

Exercices : Récursivité

Exercice 01 Répondre aux questions suivantes :

```
1 def f(n):
2     return n * f(n-1)
```

Que se passe-t-il si on exécute l'instruction `f(2)` ?

.....

```
1 def g(n):
2     if n == 0:
3         return 1
4     else:
5         return n * g(n-1)
```

Que se passe-t-il si on exécute l'instruction `g(3)` ?

.....

```
1 def h(n):
2     if n == 0:
3         return 1
4     else:
5         return n * h(n) - 1
```

Que se passe-t-il si on exécute l'instruction `h(4)` ?

.....

```
1 def p(n):
2     if n <= 0:
3         return 1
4     else:
5         return p(n-1) + p(n-3)
```

Que se passe-t-il si on exécute l'instruction `p(4)` ?

.....

Exercice 02 Quel est le rôle des fonctions suivantes ?

```
1 def f(n): # n : entier naturel
2     if n == 0:
3         return 0
4     else:
5         return 1 + f(n // 10)
```

```
1 def g(n): # n : entier naturel
2     if n == 0:
3         return 0
4     else:
5         return (n % 10) + g(n // 10)
```

.....
.....

.....
.....

```
1 def h(n): # n : entier naturel
2     if n == 0:
3         return True
4     elif n == 1:
5         return False
6     else:
7         return n - 2
```

```
1 def p(s): # s : chaîne de caractères
2     if len(s) <= 1:
3         return True
4     elif s[0] != s[-1]:
5         return False
6     else:
7         return p(s[1:-1])
```

.....
.....

.....
.....

Exercice 03 Même exercice :

```

1 def f(s,c): # s,c : chaînes de caractères
2     if len(s) == 0:
3         return 0
4     elif s[0] == c:
5         return 1 + f(s[1:], c)
6     else:
7         return f(s[1:], c)

```

```

1 def g(L): # L : liste
2     if len(L) == 0:
3         return 0
4     elif L[0] % 2 == 0:
5         return 1 + g(L[1:])
6     else:
7         return g(L[1:])

```

Exercice 04

1) Écrire une fonction récursive `dec2bin(n)` qui retourne l'écriture binaire de l'entier `n` donné, sous forme d'une chaîne de caractères.

```

>>> dec2bin(5)
'101'
>>> dec2bin(13)
'1101'

```

2) Écrire une fonction récursive `bin2dec(b)` qui retourne l'écriture décimale de nombre binaire `b` donné.

```

>>> bin2dec('101')
5
>>> bin2dec('1101')
13

```

Exercice 05 On considère la fonction récursive suivante :

```

1 def syracuse(n): # n : entier naturel
2     if n == 1:
3         return 1
4     elif n % 2 == 0:
5         return syracuse(n // 2)
6     else:
7         return syracuse(3*n + 1)

```

On admet que cette fonction se termine toujours, en renvoyant 1 (c'est la *conjecture de Syracuse*).

Par exemple, l'appel de `syracuse(5)` produit 5 appels récursifs avant de retourner 1 :

`syracuse(5) → syracuse(16) → syracuse(8) → syracuse(4) → syracuse(2) → syracuse(1) → 1`

Écrire une fonction récursive `listeSyracuse(n)` qui retourne la liste des entiers rencontrés dans les appels récursifs.

```

>>> listeSyracuse(5)
[5, 16, 8, 4, 2, 1]
>>> listeSyracuse(13)
[13, 40, 20, 10, 5, 16, 8, 4, 2, 1]

```

Exercice 06 Que dire de l'acronyme suivant : « *WINE : Wine Is Not an Emulator* » ?