

## Fiche 06 Corrigé - Boucle TANT QUE

Informatique - Syntaxe

Soit  $x$  et  $y$  deux réels supérieurs à 1. Soit  $n$  le plus grand entier tel que  $x^n \leq y$ .

1. En utilisant une boucle `while`, déterminer  $n$ . Le pseudo-code est donné ci-dessous et doit donner comme résultat  $n = 7$ . Les premières lignes du script sont aussi données ; à compléter.

```
1 x <- 3 # indique une affectation
2 y <- 5325
3 n <- 1
4 TANT QUE x**n <= y FAIRE :
5     n <- n + 1
6 FIN TANT QUE # apres la boucle, n est l'entier le plus petit tel que x**n > y
7 n <- (n - 1)
```

Début du script à compléter :

```
1 x, y = 3, 5325
2 n = ...
3 while ...
```

### Solution :

```
1 x, y = 3, 5325
2 n = 1
3 while x**n <= y:
4     n = n + 1
5 n = n - 1
```

2. Soit la fonction ci-dessous. Quelles sont les valeurs retournées par les appels suivants : `mystere(1)`, `mystere(2)`, `mystere(4)` et `mystere(8)`. Rappel : l'opérateur `//` retourne le quotient de la division entière.

```
1 def mystere(n):
2     cpt = 0
3     while n > 0:
4         cpt = cpt + 1
5         n = n // 2
6     return cpt
```

**Solution :** dans l'ordre, les appels retournent les entiers : 1, 2, 3, 4. Cela correspond au nombre de divisions euclidiennes successives de  $n$  par 2, jusqu'à arriver à un quotient nul. Remarque : c'est aussi le nombre de chiffres nécessaire pour écrire l'entier en base 2 (en binaire).

3. Ecrire une fonction `indice(texte, c)` retournant l'indice du premier caractère  $c$  dans la chaîne de caractères `texte`. On suppose ici que le caractère  $c$  est bien présent dans `texte`. Exemple d'utilisation :

```
>>> phrase = "tout le monde est lié"
>>> indice(phrase, 'u')
2
>>> indice(phrase, 't')
0
```

### Solution :

```
1 def indice(texte, c):
2     i = 0
3     while texte[i] != c:
4         i = i + 1
5     return i
```

4. Modifier le code pour traiter le cas de la recherche d'un caractère non présent dans texte. Il faut alors retourner la valeur -1. Il faut arrêter la recherche si l'on atteint la fin de la chaîne de caractères. Exemple d'utilisation :

```
>>> indice(phrase, 'x')  
-1
```

#### Solution :

```
1 def indice(texte, c):  
2     n = len(texte)  
3     i = 0  
4     while i < n and texte[i] != c: # tant que l'on n'a pas atteint la fin de texte  
5                                     # et que texte[i] différent de c  
6         i = i + 1  
7  
8     if i == n: # la fin du texte a été atteinte  
9         return -1  
10    else:  
11        return i
```

On peut aussi écrire :

```
1 def indice(texte, c):  
2     n = len(texte)  
3     i = 0  
4     while i < n # tant que l'on n'a pas atteint la fin de texte  
5         if texte[i] == c: # si le caractère est trouvé  
6             return i  
7         i = i + 1 # sinon on passe au caractère suivant  
8  
9     return -1 # le caractère n'a pas été trouvé
```