PÔLE CHATEAUBRIAND JOLIOT-CURIE

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.

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

Début du script à compléter :

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

Solution :

```
x, y = 3, 5325
n = 1
while x**n <= y:
    n = n + 1
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.

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

Solution : dans l'ordre, les appels retournent les entiers : 1, 1, 2, 3. 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 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 :

```
def indice(texte, c):
    i = 0
    while texte[i] != c:
        i = i + 1
    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 :

```
def indice(texte, c):
    n = len(texte)
    i = 0
    while i < n and texte[i] != c: # tant que l'on n'a pas atteint la fin de texte
                                   # et que texte[i] différent de c
        i = i + 1

    if i == n: # la fin du texte a été atteinte
        return -1
    else:
        return i
```

On peut aussi écrire :

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

    return -1 # le caractère n'a pas été trouvé
```