Boucles_while_seance_17_novembre

November 23, 2020

0.1 Boucle while

Exemple de boucle *while* classique avec compteur: ici on affiche 7 fois le mot "bonjour". Pour rendre le code plus facile à modifier on a utilisé une variable `max` qui correspond au nombre de tours de la boucle.

[1]:
```
max = 7
c = 0
while c < max:
    print("bonjour")
    c += 1
```

```
bonjour
bonjour
bonjour
bonjour
bonjour
bonjour
bonjour
```

Exemple de boucle sans compteur: ici c'est une condition sur la variable qu'on fait évoluer qui détermine l'arrêt de la boucle. Cette boucle va se répéter autant combien de fois qu'il faut diviser 134 (variable `val`) pour arriver à une valeur inférieure à 1. A chaque tour de boucle on affiche la valeur courante.

[2]:
```
val = 134
while val > 1:
    val = val / 2
    print (val)
```

```
67.0
33.5
16.75
8.375
4.1875
2.09375
1.046875
0.5234375
```

A partir du cas précédent: même si le nombre de tours de boucle est déterminé par autre chose

qu'un compteur, on peut ajouter un compteur: cette fois-ci la variable `c` ne sert pas à décider quand on arrête la boucle, mais simplement à compter le nombre de tours effectués.

[3]:
```
val = 78
while val > 1:
    val = val / 2
    c = c + 1
    print (val)
print("il a fallu %d tours" % c)
```

```
39.0
19.5
9.75
4.875
2.4375
1.21875
0.609375
il a fallu 14 tours
```

Exercice : Table de division entière: produire l'affichage suivant:

```
7 = 1 * 7 + 0
7 = 1 * 6 + 1
7 = 1 * 5 + 2
7 = 1 * 4 + 3
7 = 2 * 3 + 1
7 = 3 * 2 + 1
```

[4]:
```
# Version directe (voir corrigé exercice)
nb_div = 7
div = nb_div
while (div > 1):
    print("%d = %d * %d + %d" % (nb_div,nb_div//div,div,nb_div%div))
    div = div - 1
```

```
7 = 1 * 7 + 0
7 = 1 * 6 + 1
7 = 1 * 5 + 2
7 = 1 * 4 + 3
7 = 2 * 3 + 1
7 = 3 * 2 + 1
```

[5]:
```
# Version généralisée: fonction ayant comme paramètres
# le nombre dont on écrit la table et le nombre de lignes
def table_div_entiere(n,nbl):
    div = nbl
    while (div > 1):
        print("%d = %d * %d + %d" % (n,n//div,div,n%div))
        div = div - 1
```

```
table_div_entiere(12,10)
```

```
12 = 1 * 10 + 2
12 = 1 * 9 + 3
12 = 1 * 8 + 4
12 = 1 * 7 + 5
12 = 2 * 6 + 0
12 = 2 * 5 + 2
12 = 3 * 4 + 0
12 = 4 * 3 + 0
12 = 6 * 2 + 0
```

0.2 Structure conditionnelle (test)

Deux exemples très simples d'utilisation de l'instruction `if` sans ou avec la contrepartie `else`. Dans la fonction `aff_pair` on n'execute l'affichage que si `n` est pair, on ne fait rien sinon. Dans la seconde fonction, on a un comportement prévu aussi bien quand le test est positif que quand il est négatif.

[6]:
```
def aff_pair(n):
    if n % 2 == 0:
        print(n, ' est pair')

aff_pair(5)
aff_pair(6)

def aff_pair_impair(n):
    if n % 2 == 0:
        print(n, ' est pair')
    else:
        print(n, ' est impair')

aff_pair_impair(5)
aff_pair_impair(6)
```

```
6  est pair
5  est impair
6  est pair
```

Exemple: voici un programme qui affiche les diviseurs pairs (et seulement eux) du nombre 28 (variable val).

[7]:
```
val = 28
div = 2
while (div < val):
    if val % div == 0:
        if div % 2 == 0:
            print("%d est un diviseur pair de %d" % (div,val))
```

```
    div += 1
```

```
2 est un diviseur pair de 28
4 est un diviseur pair de 28
14 est un diviseur pair de 28
```

Exemple: programme qui saisit au clavier un nombre (en demandant un nombre pair) et qui vérifie que le nombre est pair.

[8]:
```
nb = int(input("Saisir un nombre pair "))
if nb % 2 != 0:
    print('On a dit un nombre pair!')
```

```
Saisir un nombre pair 89
On a dit un nombre pair!
```

Exemple: programme qui saisit au clavier un nombre et affiche si ce nombre est pair ou impair.

[9]:
```
nb = int(input("Saisir un nombre "))
if nb % 2 == 0:
    print('Vous avez saisi un nombre pair')
else:
    print("Vous avez saisi un nombre impair")
```

```
Saisir un nombre 79
Vous avez saisi un nombre impair
```

Exemple: programme qui saisit deux nombres *max* et *puiss* et affiche les *max* premières puissances de *puiss*. On réutilisera la fonction `aff_puiss()` déjà définie par ailleurs.

[10]:
```
# Définition de la fonction aff_puiss
def aff_puiss(fact,n):
    p = 1
    c = 1
    while c <= n:
        print(p)
        p = p * fact
        c += 1

max = int(input("Combien de termes ? "))
puis = int(input("Quelle puissance ? "))
aff_puiss(puis,max)
```

```
Combien de termes ? 7
Quelle puissance ? 2
1
2
4
8
16
```

```
32
64
```

Exemple: reprendre le programme qui saisit un nombre et affiche sa parité, et créer une boucle d'interaction permettant à l'utiulisateur de refaire plusieurs fois la manoeuvre.

[11]:
```python
# démo: boucle de test parité avec recommencer
rep = "o"
while rep == "o":
    n = int(input("Donnez un nombre "))
    if n % 2 == 0:
        print("Vous avez donné un nombre pair")
    else:
        print("Vous avez donné un nombre impair")
    rep = input("Voulez-vous recommencer ? ")
```

```
Donnez un nombre 87
Vous avez donné un nombre impair
Voulez-vous recommencer ? o
Donnez un nombre 44
Vous avez donné un nombre pair
Voulez-vous recommencer ? o
Donnez un nombre 9
Vous avez donné un nombre impair
Voulez-vous recommencer ? n
```

La même chose mais plus lisible car on sépare l'action qui est répétée et le mécanisme de contrôle de cette répétition.

[12]:
```python
def saisie_pair_impair():
    n = int(input("Donnez un nombre "))
    if n % 2 == 0:
        print("Vous avez donné un nombre pair")
    else:
        print("Vous avez donné un nombre impair")

rep = "o"
while rep == "o":
    saisie_pair_impair()
    rep = input("Voulez-vous recommencer ? ")
```

```
Donnez un nombre 90
Vous avez donné un nombre pair
Voulez-vous recommencer ? n
```