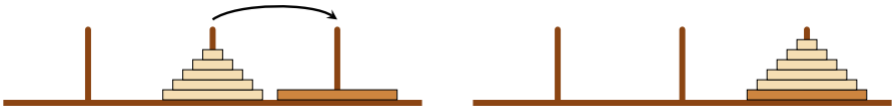
La résolution classique du problème des tours de Hanoï s'appuie sur un algorithme récursif.

1. On déplace les $n-1$ disques qui couvrent le dernier disque, sur la tige centrale.



2. On déplace le dernier disque sur la troisième tige.

3. On déplace les $n-1$ autres disques sur la troisième tige.



Ouvrir le fichier `hanoi.py`.
Dans ce fichier :

- Les tiges sont représentées par 3 listes `tige1`, `tige2`, `tige3`.
- Les disques sont représentés par des entiers : si $i < j$, le disque i a un diamètre supérieur à celui de j.
- La fonction `dessin()` permet de dessiner l'état des trois tiges.
- La fonction `lancer_hanoi(n)` permet de simuler le jeu de Hanoï pour `n` disques.

Q4. Compléter la fonction `hanoi(n, deb, mil, fin)` qui déplace n disques de la tige `deb` à la tige `fin` en utilisant la tige `mil`.

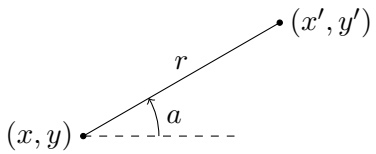
```
def hanoi(n, deb , mil, fin):
    """deb : debut
    fin : fin
    mil : en utilisant milieu
    deplace n disque de deb vers fin en utilisant milieu
    """

    if n == 1:
        a = deb.pop()
        fin.append(a)
        dessin()

    else:
        .....
        .....
        .....
```

III Un arbre fractal

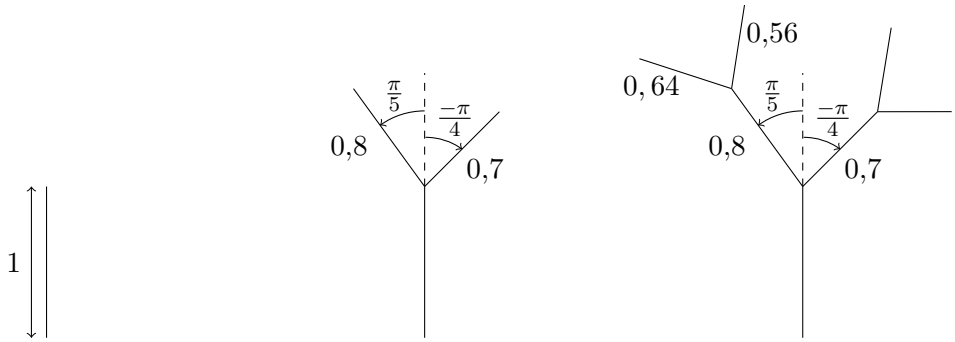
Remarque :



Exprimer x' et y' en focntion de x, y, r, a.

Un arbre fractal est un arbre dans lequel chaque branche est une réduction de l'arbre fractal.

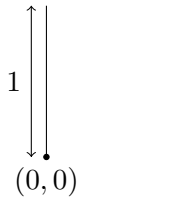
$n = 7$



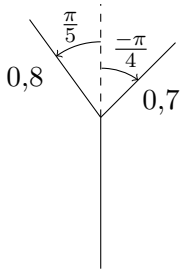
$n = 1$ $n = 2$ $n = 3$

On définit une fonction `arbre(n, a, r, x, y)` avec :

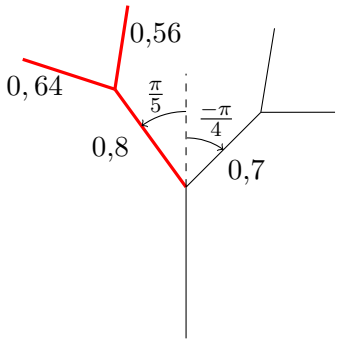
- `a` est l'angle par rapport à l'horizontale ;
- `r` est la longueur du tronc ;
- `x,y` sont les coordonnées de la base du tronc.




```
arbre(1, np.pi/2, 1, 0, 0)
```




```
arbre(2, np.pi/2, 1, 0, 0)
```
qui peut aussi s'écrire :
```
arbre(1, np.pi/2, 1, 0, 0)
arbre(1, np.pi/2+np.pi/5, 0.8, 0, 1)
arbre(.............................)
```



`arbre(3, np.pi/2, 1, 0, 0)`
qui peut aussi s'écrire :
`arbre(.....`
`arbre(2, np.pi/2+np.pi/5, 0.8, 0, 1)`
`arbre(2, .....)`

Q5. Ouvrir le fichier `arbre.py` et le compléter.