boucle_while_correction

September 11, 2019

1 Boucles non bornées : les boucles `while`

À la différence essentielle des boucles `for`, dont on peut savoir à l'avance combien de fois elles vont être exécutées, les boucles `while` sont des boucles indéterminées.
Avec donc le risque de rester infiniment bloqué à l'intérieur !

1.1 Exemple

[1]:
```
a = 0
while a < 3 :
    print("ok")
    a = a + 1
print("fini")
```

```
ok
ok
ok
fini
```

Vous pouvez visualiser l'exécution de ce code pas-à-pas sur le site pythontutor.
Remarque : le code ci-dessous va-t-il donner un résultat différent ?

[2]:
```
a = 0
while a < 3 :
    a = a + 1
    print("ok")
print("fini")
```

```
ok
ok
ok
fini
```

Conclusion : l'évaluation de la condition ne se fait pas à chaque ligne mais bien au début de chaque tour de boucle. Si la variable qui déclenchera la sortie de boucle atteint sa valeur de sortie au milieu des instructions, les lignes restantes sont quand même exécutées.

1.2 Les pièges ...

1.2.1 piège nř1 : ne jamais sortir de la boucle

[ ]:
```
a = 0
b = 0
while a < 10 :
    a = a * b
    print("flood")
    a = a + 1
print("ce texte ne s'écrira jamais")
```

1.2.2 piège nř2 : ne jamais ENTRER dans la boucle

[4]:
```
a = 0
b = 0
while a > 10 :
    print("ce texte non plus ne s'écrira jamais")
    a = a + 1

print("fini")
```

```
fini
```

1.3 Exercice 1

Trouver le plus petit nombre entier n tel que 2^n soit supérieur à 1 milliard.

[ ]:
```
n = 1
while 2**n < 10**9 :
    n = n + 1
    print("trop petit")
print("trouvé : ",n)
```

1.4 Exercice 2

Demander à l'utilisateur de taper la lettre S (puis sur la touche Entrée). Recommencer tant qu'il n'a pas obéi.

[ ]:
```
touche = ""  #chaine de caractère vide

while touche != 'S' :
    touche = input("appuyez sur S s'il vous plaît ")

print("ouf, merci !")
```